LECTURES ON THE THEORY OF SEARCH\textsuperscript{1}

by

ALFRED RENYI\textsuperscript{2}

Department of Statistics

University of North Carolina at Chapel Hill

Institute of Statistics Mimeo Series No. 600.7

MAY 1969

1. Research partially sponsored by the National Science Foundation under Grant No. - GU-2059.

\textsuperscript{2}University of North Carolina and Hungarian Academy of Sciences
CHAPTER 1. Introduction

§1.1. The aims of these lectures.

Problems of search occur in practically every field of human activity. Some typical examples of search are:

1) search for a failure in a system, e.g. a computer, a car, etc.
2) search for a lost or hidden object
3) medical diagnosis
4) chemical analysis
5) search for a mistake in a series of computations, or in a program for a computer, or in a proposed proof of a theorem, etc.
6) search for some items of information in a library
7) search for a mathematical object with prescribed properties (the root of an equation, the maximum of a function, the construction of some combinatorial design, etc.)
8) decoding a received message
9) search for the parameter of a distribution by taking a sample
10) guessing games, like the game of 20 questions

Many particular problems of search have been investigated by setting up a mathematical model of the situation. However, up to now there does not exist a general mathematical theory of search. The aim of these lectures is to deal with the theory of search in a more systematic way.
as has been done previously. We shall try to point out the outlines of a general theory, hoping that this may contribute to the development of this field of research, which, besides its obvious importance from a practical point of view, presents many challenging unsolved mathematical problems; one of the aims of these lectures is to call attention to these problems.

§1.2. The relation of search theory to other branches of mathematics.

Search theory—as interpreted in these lectures—is a chapter of information theory. If we are searching for something, even if this is a material object, our endeavour can be described by saying that we want to get certain information, about the thing searched. The process of search can be characterized as the process of collecting bits of information until the information obtained is sufficient for our purposes: after this, the information obtained has to be processed, the information needed has to be extracted from the data, i.e. the information we need separated from the information which is unnecessary for our purposes. This being the case it is clear that a close connection exists between search theory and other branches of information theory, as coding theory, the theory of pattern recognition, etc.

Problems of search are also closely connected with mathematical statistics, because the collection of information on the object searched is a special sort of sampling. To put it otherwise: searching means to make indirect observations and to extract from these the information needed. Search theory being closely connected with both information
theory and statistics, it is evident, that is is especially closely connected with the information-theoretical point of view in statistics. Looking at problems of search as statistical problems, one usually can take the Bayesian point of view.

Search theory leads to many highly interesting combinatorial problems; conversely, results of combinatorial mathematics obtained originally for other purposes can be often successfully applied in solving problems of search. Thus combinatorial mathematics is one of those branches of mathematics which are most intimately connected with search theory.

§1.3. The general mathematical model of a search process.

In what follows we shall deal with the following model of a search process:
There is given a nonempty set $S$: the set of all possible objects of search. We shall call $S$, for the sake of brevity, the basic set. We suppose that the unknown thing we are searching for is an element of the set $S$: we shall denote it by $x^*$. Thus the aim of the search is to find $x^*$ in $S$. In general we shall suppose that $x^*$ is a random variable, $x^* = x^*(\omega)$, $(\omega \in \Omega)$, defined on a probability space $(\Omega, \mathcal{A}, P)$, with values in $S$: in other words we suppose that there is given on $S$ a prior distribution of $x^*$, defined by $P(A) = P(x^* \in A)$. (In case $S$ is finite or denumerable, we suppose that $P(A)$ is defined on all subsets of $S$; in the general case $P(A)$ is defined only on some $\sigma$-algebra of subsets of $S$). We suppose further that a family $F$ of functions $f$ defined on $S$ is given, and that the search process consists in selecting one after the other certain functions $f_1, f_2, \ldots$ from the family $F$.
and observing their values at the (unknown) point \( x^* \); thus \( x^* \) has to be identified on the basis of the observed values \( f_1(x^*), f_2(x^*), \ldots \). Each function \( f \in F \) is called a test-function, and \( F \) itself the family of admissible test-functions. The search is called noiseless if the functions \( f \in F \) depend only on \( x \); it is called search in the presence of noise (or simply noisy search) if each \( f \in F \) for each fixed \( x \in S \) is a random variable i.e. \( f = f(x, \omega) \) defined on the probability space \((\Omega, A, P)\). [Of course the distribution of the random variable \( f(x, \omega) \) has to depend on \( x \), otherwise it does not give any information on \( x \).]

A strategy is a rule for selecting from \( F \) one after the other the test-functions \( f_1, f_2, \ldots \) to be observed at the point \( x^* \). A strategy is sequential if the choice of the function \( f_k \), \((k=2,3,\ldots)\), depends on the previous observations \( f_1(x^*), f_2(x^*), \ldots, f_{k-1}(x^*) \); otherwise the strategy is called non-sequential (or predetermined). Thus in the case of a non-sequential strategy the sequence \( f_1, f_2, \ldots \) is specified by the strategy in advance. (Sequential strategies may be also called strategies with feedback.) A strategy (independently whether it is sequential or not) may specify the choice of the function \( f_k \), \((k=1,2,\ldots)\), in \( F \) uniquely, or it may prescribe, at each stage, only a probability distribution on the subsets of \( F \), according to which the function \( f_k \) has to be chosen from \( F \). In the second case we call the search procedure a random search.

The above described general model will be more specified in what follows. In the following § we discuss the classification of search problems according to several points of view.
Of course if \( F \) contains at least one function which takes on different values for different points in \( S \), then observing such a function at the point \( x^* \) this single observation determines \( x^* \) uniquely, and the search is finished. However in most problems of search no such function is contained in \( F \): As a matter of fact the number of possible values of every function \( f \) in \( F \) is usually very small compared with the number \( |S| \) of elements of the basic set \( S \). Usually we shall suppose that the values of the functions \( f \in F \) are integers, and the family \( F \) is finite or denumerable.

Let us introduce already here one more definition: We call a search problem simple, if the following conditions are satisfied:

\( S \) is a finite set, and \( F \) consists only of such function \( f(x) \) (depending only on \( x \)) which take on a finite number of different values only: Without the restriction of generality we may suppose that these values are \( 0, 1, \ldots, q-1 \) for some \( q \geq 2 \). Thus a simple search problem is one in which the unknown \( x^* \) may take on a finite number of values only, the search is noiseless, and each admissible test function is capable to take on the values \( 0, 1, \ldots, q-1 \) only.

§1.4. Classification of search problems.

In this § we shall classify search problems according to ten different points of view, giving some comments on the different types of problems.

A) Search problems can be classified according to the cardinal number of the basic set \( S \). If \( S \) is finite or denumerable, we shall
call the search problem **discrete**. In what follows we shall mainly consider
discrete search problems. Evidently, if the search problem is not
discrete, and the set of possible values of each admissible function
\( f \in F \) is finite, then the search procedure can in general be not terminated
after a finite number of steps, provided we want to specify \( x^* \) exactly
(see however C)).

**B)** The basic set \( S \) may or may not possess some structure, which
is relevant for the search process. In case of discrete search problems,
usually no structure in \( S \) is supposed (of course it may be made a
completely ordered set by labelling its elements); if \( S \) is nondenumerable,
then usually there is some structure in \( S \) given, e.g. \( S \) is a metric
space or a Euclidean space with given dimension, etc.

**C)** An important distinction is whether we want to find \( x^* \) exactly,
or we are contented with localizing \( x^* \) to a "small" subset of \( S \);
[this may be expressed also as follows: We want to find the value of a
certain function \( g(x^*) \) of \( x^* \)]. This point of view is usually taken
if the problem is not discrete: In this case it may be possible to
localize \( x^* \) to a sufficiently small subset in a finite number of
steps, even if each admissible test function takes on a finite number of
values only.

**D)** One can classify search problems as **Bayesian** or **non-Bayesian**,
according to whether a prior distribution of \( x^* \) on \( S \) is given or not.
However in the case when \( S \) is finite, this distinction is not really
fundamental, because if a prior distribution of \( x^* \) is not specified,
we may suppose nevertheless that \( x^* \) is a random variable with uniform
distribution over S, i.e. that \( x^* \) is equal to any element \( x \) of \( S \) with the same prior probability \( |S|^{-1} \).

E) The important distinction between search under the presence of noise and noiseless search has been introduced already in §1.3. We want to add here only two remarks.

a) It is not a restriction to suppose that in the case of search in the presence of noise the random variables \( f(x, \omega) \) are defined on the same probability space as \( x^*(\omega) \); if these random variables were defined on different probability spaces, we could pass to the product space.

b) In the case of search in presence of noise, we shall usually suppose that for each fixed value of \( x \in S \) the random variables \( f(x, \omega) \) (where \( f \) ranges over all functions in the class \( F \)) are independent. Of course this does not mean that the observations \( f_1(x^*(\omega), \omega), f_2(x^*(\omega), \omega), \ldots \) are independent.

The following classifications concern the strategy employed.

f) We distinguish between systematic and random strategies. A systematic strategy is one that specifies at each stage exactly the element \( f \in F \) to be chosen as the next test function which has to be observed at the point \( x^* \). A random strategy is one that does not specify exactly which function has to be chosen, it specifies only a probability distribution on \( F \) according to which the next function \( f \) has to be chosen. A search process with a random strategy is called a random search. If \( F \) is finite or denumberable, and its elements are labeled by the integers, i.e. \( F = \{f_1, f_2, \ldots, f_n, \ldots\} \) a random strategy may be
described as a strategy consisting in observing one after the other the functions \( f_{k_1}, f_{k_2}, \ldots \) where \( k_1, k_2, \ldots \) are positive integer-valued random variables. Again it may be supposed without restricting the generality, that \( k_1, k_2, \ldots \) are random variables on the same probability space \((\Omega, A, P)\) on which \( x^*(\omega) \) is defined. Systematic strategies may also be called pure strategies, and random strategies mixed strategies, in analogy with the terminology of game theory.

G) Strategies can be classified—as already mentioned—as sequential or non-sequential (predetermined). If a sequential strategy is employed, the selection of functions \( f_1, f_2, \ldots \) to be observed is of course continued until the observations \( f_1(x^*), f_2(x^*), \ldots f_N(x^*) \) give us the necessary information; in case we want to determine \( x^* \) uniquely, this means that we stop at the least value of \( N \) such that the values \( f_1(x^*), f_2(x^*), \ldots f_N(x^*) \) determine \( x^* \) uniquely. In this case \( N \) usually depends on the value of \( x^* \); thus if \( x^* \) is considered as a random variable, then \( N \) is a random variable too. We call \( N \) the duration of the search. On the other hand, in the case of a non-sequential strategy \( N \) is fixed, and it is irrelevant whether the values \( f_1(x^*), \ldots, f_N(x^*) \) are observed one after the other or simultaneously. However, even in the case of a nonsequential strategy it may occur that a segment \((f_1(x^*),\ldots,f_M(x^*))\) of the sequence \((f_1(x^*),\ldots,f_N(x^*))\) is sufficient to determine \( x^* \) uniquely where the value of \( M \leq N \) depends on \( x^* \); thus the duration of the search may depend, even in the case of a non-sequential strategy, on the value of \( x^* \). A search strategy with variable duration will be called a \textit{variable-length strategy}, and a strategy with fixed duration of search a \textit{fixed-length strategy}. 
H) A strategy may aim at determining $x^*$ with certainty (with probability 1) or we may be contented with identifying $x^*$ with probability exceeding $1 - \varepsilon$, $(0 < \varepsilon < 1)$. Note that this is not the same distinction as mentioned under C): If we have identified $x^*$ with a certain probability $p > 1 - \varepsilon$, this does not mean that we have localized it with certainty to a small subset of $S$, as with probability $1-p$ it may be anywhere in $S$. Of course the two points of view may be combined: e.g. we may aim only at confining $x^*$ with probability $\geq 1 - \varepsilon$ to a "small" subset of $S$.

J) Search problems can be classified according to requirements on the strategy employed. We may aim for instance at minimizing the duration of the search, or--if the durations is random--the expected duration of the search. More generally, different costs may be attached to different test functions, and we may aim at minimizing the expectation of the overall cost of the search process. However we may have some other objective, e.g. minimizing the maximal duration (or cost) of the search process. [Of course the word "cost" is used here in the same general sense as in decision theory: Thus for instance the cost of the use of a test function may mean the duration of the observation and evaluation of this particular test function.]

K) A delicate distinction which is of importance especially in connection with random strategies is the following. If we consider a strategy, which is "good" in the sense that it leads to the identification of the unknown $x^*$ by a relatively small expected number of steps, it is not sure, that the same strategy works as well for all other values of the unknown too. Let us denote by $N(x, \omega)$ the duration of the
search for \( x \) corresponding to a given strategy; we may require that the expectation of \( N(x^*(\omega), \omega) \) should be small, or we may require more, namely that the expectation of \( \max_x N(x, \omega) \) should be small. In the first case we may call the strategy "statistically good", and in the second case "universally good".

These alternatives can be combined in various ways, which leads to a great variety of types of search problems.

§1.5. Discussion of the scope of the model of §1.3.

There are problems of search which at the first sight seem not to be covered by the model described in §1.3. In this § we want to show how some of these problems can be nevertheless treated by a model of the type described in §1.3.

In some problems of search we want to find instead of a single element of a basic set \( C \), a certain subset \( c \) of \( C \). For instance consider the well known group of problems: To find the counterfeit coin (or coins) in a set \( C \) of coins. It seems at the first sight, that our model can be used only if it is known in advance that there is exactly one counterfeit coin in the set \( C \). However if this is not the case, we may take as our basic set \( S \) the power set of \( C \) (i.e. the set \( P(C) \) of all subsets of \( C \)) and in this way the general problem of counterfeit coins is covered by our model.

In problems of pursuit one aims at finding a moving object, i.e. the location of the object searched for changes during the search process. Such problems can be described by a model as described in §1.3. by
taking as the basic set the set of all admissible trajectories of the unknown. In such a problem we may aim only at hitting the moving target after a certain number of steps, and its trajectory may be irrelevant. This means that denoting by $x^*$ a trajectory, we may be interested, instead of specifying $x^*$ exactly, only in finding the value of a function of $x^*$, say $g(x^*)$. This again fits into the model easily (see C).

There are problems of search in which different restrictions are made concerning the choice of the 1st, 2nd, 3rd, ... test function. Such restrictions may be taken into account, without changing the basic model, as additional conditions on the choice of the strategy.

§1.6. Some examples

H. Steinhaus [1] has raised the following problem: How many pairwise comparisons are needed to arrange $n$ different real numbers according to their order of magnitude? The problem is not yet completely solved; the best results have been obtained by L.R. Ford, Jr. and S.M. Johnson [2]. Denoting the least number of comparisons needed by $\xi(n)$, the exact value of $\xi(n)$ is known for $n \leq 12$ and is as follows:

<table>
<thead>
<tr>
<th>$n$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\xi(n)$</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>13</td>
<td>16</td>
<td>19</td>
<td>22</td>
<td>30</td>
</tr>
</tbody>
</table>

$\xi(13)$ is either 33 or 34, and in general upper and lower bounds are known for $\xi(n)$ for arbitrary $n$, the lower bound being the least integer $\leq \log_2 n!$, and the upper bound being larger only by $\dagger$ For $n=12$ it has been shown by Wells [4] that $\xi(12)=30$ is the least possible value. $n=12$ is the least number for which $\xi(n)$ is larger than the lower bound $\lceil \log_2 n! \rceil$. See also Césari [5].
a term $\sim cn$ where $c$ is a positive constant ($c=0.028...$). Let us consider in detail the first nontrivial case (in which not all comparisons have to be made) that is the case $n = 4$. Let us put

$$M(x, y) = \max(x, y) \quad \text{and} \quad m(x, y) = \min(x, y).$$

A possible procedure is as follows: If the 4 numbers in question are $x_1, x_2, x_3, x_4$ and the same numbers arranged in increasing order of magnitude are $x_1^* < x_2^* < x_3^* < x_4^*$; compare first $x_1$ with $x_2$, determining thereby $M(x_1, x_2)$ and $m(x_1, x_2)$; now compare $x_3$ with $x_4$, determining thereby $M(x_3, x_4)$ and $m(x_3, x_4)$; now compare $M(x_1, x_2)$ with $M(x_3, x_4)$; also compare $m(x_1, x_2)$ with $m(x_3, x_4)$; finally compare $m(M(x_1, x_2), M(x_3, x_4))$ with $M(m(x_1, x_2), m(x_3, x_4))$.

As we have clearly

$$x_1^* = m(m(x_1, x_2), m(x_3, x_4))$$

$$x_2^* = m(m(M(x_1, x_2), M(x_3, x_4)), M(m(x_1, x_2), m(x_3, x_4)))$$

$$x_3^* = M(m(M(x_1, x_2), M(x_3, x_4)), M(m(x_1, x_2), m(x_3, x_4)))$$

$$x_4^* = M(M(x_1, x_2), M(x_3, x_4))$$

by the mentioned 5 comparisons we obtain the sequence $x_1^*, x_2^*, x_3^*, x_4^*$.

Let us consider now how this problem of search can be described by a model of the type considered in §1.3. We may suppose without restricting the generality, that the $n$ numbers in question are the numbers $1, 2, \ldots n$, in some order, i.e. that the basic set $S$ is the set of all $n!$ permutations of the first $n$ natural numbers and that $x^*$ is an
element of $S$. Let $F$ denote the family of all functions $g_{i,j}(x)$ defined on $S$ ($1 \leq i < j \leq n$) as follows: If $x = (x_1, x_2, \ldots, x_n)$ then $g_{i,j}(x) = 1$ if $x_i > x_j$ and $g_{i,j}(x) = 0$ if $x_i < x_j$. The admissible strategies in the Steinhaus problem are those which consist in choosing a sequence of elements $f_k$ ($k=1,2,\ldots$) from $F$ such that the choice of $f_k$ may depend on the previously chosen functions and on their observed values. Thus e.g. for $n = 4$ the strategy given above may be described as follows:

a) $f_1 = g_{1,2}$
b) $f_2 = g_{3,4}$
c) $f_3$ is defined as follows

- if $f_1(x^*) = 0$, $f_2(x^*) = 0$ choose $f_3 = g_{24}$
- if $f_1(x^*) = 0$, $f_2(x^*) = 1$ choose $f_3 = g_{23}$
- if $f_1(x^*) = 1$, $f_2(x^*) = 0$ choose $f_3 = g_{14}$
- if $f_1(x^*) = 1$, $f_2(x^*) = 1$ choose $f_3 = g_{13}$

The choice of $f_4$ depends similarly on the values $f_1(x^*), f_2(x^*), f_3(x^*)$ and the choice of $f_5$ on the values $f_1(x^*), f_2(x^*), f_3(x^*)$ and $f_4(x^*)$.

A search problem which is closely connected to, but nevertheless different from that of Steinhaus has been considered by R.C. Bose and R.J. Nelson [3]. This problem consists in arranging a sequence of $n$ given different real numbers in ascending order by a minimal number of pairwise comparisons, so that after each comparison of two numbers if that with the smaller index is larger the two numbers are interchanged, while if that with the smaller index is smaller, the sequence is left unchanged. Denoting by $L(n)$ the minimal number of such comparisons, Bose and Nelson gave an algorithm, which can be performed by a computer,
and leads to the required permutation in $L_1(n)$ steps, where $L_1(n)$ is a rather sharp upper bound for $L(n)$. For the sake of comparison we give the first 10 values of $L_1(n)$:

<table>
<thead>
<tr>
<th>$n$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1(n)$</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>12</td>
<td>16</td>
<td>19</td>
<td>27</td>
<td>32</td>
<td>38</td>
</tr>
</tbody>
</table>

Notice that $L_1(n) = \ell(n)$ for $n \leq 4$ and $L_1(n) > \ell(n)$ for $5 \leq n \leq 11$.

The problem is of importance in digital computer programming as it often occurs that a collection of data has to be arranged according to their magnitude. To make the comparison with the Steinhaus problem easier, let us suppose, that the computer which carries out the algorithm writes down a 1 each time when two elements have been interchanged and a 0 if two elements have been compared, but no interchange was necessary. Clearly the resulting sequence of zeros and ones determines the permutation in question uniquely, and the permutations can be easily reconstructed from this sequence, reading it backwards. To describe this search problem by a model of the type introduced in §1.3, let $S$ and $F$ have the same meaning as above, and to each function $g_{i,j}$ in $F$ let correspond a transformation $G_{i,j}$ of the sequence $x = (x_1, x_2, \ldots, x_n)$ consisting in interchanging in $x$ the elements $x_i$ and $x_j$ if $x_i > x_j$ and leaving $x$ unaltered if $x_i < x_j$ ($1 \leq i < j < n$). With this notation the restrictions on the search strategy in this problem are as follows:

a) $f_i$ has to be chosen from the set $F$.

b) If $f_i = g_{i}$ and $G_i$ denotes the transformation corresponding
to $g_1$ then $f_2$ has to be of the form $f_2(x) = g_2(G_1x)$ where $g_2 \in F$.

c) Similarly if $f_1 = g_1$, $f_2 = g_2$ then $f_3$ has to be of the form $f_3 = g_3(G_2G_1x)$ where $G_i$ is the transformation corresponding to $g_i$ (i=1,2) and $g_3 \in F$, etc.

Notice that the functions $f_k$ do all belong to $F$. Moreover, $f_k$ can be any element of $F$, for each value of $k$.

Notice that for the special case $n = 4$, the solution of the Steinhaus problem is at the same time also a solution of the Bose-Nelson problem, i.e. the sequence of comparisons and corresponding transpositions (1, 2), (3, 4), (2, 4), (1, 3), (2, 3) rearranges a sequence of four numbers always into an ascending sequence, i.e. we may put $g_1 = g_{1,2}$, $g_2 = g_{3,4}$, $g_3 = g_{2,4}$, $g_4 = g_{1,3}$, $g_5 = g_{2,3}$.

The reason why for $n > 5$ more comparisons are needed in the Bose-Nelson case than in the Steinhaus case can be—rather heuristically—explained as follows: In the Steinhaus problem at each stage all the information obtained from the previous steps is available, while in the Bose-Nelson case only a compound result of the previous steps is available, i.e. we have less information; thus while at each step the set of admissible functions is the same, in the Bose-Nelson case we have less information concerning the most appropriate choice of the next test function.
§1.7. Irreducible strategies.

In this § we consider only simple sequential search, as defined at the end of §1.3. Let us consider a strategy, which consists in observing one after the other the test functions \( f_1, f_2, \ldots \) where the choice of \( f_k \) depends in general on the previously observed values \( f_i(x^*) \) \((i < k)\). In what follows when speaking about strategies, we mean only correct strategies, i.e. such that for each value of \( x^* \in S \) there exists a number \( N(x^*) \) such that the values \( f_k(x^*) \) for \( k = 1, 2, \ldots N(x^*) \) determine \( x^* \) uniquely. For such a strategy and for each \( k \) and \( x^* \in S \) such that \( k \leq N(x^*) \) let \( S_k(x^*) \) denote the set of those \( x \in S \) for which \( f_j(x) = f_j(x^*) \) for \( j = 1, 2, \ldots k \). Clearly we have, putting \( S_0(x^*) = S \) for each \( x^* \),

\[
S_{N(x^*)}(x^*) = \{x^*\}
\]

and

\[
S_k(x^*) \subseteq S_{k-1}(x^*) \text{ for } k = 1, 2, \ldots N(x^*) .
\]

A strategy is called irreducible if for every \( x^* \in S \) and for each \( k \leq N(x^*) \), \( S_k(x^*) \) is different from \( S_{k-1}(x^*) \) (i.e. is a proper subset of \( S_{k-1}(x^*) \)). Notice that this implies that \( S_k(x^*) \) has more than one element for every \( k \leq N(x^*) \); as clearly \( S_k(x^*) \) is the set of those \( x \in S \) which are not yet excluded as possible values of \( x^* \) after having observed \( f_j(x^*) \) for \( j \leq k \), this means that if we apply an irreducible strategy, we never continue the search once \( x^* \) has been identified. More generally, the irreducibility of a strategy can be interpreted as follows: A strategy is irreducible if applying this strategy no unnecessary observations of test functions are made.
References to Chapter 1.


CHAPTER 2. Simple search and its description by a search tree resp. by a code

§2.1. Description of a simple search by a tree.

Let us consider a simple search problem, i.e. let us suppose that there is given a finite basic set \( S \), and a family \( F \) of functions on the set \( S \) taking on only a finite number of values each. We suppose that the search is noiseless, and we consider only systematic sequential strategies. (A non-sequential strategy may be considered as a special sequential strategy.) Let us put \(|S| = n \geq 2\). In this case the search process consists in observing one after the other the values \( f_1(x^*), f_2(x^*), \ldots, f_N(x^*) \) where \( N \) may depend on \( x^* \). Without restricting the generality, we may suppose that each function \( f \in F \) takes on the values 0, 1, \ldots, \( q-1 \) only (\( q \geq 2 \)). After having observed the values of \( f_1(x^*), f_2(x^*), \ldots, f_r(x^*) \) where \( r \leq N = N(x^*) \), the information on \( x^* \) obtained can be summarized as follows: \( x^* \) belongs to the set of those elements \( x \) of \( S \) for which the equations \( f_k(x) = f_k(x^*) \) \((1 \leq k \leq r)\) hold. (If \( r = N(x^*) \) this set contains only \( x^* \).) Let us denote by \( S(e_1, e_2, \ldots, e_r) \) the set of those elements of \( S \) for which \( f_k(x) = e_k \) \( (1 \leq k \leq r) \), where \( e_k \) is for each \( k \leq r \) an element of the set \( \{0, 1, \ldots, q-1\} \). Let us construct a graph, the vertices of which are all the sets \( S(e_1, e_2, \ldots, e_r) \) with \( e_k \in \{0, 1, \ldots, q-1\} \) which are not empty, and the set \( S \) itself. Let us connect the vertex corresponding to the set \( S \) with all the vertices corresponding to nonempty sets of the form \( S(e_1) \), and in general let us connect the vertex corresponding to a nonempty set \( S(e_1, \ldots, e_r) \) \((1 \leq r)\) with all the vertices corresponding to nonempty sets of the form \( S(e_1, e_2, \ldots, e_r, e_{r+1}) \).
In this way we get a rooted directed tree, the root of which is the vertex corresponding to \( S \). This tree has the following properties:

a) The number of edges going out from every vertex is at most \( q \).

b) The number of endpoints of the tree is equal to \( n \), and each endpoint corresponds to a one-element set.

c) The unique path leading from the root to the endpoint corresponding to the set consisting of the single element \( x^* \) of \( S \) has the length \( N(x^*) \): The \( k \)-th point along this path corresponds to the set \( S_k(x^*) \) of those \( x \in S \) for which the conditions \( f_j(x) = f_j(x^*) \) \((1 \leq j \leq k \leq N(x^*))\) are satisfied. (The sets \( S_k(x^*) \) have been introduced already in §1.7.)

d) The set corresponding to any vertex \( V \) of the tree consists of all those elements \( x \) of \( S \) which have the property, that the unique path from the root to the vertex corresponding to the one-element set \( x \) passes through \( V \).

Thus the tree which we have constructed reflects everything which is of importance concerning the search process in question: It is even unnecessary to indicate the sets corresponding to the vertices, only the one-to-one correspondence between the elements of \( S \) and the endpoints of the tree has to be indicated.

If \( q = 2 \), we call the simple search a \textit{binary search}, and the corresponding search tree a \textit{binary tree}.

Fig. 2.1 shows the (binary) tree corresponding to the solution of the Steinhaus problem given in §1.6. for \( n = 4 \).

It can be seen from Fig. 2.1 that in 16 cases out of 24 the length of the search is 5 steps, in the remaining 8 cases however, the length is 4: That is in the case of 8 permutations already four comparisons (the first four) are sufficient.
Thus the average duration of the search process is 4,666..., provided that all 24 permutations are equiprobable.

It can be seen from Fig. 2.1 that the 8 permutations in which already 4 comparisons are sufficient are those in which the greatest and the smallest number occupy either the first two or the last two places. The reason why for these permutations the last comparison is unnecessary is the following: In these cases the pair $M(x_1, x_2), M(x_3, x_4)$ is, except for the order, necessarily the pair $(3, 4)$ and thus the pair $m(x_1, x_2), m(x_3, x_4)$, except for the order, the pair $(1, 2)$, and thus the last comparison is really unnecessary. Or, expressed otherwise, in these cases $(M(x_1, x_2) - M(x_3, x_4))$ is negative, and therefore after the fourth comparison the permutation is uniquely determined.

Let us consider now how the tree corresponding to a strategy of a simple search reflects the fact, that the strategy is irreducible. This clearly means that from each point which is not an endpoint, at least two edges are going out: Thus every point which is not an endpoint has at least valency 3, except the root, the valency of which is at least 2.

In what follows we shall call a tree, which describes an irreducible strategy of a simple search process, a search tree. Thus a search tree is a rooted tree, such that its endpoints are labeled, and the valency of each point which is different from the root and not an endpoint is at least 3, while the valency of the root is at least 2. Conversely, every such tree is the tree describing an irreducible strategy of a simple search. To show this, it is sufficient to remark, that given a search
tree as defined above the function \( N(x^*) \) and the sets \( S_k(x^*) \) are thereby uniquely defined \((1 \leq k \leq N(x^*))\) and satisfy the conditions characterizing irreducible strategies, that \( S_k(x^*) \) is a proper subset of \( S_{k-1}(x^*) \) for all \( x^* \in S \) and \( k \leq N(x^*) \), where by definition \( S_0(x^*) = S \) for all \( x^* \in S \). Labeling the edges going out from any point of a search tree by the number 0, 1, ..., \( v \) \((v \geq 1)\), let us define the function \( f_k(x) \) as follows:

\[
f_k(x^*) = j \quad \text{if} \quad k \leq N(x^*) \quad \text{and the} \quad k\text{-th edge of the unique path leading to the endpoint labeled by} \quad x^* \quad \text{is labeled by} \quad j ,
\]
and defining \( f_k(x^*) \) arbitrarily \((\text{e.g. putting} \quad f_k(x^*) = 0)\) for \( k > N(x^*) \); in this way we get an irreducible strategy, to which the given tree corresponds. Notice, that the irreducible strategy we have constructed has the property, that the choice of the function \( f_k \) does not depend on the previously observed values; nevertheless the strategy is sequential in the sense, that whether we stop after \( k \) steps, or continue the search by observing \( f_{k+1}(x^*) \), depends on \( x^* \).

A search tree is called a regular (or homogeneous) search tree of degree \( q \quad (q \geq 2) \) if its root has the valency \( q \) and all other points which are not endpoints have the valency \( q + 1 \). Clearly a binary search tree is always regular. A regular search tree describes such an irreducible strategy of a simple search in which after having already observed \( k-1 \) test functions \((k \geq 1)\) if the search has to be continued at all, then the next test function chosen according to the strategy takes on effectively exactly \( q \) different values on the set of those possible values of \( x^* \) which have not yet been excluded by the previous observations.
As well known any tree with \( N \) vertices has \( N-1 \) edges, further in every graph the sum of the valencies of all vertices is equal to the double of the total number of edges. Using these well known facts it follows immediately that if \( T \) is a regular search tree of degree \( q \) and having \( n \) endpoints, then the total number \( N \) of vertices of \( T \) is

\[
N = n + \frac{n-1}{q-1}
\]

An even simpler way to verify (2.1.1) is the following: Each edge is at the same time an outgoing and an incoming edge, therefore the total number of outgoing edges i.e. \((N-n)q\) has to be equal to the total number of incoming edges, i.e. to \( N-1 \), and thus (2.1.1) follows.

Thus such a graph can exist only if \( n-1 \) is divisible by \( q-1 \), i.e. if

\[
n = k(q-1) + 1
\]

where \( k \) is a positive integer: Conversely, if (2.1.2) is fulfilled, one can always construct a regular search-tree of degree \( q \) and having \( n \) endpoints. This is most simply proved by induction: For \( k = 1 \), we simply take a root and connect it with \( q \) endpoints. Given a regular search tree of degree \( q \) and \( k(q-1) + 1 \) endpoints, by connecting \( q \) additional points with one of the endpoints of the given tree, we get a regular search tree of degree \( q \) with \((k+1)(q-1) + 1 \) endpoints.

Of course if \( q = 2 \), the equation (2.1.1.) does not mean any restriction on \( n \): for every \( n \geq 2 \) there exist regular binary trees with \( n \) endpoints. In the next § we shall deal with the number of regular search trees of degree \( q \), with a given number \( n \) of endpoints, satisfying (2.1.2).
A search tree can also be interpreted as describing a questionary (see Picard [1]). Every point of the tree, which is not an endpoint, corresponds to a question, and the edges going out from the point as the possible answers to the question. Thus a regular search tree of degree \( q \) describes a questionary in which every question admits \( q \) possible answers: A binary search tree describes a questionary in which only questions, which can be answered by yes or no, can be asked.
§2.2. Enumeration of regular search trees.

We prove now the following:

**THEOREM 2.2.1.** Let us denote by $T(q, k)$ the total number of different regular search trees of degree $q$ and having $n = k(q-1) + 1$ endpoints ($q \geq 2$, $k \geq 1$). Then we have

\[
T(q, k) = \frac{(kq)!}{k! (q!)^k}.
\]

**Proof.** We shall prove Theorem 2.2.1 by using Prüfer's Method [2]. This method consists in coding labeled trees. Let $T$ be a tree with $n$ vertices, labeled by the numbers $1, 2, \ldots, n$. To such a tree by Prüfer's method there corresponds a sequence of length $n-2$, the elements of which are elements of the set $\{1, 2, \ldots, n\}$ (not necessarily different): This correspondence is one-to-one between the set $T_n$ of all labeled trees with $n$ vertices and all possible sequences of length $n-2$ from elements of the set $\{1, 2, \ldots, n\}$ which, incidentally, furnishes a proof, (one of the most simple known) of the result, due to A. Cayley [3], that there are $n^{n-2}$ different labeled trees with $n$ vertices. The Prüfer code is obtained as follows: a) among the endpoints of $T$ that labeled with the smallest number is removed from the tree $T$, and the number by which the unique point to which the removed endpoint was connected is labeled, is written down; b) with the remaining tree the same

* As pointed out in the previous §, in a search tree the endpoints (and only these) are labeled: Thus two regular search trees which are isomorphic, but their endpoints are labeled differently, are considered as different. (Of course if the tree has $n$ endpoints, these have to be labeled by the numbers $1, 2, \ldots, n$).
procedure is repeated, and this process is continued until there remains
only a single edge. It is easy to show that a tree can be uniquely repro-
duced from its Prüfer-code. Let us note the following fact about the
Prüfer code, which follows immediately from its definition: If \( T \) is a
labeled tree with \( n \) vertices and \( P \) one of its vertices having valency
\( v \) and labeled by the number \( k \), then \( k \) occurs in the Prüfer code of
\( T \) exactly \((v-1)\) times: Thus the numbers by which the endpoints are
labeled, and only these among the numbers \( 1, 2, \ldots, n \) are not occurring
at all in the Prüfer code of \( T \). Now let us consider all regular
search-trees of degree \( q \) and having \( n = k(q-1) + 1 \) endpoints. As
shown in the previous \( § \), such trees have besides the \( n \) endpoints a
root, having valency \( q \) and \( k-1 \) points having valency \( q + 1 \); let
us label all the points which are not endpoints of all these trees by the
numbers \( 1, 2, \ldots, k \), in every possible manner. The total number of
such labeled trees is, according to the mentioned property of the Prüfer
code, equal to the total number of all sequences of length \( n + k-2 \) formed
from the integers \( 1, 2, \ldots, k \) in which one of the numbers \( 1, 2, \ldots, k \)
occurs exactly \( q-1 \) times, and all the others \( q \) times (notice that
\( n + k-2 = q-1 + (k-1)q = kq - 1 \)). The total number \( U(q, k) \) of such
sequences is clearly

\[
U(q, k) = \frac{(kq-1)!}{(q-1)!} \frac{k}{q^r(k-1)} = \frac{(kq)!}{q^r k}.
\]

To get the number \( T(q, k) \) we have to divide \( U(q, k) \) by \( k! \), because
the labeling of the vertices other than the endpoints is irrelevant
according to the definition of a search tree. Thus from (2.2.2) we obtain (2.2.1).
For \( q = 2 \), we get from (2.2.1) that the total number of binary search trees with \( n \) endpoints is

\[
T(2, n) = 1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-3) = (2n-3)!!
\]

This result is due to A. Cayley [4], who obtained it by solving the following question: Given a non-associative operation \( A \circ B \), how many interpretations of the ambiguous symbol \( A_1 \circ A_2 \circ \ldots \circ A_n \) are possible? The answer to this question is, that the number \( C_n \) of possible interpretations is

\[
C_n = \frac{T(2, n)}{n!} 2^{n-1}
\]

and thus

\[
C_n = \frac{\binom{2n-1}{n}}{(2n-1)}
\]

To prove (2.2.4) it is sufficient to point out, that every possible interpretation of the ambiguous symbol \( A_1 \circ A_2 \circ A_3 \circ \ldots \circ A_n \) means bracketing this expression: For instance for \( n = 4 \) the possible bracketings of the "product" \( A \circ B \circ C \circ D \) are the following:

\[
((A \circ B) \circ C) \circ D, \quad A \circ (B \circ (C \circ D)), \quad (A \circ B) \circ (C \circ D),
\]

\[
(A \circ (B \circ C)) \circ D, \quad A \circ ((B \circ C) \circ D).
\]

To each bracketing there corresponds a tree in an obvious way. For instance the five bracketings of the product \( A \circ B \circ C \circ D \) listed above correspond to the trees shown on Fig. 2.2.1. It can be seen from this figure, that the four endpoints are always labeled with \( A, B, C, D \) from left to right. As in enumerating binary trees with \( n \) endpoints, any labeling of the endpoints is admissible,
Fig. 2.2.1
to obtain $C_n$ from $T(2, n)$ we have first to divide by $n!$; however as left and right make no sense with respect to a tree as a graph*, and there are $n-1$ branching points, we have to multiply by $2^{n-1}$ to obtain all bracketings of a nonassociative product of $n$ factors; this explains (2.2.4).

Let us add, that the bracketing of nonassociative products can be directly interpreted in terms of search theory. Let us recall that in a search tree to each vertex $P$ there corresponds uniquely a subset of the basic set: The set of all those elements $x$ of $S$ for which the unique path from the root the endpoint labeled by $x$ leads through $P$. The bracketing corresponding to a binary search tree exhibits all these subsets in a rather compact way: This can be seen if for instance for the trees shown by Fig. 2.2.1 one writes besides each vertex the corresponding subset of the set $A, B, C, D$ and compares this with the bracketing corresponding to the tree. On Fig. 2.2.2 this is shown for the first and the last of the trees in Fig. 2.2.1.

It is easy to see that if in a bracketing of a nonassociative product of $n$ factors only the beginnings ["("] of a bracket are retained and the endings [")"] are left out, these can be reconstructed uniquely: Thus the sequence of $n$ letters and $n-1"("$ signs determines the bracketing in question, and thus also the corresponding tree as drawn on the plane. If now instead of the sign "(" we write a zero and instead of each of the $n$ letters a one, we get a sequence consisting of $n$ ones and $n-1$ zeros; this sequence still determines

* It has a meaning only with respect to the figure of the tree drawn on the plane.
Fig. 2.2.2
the tree as drawn on the plane, except for the labeling of its endpoints. Thus the number of all such sequences is $C_n$. We call these sequences Cayley-sequences. Clearly not all (only one out of $2n-1$) possible sequences of length $2n-1$ consisting of $n$ ones and $n-1$ zeros (the total number of such sequences being $\binom{2n-1}{n} = (2n-1)C_n$) are Cayley sequences. The family of Cayley sequences of length $2n-1$ can be described as follows: We start from the sequence consisting of a single 1, and we replace it by the group of three symbols 011; after this we replace one of the ones, by the group 011, and we continue this as long as $n-1$ replacements have been made, i.e. a sequence of $n-1$ zeros and $n$ ones obtained. Thus the set of all Cayley sequences forms a formal language (see e.g. Bar-Hillel – Schützenberger [5]). For $n = 4$ the Cayley sequences of length 7 are the following:

0001111
0010111
0011011
0100111
0101011

Let us notice, that forming all cyclic permutations of these 5 Cayley sequences we get all the 35 sequences of length 7 consisting of 4 ones and 3 zeros. This is true for every $n$: taking the $C_n$ Cayley sequences of length $2n-1$ and taking all their cyclic permutations, we get all sequences of length $2n-1$ consisting on $n$ ones and $n-1$ zeros, each once.

Another way to characterize Cayley sequences is the following:

A sequence consisting of $n$ ones and $n-1$ zeros is a Cayley sequence.
if an only if among its \( k \) last terms the number of ones exceeds the number of zeros for each value of \( k \) \((1 \leq k \leq 2n-1)\). This characterization implies among others that each Cayley sequence starts with a zero and ends with at least two ones. The correspondence between a binary tree and its Cayley code can be described as follows: We consider a binary tree drawn on the plane (i.e. in which it makes sense to speak about the left-hand side edge or branch and the right-hand side edge or branch going out from any point which is not an endpoint) having \( n \) endpoints and \( n-1 \) branching points. We first label all its vertices from 1 to \( 2n-1 \) as follows: We start at the root, and label it with one: We continue the labeling by proceeding along the tree so that we proceed along the left-hand edge whenever this is possible (i.e. until we get to an endpoint. If this can not be done any more we go back until we find an outgoing edge leading to a still unlabeled point, go along this edge to this point, if possible proceed along the left-hand side edge, etc. After all points have been labeled in this way, the Cayley codeword of the tree is obtained simply as follows: The \( k \)-th digit of the codeword is one if the point labeled by the number \( k \) is an endpoint and 0 otherwise. Fig. 2.2.3 shows a binary tree, labeled according to the rule given above, and the corresponding Cayley codeword.

The characterization of Cayley sequences given above can be formulated also as follows: Each Cayley sequence, read from right to left corresponds to a sequence of \( n \) "yes" votes and \( n-1 \) "no" votes such that at each stage of the voting the number of yes votes exceeds the number of no votes. The problem of counting the number of such sequences, i.e. to determine the probability of such a sequence of votes, is called in probability theory the ballot problem: It was solved by Desirée André.
Fig. 2.2.3.

0001100111011
There are several other ways to prove Theorem 2.2.1, but we have chosen to deduce it by Prüfer's method, because we shall need this method for the solution of other enumeration problems on trees, and wanted to introduce the method anyway. We shall prove now a recursion formula for the sequence $T(q, k)$ ($k=1,2,...$) for any fixed value of $q$, which can also be used as the starting point of a quite different proof of Theorem 2.2.1.

**THEOREM 2.2.2.** Denoting by $T(q, k)$ the number of different regular search trees of degree $q$ and $n = k(q-1) + 1$ edges, the following recursion formula holds (and putting $T(q, 0) = 1$, $k \geq 1$):

\[
T(q, k) = \sum_{j=0}^{k-1} \frac{(k(q-1)+1)!}{j!(j(q-1)+1)!} \cdot \prod_{j=0}^{k-1} (T(q,j))^a_j \\
\text{where the summation has to be extended for all sequences } a_1, a_2, \ldots a_{k-1} \text{ of nonnegative integers which satisfy the conditions}
\]

\[
k-1 \sum_{j=0}^{k-1} j a_j = k-1 \quad k-1 \sum_{j=0}^{k-1} a_j = q .
\]

**Remark:** For $q = 2$, the recursion formula (2.2.6) can be written in the following simple form:

\[
T(2, k) = \frac{1}{2} \sum_{h=1}^{k} \binom{k+1}{h}(T(2, h)T(2, k-h+1)) .
\]

**Proof of Theorem 2.2.2.** To prove (2.2.6) it is sufficient to remark that in a regular search tree of degree $q$ and having $n = k(q-1) + 1$ endpoints each of the $q$ points connected with the root is the root of a regular search tree of order $q$ and with $j(q-1) + 1$ endpoints ($0 \leq j \leq k-1$) provided that we consider a single point also as a regular
search tree of order \( q \) and having one endpoint. (This explains the
convention \( T(q, 0) = 1 \). We call these trees the subtrees of the given
tree. If there are among these subtrees \( a_j \) having \( j(q-1) + 1 \) end-
points \( (j=0,1,\ldots,k-1) \) then the equations \( \sum_{j=0}^{k-1} a_j = q \) and
\( \sum_{j=0}^{k-1} j a_j = k(q-1) + 1 \) have to hold, i.e. the \( a_j \) have to
satisfy the equations \( \sum_{j=0}^{k-1} a_j = q \) and \( \sum_{j=0}^{k-1} j a_j = k-1 \). Counting in
how many ways the number \( 1, 2, \ldots, n \) can be distributed among the
subtrees, and in how many ways the subtrees can be chosen and labeled,
we get (2.2.6).

From (2.2.6) it is easy to deduce a functional equation of the
generating function of the sequence \( T(q, k) \) \((k=0,1,\ldots)\).

**THEOREM 2.2.3.** Putting

\[
F_q(x) = \sum_{k=0}^{\infty} \frac{T(q, k) x^{k(q-1)+1}}{(k(q-1) + 1)!}
\]

we have the functional equation

\[
(F_q(x))^q \quad \frac{q!}{q!} F_q(x) + x = 0.
\]

Starting from Theorem 2.2.3, an alternative proof of (2.2.1) can
be obtained, by developing the inverse function of the function
\( x = y - \frac{y^q}{q!} \) into a power series in \( x \) according to the Bürmann-Lagrange
formula (see e.g. G. Pólya and G. Szegő, [6]). We do not give the details
for arbitrary \( q \), only for \( q = 2 \), for which case only an equation of
the second degree has to be solved, as in this case we get from (2.2.8)

\[
F_2(x) = 1 - \sqrt{1-2x}
\]
and thus, by the binomial expansion

\[
F_2(x) = \sum_{n=1}^{\infty} \frac{\mathcal{T}(2,n)}{n!} x^n = \sum_{n=1}^{\infty} \frac{1}{n!} (2x)^n (-1)^{n-1} = \sum_{n=1}^{\infty} \frac{(2n-3)!}{n!} x^n.
\]

Thus we obtained a second proof of the formula (2.2.3).

As regards the general case, comparing Theorems 2.2.1 and 2.2.3, we get a combinatorial proof of the expansion

\[
y = \sum_{k=0}^{\infty} \frac{(kq)!}{k!(q!)^k} \frac{x^{k(q-1)+1}}{(k(q-1)+1)!}
\]

of the function \( y = y(x) \) defined as that solution of the equation

\[
y^q - y + x = 0 \quad \text{which vanishes for } x = 0.
\]

One way of looking at this result is, that we have obtained the power series expansion of a root of a trinomial equation of the form \( y^q + A y + B = 0 \), by a purely combinatorial argument.

From (2.2.1) we get, using Stirling's formula, the asymptotic formula

\[
T(q, k) \sim \left(\frac{k}{e}\right)^{k(q-1)} \frac{q^k}{q!} \frac{K}{\sqrt{q}}
\]

for \( k \to +\infty \) and \( q \) fixed, which shows, that the power series (2.2.12) is convergent for

\[
|x| \leq \left[ \frac{q!(q-1)^{q-1}}{q^q} \right] \frac{1}{q-1} = R_q.
\]

(Note that \( R_2 = \frac{1}{2} \), \( R_3 = \frac{2\sqrt{2}}{3} \), \( R_4 = \frac{3}{4} \frac{3\sqrt{6}}{4} > 1 \) and that \( R_q \) is increasing with \( q \) and tends to \( +\infty \) for \( q \to +\infty \), in such a way that \( \lim_{q \to +\infty} \frac{R_q}{q^q} = \frac{1}{e} \).
§2.3. Description of a simple search by a code.

Let us consider a simple search problem such that all admissible test functions take on only the values 0, 1, ... q-1, and let the number of elements of the basic set S be denoted by n. Let us consider a strategy, consisting in observing the values of the functions $f_k$ (k=1,2,...) one after the other, and let the duration of the search be denoted by $N(x^*)$. This means that the element $x^*$ is uniquely determined by the sequence $f_1(x^*), f_2(x^*), ... f_N(x^*)$, this being a sequence each element of which is one of the numbers 0, 1, ..., q-1. Thus there exists a one-to-one mapping of the elements of S into a set of n sequences formed from the numbers 0, 1, ..., q-1. We call the sequence $(f_1(x^*), f_2(x^*), ... f_N(x^*))$ the codeword corresponding to $x^*$, and the set of all such sequences the code corresponding to the search process. The code corresponding to a search process has evidently the following property:

No codeword is a continuation of another codeword (or expressed otherwise: No codeword is a segment of another codeword). Codes having this property are called in information theory prefix codes. Prefix codes have the property that if several codewords are written one after the other, the resulting sequence can be uniquely decomposed into codewords: Thus in writing codewords one after the other it is unnecessary to leave gaps between the codewords or indicate in any other way where a codeword ends and the next begins. As $N(x^*)$ in general depends on $x^*$, the prefix code corresponding to a search process is in general a variable length prefix code. If in particular $N(x^*)$ is constant, the code is a fixed length prefix code.
If the maximal number of values taken on by any of the test functions figuring in a strategy is \( q \), we may without restricting the generality suppose that these values are the numbers \( 0, 1, \ldots, q-1 \). Moreover we may suppose that if a test function figuring in the search process takes on only \( r < q \) values, then these values are \( 0, 1, \ldots, r-1 \). In what follows speaking about the code corresponding to a simple search, we shall always suppose, that the mentioned condition is satisfied: As regards the code this means that if among the code words of the code starting with a given segment of length \( a_1, a_2, \ldots, a_{k-1} \) the next symbol may take on \( r \) different values, then these values are the numbers \( 0, 1, \ldots, r-1 \) \((r \geq 1, k \geq 1)\).

The code corresponding to a binary search is clearly a binary code, i.e. such that its codewords are sequences of zeros and ones.

Let us now consider what characterizes the codes corresponding to irreducible strategies. For the sake of simplicity we deal with this question only in the binary case. We shall prove the following:

**Theorem 2.3.1.** The binary code corresponding to a simple binary search with an irreducible strategy has the property that if from one (or more) of the codewords one (or more) digits are omitted, the resulting set of codewords is no more a prefix code.

**Remark:** The statement of Theorem 2.3.1 can be expressed simply by saying that the code corresponding to an irreducible strategy in a simple binary search can not be shortened without losing the prefix property. Such a code is called an irreducible prefix code. Thus the
statement of Theorem 2.3.1 can also be expressed as follows: To an irreducible strategy of a simple binary search there corresponds an irreducible prefix binary code.

**Proof of Theorem 2.3.1.** If a code does not have the prefix property, then by leaving out a digit from any of its codewords, the code thus obtained does not have the prefix property either. Therefore in proving Theorem 2.3.1 it is sufficient to show that if we omit one digit from one of the codewords of the code corresponding to an irreducible strategy, we get a code which does not have the prefix property, that is it contains at least two codewords, such that one is a segment of the other. (In particular this may mean that the two codewords are identical, i.e. the set of zero-one sequences is not a code at all). To show this we need the following simple lemmas. For the sake of brevity in course of this proof we call a code corresponding to an irreducible strategy of a simple binary search a binary I-code.

**Lemma 2.3.1.** If a binary I-code $C$ contains a codeword $(e_1, e_2, e_k, \ldots e_n)$ $(n \geq k)$ it contains also at least one codeword starting with $(e_1, e_2, \ldots e_{k-1}, 1-e_k)$.

**Proof:** If this were not true, the strategy would not be irreducible, as after having observed the first $k-1$ test functions and obtained the values $e_1, e_2, \ldots, e_{k-1}$, the observation of the $k$-th test function would be unnecessary.
Remark: Notice that for the special case \( k = n \), the statement of Lemma 2.3.1 is as follows: If the codeword \((e_1, e_2, \ldots, e_n)\) is in \( C \) then the codeword \((e_1, \ldots, e_{n-1}, 1-e_n)\) is also in \( C \).

**Lemma 2.3.2.** Given a binary I-code \( C \) and an arbitrary finite sequence \( s \) of zeros and ones, then either \( s \) is contained in \( C \), or \( s \) is a proper segment of a codeword in \( C \) or finally there is in \( C \) one and only one codeword which is a segment of \( s \), and these three cases exclude each other.

Proof: The fact that the three cases exclude each other follows from the prefix property of the code \( C \). Thus to prove the lemma it is sufficient to show that if for a given sequence \( s \) the first two cases do not take place, then the third case takes place. Suppose namely that the sequence \( s \) does not belong to \( C \) and is not a segment of any codeword of \( C \). Let \( c \) denote that codeword in \( C \) which has the longest initial segment coinciding with the corresponding segment of \( s \); such a \( c \) always exists, because \( C \) necessarily contains codewords starting with a 0 and also codewords starting with 1. If \( c \) has the length \( n \) and its longest initial segment coinciding with the initial segment of \( s \) has length \( k \), then \( s \) necessarily has length \( \geq k + 1 \), because otherwise \( s \) would be a segment of \( c \); if \( n = k \), then \( c \) is a segment of \( s \), and that is exactly what we want to prove. Suppose therefore that \( n > k \). Let \( e \) denote the \((k + 1)\)-st digit of \( s \), then by supposition the \( k + 1 \)-st digit of \( c \) is 1-e; but then by Lemma 2.3.1 \( C \) contains at least one codeword
c' such that the first \( k \) digits of \( c' \) coincide with the corresponding digits of \( c \) and thus also with those of \( s \), and the \((k + 1)\)-st digit of \( c' \) is \( e \), i.e. it coincides with the \((k + 1)\)-st digit of \( s \). But this is in contradiction with our supposition that \( c \) has the maximal initial segment coinciding with the corresponding segment of \( s \). This contradiction proves our lemma. The statement of the following lemma is obvious:

**LEMMA 2.3.3.** Let \( C \) be a binary I-code and let us consider those codewords of \( C \) in which the initial segment of length \( k \) coincides with a given sequence \( s \) of length \( k \) of zeros and ones. Let us omit from these codewords the initial segment \( s \). The codewords thus obtained (if their set is not empty) form also a binary I-code.

In view of Lemma 2.3.3 we can suppose, without restricting the generality that we omit the first digit of a codeword \( c \) of a binary I-code \( C \). We may suppose that the omitted first digit is a \( 0 \). Let the remaining codeword be \( (e_2, e_3, \ldots, e_n) \) and let us denote it by \( S \). Let \( C \) denote the set of codewords obtained by replacing \( c \) by \( S \) in \( C \).

Let us apply now Lemma 2.3.2 to \( s \) and the binary I-code \( C \). If \( s \) is contained in \( C \) or contains as initial segment a codeword in \( C \), then clearly \( C' \) is not a prefix code. Thus it is sufficient to consider the third case, in which there is a codeword \( c^* \) in \( C \) the initial segment of which coincides with \( s \). Now two cases are possible: Either \( c^* \) may be chosen different from \( c \), in
this case $c^*$ is contained in $C'$ too, and thus $C'$ is not a prefix code, or the only possible choice of $c^*$ is $c^* = c$. But this means that $c$ has the property that if its first digit (which is a 0) is omitted, the remaining sequence coincides with the first $n-1$ digits of $c$. But this is possible only if $c$ consists of $n$ zeros. However in this case by the remark to Lemma 2.3.1, $C$ contains also the sequence starting with $n-1$ zeros and ending with 1; this sequence is different from $c$, and its initial segment of length $n-1$ coincides with $s$, in contradiction with our supposition. This contradiction proves Theorem 2.3.1.

The situation in the general case is not so simple: For instance the code consisting of the codewords 00, 01, 02, 10, 11, 12 is a prefix code, but is is not irreducible, because omitting the first digit of the third codeword we get again a prefix code, namely 00, 01, 2, 10, 11, 12. Nevertheless Theorem 2.3.1 can be generalized for regular codes of degree $q$, which will be introduced in the next §.

§2.4. Connection between the code and the tree describing a simple search.

As we have seen, a simple search can be described both by a tree and by a code. Let us now have a look at the question how these two descriptions are related to each other.

If the tree corresponding to a simple search is given, we can obtain from it several codes describing the search, by labeling all the edges going out from the root by the numbers 0, 1, ..., $v-1$ if $v$ is the
valency of the root, then labeling similarly all the edges going out (i.e. leading nearer to an endpoint) from any other point which is not an endpoint similarly. Of course this labeling can be done in several ways. If there are \( k \) points which are not endpoints, the root has valency \( v_0 \) and the other \( k-1 \) vertices the valencies \( v_1, v_2, \ldots, v_{k-1} \) then the number of possible labelings is

\[
\frac{v_0!}{v_1! v_2! \cdots v_{k-1}!} (v_1-1)! \cdots (v_{k-1}-1)!
\]

Especially if the tree is a regular search tree of degree \( q \), having \( n \) endpoints, where \( n = k(q-1) + 1 \), then the number of possible labelings is \( (q!)^k \). Thus for a binary tree with \( n \) endpoints the number of possible labelings is \( 2^{n-1} \). For any particular labeling to each element \( x^* \) of \( S \) there corresponds an endpoint of the tree, and a unique path leading to this endpoint. The corresponding codeword is defined as the sequence of numbers by which the edges of this path have been labeled, starting from the root. This correspondence can be of course used also the other way around: To every code we can construct a corresponding tree, the edges of which are labeled, as described above.

Using this correspondence between search trees and codes, we can get from results concerning the enumeration of all search trees of a given type the number of all possible codes of the corresponding type (and conversely). In counting codes we shall always suppose that a code is an ordered set of codewords, i.e. we consider two codes consisting of the same codewords, but in different order, as different codes. (This is of course a matter of convention.)

A code corresponding to a regular search tree of degree \( q \) will be called a **regular code of degree** \( q \). A binary irreducible prefix code is of course always regular.
From Theorem 2.2.1 we get the following:

**THEOREM 2.4.1.** The total number $C(q, k)$ of regular search codes of degree $q$ containing $n = k(q-1) + 1$ codewords is

$$C(q, k) = (q!)^k T(q, k) = \frac{(kq)!}{k!},$$

where $T(q, k)$ is the number of regular search trees of degree $q$ and having $n$ endpoints. In particular the total number of binary irreducible prefix codes is

$$C(2, n-1) = 2^{n-1}.(2n-3)!! = C_n n!$$

where $C_n$ is the sequence of Cayley defined by (2.2.4).

**Remark:** Clearly

$$C_n^{(q)} = \frac{C(q, n-1)}{n!} (n \equiv 1 \mod (q-1))$$

is the total number of unordered regular search codes of degree $q$ consisting of $n$ codewords. We have $C_n^{(2)} = C_n$, where $C_n$ is Cayley's sequence. Let us denote by $G_q(x)$ the generating function of the sequence $C_n^{(q)}$, i.e. let us put

$$G_q(x) = \sum_{n=1}^{\infty} C_n^{(q)} x^n \quad \text{for} \quad n \equiv 1 \mod (q-1)$$

Then in view of Theorem 2.4.1 we have

$$G_q(x) = \sum_{n=1}^{\infty} \frac{(q!)^n T(q, n-1)}{n!} x^n = \frac{1}{1 - F(q, q-1, x)}$$

$$n \equiv 1 \mod (q-1)$$
and thus \( G_q(x) \) satisfies the functional equation

\[
(2.4.6.) \quad G_q(x) = x + [G_q(x)]^q.
\]

The equation (2.4.6) can be verified also directly, by the recursion formula

\[
(2.4.7.) \quad c_n^{(q)} = \sum_{n_1 + n_2 + \ldots + n_q = n} c_{n_1}^{(q)} c_{n_2}^{(q)} \ldots c_{n_q}^{(q)}
\]

§2.5. Enumeration of all search trees having a given number of endpoints.

Let \( V(n) \) denote the total number of different search trees with \( n \) (labeled) endpoints: The aim of this § is to determine \( V(n) \).

Let \( V(n, k) \) denote the total number of different search trees having \( n \) endpoints and \( k \) other points (including the root). Then we have

\[
(2.5.1.) \quad V(n) = \sum_{k=1}^{n-1} V(n, k)
\]

as \( k \) is clearly maximal if and only if the tree is binary.

We shall determine \( V(n, k) \) by using Prüfer's method. According to the definition, in a search tree only the endpoints are labeled. Thus \( V(n, k) \) denotes the number of those trees which have \( n + k \) points, among these there are \( n \) endpoints, labeled by the numbers 1, 2, ..., \( n \); among the remaining points there is one which is distinguished as the root and has valency \( \geq 2 \), while the remaining \( k-1 \) points have valency \( \geq 3 \). If we label the root by the number 1, and the
remaining \( k-1 \) points by the numbers 2, 3, ... \( k \), in every possible way, we get \( U(n, k) = (k-1)! V(n, k) \) different graphs. Now by Prüfer's coding it is evident that \( U(n, k) \) is equal to the number of sequences of length \( n+k-2 \) in which the number 1 occurs at least once and the numbers 2, 3, ..., \( k \) occur at least twice. To determine \( U(n, k) \) let us solve first the following question: Let \( S_r(N, M) \) denote the number of those sequences of length \( N \) formed from the numbers 1, 2, ..., \( M \) in which each of these numbers occurs at least \( r \) times \((N \geq Mr)\). Then we have evidently the recursion formula

\[
S_r(N, M) = \sum_{j=r}^{N} \binom{N}{j} S_r(N-j, M-1)
\]

from which it follows that, putting

\[
\frac{1}{N^{Mr}} S_r(N, M) x^N = \delta_r(M, x)
\]

we have

\[
\delta_r(1, x) = \sum_{N=r}^{\infty} \frac{x^N}{N!} = e^x - \sum_{k=0}^{r-1} \frac{x^k}{k!}
\]

and

\[
\delta_r(M, x) = \delta_r(M-1, x) \delta_r(1, x).
\]

Thus it follows

\[
\delta_r(M, x) = (e^x - \sum_{k=0}^{r-1} \frac{x^k}{k!})^M \quad (M=1,2,...).
\]
Thus especially for \( r = 2 \) we get

\[
(2.5.6b.) \quad \sum_{N=2M}^{\infty} \frac{S_2(N, M)}{N!} x^N = (e^{x-1}-x)^M.
\]

Evidently \( S_1(N, M) \) are the Stirling numbers of the second kind.

In general \( \frac{S_r(N, M)}{M!} \) is equal to the number of partitions of a set of \( N \) elements into \( M \) parts each containing at least \( r \) elements.

We shall need the numbers \( S_2(N, M) \) because evidently

\[
(2.5.7.) \quad U(n, k) = \sum_{j=1}^{n-k} \binom{n-k+2}{j} S_2(n+k-2-j, k-1).
\]

Thus, putting

\[
(2.5.8.) \quad u(k, x) = \sum_{n=k+1}^{\infty} \frac{U(n, k)}{(n+k-2)!} x^{n+k-2}
\]

we have, in view of (2.5.6b),

\[
(2.5.9.) \quad u(k, x) = (e^x-1)(e^x-1-x)^{k-1}
\]

and therefore

\[
(2.5.10.) \quad \sum_{n=k+1}^{\infty} \frac{V(n, k)}{(n+k-2)!} x^{n+k-2} = \frac{(e^x-1)(e^x-1-x)^{k-1}}{(k-1)!}
\]

Using (2.5.10) one can determine the numbers \( V(n, k) \) and thus also \( V(n) \).

For instance we get from (2.5.10) the following values:

<table>
<thead>
<tr>
<th>( n )</th>
<th>( V(n, 1) )</th>
<th>( V(n, 2) )</th>
<th>( V(n, 3) )</th>
<th>( V(n, 4) )</th>
<th>\ldots</th>
<th>( V(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>10</td>
<td>15</td>
<td>0</td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>25</td>
<td>105</td>
<td>105</td>
<td></td>
<td>236</td>
</tr>
</tbody>
</table>
Another way for computing the numbers \( V(n) \) is the following: By considering all possible ways in which the \( n \) endpoints can be distributed between the main branches of the tree starting from the root, one obtains the recursive formula:

\[
(2.5.11.) \quad \frac{V(n)}{n!} = \sum_{h=2}^{n} \prod_{j:a_j=h} \left( \frac{v(j)}{j!} \right)^{a_j}
\]

where by definition \( V(1) = 1 \). From (2.5.11) we get for the power series

\[
(2.5.12) \quad W(x) = \sum_{n=1}^{\infty} \frac{V(n)x^n}{n!}
\]

the functional equation

\[
(2.5.13.) \quad W(x) = \frac{1}{2} \left( x + e^{W(x)} - 1 \right).
\]

It can be shown that the power series (2.5.12) is convergent for \( |x| < \ln \frac{4}{e} \). By taking successively the derivatives of both sides of (2.5.13) one obtains successively the values \( V(n) = W^{(n)}(0) \) \((n=1,2,...)\).

§2.6. **Enumeration of all codes corresponding to a search process, and having a given number of codewords.**

In the case of regular search trees of degree \( q \) and \( n \) endpoints, where \( n = k(q-1) + 1 \), it was easy to get from the number of trees the number of codes, by multiplying it by the factor \( q^k \). In the case of all search trees, one can not pass so easily to the number of all corresponding codes, because the factor by which one has to multiply depends not only on the number of endpoints, but on the structure of the tree too: The
factor is namely equal to the product \( v_0 \cdot \prod_{j=1}^{k-1} (v_j - 1)! \) where \( v_0 \) is
the valency of the root, and \( v_1, v_2, ..., v_{k-1} \) the valencies of the
other points which are not endpoints. Let \( D(n) \) denote the total
number of codes corresponding to search trees with \( n \) endpoints, and
\( D(n, k) \) the number of those of these codes, which correspond to search
trees containing besides \( n \) endpoints \( k \) further points. While one
can not directly obtain \( D(n, k) \) from \( V(n, k) \), nevertheless the same
method which led us to determine \( V(n, k) \) can be used to determine \( D(n, k) \).

Let \( P_2(N, M) \) denote the sum of the product of the occurrences
of the numbers \( 1, 2, ... M \) in a sequence of length \( N \) formed from
these numbers, where the summation is extended over all such sequences
in which each of the numbers \( 1, 2, ... M \) occurs at least twice. We
get, in analogy to (2.5.2)

\[
(2.6.1.) \quad P_2(N, M) = \sum_{j=2}^{N} \binom{N}{j} P_2(N-j, M-1)
\]

and thus, putting

\[
(2.6.2.) \quad \sum_{N=2M}^{\infty} \frac{P_2(N, M)}{N!} x^N = p_2(M, x)
\]

we obtain

\[
(2.6.3.) \quad p_2(M, x) = \left( \frac{x^2}{1-x} \right)^M
\]

Let \( E(n, k) \) denote the sum of the product of the occurrences of the
numbers \( 1, 2, ..., k \) in a sequence of \( (n + k - 2) \) elements, consisting
of the numbers \( 1, 2, ..., k \), the summation being extended over all
such sequences in which the number 1 occurs at least once and all the numbers
1, 2, ..., \( k \) at least twice. Then we have, in analogy to (2.5.7)
Thus putting

\[ E(n, k) = \sum_{j=1}^{n-k} \frac{n-k}{j!} (n+j)!(n+k-2-j)_j p_2(n+k-2-j, k-1). \]

Thus putting

\[ e(k, x) = \sum_{n=k+1}^{\infty} \frac{E(n, k)}{(n+k-2)!} x^{n+k-2} \]

we get

\[ e(k, x) = \sum_{n=k+1}^{\infty} \frac{2^{k-1}(2-x)}{x^{n+k-2}} \]

As however \( D(n, k) = \frac{E(n, k)}{(k-1)!} \) we obtain finally for \( k=1, 2, \ldots \)

\[ d(k, x) = \sum_{n=k+1}^{\infty} \frac{D(n, k)}{(n+k-2)!} x^{n+k-2} = \frac{2^{k-1}}{x^{(k-1)!}} \left( \frac{1}{(1-x)^{k+1}} + \frac{1}{(1-x)^k} \right). \]

From (2.6.7) we get

\[ D(n, k) = \frac{(n-2)!(n+k-1)!}{k!(k-1)!(n-k-1)!}. \]

Thus we obtained surprisingly an explicit formula for \( D(n, k) \); this formula can also be written in the form

\[ D(n, k) = (n-1)! \frac{n-2}{k-1} (n+k-1)_k. \]

The first four values of \( D(n, k) \) \((k=1, 2, 3, 4)\) for \( n = 2, 3, 4, 5 \) are given by the following table:

<table>
<thead>
<tr>
<th>n</th>
<th>D(n, 1)</th>
<th>D(n, 2)</th>
<th>D(n, 3)</th>
<th>D(n, 4)</th>
<th>D(n)</th>
<th>( \frac{D(n)}{n!} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>120</td>
<td>120</td>
<td>0</td>
<td>264</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
<td>1080</td>
<td>2520</td>
<td>1680</td>
<td>5400</td>
<td>45</td>
</tr>
</tbody>
</table>

In the table we have indicated also \( D(n) \) and \( \frac{D(n)}{n!} \).
There is another way to prove the formula (2.6.8). One can easily show that for $D(n, k)$ the following recursion formula holds:

$$
\frac{D(n, k)}{n!} = \sum_{e=2}^{n} \frac{D(n_1, k_1)}{n_1!} \cdot \frac{D(n_2, k_2)}{n_2!} \cdots \frac{D(n_e, k_e)}{n_e!}.
$$

Let us put

$$
\sum_{n=1}^{\infty} \sum_{k=0}^{\infty} \frac{D(n, k)}{n!} x^n y^k = d(x, y).
$$

Then it follows from (2.6.10) that

$$
d(x, y) = x + \frac{yd^2(x, y)}{1-d(x, y)}.
$$

Thus we get for $d(x, y)$ the explicit formula

$$
d(x, y) = \frac{1 + x - \sqrt{(1+x)^2 - 4x(y+1)}}{2(y+1)}
$$

and developing the function $d(x, y)$ into a double power series we obtain (2.6.10).

Now we shall prove a simple explicit formula for the generating function of the sequence $D(n)/n!$.

Recalling the meaning of $D(n)$ we get for $D(n)$ the recursion formula

$$
\frac{D(n)}{n!} = \sum_{h=2}^{n} \frac{D(n_1)}{n_1!} \cdot \frac{D(n_2)}{n_2!} \cdots \frac{D(n_h)}{n_h!} \quad (n \geq 2)
$$

from which, putting

$$
d(x) = \sum_{n=1}^{\infty} \frac{D(n)}{n!} x^n
$$
we get (taking into account that $D(1) = 1$) that

\[(2.6.16.) \quad d(x) = x + \frac{d^2(x)}{1-d(x)}\]

and thus we arrive to the following explicit formula:

\[(2.6.17.) \quad d(x) = \frac{1 + x - \sqrt{1 - 6x + x^2}}{4} \quad .\]

It is easy to see that the power series (2.6.14.) is convergent for $|x| \leq 3 - 2\sqrt{2}$ (\(x = 3 - 2\sqrt{2}\) being the singular point of the function $d(x)$ which is nearest to the origin); one can prove by a simple function-theoretic argument that the following asymptotic formula holds:

\[(2.6.18.) \quad \frac{D(n)}{n!} \sim \frac{\sqrt{3 - 2\sqrt{2}}}{4\sqrt{\pi}} \cdot \frac{(3 + 2\sqrt{2})^n}{n^{3/2}} \quad .\]

Of course $d(x, 1) = d(x)$, thus formula (2.6.17) could have been obtained also from (2.6.13) by substituting into it $y = 1$, but we wanted to show that if we are interested only in $D(n)$ and not in $D(n, k)$ we can get (2.6.17) directly.

Of course from (2.6.17) one can also get an explicit formula for $D(n)$ but it is not very useful, as it is a sum with alternating terms:

\[(2.6.19.) \quad \frac{D(n)}{n!} = \sum_{j=0}^{\lfloor n/2 \rfloor} (-1)^j \frac{(2n-2j-3)!!}{2^{j+2} j! (n-2j)!} \cdot\]

We get e.g. from (2.6.13) that

$$\frac{D(6)}{6!} = 197 \quad .$$

We shall solve now some more related problems of enumeration. Let $D_r(n)$ denote the total number of search codes having $n$ (ordered) codewords,
each of length \( \leq r \), \((r=1,2,\ldots)\) and let us put

\[
(2.6.20.) \quad d_r(x) = \sum_{n=1}^{\infty} \frac{D_r(n)}{n!} x^n.
\]

By the same argument, that led us to (2.6.9) we get the recursion formula

\[
(2.6.21.) \quad \frac{D_r(n)}{n!} = \sum_{h=2}^{\infty} \sum_{n_1+n_2+\cdots+n_h=n} \frac{D_{r-1}(n_1)}{n_1!} \frac{D_{r-1}(n_2)}{n_2!} \cdots \frac{D_{r-1}(n_h)}{n_h!}.
\]

and from this we deduce

\[
(2.6.22.) \quad d_r(x) = x + \frac{d_{r-1}(x)}{1-d_{r-1}(x)}.
\]

Thus we get

\[
\begin{align*}
  d_1(x) &= \frac{x}{1-x} \\
d_2(x) &= x + \frac{x^2}{(1-x)(1-2x)} \\
d_3(x) &= x + \frac{x^2(1-2x+2x^2)^2}{(1-x)(1-2x)(1-4x+4x^2-2x^3)}
\end{align*}
\]

e tc.

Of course we have

\[
(2.6.23.) \quad \lim_{x \to +\infty} d_r(x) = d(x).
\]

The number of codes consisting of an arbitrary number of codewords of length \( \leq r \) can be determined for search codes of degree \( q \) too. Let \( B(r,q,n) \) denote the number of (ordered) search codes of degree \( q \) consisting of codewords of length \( \leq r \). We obtain easily the recursion formula, in analogy to (2.4.7)
\[ \frac{B(r, q, n)}{n!} = \sum_{n_1 + n_2 + \cdots + n_q = n} \frac{B(r-1, q, n_1)}{n_1!} \cdot \frac{B(r-1, q, n_2)}{n_2!} \cdots \frac{B(r-1, q, n_q)}{n_q!} \]

from which, putting

\[ b(r, q, x) = \sum_{n=1 \text{ mod } (q-1)}^{\infty} \frac{B(r-1, q, n)}{n!} x^n \]

we obtain

\[ b(r, q, x) = x + (b(r-1, q, x))^q \quad \text{for } r \geq 1 \]

where

\[ b(0, q, x) = x. \]

Clearly

\[ \lim_{r \to +\infty} b(r, q, x) = \sum_{n=1 \text{ mod } (q-1)}^{n-1} \frac{\Gamma(q, \frac{n-1}{q-1}q^{-1})}{n!} x^n = \frac{1}{q^{-1}q(q-1)} x^{\frac{1}{q-1}} = G_q(x). \]

Thus from (2.6.26) by passing to the limit \( r \to +\infty \) we obtain a new proof of the formulae (2.2.8) resp. (2.4.7).

Especially for \( q = 2 \), i.e. for binary search codes we get

\[ b(r, 2, x) = x + b(r-1, 2, x)^2 \]

and thus we obtain successfully

\[ b(1, 2, x) = x + x^2 \]

\[ b(2, 2, x) = x + (x + x^2)^2 \]

\[ b(3, 2, x) = x + [x + (x + x^2)^2]^2 \]

etc.
The total number of binary search codes consisting of codewords of length \( \leq r \) irrespectively of the number of codewords is of course finite too: Denoting this number by \( B(r) \) we have

\[
B(r) = b(r, 2, 1) .
\]

Thus we get for \( B(r) \) the recursion formula

\[
B(r+1) = 1 + B^2(r) .
\]

(2.6.23.)

from which we can compute the values of \( B(r) \) successively: We have

<table>
<thead>
<tr>
<th>( r )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B(r) )</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>26</td>
<td>677</td>
<td>458330</td>
</tr>
</tbody>
</table>

( \( B(0) = 1 \) means that the empty code is considered here as the unique code consisting of codewords of length 0.)
CHAPTER III. Uniform flows in cascade graphs.

§3.1. Cascade graphs.

We shall call a directed graph \( G \), having a finite or denumerably infinite number of points, a **cascade graph**, if it has the following properties:

a) There is in \( G \) a point \( a_0 \) --called the **source**--such that for any other point \( a \) of \( G \) there is in \( G \) a directed path from \( a_0 \) to \( a \).

b) For any point \( a \) of \( G \) each directed path from \( a_0 \) to \( a \) has the same length \( r(a) \)--called the **rank** of \( a \).

c) The number of points of \( G \) having rank \( k \) is finite for every \( k \geq 1 \).

We shall denote the set of all points of \( G \) by \( V \) and the set of all points of \( G \) having rank \( k \) by \( V_k \) \((k=0,1,2,\ldots)\). The source has clearly the rank 0 and it is the only point with this property: Thus \( V_0 \) is a one-element set containing the element \( a_0 \) only, i.e. \( V_0 = \{a_0\} \). As the supposition b) has to hold for \( a = a_0 \) also, it follows that there is no directed cycle in \( G \) containing \( a_0 \).

It follows further that if there is in \( G \) an edge from the point \( a \) to the point \( b \) then \( r(b) - r(a) = 1 \), i.e. every edge starting from a point in \( V_k \) leads to a point in \( V_{k+1} \) \((k=0,1,\ldots)\). Let us denote by \( d(a) \) the number of edges of \( G \) ending at the point \( a \), i.e. the indegree of \( a \), and by \( D(a) \) the number of edges of \( G \) starting from \( a \), i.e. the outdegree of \( a \). Clearly both \( d(a) \) and \( D(a) \) are finite for every
a ∈ V. We call a point a ∈ V an endpoint of G if D(a) = 0. We shall denote the set of endpoints of G by E and the set of endpoints of rank k of G by E_k. For any finite set A let |A| denote the number of elements of A. We put N_k = |V_k| and M_k = |V_k - E_k|, i.e. N_k denotes the total number of points of rank k, and M_k the number of those points of rank k, which are not endpoints. We shall denote further by R the maximum of r(a) for a ∈ V if V is finite and put R = +∞ if V is infinite. Clearly if M_k = 0 for some value of k then N_m = 0 for m > k and R is finite, and conversely.

Let us consider now some examples of cascade graphs.

Example 1. If G is a rooted directed tree in which all edges are directed away from the root, and D(a) is finite for every point a of G, then G is a cascade graph, its source being the root of the tree. Conversely if in a cascade graph G, d(a) = 1 for all points a different from a_0 (for which of course d(a_0) = 0) then G is a rooted directed tree in which all edges are directed away from the root.

Example 2. Let S be a finite set. Let the points of the graph G be all subsets of S and connect a ⊆ S with b ⊆ S by an edge (directed from a to b) if and only if b is obtained from a by omitting one of its elements. The graph G thus obtained is a cascade graph and for every a ⊆ S one has r(a) = |S| - |a|.

Example 3. Let S be a finite set and let the points of the graph G be all non-negative integral valued functions defined on S. If f and g are two such functions, draw an edge from f to g if and only if
\( g(x) \geq f(x) \) for all \( x \in S \) and \( \sum_{x \in S} (g(x) - f(x)) = 1 \). In this way we get a cascade graph and the rank of a function \( f \) is \( r(f) = \sum_{x \in S} f(x) \).

Let us call a cascade graph simple, if it does not contain any infinite directed path. In a simple cascade graph to every point \( a \) there corresponds a nonempty set \( T(a) \) consisting of those endpoints of the graph which can be reached from \( a \) by a directed path. We call \( T(a) \) the target-set of the point \( a \). If there is a directed path from \( a \in V \) to \( b \in V \) then \( T(b) \subseteq T(a) \).

A subset \( A \) of the vertices of a cascade graph \( G \) is called an antichain, if for any two points \( a \in A \) and \( b \in A \) there does not exist in \( G \) a directed path from \( a \) to \( b \). An antichain is called saturated, if it is not a proper subset of another antichain. An antichain is called a blocking antichain, if any directed path from the source to an endpoint and every infinite directed path starting at the source, passes through a (unique) point of the antichain. Clearly every blocking antichain is saturated but a saturated antichain is not necessarily blocking. For instance in Example 2, let \( S \) be the set \( S = \{1, 2, \ldots, n\} \) where \( n \geq 3 \), and let the antichain \( A \) consist of the two sets \( \{1\} \) and \( \{2, 3, \ldots, n\} \). Then \( A \) is saturated, as every subset of \( S \) not containing the set \( \{1\} \) is a subset of the set \( \{2, 3, \ldots, n\} \) but it is not blocking. If the cascade graph \( G \) is a rooted tree (see Example 1) then a saturated antichain is always blocking if \( G \) is finite, but not necessarily if \( G \) is infinite. (See the following Example.)

Example 4. Let the points of the graph \( G \) be all finite sequences, each term of which is one of the numbers \( 0, 1, \ldots, q-1 \) where \( q \geq 2 \);
the empty sequence is also a point of $G$. Let there be an edge from
the point $a$ to the point $b$ if the sequence $b$ is obtained from the
sequence $a$ by adding to the end of the sequence $a$ one more digit, i.e.
one of the numbers 0, 1, ..., $q-1$. In this way we obtain a cascade
graph, which is a tree, and has no endpoints. Let us take $q = 2$
and let the antichain $A$ consist of the sequences 0, 10, 110, 1110, ...
$A$ is clearly a saturated antichain, but it is not blocking as it does
not block the infinite path from the source (this being the empty sequence)
leading through the points 1, 11, 111, ...

In a simple cascade graph the set of all endpoints is a blocking
antichain. In any cascade graph the set $V_k$ is an antichain and if
there are no endpoints of rank $< k$, then $V_k$ is a blocking antichain.

Let us write, for any two points $a$ and $b$ of a cascade graph,
$a < b$ if there is a directed path from $a$ to $b$. $G$ is a partially
ordered set* with respect to the order relation $<$, but not necessarily
a lattice. The following example shows that cascade graphs which are
lattices have remarkable properties.

Example 5. Let $G$ be a finite combinatorial geometry (see H. Crapo and
G.C. Rota [1]). Let $G$ be the graph, the points of which are the flats
of $G$ and there is an edge in $G$ from the flat $a$ to the flat $b$ if
and only if $b$ covers $a$. Then $G$ is a cascade graph, which is a
lattice, and the rank of any flat $a$ in $G$ is the same as in $G$.
Let us remove the cascade graph $G$ its unique element with maximal rank:

* What we call a cascade graph is considered as a partially ordered
set a graded partially ordered sets: See Birkhoff [2] and Klarner [3],
where the graded partially ordered set with a given maximal rank and
given number of points are counted.
We obtain again a cascade graph, the endpoints of which are the copoints of $G$. This cascade graph has among others the following remarkable property: The target sets corresponding to different points are different, and the target sets of points having the same rank form a Sperner-system (i.e. none of them contains any other as a subset.)

We call a simple cascade graph in which the target sets corresponding to different points are different, a search cascade. A rooted tree of Example 1 is a search cascade if and only if $D(a) \neq 1$ for all $a \in V$, i.e. if it is a search tree, describing a simple search without noise. A search cascade which is not a tree describes a search process with noise.

One can construct a search process (with noise) which is described by a given cascade graph $G$ as follows: Let $a_1, \ldots, a_q$ be the points to which there is an edge from the source. Let the corresponding sets be $T(a_i)$ ($i=1,2,\ldots,q$). Let $S$ denote the set corresponding to the source, i.e. $S$ is the set of all endpoints of the cascade graph $G$. Then $S$ is the basic set of the search problem and if the elements of $S$ are $x_1, x_2, \ldots, x_n$ let the first test function $f_1(x)$ be defined as follows: $f_1(x)$ takes on with equal probabilities those values $j$ for which $x \in T(a_j)$. Notice that the union of the sets $T$ is equal to $S$, thus $f_1(x)$ is defined for each $x \in S$. The subsequent test functions are defined similarly. For example, the search cascade shown on Fig. 3.3.1 can be interpreted as describing the following search process: Let $S$ be the set $0, 1, 2, 3$ and let $F$ consist of the family of the 6 random variables $f_{ij}$ defined as follows ($0 \leq i < j < 3$):
Let us now consider the following strategy: We first observe
f_{03}(x^*) . If f_{03}(x^*) = 0 we select f_{02}(x^*) as the second test
function. If f_{03}(x^*) = 1 we select as the second test function
f_{13}(x^*) . If we obtained as the result of the first two observations
00, we select as the third test function f_{01} ; if the results of the
first two observations were 01 or 10 we select as the third test
function f_{12} , while if the results of the first two observations
were 11 we select f_{23} as the third test function. The correspondence
between the observations and the true value of x* is as follows:
x* is equal to the sum of the three observations. We may interpret
a search process with noise like the above example as a questionary,
with multiple choice questions such that in certain cases more than one
of the possible answers are correct, but only one of them must be
answered in the affirmative. For instance observing the test function
f_{03} is equivalent to getting an answer "yes" to one and only one of
the following two questions: 1) Is x* \leq 2 ? ; 2) Is x* \geq 1 ?
If x* = 0 , then we shall get a positive answer to the first question;
if x* = 3 we shall get a positive answer to the second question, while
if x* = 2 or x* = 3 we may get any of the two possible answers.
Thus if we get a positive answer to the first question, we know that x*
is not 3 , i.e. is in the set \{ 0, 1, 2 \} , while if we get a positive
answer to the second question we know that x* is not 0 , i.e. it is
Fig. 3.3.1
in the set \( (1, 2, 3) \). Of course if we interpret such a questionary so that if more than one answer is available, one of them is selected at random, then this interpretation is not essentially different from the interpretation as a search with noise. However there is no need to suppose definite probabilities for these choices, provided that the strategy is such that it leads in a finite number of steps to finding with certainty the object of the search.

If \( a \) is any point of the cascade graph \( G \) we denote by \( \Gamma_a \) the set of all endpoints of edges starting at \( a \). If \( A \) is any set of points of \( A \) we denote by \( \Gamma_A \) the set of those points which are the endpoints of at least one edge starting at a point in \( A \), i.e. we put \( \Gamma_A = \bigcup_{a \in A} \Gamma_a \). If \( b \) is any point of \( G \) we denote by \( \Gamma^{-1}_b \) the set of those points \( a \) for which \( b \in \Gamma_a \).

§3.2. Random walks on a cascade graph.

Let us assign to each edge \( ab \) (from \( a \) to \( b \)) of a cascade graph \( G \) a non-negative number \( w(a, b) \) such that

\[
(3.2.1.) \quad \sum_{b \in \Gamma_a} w(a, b) = 1 ,
\]

for all vertices \( a \) of \( G \) which are not endpoints. Such a function \( w(a, b) \) defines a (Markovian) random walk on the edges of \( G \) as follows: The random walk starts always from the source \( a_0 \) and proceeds to a point \( a \) of rank 1 with probability \( w(a_0, a) \); after arriving to the point \( a \), the walk continues with probability \( w(a, b) \) to a point \( b \), etc. Thus the random walk proceeds always along a directed
path of $G$, until it reaches an endpoint, while if the path is infinite, the walk continues indefinitely. Such a random walk defines uniquely a probability measure $P$ on the power set of the set of all paths starting from the source (this set being finite or denumerable). Let $B_{a}$ denote the event that the random walk arrives eventually to the point $a$. (In other words let $B_{a}$ denote the set of all paths containing the point $a$.) Let us put

\[ w(a) = P(B_{a}) \]

Let $A$ be any antichain, then by definition the events $B_{a}$ ($a \in A$) are mutually exclusive. Thus we have for every antichain $A$

\[ \sum_{a \in A} w(a) \leq 1. \]

If $A$ is a blocking antichain, then the events $B_{a}$ ($a \in A$) form a complete set of events (i.e. the sets $B_{a}$ of paths are disjoint and their union is the set of all paths), and thus we have

\[ \sum_{a \in A} w(a) = 1. \]

§3.3. Normal cascade graphs.

We shall call a cascade graph $G$ normal if the transition-probabilities $w(a, b)$ can be chosen in such a way that $w(a)$ depends only on the rank $r(a)$ of $a$, i.e.

\[ w(a) = f(r(a)) \]

where $f(x)$ is a function defined on the set of non-negative integers.
Let $B_k$ denote the event that the random walk does not stop before arriving to a point of rank $k$, and let $C_k$ denote the event that the random walk does not stop at an endpoint of rank $k$. Then we have evidently *

\[(3.3.2.) \quad P(B_{k+1}) = P(B_k)P(C_k | B_k).\]

Now let $G$ be a normal cascade graph, and suppose that the transition probabilities have been chosen so that (3.3.1) holds: In this case we call the random walk a uniform flow. In case we have a uniform flow on $G$, clearly

\[(3.3.3.) \quad P(B_k) = N_k f(k)\]

and

\[(3.3.4.) \quad P(C_k | B_k) = \frac{M_k}{N_k} .\]

It follows

\[(3.3.5.) \quad f(k+1) = f(k) \frac{M_k}{N_{k+1}}\]

and thus, as $f(0) = 1$, we get

\[(3.3.6.) \quad f(k) = \frac{1}{N_k} \prod_{j<k} \frac{M_j}{N_j} \quad \text{for } k \geq 1\]

where an empty product is by definition equal to 1. Especially if the cascade graph is finite and it has no endpoints of less than maximal rank, (i.e. for every endpoint $e$ one has $r(e) = R$), or if the

* $P(C_k | B_k)$ denotes the conditional probability of $C_k$ under condition $B_k$. 
cascade graph is infinite and it does not contain any endpoints, then
we have \( M_j = N_j \) for \( j < R \) and thus

\[
(3.3.7.) \quad f(k) = \frac{1}{N_k}.
\]

The following theorem is an immediate consequence of (3.2.3)
and (3.3.6):

**THEOREM 3.3.1.** Let \( G \) be a normal cascade graph. Let \( N_k \) denote
the total number of points of rank \( k \) of \( G \) and \( M_k \) the number of those
points of rank \( k \) which are not endpoints. Let \( A \) be an antichain in
\( G \) and let \( n_k \) denote the number of points of rank \( k \) in \( A \). Then
the inequality

\[
(3.3.8.) \quad \sum_{k=0}^{R} \frac{n_k}{N_k} \leq 1
\]

holds; if \( A \) is a blocking antichain there is equality in (3.3.8).

**COROLLARY:** If \( M_k = N_k \) for \( k < R \), where \( R \) is the maximal
rank of points of \( G \), then for every antichain \( A \) we have

\[
(3.3.9.) \quad \sum_{k=0}^{R} \frac{n_k}{N_k} \leq 1
\]

with equality standing in (3.3.9) if \( A \) is a blocking antichain.

**Remark:** Notice that (3.3.9) can be written also in the equivalent form

\[
(3.3.10.) \quad \sum_{a \in A} \frac{1}{N_r(a)} \leq 1.
\]

We shall refer to Theorem 1. as the uniform flow theorem. In the next §
we shall give a necessary and sufficient condition for the normality of
a cascade graph.
3.4. A necessary and sufficient condition for the normality of a cascade graph.

**THEOREM 3.4.1.** A cascade graph $G$ is normal if and only if it satisfies the following condition: For every $k \geq 1$ and for every subset $A$ of the set $V_k - E_k$ of points of rank $k$ which are not endpoints, one has

$$M_k |\Gamma A| \geq N_{k+1} |A|$$

where $N_{k+1}$ is the number of points of rank $k+1$, and $M_k$ the number of points of rank $k$ which are not endpoints.

**Proof** of Theorem 3.4.1. Clearly for every random walk (i.e. probability flow) on $G$ and for every set of points $A \subseteq V-E$ one has

$$\sum_{a \in A} w(a) \leq \sum_{b \in \Gamma A} w(b)$$

Thus if $A \subseteq V_k - E_k$ and the flow is uniform then (3.4.1) holds, i.e. the condition is necessary. Let us prove now its sufficiency, i.e. that if (3.4.1) holds, one can choose the transition probabilities $w(a, b)$ so that the flow should be uniform. We shall prove the existence of such transition probabilities $w(a, b)$ step by step, i.e. by induction on the rank $k$ of $a$. Clearly if we put for every point $b$ of rank 1 $w(a_0, b) = \frac{1}{N_1}$, then we have $w(b) = \frac{1}{N_1}$ for every point $b$ of rank 1. Let us suppose that we have already determined $w(a, b)$ for all points $a$ of rank $< k$ so that (3.3.1) holds for all $a \in V_k$. We have to show that one can choose the values of $w(a, b)$ for all

* The proof given here is due to Dr. G. Katona.
a ∈ V_k so that (3.3.1) holds with k + 1 instead of k too.

In other words, we have to choose the transition probabilities

\[ w(a, b) \] for \( a ∈ V_k - E_k \) in such a way that they should be non-negative

and should satisfy the following two sets of equations:

\[
(3.4.3.) \quad \sum_{b ∈ \Gamma a} w(a, b) = 1 \quad \text{for all} \quad a ∈ V_k - E_k
\]

and

\[
(3.4.4.) \quad \sum_{a ∈ \Gamma^{-1} b} w(a, b) = \frac{M_k}{N_{k+1}} \quad \text{for all} \quad b ∈ V_{k+1}
\]

Let \( a_1, a_2, \ldots, a_{M_k} \) and \( b_1, b_2, \ldots, b_{N_{k+1}} \) denote the elements

of the sets \( V_k - E_k \) and \( V_{k+1} \) respectively. We consider the auxiliary

graph \( G^* \) defined as follows: \( G^* \) has \( 2M_k N_{k+1} \) points which we denote

by \( a_{i,j} \) (\( 1 ≤ i ≤ M_k ; 1 ≤ j ≤ N_{k+1} \)) and \( b_{u,v} \)

(\( 1 ≤ u ≤ N_{k+1} ; 1 ≤ v ≤ M_k \)). We connect the points \( a_{i,j} \) and \( b_{u,v} \)

in \( G^* \) if and only if there is in \( G \) and edge from \( a_{i,j} \) to \( b_u \).

Thus \( G^* \) is a bipartite graph with the two classes of points

\( A = \{a_{i,j}\} \) and \( B = \{b_{u,v}\} \). Let for any subset \( A^* \) of the set \( A \)

denote the subset of those \( b_{u,v} \) which are connected by at least one

\( a_{i,j} \) in \( G^* \). Let \( A \) denote the set of those \( a_i ∈ V_k - E_k \) for which

\( a_{i,j} \) is for at least one \( j \) contained in \( A^* \). Then we have

\[
(3.4.5.) \quad |A^*| ≤ N_{k+1} |A|
\]

and

\[
(3.4.6.) \quad |\Gamma^* A^*| = M_k |\Gamma A|
\]
Thus (4.1) implies

\[(3.4.7.) \quad |\Gamma A^*| = M_k |\Gamma A| \geq N_{k+1} |A| \geq |A^*|.
\]

But (3.4.7) means that the conditions of the marriage-theorem (see e.g. Harper, L. and Rota, G.C. [4]) for the existence of a one-to-one matching between the sets \(A\) and \(B\) so that each \(a_{i,j}\) is matched with such a \(b_{u,v}\) with which it is connected in \(G^*\), are fulfilled, and thus such a matching exists. Let us take such a matching and let \(s(i, u)\) denote the number of \(a_{i,j}\) \((1 \leq j \leq N_{k+1})\) which are matched to \(b_{u,v}\) \((1 \leq v \leq M_k)\).

Then we have evidently

\[(3.4.8.) \quad \sum_{u=1}^{N_{k+1}} s(i, u) = N_{k+1} \text{ for } 1 \leq i \leq M_k\]

and

\[(3.4.9.) \quad \sum_{i=1}^{M_k} s(i, u) = M_k\]

Further \(s(i, u) = 0\) if \(a_i\) and \(b_u\) are not connected in \(G\). Thus putting

\[(3.4.10.) \quad w(a_i, b_u) = \frac{s(i, u)}{N_{k+1}} \text{ (1 \leq i \leq M_k, } b_u \in \Gamma a_i)\]

the equations (3.4.3) and (3.4.4) hold. Thus Theorem 2 is proved.

From Theorem 3.4.1 one can easily deduce the following:

**COROLLARY:** If in a cascade graph \(G\) the outdegree \(D(a)\) of a point \(a\), which is not an endpoint, depends only on the rank \(r(a)\) of \(a\), and the indegree \(d(b)\) of any point \(b\) depends only on the rank \(r(b)\) of \(b\) then \(G\) is normal.
Proof: Let us denote the outdegree of a point of rank \( k \) which is not an endpoint by \( D_k \), and the indegree of a point of rank \( k \) by \( d_j \). As the total number of edges starting from some point of rank \( k \) is equal to the total number of edges leading to a point of rank \( k + 1 \), we have

\[
(3.4.11.) \quad M_k \cdot D_k = N_k + 1 \cdot d_{k+1}.
\]

Now let \( A \) be any subset of \( V_k - E_k \). As the number of edges going out from one of the points in \( A \) can not be larger than the number of edges arriving to a point in \( \Gamma A \), we have

\[
(3.4.12) \quad d_{k+1} |\Gamma A| \geq D_k |A|.
\]

Multiplying both sides of (3.4.12) by \( N_k + 1 \), and using (3.4.11) we get that (3.4.1) holds, i.e. that \( G \) is normal.

Remark. Instead of deducing the above corollary from Theorem 2 one can prove its statement by constructing effectively the uniform flow on the cascade graph. As a matter of fact, if for every \( a \) which is not an endpoint we put \( w(a, b) = |\Gamma a|^{-1} \), then we get a uniform flow on the cascade graph \( G \) satisfying the conditions of the corollary.

The cascade graphs satisfying the conditions of the above Corollary of Theorem 3.4.1 are called semiregular cascade graphs. As by the Corollary every semiregular cascade graph is normal, it follows that the statement of Theorem 3.3.1 holds for every antichain of a semiregular cascade graph; this special case of Theorem 3.3.1 is due to Kirby A. Baker [5],
who has formulated this result in a slightly different terminology, but the result expressed in our terminology is just the statement of Theorem 3.3.1 for semiregular cascade graphs.

§3.5. Kraft's inequality and Sperner's theorem as special cases of the uniform flow theorem.

In spite of the extreme simplicity of its proof, Theorem 3.1.1 is a common source of several known results, thus for example Kraft's (see e.g., Feinstein [6]) inequality which we will need in the next chapter, and Sperner's theorem (see e.g. Lubell [7]). In this § we shall deduce these theorems and some of their generalizations as special instances of the uniform flow theorem.*

Let us deal first with the Kraft inequality. Let us consider the cascade graph described in Example 4: Let the points of $G$ be all finite sequences which can be formed from the digits $0, 1, \ldots, q-1$ ($q \geq 2$) and let us draw a directed edge from the sequence $a$ to the sequence $b$ if and only if $b$ is obtained by adding one more digit to the end of $a$. Let the source be the empty sequence. It is easy to see that the cascade graph thus obtained is a tree, moreover a regular search tree in which there are $q$ edges going out from every point and no endpoints. Let $A$ be an antichain in $G$; thus $A$ is a finite or denumerable family of finite sequences formed from the digits $0, 1, \ldots, q-1$ such that no sequence in $A$ is an initial segment (prefix) of another sequence in $A$. Such families of sequences

* I want to mention that I obtained the uniform flow theorem by analysing Lubell's elegant proof of Sperner's theorem [7].
are called in information theory q-ary prefix codes, and its elements codewords. Evidently G is a normal cascade graph; to show this we do not need Theorem 2, because it is easy to see that putting \( w(a, b) = q^{-1} \) for every edge \( a, b \) of G we get \( w(a) = q^{-r(a)} \), i.e. a uniform flow.

Applying Theorem 1 we obtain the following result, called Kraft's inequality:

\[
\text{If } A \text{ is a q-ary prefix code } (q \geq 2) \text{ and } n_k \text{ denotes the number of codewords of length } k \text{ in the code, then the inequality}
\]

\[
\sum_{k=0}^{\infty} \frac{n_k}{q^k} \leq 1
\]

holds; if \( A \) has the property that every infinite sequence consisting of the digits 0, 1, ..., \( q-1 \) contains one of the codewords from \( A \) as an initial segment, then there is equality in (3.5.1).

Clearly (3.5.1) can be formulated also in another form, speaking about trees instead of codes. It is natural to ask in general which rooted trees, considered as cascade graphs, are normal. The answer is very simple: A rooted tree is a normal cascade graph if and only if it is semiregular, i.e. the outdegree of any point which is not an endpoint depends only on the rank of the point. If \( D(a) = q_k \) for all \( a \in V_k - E_k \) then

\[
N_{k+1} = M_k \cdot q_k
\]

and thus we get from the uniform flow theorem the following result, which reduces to the Kraft inequality in the special case when \( q_k = q \) for all \( k \).

\[
\text{If } A \text{ is an antichain in a semiregular rooted tree such that } D(a) = q_k \text{ if } a \in V_k - E_k \text{ and if } n_k \text{ denotes the number of elements of the antichain } A \text{ having rank } r, \text{ then the inequality}
\]

\[
\sum_{k=0}^{\infty} \frac{n_k}{q^k} \leq 1
\]
(3.5.3.) \[
\sum_{k=0}^{\infty} \frac{n_k}{q_1 q_2 \cdots q_k} \leq 1
\]

holds.

Of course Kraft's inequality (as well as its generalization) can be proved quite easily directly, but it is instructive to consider this inequality as a particular instance of the uniform flow theorem.

Now let us consider how Sperner's theorem is obtained from Theorem 3.3.1. Let \( G \) be the cascade graph of Example 2 of § 3.1. Let the points of \( G \) be all subsets of an \( n \)-element set \( S \), and let there be an edge in \( G \) from the set \( a \subseteq S \) to the set \( b \subseteq S \) if and only if \( b \) is obtained from \( a \) by omitting one of its elements.

To show that the cascade graph \( G \) is normal it is sufficient to point out that if we put

\( (3.5.3.) \quad w(a, b) = \frac{1}{n-r(a)} \) for every \( a \) with \( r(a) < n \)

(notice that \( r(a) = n \) only if \( a \) is the empty set and this is the (unique) endpoint of \( G \)), then we get, taking into account that there are \( (n-k)! \) paths from the source to a point \( a \) for which \( r(a) = n - k \) i.e. to set \( a \) having \( k \) elements, it follows that

\( (3.5.4.) \quad w(a) = \frac{1}{n \choose r(a)} \) for all points \( a \),

i.e. there exists a uniform flow in \( G \). (This follows also from the corollary of Theorem 3.4.1.) It is easy to see further that a subset \( A \) of the points of \( G \), i.e. a family of subsets of \( S \) is an antichain if for any two different sets \( a \subseteq S \) and \( b \subseteq S \) belonging to \( A \), \( a \)
is not a subset of \( b \), i.e. if \( A \) is a Sperner-system of subsets \( S \).

Thus the uniform flow theorem yields the following result:

If \( A \) is a Sperner system of subsets of an \( n \)-element set, and \( A \) contains \( n_k \) sets having \( k \) elements \( (k=0,1,...) \) then the inequality

\[
\sum_{k=0}^{n} \frac{n_k}{\binom{n}{k}} \leq 1
\]

holds.

As

\[
\max_{k} \binom{n}{k} = \left( \begin{array}{c} n \\ \left\lfloor \frac{n}{2} \right\rfloor \end{array} \right)
\]

we obtain from (3.5.6) the usual (though slightly weaker) form of Sperner's theorem:

\[
|A| \leq \left( \begin{array}{c} n \\ \left\lfloor \frac{n}{2} \right\rfloor \end{array} \right).
\]

Before going further let us add a remark. Comparing the two special cases just discussed, it turns out that the property of a system of sets being a Sperner-system plays the same role in Sperner's theorem as the prefix property of a code in Kraft's inequality. As a matter of fact, there is a real connection between these two concepts, not only a superficial analogy.

As mentioned earlier the prefix property of a code implies that if the codewords of such a code are written one after the other, without indicating where one codeword ends and the next begins, and if the code has the prefix property, the sequence of symbols can be uniquely decoded,
i.e. the codewords can be unambiguously separated from another. Now let us impose on the code the auxiliary restriction that a sequence of codewords should be uniquely decodable even in the case when the letters within each codeword are arbitrarily rearranged, i.e. if the codewords are unordered sets of letters, and not ordered sets as usual. If we require further that the same letter should not occur more than once in any codeword, then such a code is uniquely decodable if the codewords (considered as unordered sets of letters) form a Sperner system.

Expressed in the language of search theory, a Sperner system corresponds to such a strategy of search in which the values of the test functions are obtained simultaneously and one does not know which value comes from which test function.

For example let us consider the 19 numbers

0, 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 14, 16, 17, 18, 19, 23, 24, 29

These numbers can be uniquely characterized by their residues mod 2, mod 3, mod 5, even if these three residues are given in a random order. For instance if we are told that the three residues are 0, 0, and 2, it is easy to see that among the 19 numbers listed above only 20 has these residues, namely it is congruent to 0 mod 2 and mod 5 and to 2 mod 3.

As a further application of the uniform flow theorem, we prove the following generalization of Sperner's theorem:

THEOREM 3.5.1. Let \( \mathcal{A} \) be a family of ordered \( r \)-tuples \((r \geq 1)\) of disjoint subsets of an \( n \)-element set \( S \), such that if \((A_1, A_2, \ldots, A_r)\) and \((B_1, B_2, \ldots, B_r)\) both belong to the family \( \mathcal{A} \) then...
A then the relations \( A_j \subseteq B_j \) \((j=1,2,\ldots,r)\) cannot hold simultaneously. Then the number of elements of the family \( A \) satisfies the inequality

\[
|A| \leq r \left\lfloor \frac{r(n+1)}{r+1} \right\rfloor \cdot \left( \frac{n}{\left\lfloor \frac{r(n+1)}{r+1} \right\rfloor} \right).
\]

which is best possible, i.e. equality is possible in (5.8) by a suitable choice of the class \( A \).

Proof of Theorem 3.5.1. Let us construct a cascade graph \( G \) as follows:

The points of \( G \) are all possible \((r+1)^n\) ordered \( r \)-tuples 
\((A_1, A_2, \ldots, A_r)\) of disjoint subsets of \( S \). The source of \( G \) is the \( r \)-tuple each element of which is the empty set. The rank of an \( r \)-tuple \( a = (A_1, A_2, \ldots, A_r) \) is \( r(a) = \sum_{j=1}^{r} |A_j| \) and there is an edge from \( a = (A_1, A_2, \ldots, A_r) \) to \( b = (B_1, B_2, \ldots, B_r) \) if and only if the following conditions are satisfied: \( A_j \subseteq B_j \) for \( j=1,2,\ldots,r \) and \( r(b) = r(a) + 1 \); in other words there is in \( G \) an edge from \( a \) to \( b \) if \( A_j = B_j \) for all but one value of \( j \) \((1 \leq j \leq r)\)--say except for \( i \)--and \( B_i \) is obtained from \( A_i \) by adding one more element of \( S \) to \( A_i \). Clearly the conditions of Theorem 3.5.1. mean that \( A \) should be an antichain in \( G \). Thus if we show that \( G \) is normal, we can apply Theorem 1

We shall prove the normality of \( G \) by verifying that the conditions of the corollary to Theorem 3.3.1 are fulfilled. Let \( a = (A_1, A_2, \ldots, A_r) \) be any point in \( G \) which is not an endpoint; then we have \( r(a) < n \) because the endpoints of \( G \) are the points with \( r(a) = n \). Thus there are \( r - r(a) \) elements of \( S \) which can be added to one of the sets
A \_1 to get a point \_b to which there leads an edge from \_a: The number of possible choices is thus \( r \cdot (n-r(a)) \). Thus \( D(a) = r \cdot (n-r(a)) \) depends only on the rank of \( a \in V - E \). Let now \( b \) be any point of \( G \). By a similar argument we get that \( D(b) = r(b) \). Thus all conditions of the Corollary of Theorem 3.3.1 are satisfied, and therefore \( G \) is normal. Thus we can apply Theorem 3.3.1 to the antichain \( A \) and we get, in view of \( N_k = r^k \binom{n}{k} \) and \( M_k = N_k \) if \( k < n \), that

\[
\sum_{k=0}^{n} \frac{n_k}{\binom{n}{k}} \leq 1,
\]

(3.5.9.)

where \( n_k \) denotes the number of elements of \( A \) having rank \( k \).

As however

\[
\max_{0 \leq k \leq n} n^k \binom{n}{k} = n^r \binom{n}{r+1}
\]

(3.5.10.)

it follows that (3.5.8) holds.

Clearly for \( r = 1 \) Theorem 3.5.1 is nothing else than Sperner's theorem. To show that the inequality (3.5,8) is best possible take for \( A \) the family of all \( r \)-tuples \( (A_1, A_2, \ldots, A_r) \) of disjoint subsets of \( S \) such that \( \sum_{j=1}^{r} |A_j| = \frac{(n+1)r}{r+1} \) (i.e. all points of rank \( \frac{(n+1)r}{r+1} \) of \( G \)); \( A \) clearly satisfies the requirements of the Theorem 3.5.1 and \( |A| \) is equal to the right hand side of (3.5.8).

Other generalizations of Sperner's theorem can also be obtained from Theorem 3.3.1.
References to Chapter III.


