TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Discrete Mathematics

Nguyễn Khánh Phương

Department of Computer Science
School of Information and Communication Technology
E-mail: phuongnk@soict.hust.edu.vn

PART 1

COMBINATORIAL THEORY

(Lý thuyết tổ hợp)

PART 2

GRAPH THEORY

(Lý thuyết đồ thị)

Contents of Part 1: Combinatorial Theory

Chapter 1. Counting problem

- This is the problem aiming to answer the question: "How many ways are there that satisfy given conditions?" The counting method is usually based on some basic principles and some results to count simple configurations .
- Counting problems are effectively applied to evaluation tasks such as calculating the probability of an event, calculating the complexity of an algorithm (how long the algorithm will take to run), ...

Street art

Given N paintings in a row over a distance of M centimeters.
Each painting i ($1 \le i \le N$) will be drawn on a length of t_i cm, so $t_1+t_2+..+t_n = M$.
The K city's most famous artists have been selected to do this work, each artist will be assigned to draw at least one painting. To facilitate the artist's work, if someone is assigned to draw more than one painting, the paintings must be adjacent to each other on the street art

Contents of Part 1: Combinatorial Theory

Chapter 1. Counting problem

- This is the problem aiming to answer the question: "How many ways are there that satisfy given conditions?" The counting method is usually based on some basic principles and some results to count simple configurations .
- Counting problems are effectively applied to evaluation tasks such as calculating the probability of an event, calculating the complexity of an algorithm

Chapter 2. Existence problem

In the counting problem, configuration existence is obvious; in the existence problem, we need to answer the question: "Is there a combinatorial configuration that satisfies given properties ?"

Chapter 3. Enumeration problem

This problem is interested in giving all the configurations that satisfy given conditions.

Chapter 4. Combinatorial optimization problem

- Unlike the enumeration problem, this problem only concerns the "best" configuration in a certain sense.
- In the optimization problems, each configuration is assigned a numerical value (which is the use value or the cost to construction the configuration), and the problem is that among the configurations that satisfy the given conditions, find the configuration with the maximum or minimum value assigned to it

Contents of Part 1

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chapter 1

COUNTING PROBLEM

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Contents

- Combinatorics is the branch of mathematics devoted to calculate the number of ways in which a specified process can be carried out. Problems of this type often occur in computing. A typical problem is to determine how many times a particular sequence of steps in an algorithm will be executed. You might need to know this in order to estimate how long the algorithm will take to run.
- For example: determine the shortest path for a signal to travel through a communication network
  - Algorithm 1 (simplest): calculate the lengths of all the possible paths through the network.
  - Algorithm 2: just calculate the lengths of some **selected paths**

➔ In order to determine how efficient these algorithms are, and to be able to compare them with each other and with other algorithms, we must first answer the question: 'How many such paths are there altogether?'

Contents

1. Basic counting principles

1.1. The sum rule

1.2. The product rule

1.3. Tree diagram

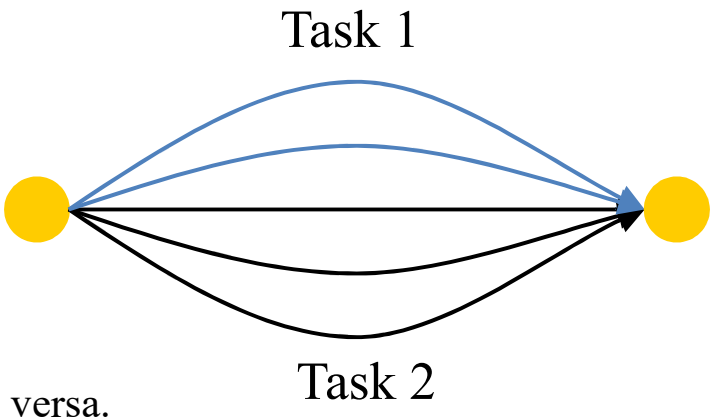
1.1. The sum rule

Example:

- 40 textbooks on Mathematics; 50 textbooks on English
- to select 1 book: 40+50 choices
- What about selecting 2 books?

problem → decompose → combine

Independent tasks



Let us consider two tasks:

m_1 is the number of ways to do **task 1**

m_2 is the number of ways to do **task 2**

Tasks are <u>independent</u> of each other, i.e.,

Performing **task 1** does not accomplish **task 2** and vice versa.

<u>Sum rule</u>: the number of ways that "**either** task 1 **or** task 2 can be done, but **not both**", is $\boldsymbol{m_1 + m_2}$.

Generalizes to multiple tasks ...

1.1. The sum rule

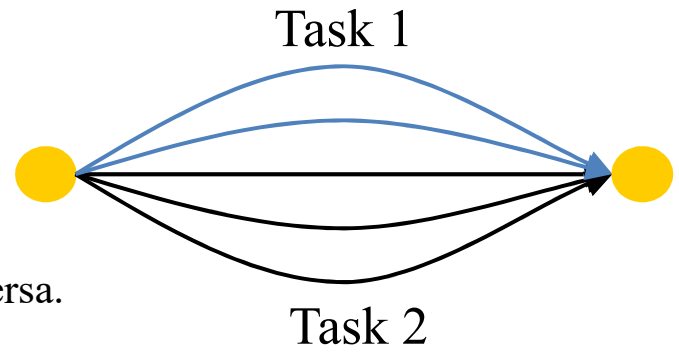
Let us consider two tasks:

m_1 is the number of ways to do **task 1**

m_2 is the number of ways to do **task 2**

Tasks are <u>independent</u> of each other, i.e.,

Performing **task 1** does not accomplish **task 2** and vice versa.



<u>*Sum rule*</u>: the number of ways that "**either** task 1 **or** task 2 can be done, but **not both**", is $\boldsymbol{m_1 + m_2}$.

Generalizes to multiple tasks ...

Things	1	2	3	…	*k*
ways	m_1	m_2	m_3	…	m_k

select one of them: $m_1 + m_2 + m_3 + ... + m_k$ ways

1.1. The sum rule

Example:

- 40 textbooks on Mathematics; 50 textbooks on English
- to select 1 book: 40+50 choices
- What about selecting 2 books?

problem ⟶ decompose ⟶ combine

Independent tasks

Generalized sum rule:

Tasks	1	2	3	…	*k*
ways	m_1	m_2	m_3	…	m_k

select one of them: $m_1 + m_2 + m_3 + ... + m_k$ ways

1.1. The sum rule

Example: How many 6-character strings are there, each of which is taken from the set {A, B, C, D, E} containing **at least** 3 different characters? For example, the string ABCCAC is the string that satisfies the problem condition, while the string ABABAB does not.

problem ⟶ decompose ⟶ combine

Independent tasks

Generalized sum rule:

Tasks	1	2	3	…	*k*
ways	m_1	m_2	m_3	…	m_k

select one of them: $m_1 + m_2 + m_3 + ... + m_k$ ways

1.1. The sum rule

Generalized sum rule: If we have tasks $T_1, T_2, \ldots, T_k$ that can be done in $m_1, m_2, \ldots, m_k$ ways, respectively, and any two of these tasks can not be done at the same time, then there are $m_1 + m_2 + \ldots + m_k$ ways to do one of these tasks.

➔The sum rule can also be phrased in terms of *set theory:* The size of the union on k finite pair wise disjoint sets is the sum of their sizes:

- Let $A_1, A_2, \ldots, A_k$ be disjoint sets. Then the number of ways to choose any element from one of these sets is

$$|A_1 \cup A_2 \cup \ldots \cup A_k| = |A_1| + |A_2| + \ldots + |A_k|.$$

1.1. The sum rule

Example 1: A student can choose a computer project from one of three lists. The three lists contain 23, 15, and 19 possible projects respectively. How many possible projects are there to choose from?

23+15+19 ☺

Example 2: How many strings of 4 decimal digits, have exactly three digits that are 9s?

The string can have:

- The non-9 as the first digit (x999)
- OR the non-9 as the second digit (9x99)
- OR the non-9 as the third digit (99x9)
- OR the non-9 as the fourth digit (999x)

Thus, we use the sum rule:

- For each of those cases, there are 9 possibilities for the non-9 digit (any number other than 9)
- Thus, the answer is 9+9+9+9 = 36

1.1. The sum rule

Example 3: What is the value of *k* once this program is executed?

```
n1=10; n2=20; n3=30;
k=0;
for (i1= 1; i1<=n1;i1++)   k=k+1;
for (i2= 1; i2<=n2;i2++)   k=k+1;
for (i3= 1; i3<=n3;i3++)   k=k+1;
```

Answer: At the beginning, the value of **k** is assigned to 0. There are 3 independent "for" loops. After each for loop, the value of **k** is increased by one:

- The first "for" loop iterates 10 times,
- The second "for" loop iterates 20 times,
- The last "for" loop iterates 30 times

Therefore, at the end, the value of **k** = 10 + 20 + 30 = 60

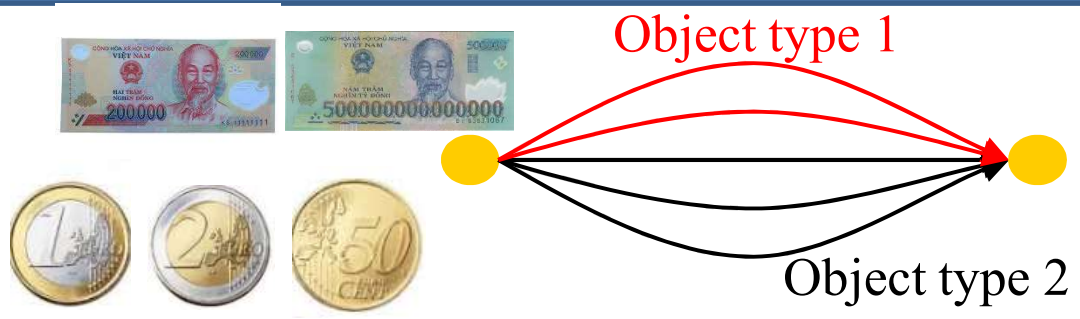
1. Basic counting principles

1.1. The sum rule

1.2. The product rule

1.3. Tree diagram

The sum rule



Consider 2 types of objects:

m_1 is number of object type 1

m_2 is number of object type 2

The decision to choose object type 1 or 2 is independent of each other, that is,

- the decision to choose object type 1 does not affect the decision to choose object type 2, and vice versa.

Sum rule: The number of ways to choose an object type 1 or an object type 2 is $\boldsymbol{m_1 + m_2}$

Generalize to multiple types of objects:

Object type	1	2	3	…	*k*
Quality	m_1	m_2	m_3	…	m_k

The number of ways to choose one object from these *k* types of objects
$= m_1 + m_2 + m_3 + ... + m_k$

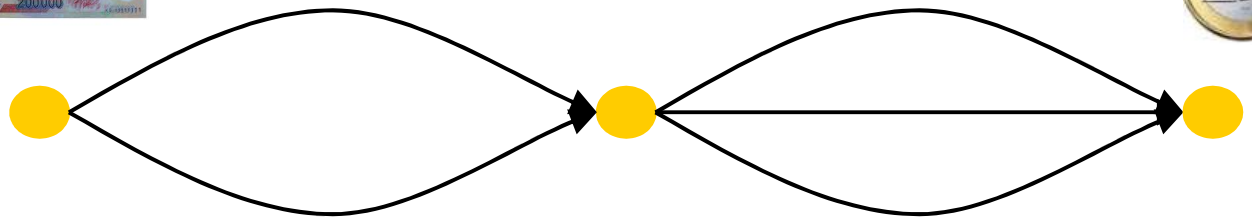
The product rule

- Consider 2 types of objects:
  - m_1 is number of object type 1
  - m_2 is number of object type 2

The decision to choose object type 1 or 2 is independent of each other, that is,

- the decision to choose object type 1 does not affect the decision to choose object type 2, and vice versa.

Product rule: The number of ways to choose an object type 1 and an object type 2 is $\boldsymbol{m_1 m_2}$

- Generalize to multiple types of objects:

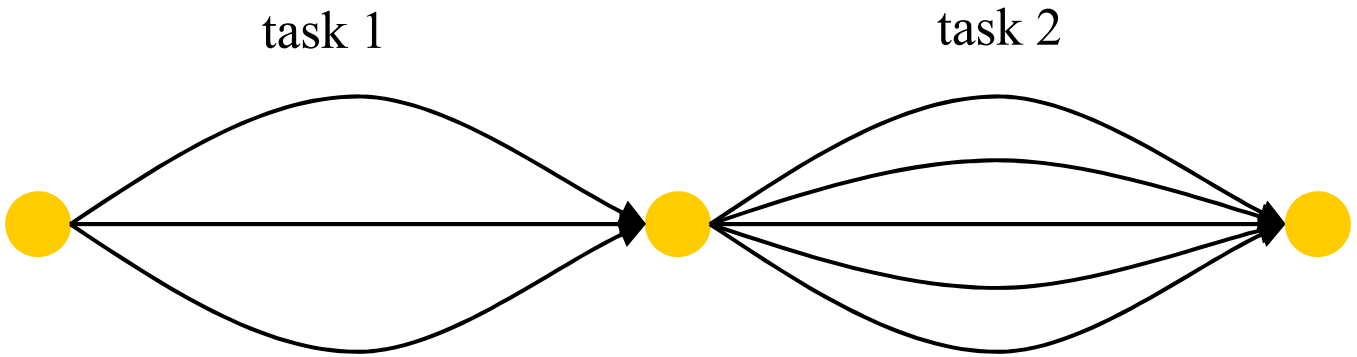
Object type	1	2	3	…	*k*
Quality	m_1	m_2	m_3	…	m_k

The number of ways to select exact k objects, not any 2 objects of same type
$= m_1 m_2 m_3 \ldots m_k$

The product rule

- Consider two tasks:
  - m_1 is the number of ways to do task 1
  - m_2 is the number of ways to do task 2
  - Tasks are independent of each other, i.e.,
    - Performing task 1does not accomplish task 2 and vice versa.

Product rule: the number of ways that "both tasks 1 and 2 can be done" in $m_1 m_2$.



- Generalize to multiple tasks ...

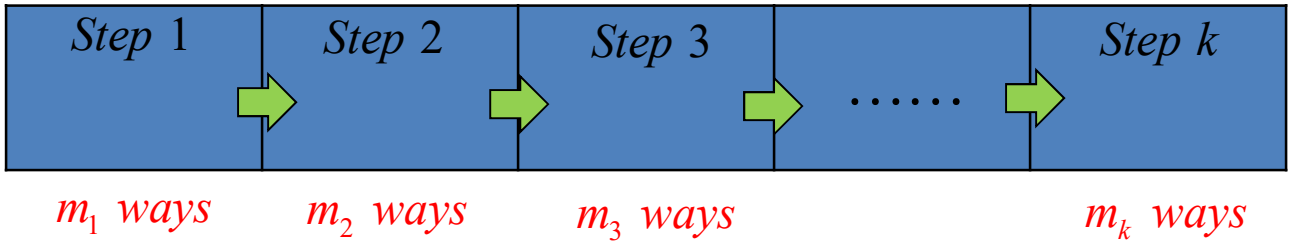
Analysis

License Plate LL-DDDD 2 letters – 4 digits

# of possible plates = ?

1.2. The product rule

Suppose a **procedure** can be constructed by a series of **steps**



Number of possible ways to complete the procedure is

$$m_1 * m_2 * \cdots * m_k$$

Analysis

License Plate **LL-DDDD** 2 letters – 4 digits

# of possible plates = ?

Procedure:

Step 1:

Step 2:

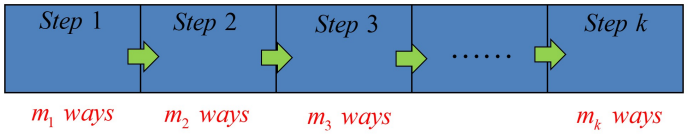
Step 3:

Step 4:

Step 5:

Step 6:

Suppose a procedure can be constructed by a series of steps



Number of possible ways to complete the procedure is

$$m_1 * m_2 * \cdots * m_k$$

1.2. The product rule

Generalized product rule: If we have a procedure consisting of sequential tasks $T_1, T_2, \ldots, T_k$ that can be done in $m_1, m_2, \ldots, m_k$ ways, respectively, then there are $m_1 * m_2 * \ldots * m_k$ ways to carry out the procedure.

➔The product rule can also be phrased in terms of *set theory:* Let $A_1, A_2, \ldots, A_k$ be finite sets. Then the number of ways to choose one element from each set in the order of $A_1, A_2, \ldots, A_k$ is

$$|A_1 \text{ x } A_2 \text{ x } \ldots \text{ x } A_k| = |A_1| * |A_2| * \ldots * |A_k|.$$

The product rule

- If each element a_i of k- tuple $(a_1, a_2, ..., a_k)$ has m_i number of ways to select ($i = 1, 2, ..., k$), then the number of tuples could be generated is the product of these $m_1 m_2 ... m_k$

The product rule

In many counting problems, it is only after building the first element, we just know how to build the second element; after building the first two elements we only know how to build the third element,… In that case, we use the general product rules:

Suppose we construct a k-tuple $(a_1, a_2, ..., a_k)$ by building each element in turnand

- a_1 can be chosen by m_1 way;
- After a_1 is selected, a_2 can be chosen by m_2 way;
- ...
- After $a_1, a_2, ..., a_{k-1}$ are chosen, a_k can be chosen by m_k ways;

Then the number of k-tuples generated is $\boldsymbol{m_1 m_2 \ldots m_k}$

Example: 6 persons are competing for 4 prizes. How many different outcomes are possible?

A 4-tuple = (1st prize, 2nd prize, 3rd prize, 4th prize)

1.2. The product rule

Example 1: There are 18 math majors and 325 CS majors. How many ways are there to pick one math major and one CS major?

Total is 18 * 325 = 5850

Example 2: The license plate: 2 letters-4 digits

(a) no letter or digit can be repeated

$$26 \times 25 \times 10 \times 9 \times 8 \times 7$$

(b) with repetitions allowed

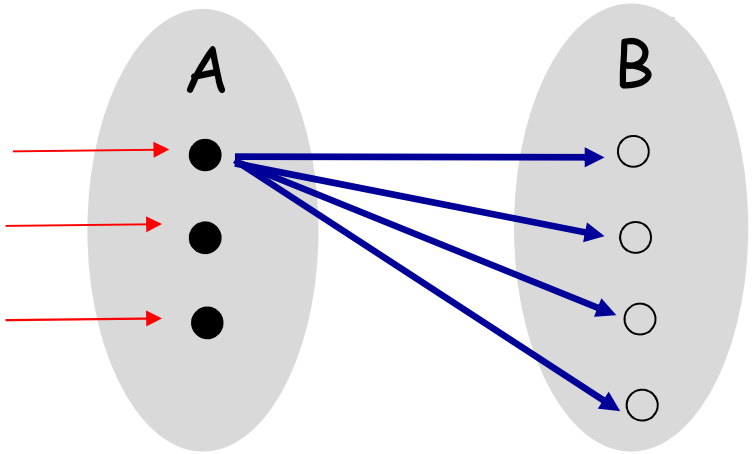
$$26 \times 26 \times 10 \times 10 \times 10 \times 10$$

(c) same as (b), but only vowels and even digits

$$5^2 x 5^4$$

1.2. The product rule

Example 3: How many functions are there from set A to set B?



So, how many Boolean functions on *n* variables?

$$2^{2^n}$$

To define each function we have to make 3 choices, one for each element of A.
Each has 4 options (to select an element from B).

How many ways can each choice be made?

$4 \times 4 \times 4$

$4^3 = 64 = |B|^{|A|}$

1.2. The product rule

Example 4: How many strings of 4 decimal digits

(a) do not contain the same digit twice?

- We want to choose a digit, then another that is not the same, then another…
  - 1st digit: 10 possibilities (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
  - 2nd digit: 9 possibilities (all but not first digit)
  - 3rd digit: 8 possibilities
  - 4th digit: 7 possibilities
- Total = 10*9*8*7 = 5040

(b) end with an even digit?

- First three digits have 10 possibilities (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- Last digit has 5 possibilities (0, 2, 4, 6, 8)
- Total = 10*10*10*5 = 5000

1.2. The product rule

Example 5: Consider the following nested loop:

```
for (i=1;i<=9;i++)
        for (j=1;j<=7;j++) {
                [ Statement 1;
                  Statement 2; ]
        }
```

How many times the statements in the inner loop will be executed?

Solution: $9 \times 7 = 63$ times (*based on the product rule*)

More complex counting problems

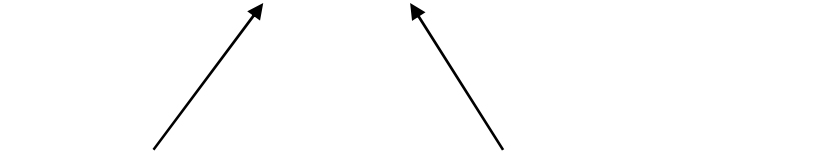
- **Combining the product rule and the sum rule.**
- **Thus we can solve more interesting and complex problems.**

More complex counting problems

Example 1: How many BASIC variables: single letter

or single letter+single digit

$26 + 26 * 10 = 286$



rule of sum rule of product

Example 2: How many integers from 1 through 999 do not have any repeated digits?

Let A = integers from 1 to 999 not having repeated digits.

Partition A into 3 sets:

- A_1=one-digit integers not having repeated digits;
- A_2=two-digit integers not having repeated digits;
- A_3=three-digit integers not having repeated digits.

➔ $|A| = |A_1| + |A_2| + |A_3|$ (*by the sum rule*)

$= 9 + 9 \times 9 + 9 \times 9 \times 8 = 738$ (*by the product rule*)

Count the number of ways to put things together into various combinations.

Example 3: Passwords consist of character strings of 6 to 8 characters. Each character is an upper case letter or a digit.

26 (A-Z) **10** (0-9)

Each password must contain at least one digit. How many passwords are possible?

constraint

- Let P – total number of possible passwords
- P_i – total number of passwords of length i, $i = 6,7,8$

➔ $P = P_6 + P_7 + P_8$ (the sum rule)

P_i – computing it directly is tricky (hmm…) How??

- **"popular" counting trick: let's calculate all of them, including those with no digits and then subtract the ones with no digits.**

P6 = (# possibilities without constraint) – (# passwords with no digits)

= 36^6 – 26^6

$P7 = 36^7 - 26^7$

$P8 = 36^8 - 26^8$

➔ $P = P6 + P7 + P8 = 36^6-26^6 + 36^7 - 26^7 + 36^8 - 26^8 = 2,684,483,063,360$

Count the number of ways to put things together into various combinations.

Example 3: Passwords consist of character strings of 6 to 8 characters. Each character is an upper case letter or a digit.

26 (A-Z) **10** (0-9)

Each password must contain at least one digit. How many passwords are possible?

constraint

- Let P – total number of possible passwords

➔ $P = P6 + P7 + P8 = 36^6 - 26^6 + 36^7 - 26^7 + 36^8 - 26^8 = 2{,}684{,}483{,}063{,}360$

Comment: If your computer can try 200 million passwords per second, then how long can the password be determined, so the intruder could penetrate this computer system?

(2 684 483 063 360/200 000 000) / (60 * 60) hours

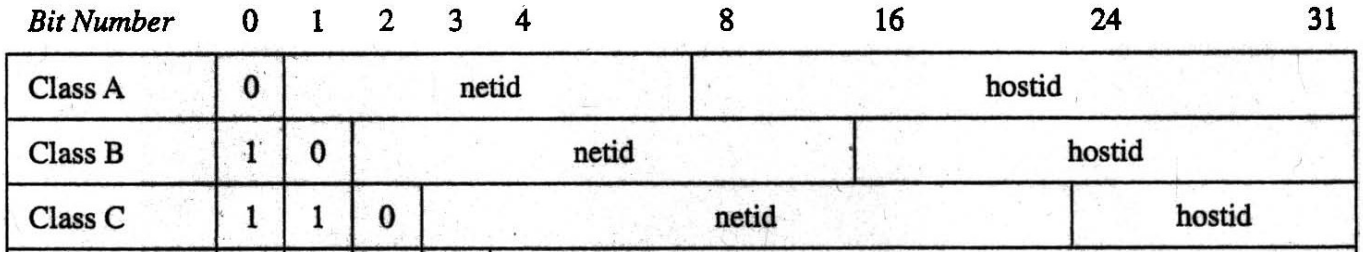
Nearly 4 hours!

IP Address Example (Internet Protocol v. 4)

Example 4: An address is a string of 32 bits – it begins with a network id (netid), followed by a host number (hostid), which identifies a computer as a member of a particular network.

- Main computer addresses are in one of 3 types:
  - *Class A (largest networks):* address contains a 0 followed by 7-bit "netid", and a 24-bit "hostid"
  - *Class B (medium networks):* address contains a 10 followed by a 14-bit netid and a 16-bit hostid.
  - *Class C (smallest networks):* address contains a 110 has 21-bit netid and an 8-bit hostid.

Netids all 1s are **not allowed**. Hostids that are all 0s or all 1s are **not allowed**.



How many valid IP addresses are there?

Wedding picture

Example 5: Consider a wedding picture of 6 people

- There are 10 people, including the bride and groom

(a) How many possibilities are there if the bride must be in the picture?

- Product rule: place the bride AND then place the rest of the party
- First place the bride:
  - She can be in one of 6 positions
- Next, place the other 5 people via the product rule:
  - There are 9 people to choose for the second person, 8 for the third, etc. ➔ Total = 9*8*7*6*5 = 15120

➔ Product rule yields 6 * 15120 = 90,720 possibilities

Wedding picture

Example 5: Consider a wedding picture of 6 people

- There are 10 people, including the bride and groom

b) How many possibilities are there if the bride and the groom must be in the picture?

- Product rule: place the bride/groom AND then place the rest of the party
- First place the bride and groom:
  - She can be in one of 6 positions
  - He can be in one of 5 remaining positions

  ➔ Total of 6*5 = 30 possibilities
- Next, place the other 4 people via the product rule:
  - There are 8 people to choose for the 3rd, 7 for the 4th, etc.

  ➔ Total = 8*7*6*5 = 1680

➔ Product rule yields 30 * 1680 = 50,400 possibilities

Wedding picture

Example 5: Consider a wedding picture of 6 people

- There are 10 people, including the bride and groom

c) How many possibilities are there if only one the bride or the groom must be in the picture?

Sum rule: place only the bride

- Product rule: place the bride AND then place the rest of the party
  - First place the bride
    - She can be in one of 6 positions
  - Next, place the other 5 people via the product rule (We can't choose the groom!!!)
    - There are 8 people to choose for the second person, 7 for the third, etc.

    ➔ Total = 8*7*6*5*4 = 6720

➔ Product rule yields 6 * 6720 = 40,320 possibilities

OR place only the groom (hmm… quickly, how many?)

- Same possibilities as for bride: 40,320

➔ Sum rule yields 40,320 + 40,320 = 80,640 possibilities

Alternative way to get the answer ???

Wedding picture

Example 5: Consider a wedding picture of 6 people

- There are 10 people, including the bride and groom

a) How many possibilities are there if the bride must be in the picture?

90,720 possibilities

b) How many possibilities are there if the bride and the groom must be in the picture?

50,400 possibilities

c) How many possibilities are there if only one the bride or the groom must be in the picture?

- Total ways to place the bride (with or without groom): 90,720
  - See (a).
- Total ways for both the bride and groom: 50,400
  - See (b).
- Total ways to place ONLY the bride:

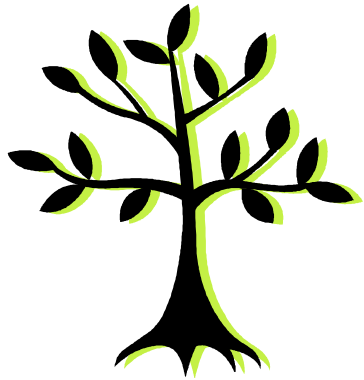
  90,720 – 50,400 = 40,320
- Same number for the groom
- Total = 40,320 + 40,320 = 80,640

1. Basic counting principles

1.1. The sum rule

1.2. The product rule

1.3. Tree diagram



1.3. Tree diagram

In many cases, we can solve the counting problems by construction of "Tree diagrams".

Example 1: In a tennis match, the player wins the game if he is the first player wins two sets. Question: What is the probability that player A will win the game in 3 sets?

Solution: Construct tree diagram:

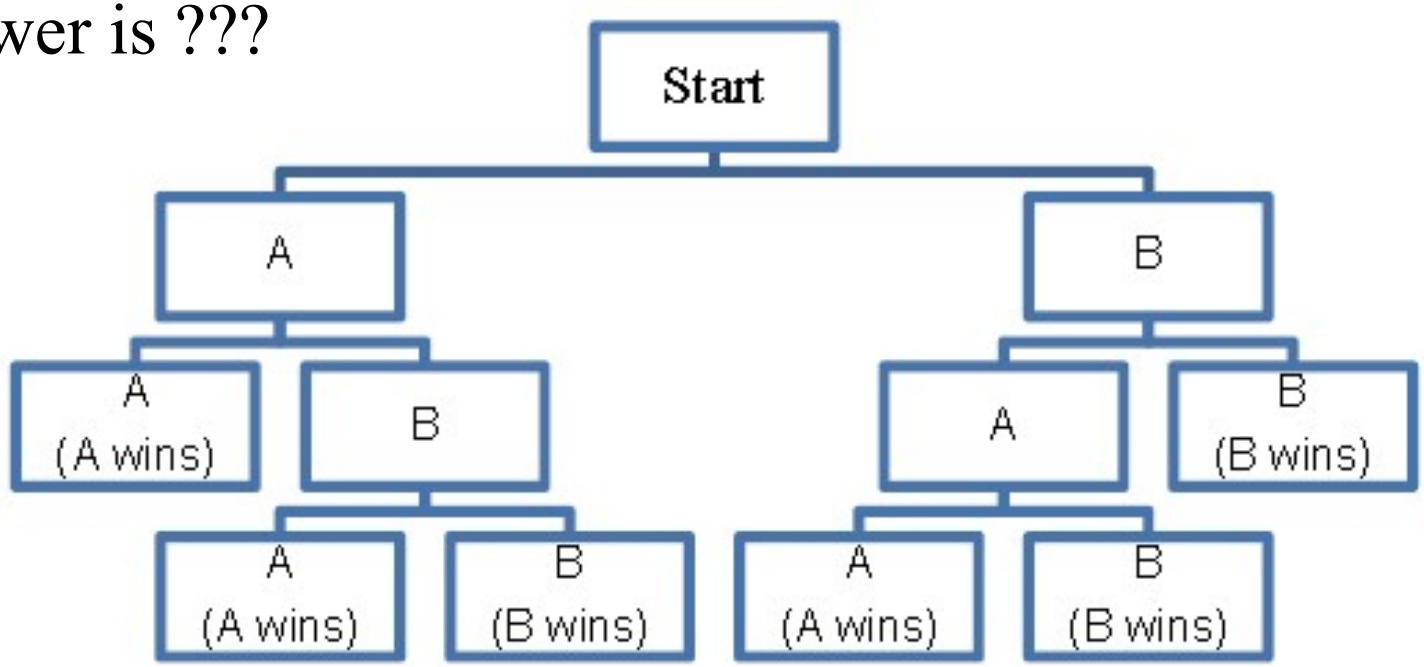
Thus, the answer is ???

2/6

1st set winner

2nd set winner

3rd set winner



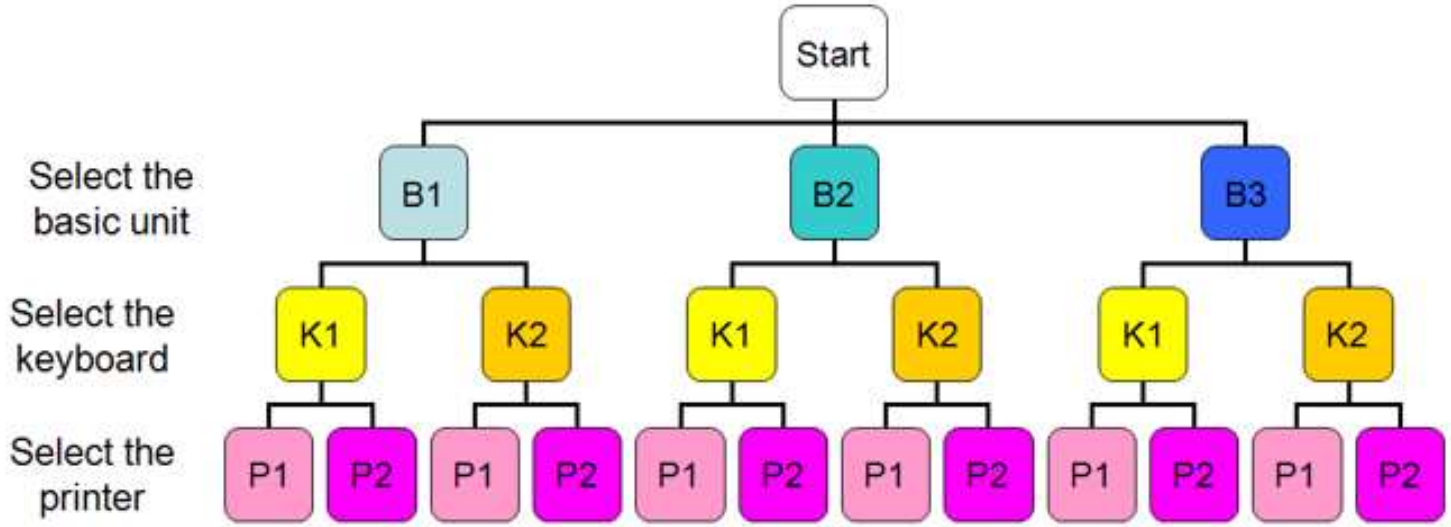
1.3. Tree diagram

Example 2:When buying a PC system, you have the choice of

- 3 models of the basic unit: B1, B2, B3;
- 2 models of keyboard: K1, K2;
- 2 models of printer: P1, P2.

Question: How many distinct systems can be purchased?

Solution: Construct tree diagram:



The number of distinct systems is:

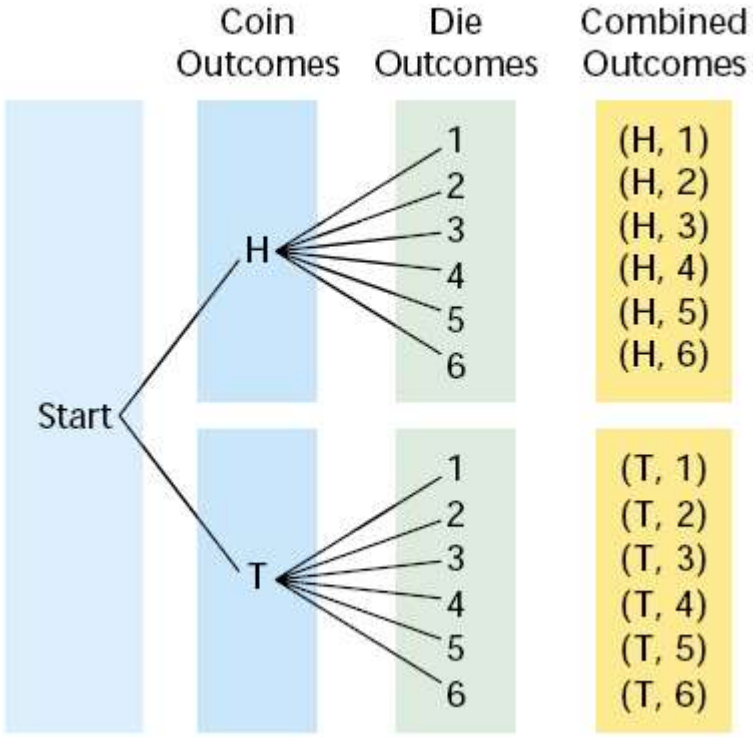
$3 \times 2 \times 2 = 12$

1.3. Tree diagram

Example 3: Suppose we flip a coin and then throw a single die. What are the possible combined outcomes?

Solution: Construct tree diagram:

Coin and die outcomes.



There are 12 possible combined outcomes: 2 ways in which the coin can come up followed by 6 ways in which the die can come up

Contents

2. Elementary combinatorial configuration

2.1. Permutation

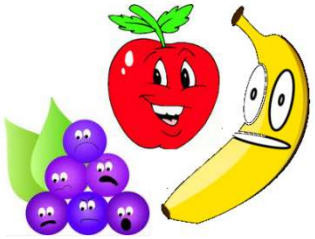
2.2. Combination

2. Elementary combinatorial configuration

2.1. Permutation

2.2. Combination

Fruit salad is a combination of apples, grapes and bananas. We don't care what order the fruits are in.



The permutation that will open the lock is 942, we do care about the order.

2. Elementary combinatorial configuration

2.1. Permutation

2.1.1. Permutation

Example 1: A PIN is defined as a sequence of any 4 digits from the set {0, 1, …, 9}.

Question 1. How many different PINs are possible if ***repetitions are allowed***?

Solution. Choosing a PIN is a 4-step operation:

- *Step 1:* Choose the 1st symbol (10 different ways).
- *Step 2:* Choose the 2nd symbol (10 different ways).
- *Step 3:* Choose the 3rd symbol (10 different ways).
- *Step 4:* Choose the 4th symbol (10 different ways).

➔ Based on the *product rule:*

$10 \times 10 \times 10 \times 10 = 10{,}000$ PINs are possible.

Question 2. How many different PINs are possible if ***repetitions are NOT allowed***?

Solution. Choosing a PIN is a 4-step operation:

- *Step 1:* Choose the 1st symbol (10 different ways).
- *Step 2:* Choose the 2nd symbol (9 different ways).
- *Step 3:* Choose the 3rd symbol (8 different ways).
- *Step 4:* Choose the 4th symbol (7 different ways).

➔ Based on the *product rule:*

$10 \times 9 \times 8 \times 7 = 5{,}040$ PINs are possible.

2.1.1. Permutation

- A *permutation* of a set A of objects is an ordered arrangement of the elements of A where each element appears only once

Example: If $A = \{a, b, c\}$, then the permutations of A are

1. *abc*
2. *acb*
3. *bac*
4. *bca*
5. *cab*
6. *cba*

- *The number of permutations of any set with n elements is*

$$P(n) = n! = n(n-1) \cdot \cdots \cdot 2 \cdot 1$$

(*Note that by definition* $0! = 1$)

Proof:

2.1.1. Permutation

- *The number of permutations of any set with n elements is*

$$P(n) = n! = n(n-1) \cdots 2 \cdot 1$$

Proof: (by product rule)

Forming a permutation is an *n*-step operation:

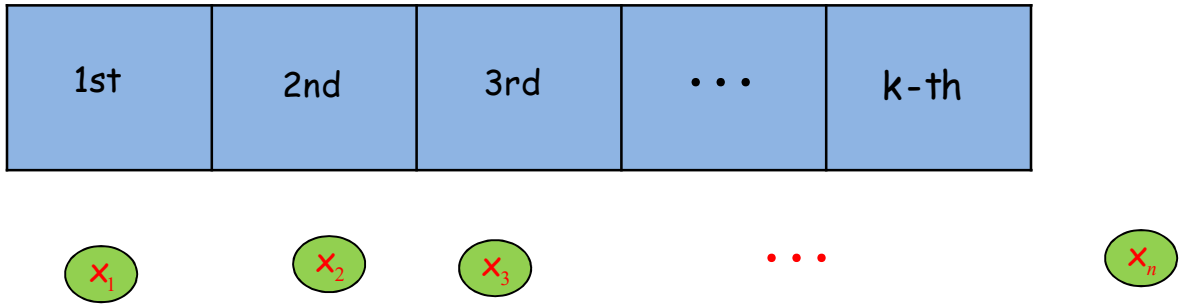
- *Step* 1: Choose the 1st element (*n* different ways).
- *Step* 2: Choose the 2nd element (*n*-1 different ways).
- …
- *Step n*: Choose the n^{th} element (1 way).

➔Based on the product rule, the number of permutations is

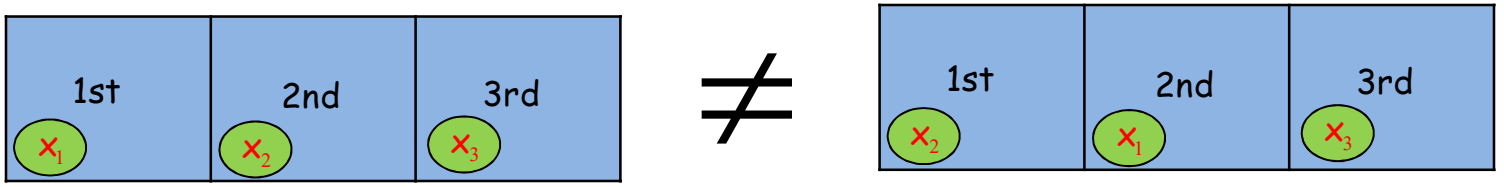
$$n \times (n\text{-}1) \times \ldots \times 2 \times 1 = n!$$

2.1.2. *k*-permutation

- A ***permutation*** of a set A of n objects $x_1, x_2, .., x_n$ is an ordered arrangement of the elements of A where each element appears only once.
- A ***k-permutation*** of a set A of n objects $x_1, x_2, .., x_n$ is an ordered arrangement of k distinct elements of A.



Example: 3-permutation of a set A of 4 objects x_1, x_2, x_3, x_4



2.1.2. *k*-permutation

- A ***permutation*** of a set A of n objects $x_1, x_2, .., x_n$ is an ordered arrangement of the elements of A where each element appears only once.
- A ***k-permutation*** of a set A of n objects $x_1, x_2, .., x_n$ is an ordered arrangement of k distinct elements of A.

➔ "permutations" means "*n*-permutations".

2.1.2. *k*-permutation

- A ***k-permutation*** of a set A of n objects $x_1, x_2, .., x_n$ is an ordered arrangement of k distinct elements of A.

➔ The number of k-permutations of a set A with $n=|A|$ elements is

$$P(n, k) = n(n-1)\ldots(n-k+1) = n! / (n-k)!$$

Example 1: 6 persons are competing for 4 prizes. How many different outcomes are possible?

$$P(6,4) = 6*5*4*3 = \frac{6!}{2!}$$

2.1.2. *k*-permutation

Example 2: Suppose you "have" time to listen to 10 songs on your daily jog around campus. There are 6 A tunes, 8 B tunes, and 3 C tunes to choose from.

Finally, suppose you still want 4 A, 4 B, and 2 C tunes, and the order of the groups doesn't matter, but you get dizzy and fall down if all the songs by any one group aren't played together.

How many playlists are there?

$P(n, k)$: k unique choices out of n objects, order matters

2.1.2. *k*-permutation

Example 3. How many ways are there to arrange 4 students to sit on the bench of 10 seats provided that sitting on the lap of each other is not allowed.

Solution. The students are numbered 1 to 4, seats 1 through 10. Each placement of the students can be represented by an ordered set (s_1, s_2, s_3, s_4), where $s_i \in \{1, 2, ..., 10\}$ are the seat of student *i*th.

As $s_i \neq s_j$, $i \neq j$, each placement is a 4-permutation of 10: P(10, 4) = 10*9*8*7=5040.

Note: We can also argue by directly using the Product rule:

We take turns putting the students into their seats.

- 1st student: there are 10 seats to be selected to sit
- 2nd student: put him one of the remaining 9 seats,…

According to the Product rule, there are 10*9*8*7 ways of placement

2. Elementary combinatorial configuration

2.1. Permutation

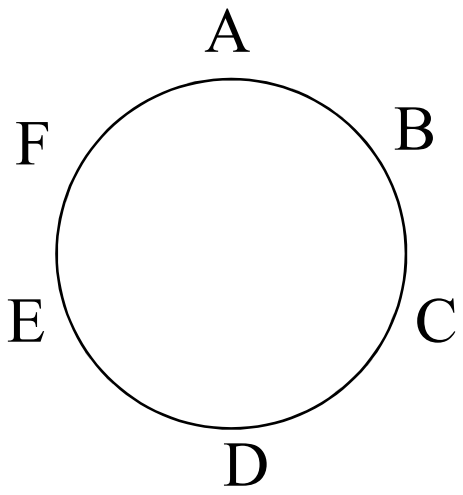
2.2. Combination

2.1.3. Circulation permutation

Circulation permutation $n!/n=(n-1)!$

Example 3: 6 people A, B, C, D, E, F are seated around a round table, how many different circular arrangements are possible, if arrangements are considered the same when one can be obtained from the other by rotations?

ABCDEF,
BCDEFA,
CDEFAB,
DEFABC,
EFABCD,
FABCDE



are the same arrangements circularly

➔ There are 6! ways to seat 6 people around the table

For each seating, there are 6 “rotations” of the seating

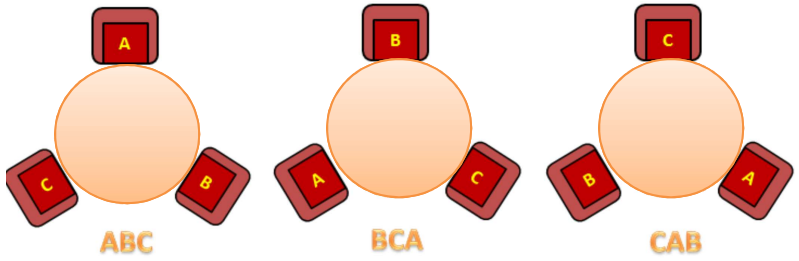
Thus, the final answer is 6!/6 = 5! = 120

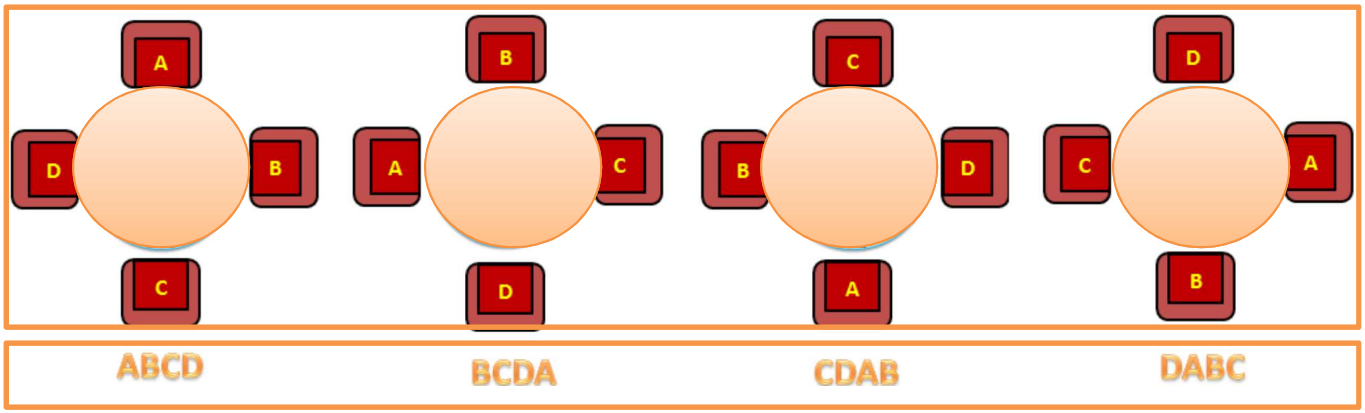
Hoán vị vòng tròn

- A *circulation permutation* of a set *A of n* objects is an ordered of all *n* objects of *A* arranged as a circle; there is not the beginning object and the ending object.

"The number of circulation permutations of an n-set equals n!/n = (n-1)!"

Proof ????





1 circulation permutation of set 4 objects A, B, C, D

4 permutations of set 4 objects A, B, C, D

2.1.3. Circulation permutation

- A *circulation permutation* of a set *A of n* objects is an ordered of all *n* objects of *A* arranged as a circle; there is not the beginning object and the ending object.

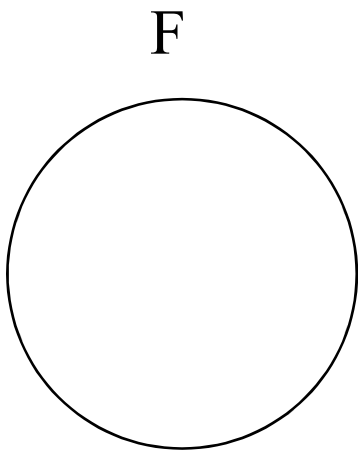
"The number of circulation permutations of an n-set equals n!/n = (n-1)!"

- A *circular k-permutation* of a set *A of n* objects is an ordered *k* objects of *A* arranged as a circle; there is not the beginning object and the ending object. Prove the following result:

"The number of circular k-permutations of an n-set equals P(n,k)/k".

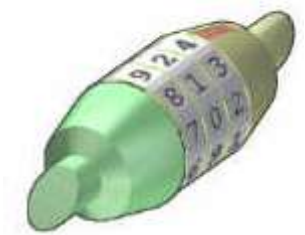
2.1.3. Circulation permutation

Example 4: How many ways to arrange 3 couples in a round table with alternating sex



2. Elementary combinatorial configuration

2.1. Permutation



2.2. Combination

2.1.4. Permutations of multisets

- A *multiset* M is a collection whose members need not be distinct.

Example: The collection

$$M = (a, a, a, b, b, c, d, d, d, 1, 2, 2, 2, 3, 3, 3, 3)$$

is a multiset; and sometimes it is convenient to write

$$M = (3a, 2b, c, 3d, 1, 3\bullet 2, 4\bullet 3).$$

- A multiset M over a set S can be viewed as a function $v : S \to \mathbb{N}$ from S to the set $\mathbb{N}$ of nonnegative integers; each element $x \in S$ is repeated $t(x)$ times in M; we write $M = (S; t)$.
- Let M be a multiset and $|M| = n$.
  - A *k-permutation* of M is an ordered arrangement of k objects selected from n objects of M.
  - A n-permutation of M is called a *permutation* of M.

2.1.4. Permutation of multisets

- Let M be a multiset and $|M| = n$.
  - A *k-permutation* of M is an ordered arrangement of k objects selected from n objects of M.
  - A n-permutation of M is called a *permutation* of M.

Proposition 1. *Let M be a multiset of r different types where each type has infinitely elements. Then the number of k-permutations of M equals* r^k

Example 1. What is the number of binary numerals with at most 4 digits?

Solution: The question is to find the number of 4-permutations of the multiset $(\infty 0, \infty 1)$. Thus the answer is $2^4 = 16$.

Example 2. What is the number of ternary numerals with at most 4 digits?

- *Solution:* The question is to find the number of 4-permutations of the multiset $(\infty 0, \infty 1, \infty 2)$. Thus the answer is $3^4 = 81$.

2.1.4. Permutation of multisets

- Let M be a multiset and $|M| = n$.
  - A *k-permutation* of M is an ordered arrangement of k objects selected from n objects of M.
  - A n-permutation of M is called a *permutation* of M.

Proposition 1. *Let M be a multiset of r different types where each type has infinitely elements. Then the number of k-permutations of M equals* r^k

1st position **2nd position** **3rd position ….** **kth position**

r choice r choice r choice

The total = r^k

2.1.4. Permutation of multisets

Proposition 2. *Let M be a multiset of r different types with repetition numbers n_1, n_2,…, n_r respectively. Let $n = n_1 + n_2 + \ldots + n_r$. Then the number of permutations of M equals*

$$\frac{\boldsymbol{n!}}{\boldsymbol{n_1!\, n_2! \ldots n_r!}}$$

Proof.

- List the elements of M as $\underbrace{a,\ldots,a}_{n_1}, \underbrace{b,\ldots,b}_{n_2}, \ \ldots \ , \underbrace{d,\ldots,d}_{n_r}$

- Let S be the set consisting of the elements

$$a_1, a_2, \ldots, a_{n_1}, b_1, b_2, \ldots, b_{n_2}, \ldots, d_1, d_2, \ldots, d_{n_r},$$

Let X be the set of all permutations of S, and let Y be the set of all permutations of M.

There is a map $f: X \to Y$, sending each permutation of S to a permutation of M by removing the subscripts of the elements. Note that for each permutation π of M there are $n_1!, n_2!, \ldots, n_r!$ ways to put the subscripts of the first, the second, ..., and the rth type elements back, respectively.

Thus there are $n_1! n_2! \ldots n_r!$ elements of X sent to Y by f. Therefore

$$|Y| = \frac{|X|}{n_1!\, n_2! \ldots n_r!} = \frac{n!}{n_1!\, n_2! \ldots n_r!}$$

2.1.4. Permutation of multisets

Proposition 2. *Let M be a multiset of r different types with repetition numbers $n_1, n_2, \ldots, n_r$ respectively. Let $n = n_1 + n_2 + \ldots + n_r$. Then the number of permutations of M equals*

$$\frac{\boldsymbol{n!}}{\boldsymbol{n_1!\, n_2! \ldots n_r!}}$$

Example 3: *The number of* 0-1 *words of length n with exactly m ones and (n – m) zeros equals to*

$$\frac{n!}{m!(n-m)!} = C_n^m = \binom{n}{m}.$$

2.1.4. Permutation of multisets

Example 4: Find the number of 8-permutations of the multiset $M = (a, a, a, b, b, c, c, c, c) = (3a, 2b, 4c)$.

Solution:

How to select 8 elements from (3+2+4) = 9 elements ???

(2a,2b,4c)
(3a, b, 4c)
(3a, 2b, 3c)

- The number of 8-permutations of $(2a, 2b, 4c)$: $8!/(2!2!4!)$
- The number of 8-permutations of $(3a, b, 4c)$: $8!/(3!1!4!)$
- The number of 8-permutations of $(3a, 2b, 3c)$: $8!/(3!2!3!)$

➔ Thus the answer is

$8!/(2!2!4!) + 8!/(3!1!4!) + 8!/(3!2!3!) = 420 + 280 + 560 = 1260$

2. Elementary combinatorial configuration

2.1. Permutation

2.2. Combination

2.2. Combination

2.1.1. Definitions

2.2.2. Binomial coefficients

2.2.3. Combinations of Multisets

2.2.4. Multinomial coefficients

2.2.1. Definitions

- A *k*-combination of a set of *n* elements is a subset of size *k of n* elements.

(Note: A permutation is a sequence while a combination is a set)

Example:

- The 2-permutation (sequence) of SOHN is:

SO, SH, SN, OH, ON, OS, HN, HS, HO, NS, NO, NH

- The 2-combination (set) of SOHN is:

{S,O},{S,H},{S,N},{O,H},{O,N},{H,N}

2.2.1. Definitions

- The number of all k-combinations of a set of n elements denoted

$$\binom{n}{k},\ C_n^k \ \text{ or } \ C(n,k)$$

and read "*n choose k*".

$$\binom{n}{k} = \frac{n!}{k!\cdot(n-k)!}$$

Proof ???

This number is also called a *binomial coefficient* because such numbers occur as coefficients in the expansions of powers of binomial expressions such as $(a+b)^n$

k-combination

Example: $X = \{a, b, c\}$; $n = 3$, $r = 2$

k-permutation

select 2 from 3 elements of X:

1. (a, b)
2. (a, c)
3. (b, a)
4. (b, c)
5. (c, a)
6. (c, b)

Ordered sequence

$2!$

k-combination

select 2 from 3 elements of X:

1. $\{a,b\}$
2. $\{a, c\}$
3. $\{b, c\}$

Order is not the matter

$$P_n^k = n(n-1)...(n-k+1) = \frac{n!}{(n-k)!}$$

$$C_n^k = \frac{n!}{k!(n-k)!}$$

k-combination

Example: $X = \{a, b, c\}$; $n = 3$, $r = 2$

k-permutation

Select 2 from 3 elements of X:

1. (a, b)
2. (a, c)
3. (b, a)
4. (b, c)
5. (c, a)
6. (c, b)

Ordered sequence

$$P_n^k = n(n-1)...(n-k+1) = \frac{n!}{(n-k)!}$$

k-combination

Select 2 from 3 elements of X:

1. $\{a,b\}$
2. $\{a, c\}$
3. $\{b, c\}$

Order is not the matter

Consider the set A of all *k*-permutation of *n* elements. Divide them into subset so that the two permutations belonging to the same subset differ only in order. Obviously, these subsets are a partition on the set A, and each such subset corresponds to a *k*-combination of *n* elements. There are the same number of *k*-permutation in each subset, and is equal to *k*! (number of permutations). The number of subsets is the number of *k*-combination of *n*. According to the sum rule, product of *k*! and the number of subsets is equal to the number of *k*-permutation of n, that is, equal to *n* (*n*-1) ... (*n* - *k* + 1). Therefore, the number of *k*-combination of *n* is:

$$\frac{n(n-1)(n-2)...(n-k+1)}{k!} \quad \text{or} \quad \frac{n!}{k!(n-k)!}$$

k-combination of a set

Example: Consider the following nested loop

```
for a:=1 to n
    for b:=1 to a-1
        for c:=1 to b-1
                [Statements]
        endfor
    endfor
endfor
```

Question: How many times the statements in the innermost loop will be executed?

Answer: Each iteration corresponds to a triple of integers (a, b, c) where $a > b > c$. The set of all this kind of triples corresponds to all **3**-combinations of $\{1, \ldots, n\}$. Thus, the total number of iterations is $\boldsymbol{C(n,3)}$.

Difference between permutation and combination

- A combination is a set while a permutation is a sequence.

➔Let's consider the difference between unordered and ordered selections:

1. Two *ordered selections* are the same if
   - the elements chosen are the same;
   - the elements chosen are in the same order.
- Ordered selections correspond to *k-permutations*.
- The number of all *k*-permutations is $P(n, k) = n! / (n-k)!$

2. Two *unordered selections* are the same if
   - the elements chosen are the same (regardless of the order in which the elements are chosen)
- Unordered selections correspond to *k-combinations*
- The number of all *k*-combinations is $C(n, k) = n! / k! (n-k)!$

Difference between permutation and combination

When dealing with any counting problem, we should **ask ourselves about the importance of order in the problem**:

- When order is relevant, we think in terms of permutations and arrangements and the rule of product.
- When order is not relevant, combinations could play a key role in solving the problem.

Difference between permutation and combination

Example: A club has 25 members.

a. How many ways are there to choose 4 members of the club to serve on an executive committee?

b. How many ways are there to choose a president, vice president, secretary, and treasurer of the club?

2.2. Combination

2.1.1. Definitions

2.2.2. Binomial coefficients

2.2.3. Combinations of Multisets

2.2.4. Multinomial coefficients

Binomial Coefficients

$$(a+b)^4 = (a+b)(a+b)(a+b)(a+b)$$
$$= \binom{4}{0}a^4 + \binom{4}{1}a^3b + \binom{4}{2}a^2b^2 + \binom{4}{3}ab^3 + \binom{4}{4}b^4$$

Binomial Theorem: Let x and y be variables, and let n be any nonnegative integer. Then

$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j} y^j$$

Binomial Coefficients

$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j} y^j$$

What is the coefficient of a^8b^9 in the expansion of $(3a+2b)^{17}$?

What is n? 17

What is j? 9

What is x? 3a

What is y? 2b

$$\binom{17}{9}(3a)^8(2b)^9 = \binom{17}{9} 3^8 2^9 a^8 b^9$$

Binomial Coefficients

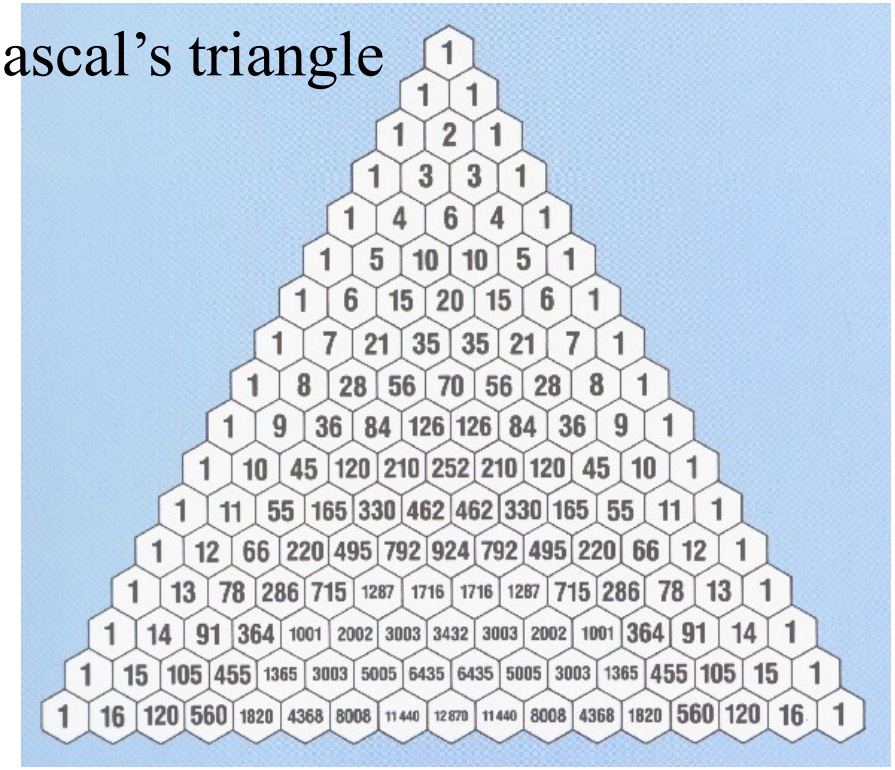
$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j} y^j$$

$(a + b)^2 = a^2 + 2ab + b^2$

$(a + b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$

$(a + b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$

Pascal's triangle



What is coefficient of a^9b^3 in $(a + b)^{12}$?

A. 36
B. 220
C. 15
D. 6
E. No clue

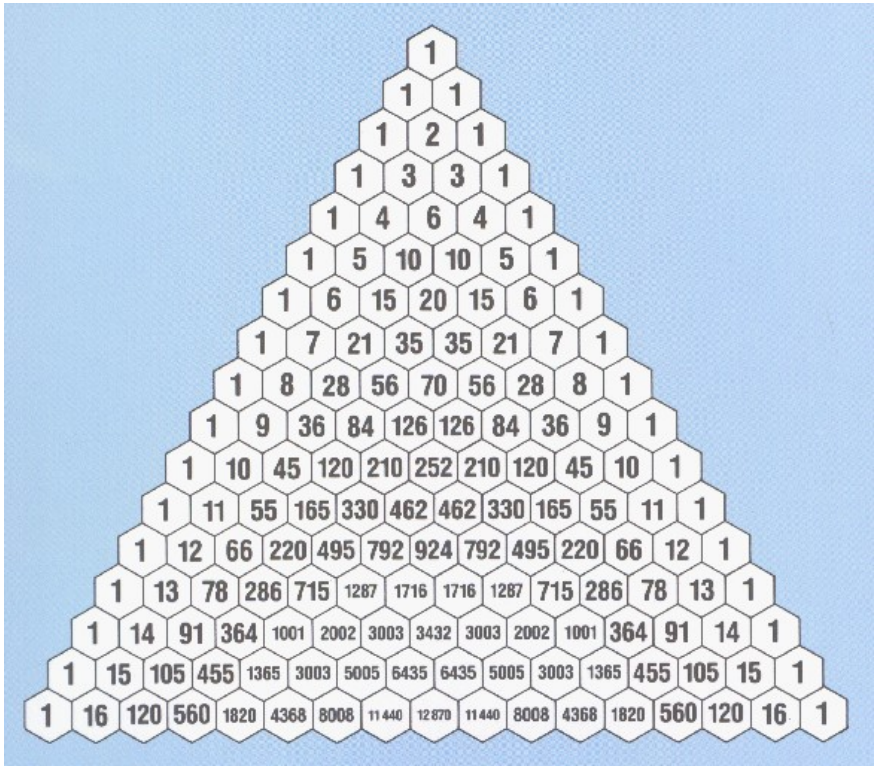
NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Binomial Coefficients

Sum each row of Pascal's Triangle: $\sum_{j=0}^{n} \binom{n}{j} = 2^n$

Powers of 2

Proof ???



Suppose you have a set of size n. How many subsets does it have?

2^n

How many subsets of size 0 does it have?

$\binom{n}{0}$

How many subsets of size 1 does it have?

$\binom{n}{1}$

How many subsets of size 2 does it have?

$\binom{n}{2}$

Add them up we have the result.

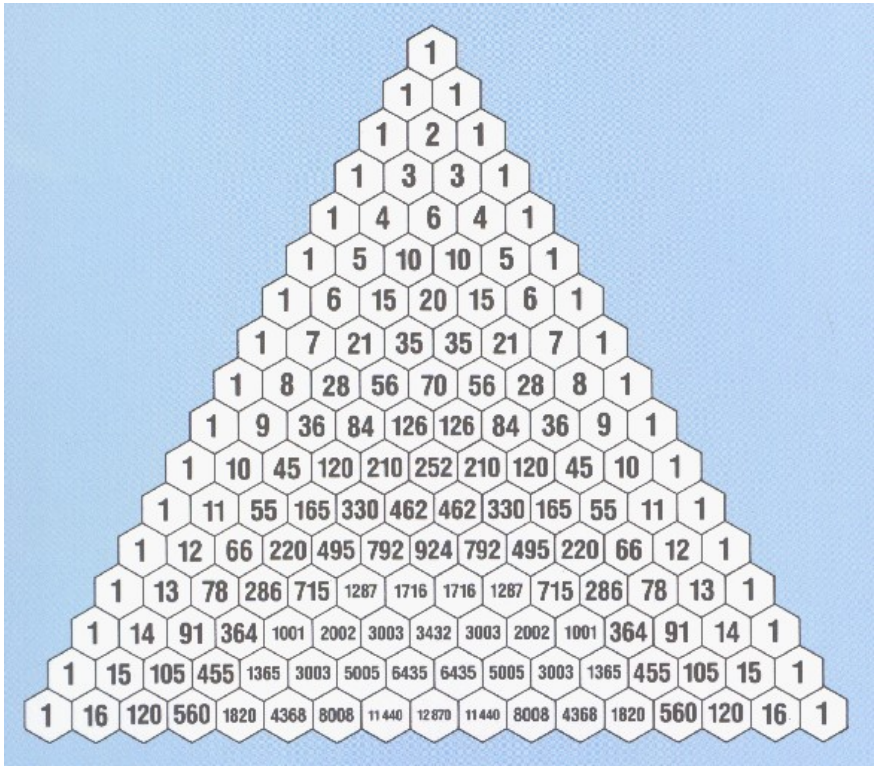
Binomial Coefficients

Sum each row of Pascal's Triangle: $\sum_{j=0}^{n} \binom{n}{j} = 2^n$

Proof ???

Alternative (clever) proof? Look at binomial theorem…

$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j} y^j$$



x and y are variables; can pick any numbers… hmm…

Pick x=1 and y=1 !

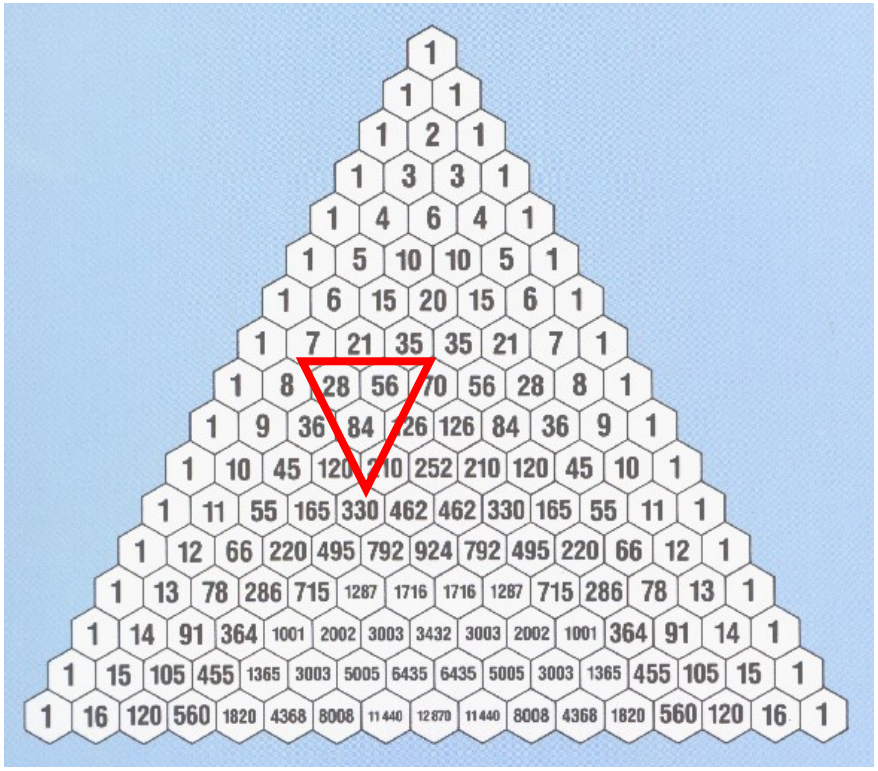
$$\sum_{j=0}^{n} \binom{n}{j} 1^{n-j} 1^j = (1+1)^n$$

$$\sum_{j=0}^{n} \binom{n}{j} = 2^n$$

Pascal Identity

A relationship between the entries in Pascal's triangle:

$$\binom{n}{j} = \binom{n-1}{j-1} + \binom{n-1}{j}$$



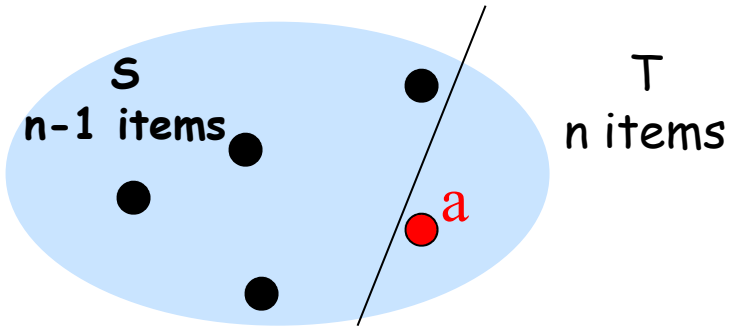
Suppose T is a set, $|T|=n$. Let a be an element in T, and let $S = T - \{a\}$. So, $|S| = n-1$.

Let's count the $\binom{n}{j}$ subsets of size j.

Note that some of these contain a, and some don't.

How many contain a? $\binom{n-1}{j-1}$

How many don't? $\binom{n-1}{j}$



2.2. Combination

2.1.1. Definitions

2.2.2. Binomial coefficients

2.2.3. Combinations of Multisets

2.2.4. Multinomial coefficients

2.2.3. Combinations of multisets

- Let M be a multiset $\{\infty a_1, \infty a_2, \ldots, \infty a_n\}$ (M has n distinct objects):
  - A *k-combination* of M is an unordered collection of k objects selected from n types of objects of M.
  - A *k-combination* of M is also called an k-combination with repetition allowed.
  - The number of k-combination of M is $C(n+k-1, k) = C(n+k-1, n-1)$

(the number of selections, ***with repetitions***, of k objects from n distinct objects)

Need to divide k candies for n kids $B_1, B_2, \ldots, B_n$. How many different ways to divide?

Let t_j be the number of candies for kid B_j, $j=1,\ldots,n$. At this point, the above problem leads to the problem:

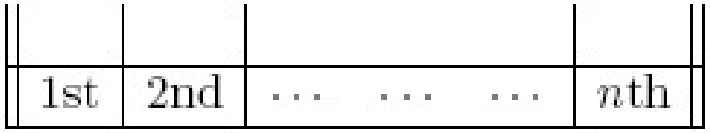
Let k and n be non-negative integers. How many non-negative integers in the following equation have?

$$t_1 + t_2 + t_3 + \cdots + t_n = k$$

$$t_1, t_2, \cdots, t_n \in Z^+$$

The number of r-combination of M is $C(n+r-1, r)$

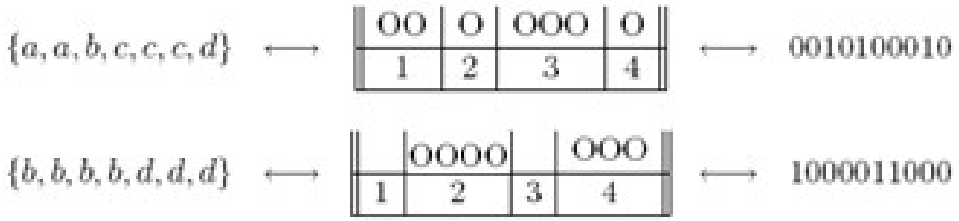
Proof: When k objects are selected from the multiset M, we put them into the following n boxes



so that the ith type objects are contained in the ith box, $1 \le i \le n$.

Since the objects of the same type are identical, we may use the symbol "O" to denote any object in the boxes, and the objects in different boxes are separated by a stick "|".

For example, for $n = 4$ and $k = 7$:



Convert the symbol "O" to zero 0 and the stick "|" to one 1, any such placement is converted into a 0-1 sequence of length $k+n-1$ with exactly k zeros and $n-1$ ones

➔ Now the problem becomes counting the number of 0-1 words of length $k+(n-1)$ with exactly k zeros and $n-1$ ones

$$= C(n+k-1, k) = C(n+k-1, n-1)$$

2.3.3. Combinations of multisets

Example 1: Example 1: How many ways to divide 10 candies for 4 kids (there may be kids without any candies)

How many solutions are there to the equation $x_1 + x_2 + x_3 + x_4 = 10$

When the variables are nonnegative integers ($x_i \geq 0$)**?**

Answer 1: 11 locations for bars. Pick 3 locations allowing repetitions

➔ *k*-combination with repetition allowed (*n*=11, *k* = 3). ➡ **C(11+3-1,3)**

$$1+3+6+0=10$$

Answer 2: (bài toán chia kẹo)

Select with repetition from x_1, x_2, x_3, x_4 10 times.

For example if x_1 is selected twice then x_1= 2 in the final solution.

Therefore: *k*-combination with repetition allowed ($n = 4$, $k = 10$):

C(4+10-1,10)

2.3.3. Combinations of multisets

Example 2: How many nonnegative integer solutions are there to the inequality

$$x_1 + x_2 + \cdots + x_6 < 10\,?$$

It is equivalent to

$$x_1 + x_2 + \cdots + x_6 + x_7 = 10,\ 0 \le x_i,\ 1 \le i \le 6,\ 0 < x_7$$

which can be transformed to

$$y_1 + y_2 + \cdots + y_6 + y_7 = 9,\quad y_i \ge 0 \quad \text{integer}$$

where $y_i = x_i$ for $1 \le i \le 6$

and $y_7 = x_7 - 1$

➔ Ans1: 10 locations for bars, pick 6 locations allowing repetitions [$k = 6$, $n = 10$]

➔ k-combination with repetition C(6+10-1,6)=C(15,6) =5005

➔ Ans2: 7 types of objects to select (y_1,..,y_7) and 9 times of selections [$k = 9$, $n = 7$]

➔ k-combination with repetition C(7+9-1,9)= C(15, 9) = 5005

The number of k-combination of M is C(n+ k-1, k)

Equivalence of the following: The number $C(n+k-1, k)$ equals

- the number of selections, with repetition, of size k from a collection of n types objects
- the number of nonnegative integer solutions of the equation

$$x_1 + x_2 + \ldots + x_n = k$$

- the number of ways to place k identical objects into n distinct containers.
- the number of non decreasing sequences of length k whose terms are taken from the set $\{1, 2, \ldots, n\}$.

2.2. Combination

2.1.1. Definitions

2.2.2. Binomial coefficients

2.2.3. Combinations of Multisets

2.2.4. Multinomial coefficients

2.2.4. Multinomial coefficients

The number of ordered arrangements of n objects, in which there are k_1 objects of type 1, k_2 objects of type 2, ..., and k_m objects of type m and where $k_1+k_2+ \ldots +k_m = n$, is

$$\binom{n}{k_1, k_2, \ldots, k_m} = \frac{n!}{k_1!k_2!\ldots k_m!}$$

2.2.4. Multinomial coefficients

- ***The Multinomial Theorem:***

$$(x_1 + x_2 + \ldots x_m)^n = \sum \binom{n}{k_1, k_2, \ldots, k_m} x_1^{k_1} x_2^{k_2} \ldots x_m^{k_m}$$

where the summation is over all sequences of non-negative integers $(k_1, k_2, \ldots, k_m)$ such that $k_1 + k_2 + \ldots + k_m = n$.

Summary: select or order *k* objects from *n* distinct objects

	Order matters	Order does not matter
Repetition not allowed	1 $P(n,k) = \frac{n!}{(n-k)!}$ (*k*-permutation of *n*-set)	2 $C(n,k) = \frac{n!}{k!\,(n-k)!}$ (*k*-combination of *n*-set)
Repetition allowed	3 n^k (*k*-permutation with repetition allowed)	4 $C(n+k-1,k) = \frac{(n+k-1)!}{k!\,(n-1)!}$ (*k*-combination with repetition allowed)

The set A = {S,O,H,N}

The 2-permutation (sequence) of set A is: P(4,2) = 12
SO, SH, SN, OH, ON, OS, HN, HS, HO, NS, NO, NH
1

The 2-combination (set) of set A is: C(4, 2) = 6
{S,O},{S,H},{S,N},{O,H},{O,N},{H,N}
2

The multi set B = {∞S, ∞O, ∞H, ∞N}

The 2-permutation with repetition allowed of set B is: $4^2 = 16$
SS, OO, HH, NN,
SO, SH, SN, OH, ON, OS, HN, HS, HO, NS, NO, NH
3

The 2-combination with repetition allowed of set B is: C(4+2-1,2)=10

{S, S}, {O, O}, {H, H}, {N, N},
{S,O},{S,H},{S,N},{O,H},{O,N},{H,N}
4

Contents

3. Inclusion-exclusion principle

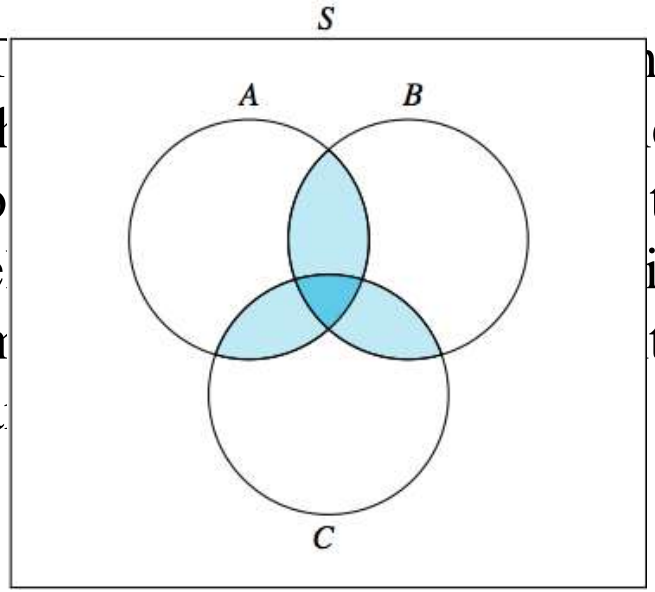
3.1. Inclusion-exclusion principle

3.2. Derangement

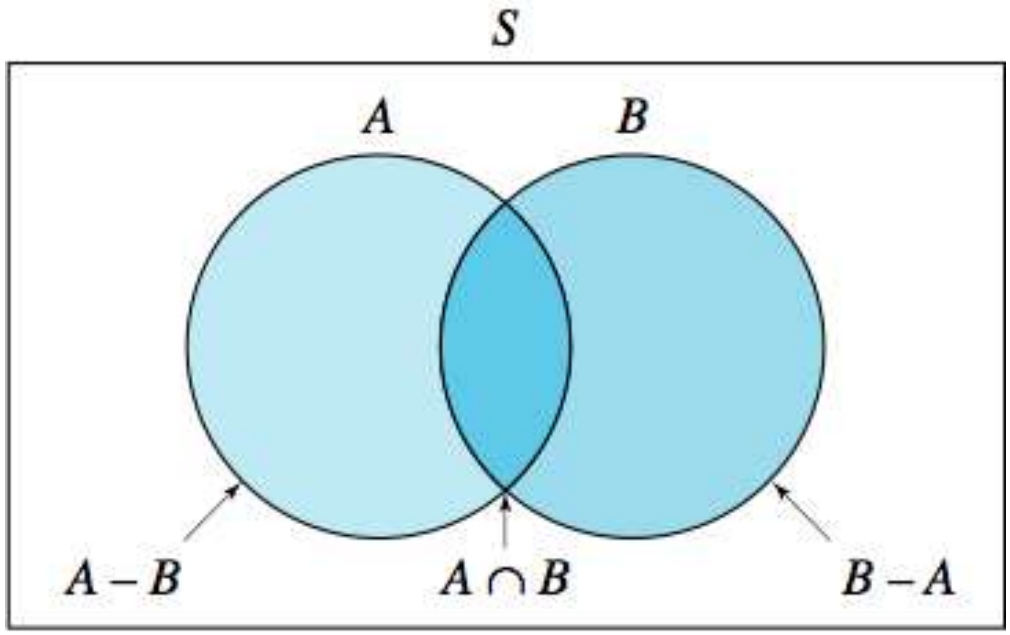
3.1. Inclusion-exclusion principle

The inclusion–exclusion principle is an equation relating the sizes of two sets and and their union. It states that if A and B are two (finite) sets, then

$$|A \cup B| = |A| + |B| - |A \cap B|$$



T ıt is that
th e union
o the
e ively,
n ts that
a



Similarly, for three sets A, B and C:

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

This can be seen by counting how many times each region in the figure is included in the right hand side.

3.1. Inclusion-exclusion principle

More generally, for finite sets $A_1, A_2, \ldots, A_m$

$$|A_1 \cup A_2 \cup \cdots \cup A_m| = N_1 - N_2 + \cdots + (-1)^{m+1} N_m$$

where:

$$N_k = \sum_{1 \le i_1 < i_2 < \ldots < i_k \le m} | A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_k} |, \quad k = 1, 2, \ldots, m$$

Note: N_k is the sum of the cardinalities of all intersections of k from m given sets. For example:

$$|A_1 \cup A_2 \cup A_3 \cup A_4| = ?$$

$$N_1 = |A_1| + \ldots + |A_m|$$

$$N_m = |A_1 \cap A_2 \cap \ldots \cap A_m|$$

$$| A_1 \cup A_2 \cup \ldots \cup A_m | = \sum_{i=1}^{m} | A_i | - \sum_{pairs\ (ij)} | A_i \cap A_j | + \sum_{triples\ (ijk)} | A_i \cap A_j \cap A_k | - \ldots$$

$$+ (-1)^{m+1} | A_i \cap A_j \cap A_k \cap \ldots \cap A_m |$$

3.1. Inclusion-exclusion principle

Example 1: How many integers from 1 to 1000 are either multiples of 3 or multiples of 5?

- A = set of all integers from 1 to 1000 that are multiples of 3.
- B = set of all integers from 1 to 1000 that are multiples of 5.
- $A \cup B$ = ?

The set of all integers from 1 to 1000 that are multiples of either 3 or 5.

- $A \cap B$ = ?

The set of all integers that are both multiples of 3 and 5, which also is the set of integers that are multiples of 15.

- To use the inclusion-exclusion principle to obtain $|A \cup B|$, we need
  - $|A|$
  - $|B|$
  - $|A \cap B|$

3.1. Inclusion-exclusion principle

- A = set of all integers from 1 to 1000 that are multiples of 3 ➔ |A| = ?

From 1 to 1000, every third integer is a multiple of 3, each of this multiple can be represented as $3p$, for any integer p from 1 through [1000/3]=333, Hence |A| = 333.

- B = set of all integers from 1 to 1000 that are multiples of 5➔ |B| = ?

Similarly for multiples of 5, each multiple of 5 is of the form $5q$ for some integer q from 1 through [1000/5]=200. Hence, we have |B| = 200.

- $A \cap B$: set of all integers from 1 to 1000 that are multiples of 15➔ $|A \cap B|$=?

To determine the number of multiples of 15 from 1 through 1000, each multiple of 15 is of the form $15r$ for some integer r from 1 through [1000/15]=66. ➔ $|A \cap B| = 66$.

- From the principle, we have the number of integers either multiples of 3 or multiples of 5 from 1 to 1000 given by

$$|A \cup B| = |A| + |B| - |A \cap B|$$
$$= 333 + 200 - 66 = 467$$

3.1. Inclusion-exclusion principle

Example 2: There are 350 applicants to a job, and

- (i) 220 with major in CS
- (ii) 147 with major in Business
- (iii) 51 with major in both CS and Business

How many applicants have major neither in CS nor Business ?

Answer:

- A: set of applicants with major in CS
- B: set of applicants with major in Bussiness

➔ A∩B:

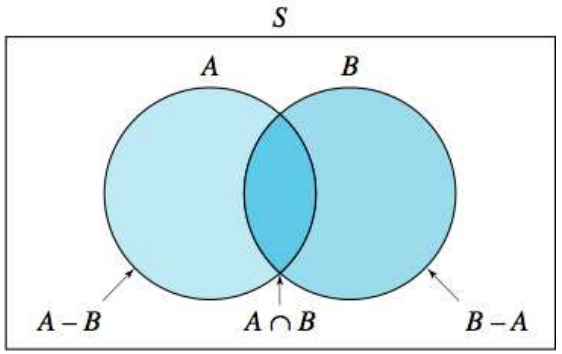
Set of applicants with major both in CS and bussiness

➔ A∪B:

Set of applicants with major in CS or business or both

$|A \cup B| = |A| + |B| - |A \cap B| = 220 + 147 - 51 = 316$

➔ The desired answer : 350 – 316 = 34



3.1. Inclusion-exclusion principle

Example 3: In a class of students undergoing a computer course the following were observed:

- Out of a total of 50 students: 30 know Java, 18 know C#, 26 know C, 9 know both Java and C#, 16 know both Java and C, 8 know both C# and C, 47 know at least one of the three languages.

From this we have to determine:

a. How many students know none of these languages?

b. How many students know all three languages?

a. $50 - 47 = 3$

b.

A = All the students who know Java in class.

B = All the students who know C# in the class.

C = All the students who know C in class.

➔ Students know all three languages, so we need to find $|A \cap B \cap C|$

We have to derive the inclusion-exclusion formula for three sets

$$|A \cup B \cup C| = (|A| + |B| + |C|) - (|B \cap C| + |A \cap B| + |A \cap C|) + |A \cap B \cap C|$$

3.1. Inclusion-exclusion principle

- Out of a total of 50 students: 30 know Java, 18 know C#, 26 know C, 9 know both Java and C#, 16 know both Java and C, 8 know both C# and C, 47 know at least one of the three languages.

A = All the students who know Java in class.

B = All the students who know C# in the class.

C = All the students who know C in class.

➔ Students know all three languages, so we need to find $|A \cap B \cap C|$.

$$|A \cup B \cup C| = (|A| + |B| + |C|) - (|B \cap C| + |A \cap B| + |A \cap C|) + |A \cap B \cap C|$$

$$47 = (30+18+26) - (8+9+16) + |A \cap B \cap C|$$

➔ $|A \cap B \cap C| = 6$

3.1. Inclusion-exclusion principle

Example 4: How many binary strings of length 10 such that either start with 00 or end with 11?

Answer:

A: set of binary strings of length 10 that start with 00

B: set of binary strings of length 10 that end with 11

→ Need to calculate $|A \cup B|$

00xxxxxxxx

8 positions
Each position: 0/1

The inclusion-exclusion formula for 2 sets:

$$|A \cup B| = |A| + |B| - |A \cap B|$$

3. Inclusion-exclusion principle

3.1. Inclusion-exclusion principle

3.2. Derangement

3.2. Derangement

A ***derangement*** of $\{1, 2, \ldots, n\}$ is a permutation on the set such that none of elements i is placed at position ith in the permutation.

(In other words, derangement is a permutation that has no fixed points)

Example: $n = 5$

- Permutation (2,3,5,**4**,1) is not a derangement
- Permutation (2,3,5,1,4) is a derangement

Denote D_n the number of derangements of $\{1, 2, \ldots, n\}$ ➔ D_n = ??

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

3.2. Derangement

- Denote D_n the number of derangements of $\{1, 2, \ldots, n\}$ ➔ D_n = ??
- Let S = set of all permutations of $\{1, 2, \ldots, n\}$ ➔ $|S| = n!$
- Let A_i = subset of permutations of $\{1, 2, \ldots, n\}$ such that the ith element = i in the permutation. $|A_1 \cup A_2 \cup \cdots \cup A_n|$???

➔ $|A_1 \cup A_2 \cup \cdots \cup A_n|$ counts the number of permutations in which at least one object i of the n objects appears in the ith position (its original position).

➔ D_n: number of permutations such that none of the n objects appears in their original positions. Therefore:

$$D_n = n! - |A_1 \cup A_2 \cup \cdots \cup A_n|$$

Remember the inclusion-exclusion principle:

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{i=1}^{n} |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k| - \sum_{i<j<k<\ell} |A_i \cap A_j \cap A_k \cap A_\ell| + \ldots + (-1)^{n+1} |A_1 \cap A_2 \cap \cdots \cap A_n|$$

3.2. Derangement

Let A_i = subset of permutations of $\{1, 2, \ldots, n\}$ such that the ith element = i in the permutation.

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{i=1}^{n} |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k|$$

$$- \sum_{i<j<k<\ell} |A_i \cap A_j \cap A_k \cap A_\ell| + \ldots + (-1)^{n+1} |A_1 \cap A_2 \cap \cdots \cap A_n|$$

- $|A_i| = (n\text{-}1)!$ since if exactly one of the n objects is placed at its original position, that leaves the other $(n\text{-}1)$ objects to be freely permuted in $(n\text{-}1)!$ possible ways. Hence, $\sum_{i=1}^{n} |A_i| = n \cdot (n-1)! = n!\,,$

  since there are n terms in the sum.

- $|A_i \cap A_j| = (n\text{-}2)!$ since if exactly two of the n objects are placed at their original positions, that leaves the other $(n\text{-}2)$ objects to be freely permuted in $(n\text{-}2)!$ possible ways. Hence, $\sum_{i<j} |A_i \cap A_j| = (n-2)!\, C(n,2) = (n-2)! \cdot \frac{n(n-1)}{2!} = \frac{n!}{2!}$

  since there are C(n,2) terms in the sum

3.2. Derangement

Let A_i = subset of permutations of $\{1, 2, \ldots, n\}$ such that the ith element = i in the permutation.

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{i=1}^{n} |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k| \quad (1)$$

$$- \sum_{i<j<k<\ell} |A_i \cap A_j \cap A_k \cap A_\ell| + \ldots + (-1)^{n+1} |A_1 \cap A_2 \cap \cdots \cap A_n|$$

- $|A_i \cap A_j \cap A_k| = (n\text{-}3)!$ since if exactly three of the n objects are placed at their original positions, that leaves the other $(n\text{-}3)$ objects to be freely permuted in $(n\text{-}3)!$ possible ways. Hence,

$$\sum_{i<j<k} P(A_i \cap A_j \cap A_k) = (n-3)!\, C(n,3) = (n-3)! \cdot \frac{n(n-1)(n-2)}{3!} = \frac{n!}{3!},$$

since there are C(n,3) terms in the sum.

The pattern should be clear. When we reach the final term of (1), we have

$$|A_1 \cap A_2 \cap \ldots \cap A_n| = 1$$

which corresponds to all n objects being placed at their original positions.

3.2. Derangement

Let A_i = subset of permutations of $\{1, 2, \ldots, n\}$ such that the ith element = i in the permutation.

$$|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{i=1}^{n} |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k|$$

$$- \sum_{i<j<k<\ell} |A_i \cap A_j \cap A_k \cap A_\ell| + \ldots + (-1)^{n+1} |A_1 \cap A_2 \cap \cdots \cap A_n| \quad (1)$$

$$= n! \left[1 - \frac{1}{2!} + \frac{1}{3!} - \frac{1}{4!} + \cdots + (-1)^{n+1} \frac{1}{n!} \right]$$

$$= n! \sum_{k=1}^{n} \frac{(-1)^{k+1}}{k!}$$

We have:

$$D_n = n! - |A_1 \cup A_2 \cup \cdots \cup A_n| = n! \sum_{k=0}^{n} \frac{(-1)^k}{k!}$$

3.2. Derangement

- An example of derangements arises in a very famous problem called the **hat-check problem:**

*"n people walk into a party and give their hats to **hat-check** girl. Unfornately, she completely loses track of which of n hats belong to which owners. Therefore, when the party finishes, their hats are returned at random."*

Question: What is the probability that nobody receives their own hat back?

➔ This is equivalent to asking for the probability that a permutation of n objects is a derangement.

➔ $= \frac{D_n}{n!}$ What is the value in number????

since there are D_n possible derangements and $n!$ possible permutations.

3.2. Derangement

It is amusing to note that as $n \to \infty$, the probability that a permutation of n values is a derangement is given by

$$\lim_{n\to\infty} P(\text{derangement}) = \lim_{n\to\infty} \frac{D_n}{n!} = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!} = \frac{1}{e} : \simeq 0.367879.$$

➔ if n is large (in practical applications, any n greater than about 10 can be considered to be large), then the probability that the permutation of the n values is a derangement is approximately 1/e almost independently of the precise value of n.

n	n!	Probability	#Derangements
0	1	1	
1	1	0	0
2	2	0.5	1
3	6	0.333333333	2
4	24	0.375	9
5	120	0.3666666667	44
6	720	0.368055556	265
7	5040	0.367857143	1854
8	40320	0.367881944	14833
9	362880	0.367879189	133496
10	3628800	0.367879464	1334961
Note: 1/e ≅ 0.367879441			

Contents

4. Recurrence relations

4.1. Recurrence relations

4.2. Solve recurrence relations

4.1. Recurrence relations

Definition:

A recurrence relation for the sequence $\{a_n\}$ is the equation that expresses a_n in terms of one or more of the previous terms in the sequence namely, a_0, a_1, …, a_{n-1}, for all integers n with $n \geq n_0$, where n_0 is a nonnegative integer.

A sequence is called a **solution** to a recurrence relation if its terms satisfy the recurrence relation.

Example: Consider the recurrence relation

$a_n = 2a_{n-1} - a_{n-2}$ for $n = 2, 3, 4, \ldots$ The sequence $a_n=n+1$??

- The sequence $a_n=3n$ is a solution of this recurrence relation?

For $n \geq 2$ we see that: $2a_{n-1} - a_{n-2} = 2(3(n-1)) - 3(n-2) = 3n = a_n$

Therefore, the sequence $a_n=3n$ is a solution of the recurrence relation

- The sequence $a_n=5$ is also a solution of the this recurrence relation?

For $n \geq 2$ we see that: $2a_{n-1} - a_{n-2} = 2 \cdot 5 - 5 = 5 = a_n$

Therefore, the sequence $a_n=5$ is also a solution of the recurrence relation

the same recurrence relation can have **multiple solutions**. ➔ Why???

4.1. Recurrence relations

A recurrence **without specifying any initial values (initial conditions)**.

➔ can have (and usually has) **multiple solutions**.

Example: $a_n = 2a_{n-1} - a_{n-2}$ for $n = 2, 3, 4$. It has the following sequences a_n as solution:

- $a_n=5$
- $a_n=3n$
- $a_n=n+1$

If **both** the initial conditions and the recurrence relation are specified, then the sequence is **uniquely** determined.

Example: $a_n = 2a_{n-1} - a_{n-2}$ for $n = 2, 3, 4$

where $a_0=0$; $a_1 = 3$

➔ The sequence $a_n=5$ is not the solution

➔ The sequence $a_n=3n$ is the unique solution

4.1. Recurrence relations

Some applications of recurrence relations:

1. To solve many counting problems we can try construct recurrence relations.
2. Complexity analysis of the recursive algorithm is a recurrence relation on the number of operations.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Modeling with Recurrence Relations

Example 1: Someone deposits $10,000 in a savings account at a bank yielding 5% per year with interest compounded annually. How much money will be in the account after 30 years?

$$P_{30} = (1.05)^{30} \cdot 10{,}000 = 43{,}219.42$$

Solution:

- Let P_n denote the amount in the account after n years.

How can we determine P_n on the basis of P_{n-1}?

- We can derive the following **recurrence relation**:

$$P_n = P_{n-1} + 0.05P_{n-1} = 1.05P_{n-1}.$$

- The initial condition is $P_0 = 10{,}000$.

Then we have:

- $P_1 = 1.05P_0$
- $P_2 = 1.05P_1 = (1.05)^2P_0$
- $P_3 = 1.05P_2 = (1.05)^3P_0$
- …
- $P_n = 1.05P_{n-1} = (1.05)^nP_0$

➔ We now have a **formula** to calculate P_n for any natural number n and can avoid the iteration.

Example 2: Fibonacci

A pair of rabbits is placed on an island. After 2 months old, a pair of rabbits produces another pair each month. How many rabbits are there after n months?

- $n = 1: f_1 = 1$
- $n = 2: f_2 = 1$ **Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233,etc.**
- $n > 2: f_n = f_{n-1} + f_{n-2}$

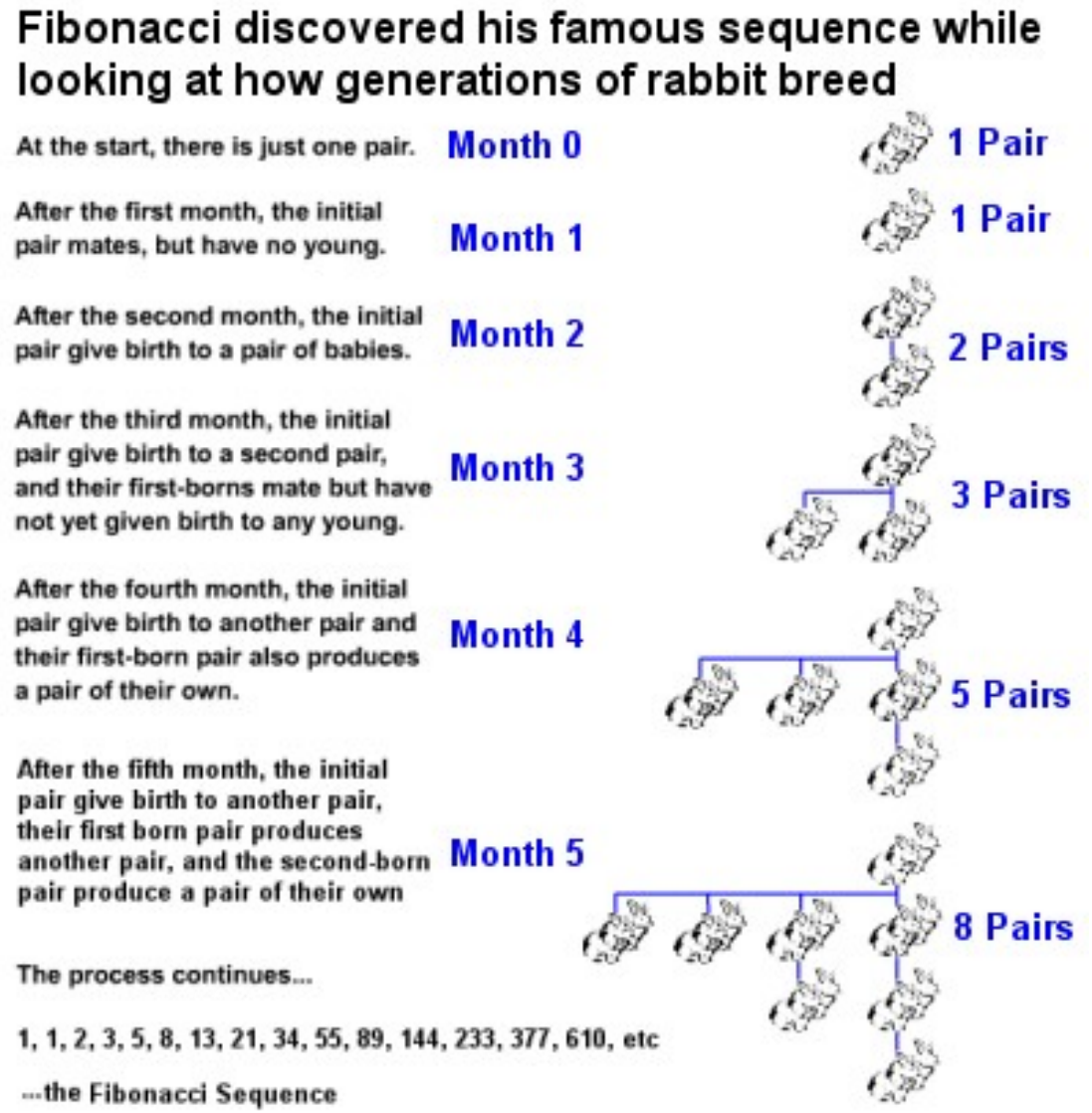
The $n > 2$ formula is true since each new pair comes from a pair at least 2 months old.

New-born rabbits	Month	1-month rabbits	>=2-month rabbits	Total f_n
	1	1	0	1
	2	0	1	1
	3	1	1	2
	4	1	2	3
	5	2	3	5
	6	3	5	8
	7	5	8	13
	8	8	13	21

Modeling with Recurrence Relations

Example 2: Fibonacci

A pair of rabbits is placed on an island. After 2 months old, a pair of rabbits produces another pair each month. How many rabbits are there after n months?



- $n = 1: f_1 = 1$
- $n = 2: f_2 = 1$
- $n > 2: f_n = f_{n-1} + f_{n-2}$

The $n > 2$ formula is true since each new pair comes from a pair at least 2 months old.

Modeling with Recurrence Relations

Example 3: Let a_n denote the number of bit strings of length n that do not have two consecutive 0s ("valid strings"). Find a recurrence relation and give initial conditions for the sequence $\{a_n\}$.

Solution:

Idea: The number of valid strings equals the number of valid strings starting with a 0 plus the number of valid strings starting with a 1.

- The sum rule: $a_n = |A| + |B|$.
- We have:
  - A = {**1**??????} → $|A| = a_{n-1}$
  - B = {**0**??????} → 2nd place is **1** → $|B| = a_{n-2}$
- Therefore

$$a_n = a_{n-1} + a_{n-2}, \quad n > 2 \qquad\qquad f_n = f_{n-1} + f_{n-2}, \; n > 2$$

$$a_1 = 2; \; a_2 = 3; \qquad\qquad f_1 = 1, f_2 = 1$$

This sequence satisfies the same recurrence relation as the **Fibonacci sequence**.
Since $a_1 = f_3$ and $a_2 = f_4$, we have $a_n = f_{n+2}$

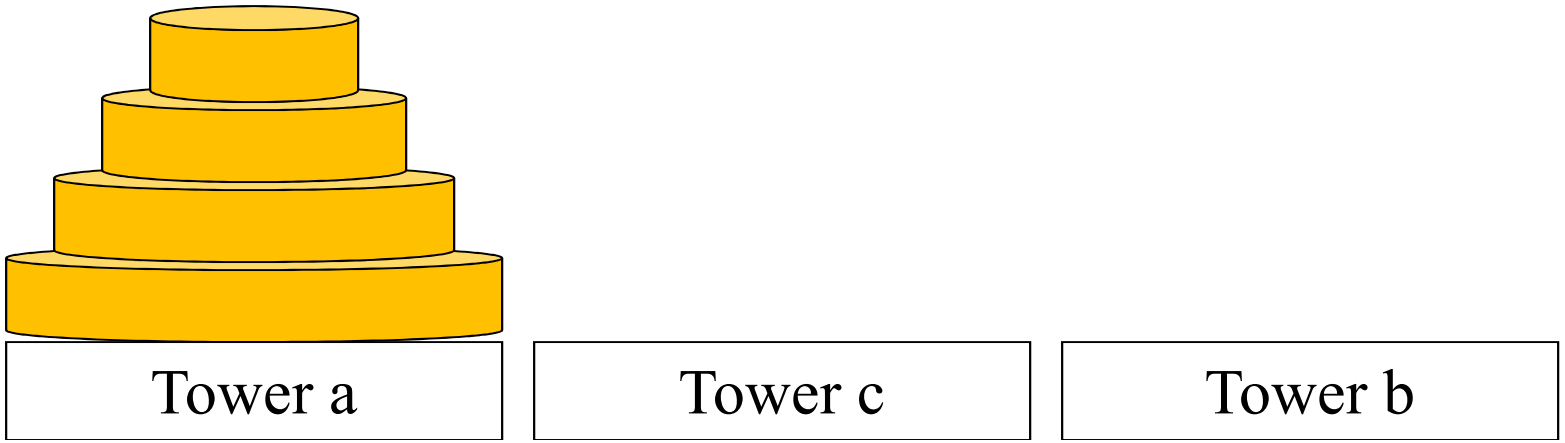
Modeling with Recurrence Relations

Example 4: The Tower of Hanoi, consists of three towers (a), (b), (c) together with n disks of different sizes. Initially these disks are stacked on the tower (a) in an ascending order, i.e. the smaller one sits over the larger one.

The objective of the game is to move all the disks from tower (a) to tower (c), following 3 rules:

- Only one disk can be moved at a time.
- Only the top disk can be moved
- No large disk can be sit over a smaller disk.

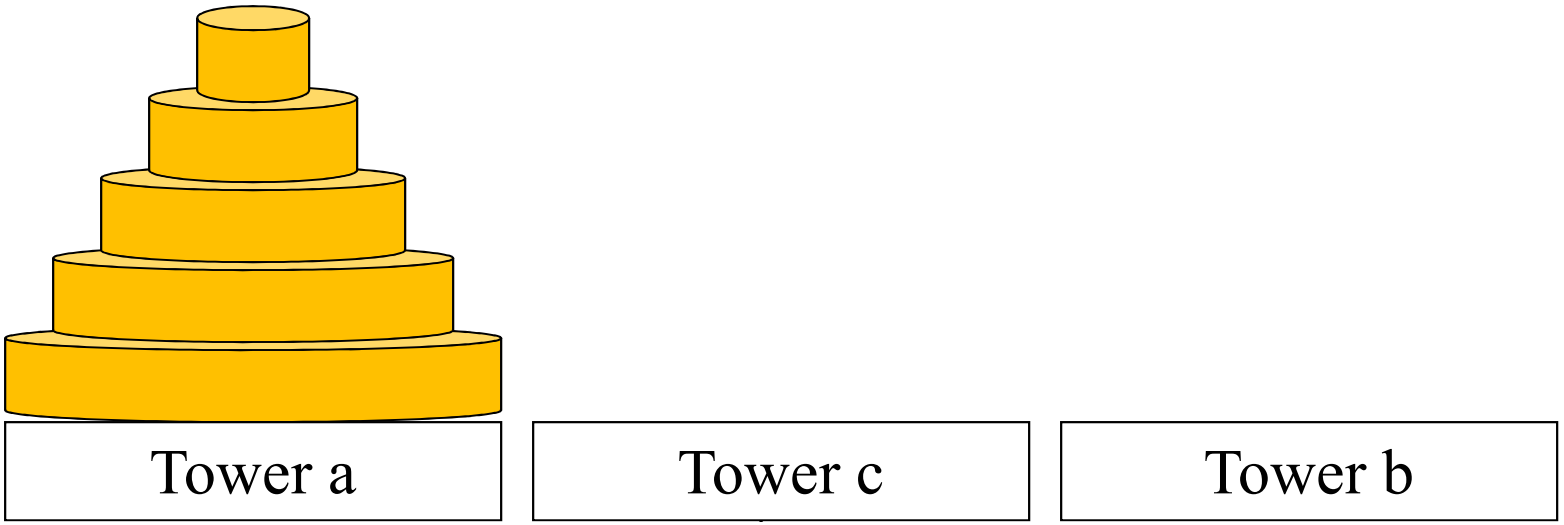
Let h_n denote the minimum number of moves needed to solve the Tower of Hanoi problem with n disks. What is the recurrence relation for h_n?



Tower of Hanoi: *n*=5

The objective of the game is to move all the disks from tower (a) to tower (c), following 3 rules:

1. Only one disk can be moved at a time.
2. Only the top disk can be moved
3. No large disk can be sit over a smaller disk.



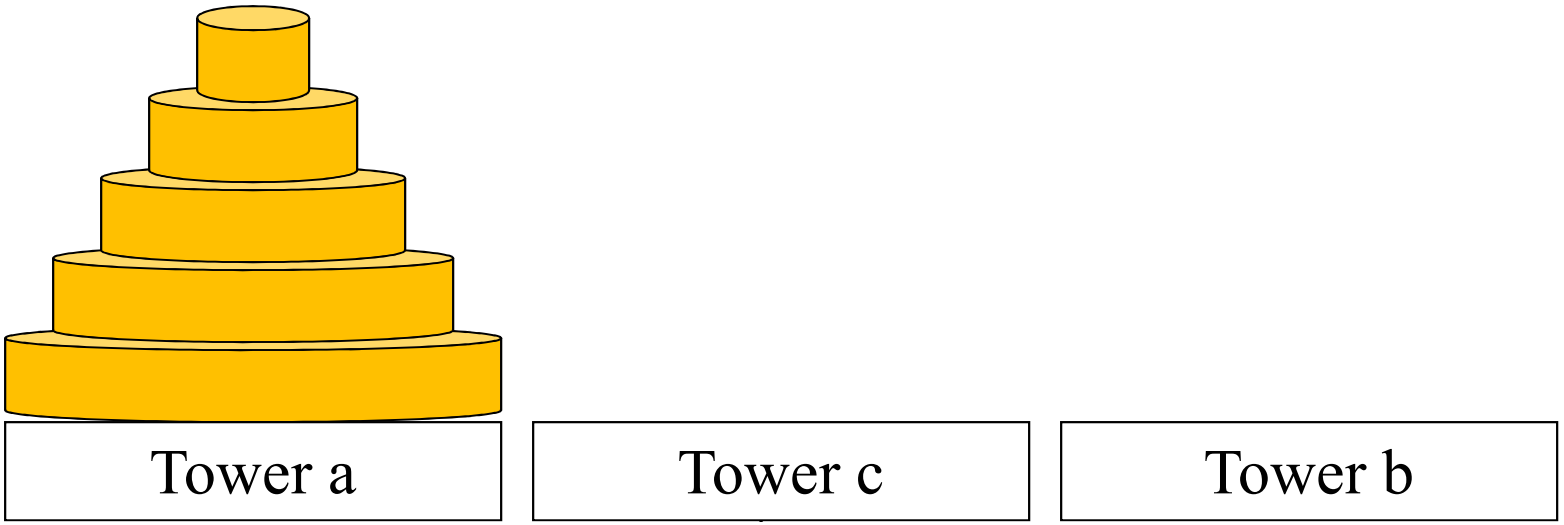
Tower of Hanoi

- $h_1 = 1$
- For $n \geq 2$, we need to do 3 following steps to transfer all disks from tower (a) to tower (c):

  (1) Move the top n - 1 disks (following the rules of the game) from tower (a) to tower (b)

  (2) Move the largest disk to the tower (c)

  (3) Move the n-1 disks (following the rules of the game) from tower (b) to tower (c), placing them on top of the largest disk



Tower of Hanoi

- $h_1 = 1$
- For $n \geq 2$, we need to do 3 following steps to transfer all disks from tower (a) to tower (c):

(1) Move the top n - 1 disks (following the rules of the game) from tower (a) to tower (b)

The problem of n-1 disks → #moves = h_{n-1}

(2) Move the largest disk to the tower (c)

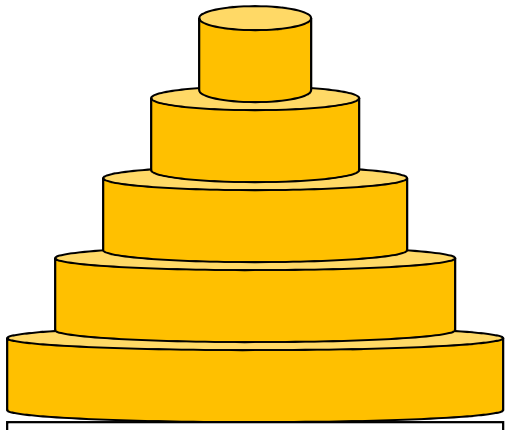
→ #moves = 1

(3) Move the n-1 disks (following the rules of the game) from tower (b) to tower (c), placing them on top of the largest disk

The problem of n-1 disks→ #moves = h_{n-1}

$$h_n = 2h_{n-1} + 1, n \geq 2$$

$$h_1 = 1$$



Tower of Hanoi: n=5

(1) Move the top n - 1 disks (following the rules of the game) from tower (a) to tower (b)

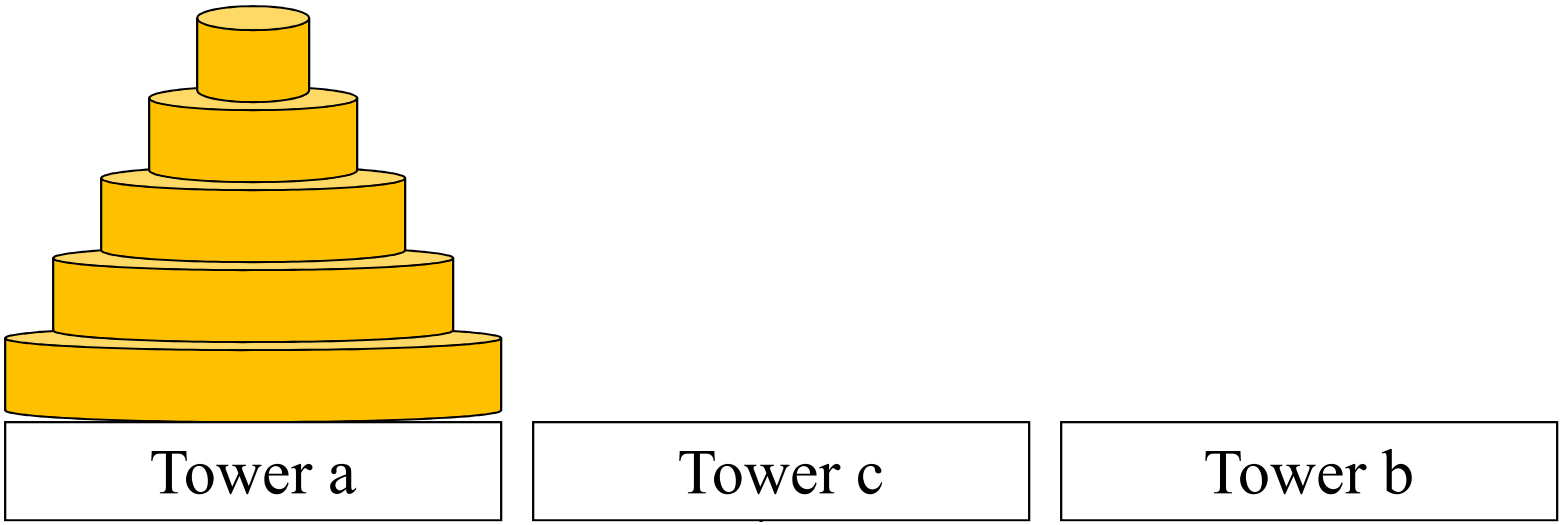
The problem of n-1 disks → #moves = h_{n-1}

(2) Move the largest disk to the tower (c)

→ #moves = 1

(3) Move the n-1 disks (following the rules of the game) from tower (b) to tower (c) , placing them on top of the largest disk

The problem of n-1 disks → #moves = h_{n-1}



Tower of Hanoi

Find the **formula** to calculate h_n for any natural number n and can avoid the iteration.

$$h_n = 2\,h_{n-1} + 1$$

$$= 2\,(2\,h_{n-2} + 1) + 1 \qquad = 2^2\,h_{n-2} + 2 + 1$$

$$= 2^2(2\,h_{n-3} + 1) + 2 + 1 \quad = 2^3\,h_{n-3} + 2^2 + 2 + 1$$

$$\ldots$$

$$= 2^{n-1}\,h_1 + 2^{n-2} + \ldots + 2 + 1$$

$$= 2^{n-1} + 2^{n-2} + \ldots + 2 + 1 \qquad \text{(because } h_1 = 1\text{)}$$

$$= 2^n - 1$$

History "Tower of Hanoi"

Legend: a group of Eastern monks are the keepers of three towers on which sit 64 golden rings. Originally all 64 rings were stacked on one tower with each ring smaller than the one above. The monks are to move the rings from this first tower to the third tower one at a time but never moving a larger ring on top of a smaller one. Once the 64 rings have all been moved, the world will come to an end.

➔ Number of moves = 2^{64}-1 = 18 446 744 073 709 551 615

➔ which would require more than *five billion centuries!*

It was invented by the French mathematician Eduoard Lucas in 1883

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

4.1. Recurrence relations

Some applications of recurrence relations:

1. To solve many counting problems we can try construct recurrence relations.
2. Complexity analysis of the recursive algorithm is a recurrence relation on the number of operations.

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

4.1. Recurrence relations

Example 5. Recall the factorial function

$$n! = \begin{cases} 1 & \text{if } n = 1 \\ n \cdot (n-1)! & \text{if } n > 1 \end{cases}$$

Consider the following (recursive) algorithm for computing $n!$

Algorithm (FACTORIAL)

```
    INPUT     : n ∈ ℕ
    OUTPUT    : n!
1   IF n = 1 THEN
2       return 1
3   END
4   ELSE
5       return FACTORIAL(n − 1) × n
6   END
```

4.1. Recurrence relations

Factorial(n);

Algorithm (FACTORIAL)

```
   INPUT        : n ∈ ℕ
   OUTPUT       : n!
1  IF n = 1 THEN
2        return 1
3  END
4  ELSE
5        return FACTORIAL(n − 1) × n
6  END
```

How many times is function Factorial called when we execute the statement **Factorial(n);** ?

- When $n = 1$, Factorial is called 1 ➔ T(1) = 1
- Otherwise:
  - We perform 1.
  - Plus the number of calls in the recursive call in Factorial($n - 1$) ➔ T(n-1)

T(n) = 1 + T(n-1)

This can be expressed as a formula (similar to the definition of n!):

$$T(1) = 1$$

$$T(n) = 1 + T(n-1),\ n > 1$$

This is known as a recurrence relation.

Find the **formula** to calculate T(n) for any natural number n and can avoid the iteration. **explicit formula**

4.1. Recurrence relations

Factorial(n);

Algorithm (FACTORIAL)

```
   INPUT        : n ∈ ℕ
   OUTPUT       : n!
1  IF n = 1 THEN
2        return 1
3  END
4  ELSE
5        return FACTORIAL(n − 1) × n
6  END
```

$T(n) = T(n-1) + 1, n > 1$

$= [T(n\text{-}2) + 1] + 1$ [has 2 ones]

$= [[T(n\text{-}3)+1]+1]+1$ [has 3 ones]

$= \ldots$

$= T(n - (n\text{-}1)) + (n\text{-}1)$ [has n-1 ones]

$= T(1) + (n\text{-}1)$

$= 1 + (n\text{-}1)$

$= n$

4.1. Recurrence relations

Algorithm (FACTORIAL)

```
   INPUT        : n ∈ ℕ
   OUTPUT       : n!
1  IF n = 1 THEN
2        return 1
3  END
4  ELSE
5        return FACTORIAL(n − 1) × n
6  END
```

How many multiplications M(n) does Factorial perform?

- When n = 1 we don't perform any ➔ M(1) = 0
- Otherwise:
  - We perform 1.
  - Plus the number of multiplications we perform in the recursive call, Factorial(n − 1) ➔ M(n-1)

This can be expressed as a formula (similar to the definition of n!):

$$M(0) = 0;\ M(1) = 0$$

$$M(n) = 1 + M(n-1),\ n > 1$$

This is known as a recurrence relation.

Find the **formula** to calculate T(n) for any natural number n and can avoid the iteration.

4.1. Recurrence relations

Example 6. Performance of Recursive Binary Search

Input: An array *A* consists of *n* elements: A[0],…,A[*n*-1] in ascending order; Value ***key*** with the same data type as array *A*.

Output: the index in array if ***key*** is found, -1 if ***key*** is not found

To solve this problem, the divide and conquer technique is built based on the following argument: The given `key` is either

equal to the element at the **middle** of array A
or equal to the element at the **left (L) half** of array A
or equal to the element at the **right (R) half** of array A.
or not in array

4.1. Recurrence relations

Input: An array *A* consists of *n* elements: A[0],…,A[*n*-1] in ascending order;
Value ***key*** with the same data type as array *A*.

Output: the index in array if ***key*** is found, -1 if ***key*** is not found

To solve this problem, the divide and conquer technique is built based on the following argument: The given `key` is either

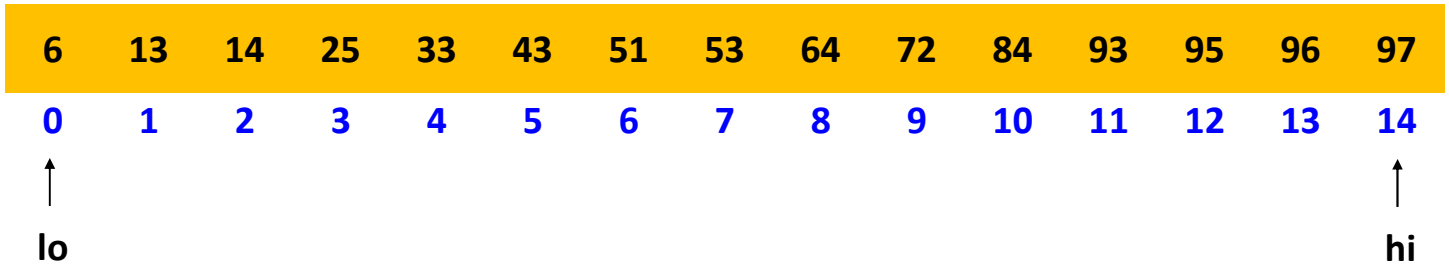
equal to the element at the **middle** of array A
or equal to the element at the **left (L) half** of array A
or equal to the element at the **right (R) half** of array A.
or not in array

```
int binsearch(int low, int high, int A[], int key)
{
   if (low <= high)
   {
       mid = (low + high) / 2;
       if (A[mid]==key) return mid;
       else if (key < A[mid])
           return binsearch(low, mid-1, A, key);
       else
           return binsearch(mid+1, high, A, key);
    }
    else return -1;
}
```

Example: Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
   if (low <= high)
   {
       mid = (low + high) / 2;
       if (A[mid]== key) return mid;
       else if (key < A[mid])
           return binsearch(low, mid-1, A, key);
       else
           return binsearch(mid+1, high, A, key);
    }
    else return -1;
}
```

key=33

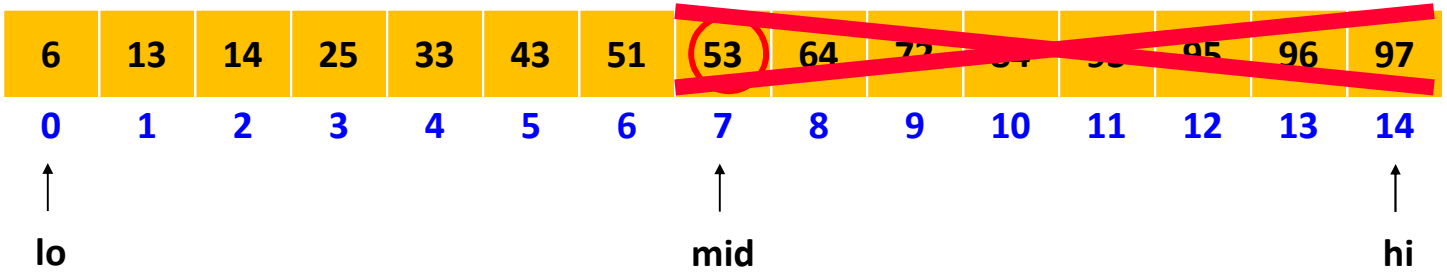


binsearch(0, 14, A, 33);

Example: Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
    if (low <= high)
    {
        mid = (low + high) / 2;
        if (A[mid]== key) return mid;
        else if (key < A[mid])
            return binsearch(low, mid-1, A, key);
        else
            return binsearch(mid+1, high, A, key);
     }
     else return -1;
}
```

key=33



binsearch(0, 14, A, 33);

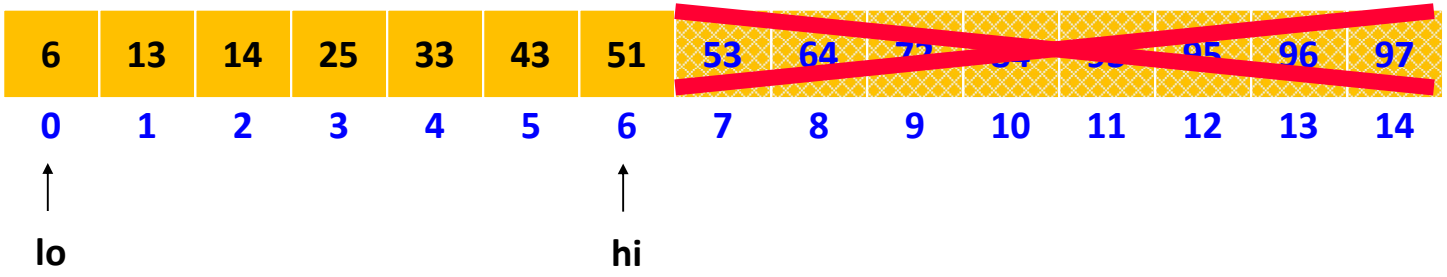
binsearch(0, 6, A, 33);

The section to be investigated is halved after each iteration

Example: Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
    if (low <= high)
    {
        mid = (low + high) / 2;
        if (A[mid]== key) return mid;
        else if (key < A[mid])
            return binsearch(low, mid-1, A, key);
        else
            return binsearch(mid+1, high, A, key);
     }
     else return -1;
}
```

key=33



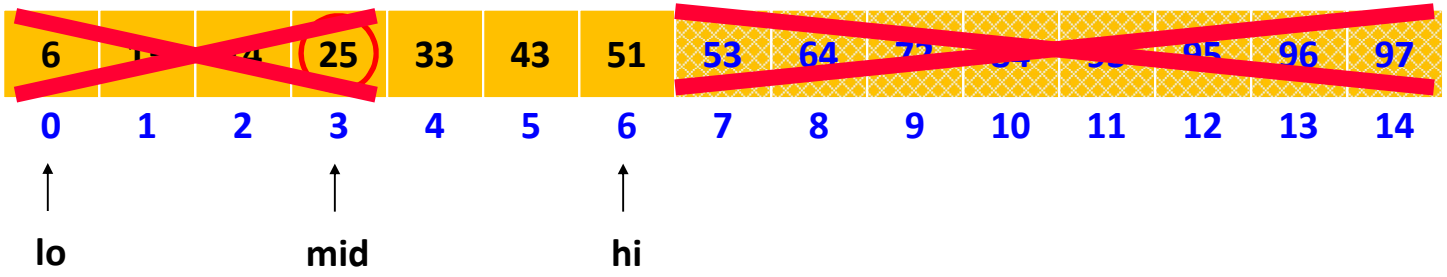
binsearch(0, 14, A, 33);

binsearch(0, 6, A, 33);

Example: Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
    if (low <= high)
    {
        mid = (low + high) / 2;
        if (A[mid]== key) return mid;
        else if (key < A[mid])
            return binsearch(low, mid-1, A, key);
        else
            return binsearch(mid+1, high, A, key);
     }
     else return -1;
}
```

key=33



binsearch(0, 14, A, 33);

binsearch(0, 6, A, 33);

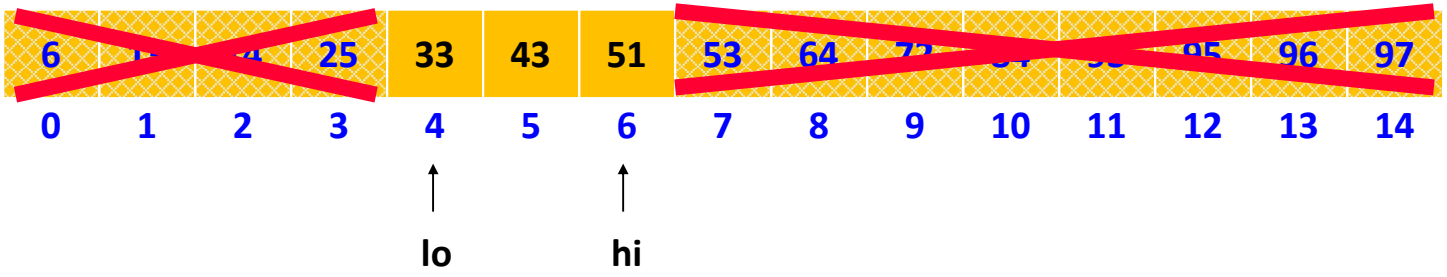
binsearch(4, 6, A, 33);

The section to be investigated is halved after each iteration

Example: Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
   if (low <= high)
   {
       mid = (low + high) / 2;
       if (A[mid]== key) return mid;
       else if (key < A[mid])
           return binsearch(low, mid-1, A, key);
       else
           return binsearch(mid+1, high, A, key);
    }
    else return -1;
}
```

key=33



binsearch(0, 14, A, 33);

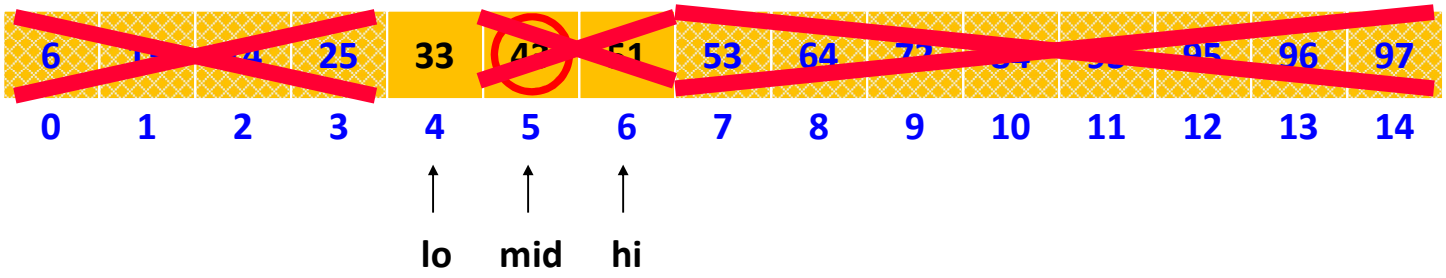
binsearch(0, 6, A, 33);

binsearch(4, 6, A, 33);

Example: Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
   if (low <= high)
   {
       mid = (low + high) / 2;
       if (A[mid]== key) return mid;
       else if (key < A[mid])
           return binsearch(low, mid-1, A, key);
       else
           return binsearch(mid+1, high, A, key);
    }
    else return -1;
}
```

key=33



binsearch(0, 14, A, 33);

binsearch(0, 6, A, 33);

binsearch(4, 6, A, 33);

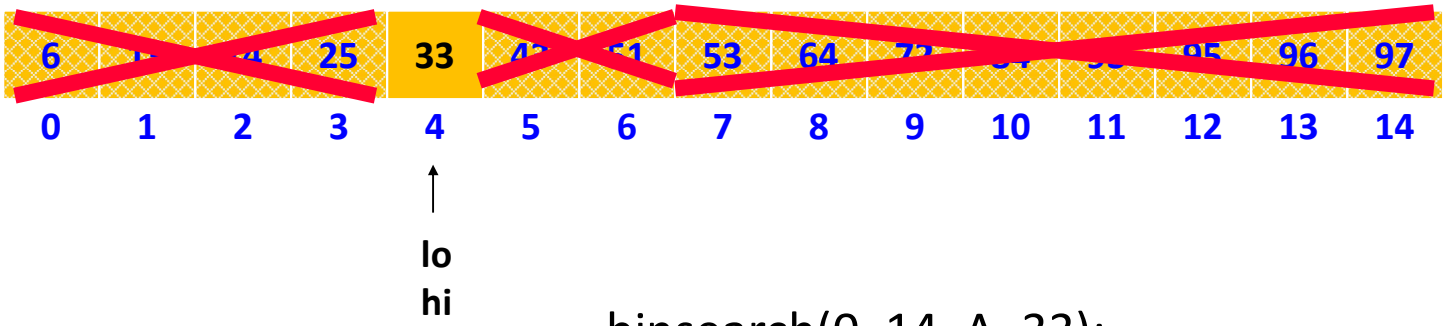
binsearch(4, 4, A, 33);

The section to be investigated is halved after each iteration

Example: Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
   if (low <= high)
   {
       mid = (low + high) / 2;
       if (A[mid]== key) return mid;
       else if (key < A[mid])
           return binsearch(low, mid-1, A, key);
       else
           return binsearch(mid+1, high, A, key);
    }
    else return -1;
}
```

key=33



binsearch(0, 14, A, 33);

binsearch(0, 6, A, 33);

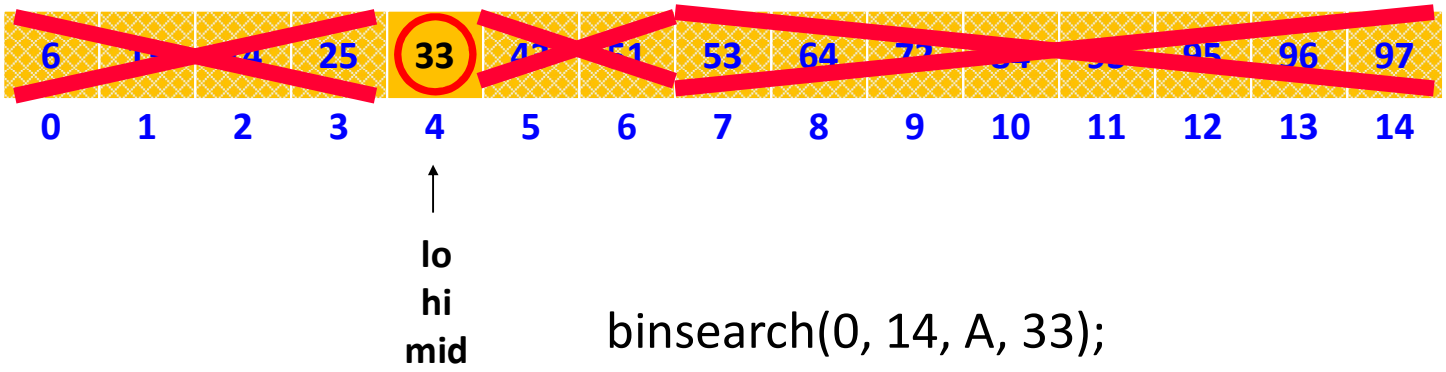
binsearch(4, 6, A, 33);

binsearch(4, 4, A, 33);

Example: Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
    if (low <= high)
    {
        mid = (low + high) / 2;
        if (A[mid]== key) return mid;
        else if (key < A[mid])
            return binsearch(low, mid-1, A, key);
        else
            return binsearch(mid+1, high, A, key);
     }
     else return -1;
}
```

key=33



binsearch(0, 14, A, 33);

binsearch(0, 6, A, 33);

binsearch(4, 6, A, 33);

binsearch(4, 4, A, 33);

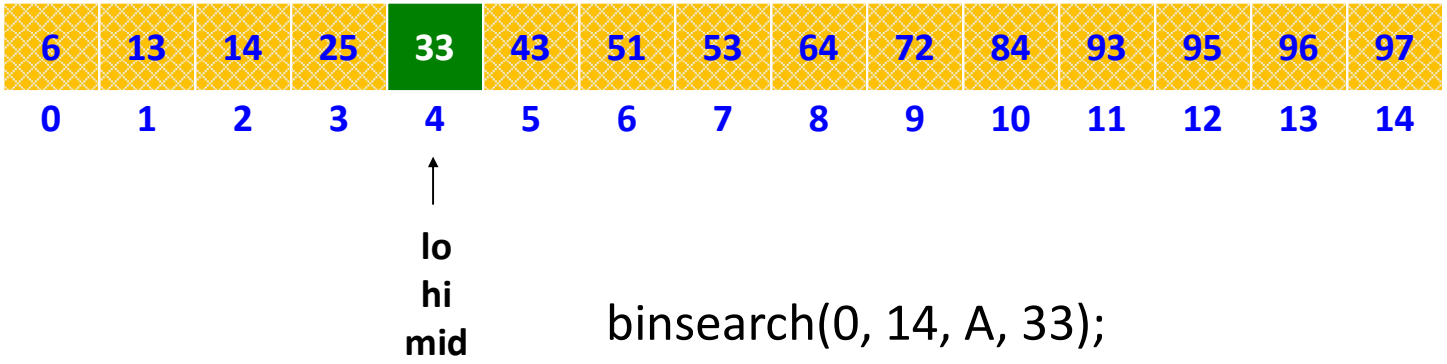
Example: Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
   if (low <= high)
   {
       mid = (low + high) / 2;
       if (A[mid]== key) return mid;
       else if (key < A[mid])
           return binsearch(low, mid-1, A, key);
       else
           return binsearch(mid+1, high, A, key);
    }
    else return -1;
}
```

key=33

Value 33 is located @index=4

key=31??



binsearch(0, 14, A, 33);

binsearch(0, 6, A, 33);

binsearch(4, 6, A, 33);

binsearch(4, 4, A, 33);

4.1. Recurrence relations

Example 6. Performance of Recursive Binary Search

Input: An array *A* consists of *n* elements: A[0],…,A[*n*-1] in ascending order; Value ***key*** with the same data type as array *A*.

Output: the index in array if ***key*** is found, -1 if ***key*** is not found

```
int binsearch(int low, int high, int A[], int key)
{
   if (low <= high)
   {
       mid = (low + high) / 2;
       if (A[mid]==key) return mid;
       else if (key < S[mid])
           return binsearch(low, mid-1, A, key);
       else
           return binsearch(mid+1, high, A, key);
    }
    else return -1;
}
```

➔ binsearch(0, *n*-1, *A*, *key*);

How many times is binsearch called in the worst case ?

4.1. Recurrence relations

Example 6. Performance of Recursive Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
    if (low <= high)
    {
        mid = (low + high) / 2;
        if (A[mid]==key) return mid;
        else if (key < S[mid])
            return binsearch(low, mid-1, A, key);
        else
            return binsearch(mid+1, high, A, key);
     }
     else return -1;
}
```

➔ T(0) = ?

➔ binsearch(0, -1, *A*, *key*);

➔ T(1) = ?

➔ binsearch(0, 0, *A*, *key*);

➔ binsearch(0, -1, *A*, *key*);

➔ binsearch(1, 0, *A*, *key*);

➔ binsearch(0, *n*-1, *A*, *key*);

Let T(*n*): the number of times that binsearch is called in the worst case when array *A* has *n* elements

- $T(0) = 1$
- $T(1) = 2$
- $T(2) = T(1) + 1 = 3$
- $T(4) = T(2) + 1 = 4$
- $T(8) = T(4) + 1 = 4 + 1 = 5$
- $T(n) = T(n/2) + 1$

Recurrence relation:

$T(n) = T(n/2) + 1$

$T(0) = 1$

$T(1) = 2$

4. Recurrence relations

4.1. Recurrence relations

4.2. Solving recurrence relations

4.2. Solving Recurrence Relations

Solving the recurrence relation for a sequence $a_0, a_1, a_2, a_3, \ldots a_n, \ldots$ is to give the explicit formula to compute the value for the general term a_n, i.e., to find an expression for an that does not involve any other a_i

Example: Given the recurrence relation:

$$a_n = 2a_{n-1} - a_{n-2} \text{ where } n = 2, 3, 4, \ldots$$

$$a_0=0; a_1 = 3$$

➔ The explicit formula for the above recurrence relation is $a_n=3n$

➔ $a_n=3n$ is the solution to the above recurrence relation

- Does not exist the method to solve all types of recurrence relation.
- Consider method to solve the recurrence relation with following types:
  - A linear homogeneous recurrence relation of degree k with constant coefficients
  - A linear nonhomogeneous recurrence relation of degree k with constant coefficients

4.2. Solving Recurrence Relations

Definition: A linear homogeneous recurrence relation of degree k with constant coefficients is a recurrence relation of the form:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$$

where $c_1, c_2, \ldots, c_k$ are real number constants, and $c_k \neq 0$.

• A sequence satisfying such a recurrence relation is uniquely determined by the recurrence relation if it also satisfying k initial conditions:

$$a_0 = C_0, a_1 = C_1, a_2 = C_2, \ldots, a_{k-1} = C_{k-1}$$

where $C_0, C_1, ..., C_{k-1}$ are constants.

4.2. Solving Recurrence Relations

Definition: A linear homogeneous recurrence relation of degree k with constant coefficients is a recurrence relation of the form:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$$

where $c_1, c_2, \ldots, c_k$ are real number constants, and $c_k \neq 0$.

Explain:

- Linear (tuyến tính): the right-hand side is the sum of the terms before the term a_n in the sequence where the coefficients (c_1, c_2, ..,c_k) are constant (not a function dependent on n)
- Homogeneous (thuần nhất): the right-hand side has no additional terms other than the terms a_i of the sequence
- Degree k: the right hand side has the (n-k)th term of the sequence

A linear homogeneous recurrence relation of degree k with constant coefficients

Example 1: Which one is the linear homogeneous recurrence relation of degree k with constant coefficients

1) $a_n = 4a_{n-1} + 2na_{n-3}$

2) $h_n = 2h_{n-1} + 1$

3) $b_n = 5b_{n-2} + 2(b_{n-3})^2$

4) $q_n = 3\, q_{n-6} + q_{n-8}$

A linear homogeneous recurrence relation of degree k with constant coefficients

- We try to find solution of the form $a_n = r^n$, where r is a constant.
- The sequence $\{a_n = r^n\}$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}$$

if and only if r satisfying:

$$r^n = c_1 r^{n-1} + \ldots + c_k r^{n-k}, \text{ or}$$

$$\boxed{r^k - c_1 r^{k-1} - \ldots - c_k = 0}$$

(substract the right-hand side from the left and × by r^{k-n})

this is called the **characteristic equation** of the recurrence relation, and its solution is called **characteristic roots** of the recurrence relation.

- We will use characteristic roots to obtain explicit formula for the sequence.

A linear homogeneous recurrence relation of degree k with constant coefficients

- **Theorem 1.** Let c_1 and c_2 be real numbers.

Suppose that $r^2 - c_1 r - c_2 = 0$ *has 2 distinct roots* r_1 *and* r_2. *Then the sequence* $\{a_n\}$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

if and only if

$$a_n = \alpha_1 (r_1)^n + \alpha_2 (r_2)^n \qquad (1)$$

$n = 0, 1, \ldots,$ *where* α_1 *and* α_2 *are constants.*

Example 1

Fibonacci sequence is given by the following recurrence relation:

$$F_n = F_{n-1} + F_{n-2}, n \geq 2,$$
$$F_0 = 0, \; F_1 = 1.$$

Find the explicit formula for F_n.

Solution: Solve the characteristic equation:

$$r^2 - r - 1 = 0,$$

its characteristic roots are:

$$r_1 = \frac{1+\sqrt{5}}{2}; \quad r_2 = \frac{1-\sqrt{5}}{2}$$

Leonardo Fibonacci
1170-1250

Recurrence relation: $a_n = c_1 a_{n-1} + c_2 a_{n-2}$

Characteristic equation: $r^2 - c_1 r - c_2 = 0$

Example 1

- The explicit formula:

$$F_n = \alpha_1.(r_1)^n + \alpha_2.(r_2)^n$$

where α_1, α_2 are constants and could be determined by using initial conditions F_0, F_1.

$F_0 = \alpha_1 + \alpha_2 = 0$

$F_1 = \alpha_1 r_1 + \alpha_2 r_2 = 1$

Solving these two equations, we have: $\alpha_1 = \frac{1}{\sqrt{5}}$; $\alpha_2 = -\frac{1}{\sqrt{5}}$

Therefore

Muavre formula

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right), \quad n \geq 0.$$

The case: one characteristic root with multiplicities 2

But what happens if the characteristic equation has only one root?

Theorem 2: Let c_1 and c_2 be real numbers with $c_2 \neq 0$. Suppose that $r^2 - c_1 r - c_2 = 0$ has only one root r_0. Then a sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$

if and only if

$$a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$$

$n = 0, 1, \ldots,$ *where* α_1 , α_2 *are constants.*

Example 2

Solving the following recurrence relation

$$a_n = 6\,a_{n-1} - 9\,a_{n-2}$$

with initial conditions $a_0 = 1$ and $a_1 = 6$.

Solution:

Characteristic equation:

$r^2 - 6r + 9 = 0$ has one root $r = 3$. The solution is:

$$a_n = \alpha_1 3^n + \alpha_2 n 3^n$$

To determine α_1, α_2 , using the initial conditions, we have:

$$a_0 = 1 = \alpha_1 ,$$

$$a_1 = 6 = \alpha_1 * 3 + \alpha_2 * 1 * 3$$

Solving these equations, we have $\alpha_1 = 1$ and $\alpha_2 = 1$.

Therefore, the solution of the recurrence relation is:

$$\boldsymbol{a_n = 3^n + n\,3^n}$$

Recurrence relation: $a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}$

Characteristic equation: $r^k - c_1 r^{k-1} - \ldots - c_k = 0$

General case

Theorem 3. Let $c_1, c_2, \ldots, c_k$ be real numbers. Assume the characteristic equation

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \ldots - c_k = 0$$

has k distinct roots $r_1, r_2, \ldots, r_k$. *Then a sequence* $\{a_n\}$ *is a solution of the recurrence relation:*

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k},$$

if and only if

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \ldots + \alpha_k r_k^n$$

where $n = 0, 1, 2, \ldots$, *and* $\alpha_1, \alpha_2, \ldots, \alpha_k$ *are constants*

Example 3

Solving the recurrence relation:

$$a_n = 6\,a_{n-1} - 11\,a_{n-2} + 6\,a_{n-3}$$

where initial conditions

$$a_0 = 2,\ \ a_1 = 5,\ \ a_2 = 15.$$

Solution: Characteristic equation

$$r^3 - 6\,r^2 + 11\,r - 6 = 0$$

has 3 distinct roots $r_1 = 1,\ \ r_2 = 2,\ r_3 = 3.$

Therefore, the solution is

$$a_n = \alpha_1\,1^n + \alpha_2\,2^n + \alpha_3\,3^n \textit{ for some constants } \alpha_1, \alpha_2, \alpha_3$$

Recurrence relation: $a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}$

Characteristic equation: $r^k - c_1 r^{k-1} - \ldots - c_k = 0$

Example 3

Using the initial conditions, we have following equations to determine the value for constants α_1, α_2, α_3:

$$a_0 = 2 = \alpha_1 + \alpha_2 + \alpha_3$$

$$a_1 = 5 = \alpha_1 + \alpha_2.2 + \alpha_3.3$$

$$a_2 = 15 = \alpha_1 + \alpha_2.4 + \alpha_3.9.$$

Solving above equations, we have

$$\alpha_1 = 1,\ \alpha_2 = -1 \text{ and } \alpha_3 = 2.$$

Therefore, the solution of the recurrence relation is

$$a_n = 1 - 2^n + 2.\ 3^n$$

General case

Given linear homogeneous recurrence relation of degree k with constant coefficients

$$a_n = \sum_{i=1}^{k} c_i a_{n-i}$$

Its characteristic equation is:

$$r^k - \sum_{i=1}^{k} c_i r^{k-i} = 0$$

Theorem 4: If the characteristic equation has t distinct roots $r_1,\ldots,r_t$ with multiplicities $m_1,\ldots,m_t$ ($m_1+\ldots+m_t = k$). Then:

$$a_n = (\alpha_{1,0} + \alpha_{1,1} n + \ldots + \alpha_{1,m_1-1} n^{m_1-1}) r_1^n +$$
$$(\alpha_{2,0} + \alpha_{2,1} n + \ldots + \alpha_{2,m_2-1} n^{m_2-1}) r_2^n + \ldots$$
$$(\alpha_{t,0} + \alpha_{t,1} n + \ldots + \alpha_{t,m_t-1} n^{m_t-1}) r_t^n$$

$$a_n = \sum_{i=1}^{t} \left(\sum_{j=0}^{m_i-1} \alpha_{i,j} n^j \right) r_i^n$$

where $n \geq 0$, and α_{ij} are constants.

Example 4

Solve the following recurrence relation:

$$c_n = -4c_{n-1} + 3c_{n-2} + 18c_{n-3}, \quad n \geq 3,$$

$$c_0 = 1;\ c_1 = 2;\ c_2 = 13.$$

Solution: Characteristic equation

$$r^3 + 4r^2 - 3r - 18 = (r-2)(r+3)^2 = 0$$

Hence, the solution of the recurrence relation:

$$c_n = \alpha_{10}\, 2^n + (\alpha_{20} + \alpha_{21}\, n)(-3)^n$$

where α_{10}, α_{20}, α_{21} are constants

Example 4

These constants are determined by using initial conditions:

$0 = c_0 = \alpha_{10}\, 2^0 + (\alpha_{20} + \alpha_{21}.0)\,(-3)^0 = \alpha_{10} + \alpha_{20}$

$2 = c_1 = \alpha_{10}\, 2^1 + (\alpha_{20} + \alpha_{21}.1)\,(-3)^1 = 2\alpha_{10} - 3\alpha_{20} - 3\alpha_{21}$

$13 = c_2 = \alpha_{10}\, 2^2 + (\alpha_{20} + \alpha_{21}.2)\,(-3)^2 = 4\alpha_{10} + 9\alpha_{20} + 18\alpha_{21}$

Solving three equations above, we have: $\alpha_{10} = 1$; $\alpha_{20} = -1$; $\alpha_{21} = 1$

The solution of recurrence relation is:

$$c_n = 2^n + (-1 + n)(-3)^n$$

Example 5

Solving the recurrence relation

$$a_n = 7a_{n-1} - 16a_{n-2} + 12a_{n-3},\ n \geq 3;$$

$$a_0 = 1,\ a_1 = 5,\ a_2 = 17.$$

Linear nonhomogeneous recurrence relation with constant coefficients

- Linear *nonhomogeneous* recurrence relation with constant coefficients) has the term $F(n)$ dependent on n (and independent on any value of a_i) :

$$a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k} + F(n)$$

Linear homogeneous recurrence relation

Nonhomogeneous term

Solving Linear nonhomogeneous recurrence relation

The following result is the basis for solving the nonhomogeneous recurrence relation:

- If $a_n = p(n)$ is a particular solution of linear nonhomogeneous recurrence relation

$$a_n = \left(\sum_{i=1}^{k} c_i a_{n-i} \right) + F(n)$$

Then the solution of linear nonhomogeneous recurrence relation is of the form:

$$a_n = p(n) + h(n),$$

where $a_n = h(n)$ is the solution of the linear homogeneous part:

$$a_n = \sum_{i=1}^{k} c_i a_{n-i}$$

Solving Linear nonhomogeneous recurrence relation

Then, we have the method for solving linear nonhomogeneous recurrence relation:

1) Find the solution $h(n)$ of linear homogeneous part.
2) Find particular solution $p(n)$ of linear nonhomogeneous recurrence relation.
3) The solution of linear nonhomogeneous recurrence relation is of the form: $a_n = h(n) + p(n)$
4) Determine the constants from equations obtained by initial conditions

Solving Linear nonhomogeneous recurrence relation

How to find particular solution $p(n)$ of linear nonhomogeneous recurrence relation?

$$a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k} + F(n)$$

$c_1 a_{n-1} + \ldots + c_k a_{n-k}$: Linear homogeneous recurrence relation

$F(n)$: Nonhomogeneous term

There is no known general method for solving nonhomogeneous linear recurrence relation. However, we can develop a method for solving the special case:

$$F(n) = G(n) \times s^n = (b_t n^t + b_{t-1} n^{t-1} + \ldots + b_0) s^n$$

where s is a constant; $G(n)$ is polynomial in n

- If s is not equal to characteristic root, then particular solution $p(n)$ is of the form $F(n)$
- If s is equal to characteristic root with multiplicities m, then particular solution $p(n)$ is the form of $\boldsymbol{n^m \times Q(n) \times s^n}$ where $Q(n)$ is of the form $G(n)$

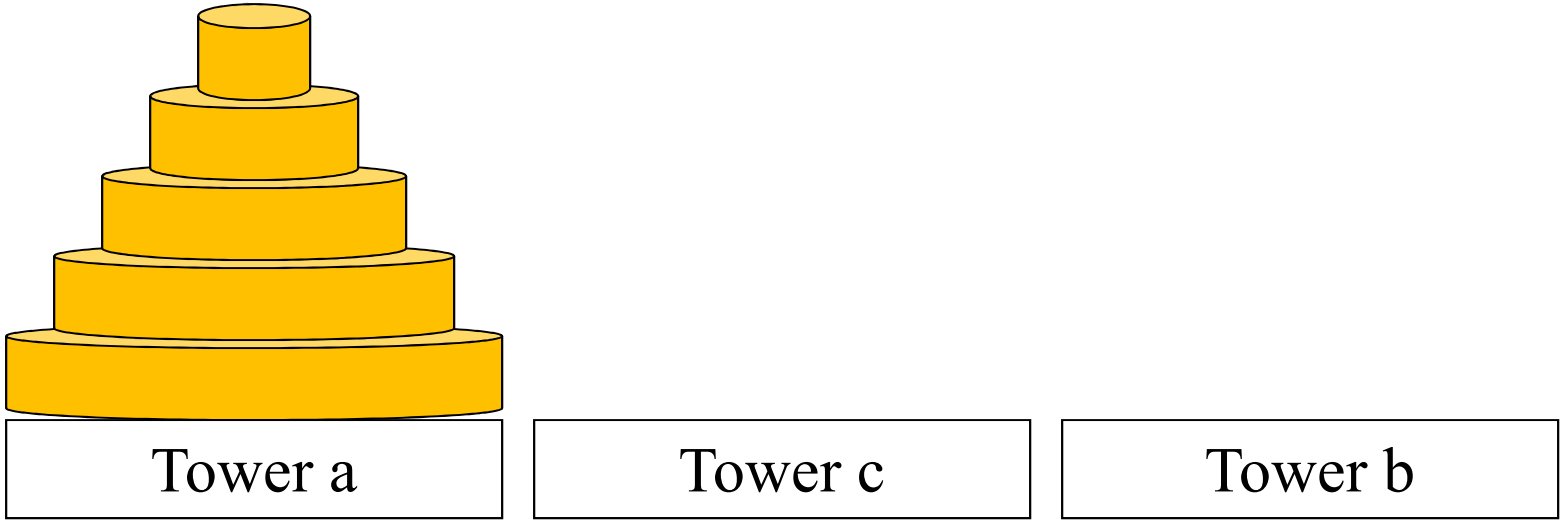
Example 1: Tower of Hanoi

Example 4: The Tower of Hanoi, consists of three towers (a), (b), (c) together with n disks of different sizes. Initially these disks are stacked on the tower (a) in an ascending order, i.e. the smaller one sits over the larger one.

The objective of the game is to move all the disks from tower (a) to tower (c), following 3 rules:

- Only one disk can be moved at a time.
- Only the top disk can be moved
- No large disk can be sit over a smaller disk.

Let a_n denote the minimum number of moves needed to solve the Tower of Hanoi problem with n disks. What is the recurrence relation for a_n?



Example 1: Tower of Hanoi

To solve the recurrence relation, we could use the substitution method:

$$a_n = 2\,a_{n-1} + 1$$

$$= 2\,(2\,a_{n-2} + 1) + 1 \qquad = 2^2\,a_{n-2} + 2 + 1$$

$$= 2^2(2\,a_{n-3} + 1) + 2 + 1 \quad = 2^3\,a_{n-3} + 2^2 + 2 + 1$$

$$\ldots$$

$$= 2^{n-1}\,a_1 + 2^{n-2} + \ldots + 2 + 1$$

$$= 2^{n-1} + 2^{n-2} + \ldots + 2 + 1 \qquad (\text{as } a_1 = 1)$$

$$= 2^n - 1$$

Example 1: Tower of Hanoi

Solving:

$a_n = 2a_{n-1} + 1, n \geq 1;$

$a_1 = 1.$

Solution:

Then, we have the method for solving linear nonhomogeneous recurrence relation:

1) Find the solution $h(n)$ of linear homogeneous part.
2) Find particular solution $p(n)$ of linear nonhomogeneous recurrence relation.
3) The solution of linear nonhomogeneous recurrence relation is of the form: $a_n = h(n) + p(n)$
4) Determine the constants from equations obtained by initial conditions

$$\underbrace{a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}}_{\text{Linear homogeneous recurrence relation}} + \underbrace{F(n)}_{\text{Nonhomogeneous term}}$$

(1) Linear homogeneous part: $a_n = 2a_{n-1}$

➔ Characteristic equation: $r - 2 = 0$ has one root $r = 2$

➔ Solution of linear homogeneous part is: $h(n) = c_1 2^n$

(2) Linear Nonhomogenenous part $F(n) = 1$, then the particular solution is of the form

$$p(n) = C$$

If $F(n) = G(n) \times s^n$ where $G(n)$ is polynomial in n, and s is a constant and not equal to characteristic root, then particular solution $p(n)$ is of the form $F(n)$

Substitute to the nonhomogeneos recurrent relation: $C = 2C+1$ ➔ $C = -1$

Hence the particular solution is: $p(n) = -1$.

Example 1: Tower of Hanoi

Solving:

$a_n = 2a_{n-1} + 1, n \geq 1;$

$a_1 = 1.$

Then, we have the method for solving linear nonhomogeneous recurrence relation:

1) Find the solution $h(n)$ of linear homogeneous part.
2) Find particular solution $p(n)$ of linear nonhomogeneous recurrence relation.
3) The solution of linear nonhomogeneous recurrence relation is of the form: $a_n = h(n) + p(n)$
4) Determine the constants from equations obtained by initial conditions

$$\underbrace{a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}}_{\text{Linear homogeneous recurrence relation}} + \overbrace{F(n)}^{\text{Nonhomogeneous term}}$$

(3) The solution of linear nonhomogeneous recurrence relation:

$$a_n = h(n) + p(n) = c_1 2^n - 1$$

(4) Determnine c_1 by using initial condition:

$$a_1 = c_1 2^1 - 1 = 1$$

➔ $c_1 = 1$

Hence, the solution of linear nonhomogeneous recurrence relation:

$$a_n = 2^n - 1, n \geq 1.$$

Example 2

Solving the recurrence rela

$a_n = 5a_{n-1} - 6a_{n-2} + 7^n$, $n \geq 2$,

$a_0 = 0$; $a_1 = 1$

Solution:

Then, we have the method for solving linear nonhomogeneous recurrence relation:

1) Find the solution $h(n)$ of linear homogeneous part.
2) Find particular solution $p(n)$ of linear nonhomogeneous recurrence relation.
3) The solution of linear nonhomogeneous recurrence relation is of the form: $a_n = h(n) + p(n)$
4) Determine the constants from equations obtained by initial conditions

$$\underbrace{a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}}_{\text{Linear homogeneous recurrence relation}} + \underbrace{F(n)}_{\text{Nonhomogeneous term}}$$

(1) Linear homogeneous part: $a_n = 5a_{n-1} - 6a_{n-2}$

Characteristic equation: $r^2 - 5r + 6 = 0$ has 2 distinct roots $r_1 = 3$, $r_2 = 2$

Thus the solution of linear homogeneous part is:

$$h(n) = c_1 3^n + c_2 2^n$$

(2) The nonhomogeneous part: $F(n) = 7^n$ and 7 is not characteristic root, thus the particular solution is of the form:

$p(n) = C.7^n$

If $F(n) = G(n) \times s^n$ where $G(n)$ is polynomial in n, and s is a constant and not equal to characteristic root, then particular solution $p(n)$ is of the form $F(n)$

Substitute to the recurrence relation:

$$C7^n = 5C7^{n-1} - 6C7^{n-2} + 7^n \rightarrow C = 49/20$$

Hence the particular solution is $p(n) = (49/20)7^n$

Example 2

Solving the recurrence rela

$a_n = 5a_{n-1} - 6a_{n-2} + 7^n$, $n \geq 2$,

$a_0 = 0$; $a_1 = 1$

Solution:

Then, we have the method for solving linear nonhomogeneous recurrence relation:

1) Find the solution $h(n)$ of linear homogeneous part.
2) Find particular solution $p(n)$ of linear nonhomogeneous recurrence relation.
3) The solution of linear nonhomogeneous recurrence relation is of the form: $a_n = h(n) + p(n)$
4) Determine the constants from equations obtained by initial conditions

$$a_n = \underbrace{c_1 a_{n-1} + \ldots + c_k a_{n-k}}_{\text{Linear homogeneous recurrence relation}} + \underbrace{F(n)}_{\text{Nonhomogeneous term}}$$

(3) The solution of linear nonhomogeneous recurrence relation is

$$a_n = p(n) + h(n) = (49/20)7^n + c_1 3^n + c_2 2^n$$

(4) The constants c_1, c_2 are determined by using initial conditions:

$$a_0 = c_1 + c_2 + 49/20 = 0$$

$$a_1 = 3c_1 + 2c_2 + (49/20).7 = 1$$

➔ $c_1 = ?$, $c_2 = ?$

Hence, the solution of linear nonhomogeneous recurrence relation is

$$a_n = \ldots$$

Example 3

Solving recurrence relation:

$$a_n = a_{n-1} + n,\ n \geq 1;\ \ a_1 = 2$$

Solution:

Then, we have the method for solving linear nonhomogeneous recurrence relation:

1) Find the solution $h(n)$ of linear homogeneous part.
2) Find particular solution $p(n)$ of linear nonhomogeneous recurrence relation.
3) The solution of linear nonhomogeneous recurrence relation is of the form: $a_n = h(n) + p(n)$
4) Determine the constants from equations obtained by initial conditions

$$\underbrace{a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}}_{\text{Linear homogeneous recurrence relation}} + \underbrace{F(n)}_{\text{Nonhomogeneous term}}$$

(1) Linear homogeneous part: $a_n = a_{n-1}$

Characteristic equation $r - 1 = 0$ has the root $r = 1$

➔ The solution of linear homogeneous part is: $h(n) = c_1 1^n$

(2) Nonhomogeneous part $F(n) = n \times 1^n$, and 1 is the root of multiplicities 1, thus the particular solution is of the form

$$p(n) = n^1(C_2 + C_3 n)1^n$$

If $F(n) = G(n) \times s^n$ where $G(n)$ is polynomial in n, and s is a constant and equal to characteristic root with multiplicities m, then particular solution $p(n)$ is of the form $\boldsymbol{n^m \times Q(n) \times s^n}$ where $Q(n)$ is of the form $G(n)$

Substitute to the recurrence relation, we have:

$$n\,(C_2 + C_3 n) = (n\text{-}1)\,[C_2 + C_3(n\text{-}1)] + n$$

Thus, $C_2 = ½$ and $C_3 = ½$. The particular solution is

$$p(n) = (n+1)n/2$$

Example 3

Solving recurrence relation:

$$a_n = a_{n\text{-}1} + n,\ n \geq 1;\ \ a_1 = 2$$

Then, we have the method for solving linear nonhomogeneous recurrence relation:

1) Find the solution $h(n)$ of linear homogeneous part.
2) Find particular solution $p(n)$ of linear nonhomogeneous recurrence relation.
3) The solution of linear nonhomogeneous recurrence relation is of the form: $a_n = h(n) + p(n)$
4) Determine the constants from equations obtained by initial conditions

$$\underbrace{a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}}_{\text{Linear homogeneous recurrence relation}} + \overbrace{F(n)}^{\text{Nonhomogeneous term}}$$

(3) The solution of linear nonhomogeneous recurrence relation is

$$a_n = c_1 + (n+1)n/2$$

(4) The constant c_1 is determined by using initial condition:

$$a_1 = c_1 + 1 = 2$$

➔ $c_1 = 1$

The solution of linear nonhomogeneous recurrence relation is

$$a_n = 1 + (n+1)n/2,\ n \geq 1$$

Example 4

Solving the recurrence relation: $a_n = 6a_{n-1} - 9a_{n-2} + F(n)$

Characteristic equation: $r^2 - 6r + 9 = (r - 3)^2 = 0$

➔ Characteristic root with multiplicities 2: $r = 3$

Consider some cases $F(n) = G(n)s^n$ where $G(n)$ is polynomial of n:

- (1) Assume $F(n) = n^2 2^n$
  - As: $F(n) = G(n)s^n = n^2 2^n$

  ➔ Degree of $G(n)$ is $t = 2$, $s = 2$ is not the characteristic root
  - Thus, the particular solution is of the form:

$$(p_t n^t + p_{t-1} n^{t-1} + \ldots + p_0)s^n$$
$$= (p_2 n^2 + p_1 n + p_0)2^n$$

If $F(n) = G(n) \times s^n$ where $G(n)$ is polynomial in n, and s is a constant and not equal to characteristic root, then particular solution $p(n)$ is of the form $F(n)$

- (2) Assume $F(n) = n^2 3^n$
  - As $F(n) = G(n)s^n = n^2 3^n$

  ➔ Degree of $G(n)$ is $t = 2$, $s = 3$ is characteristic root with multiplicities 2
  - Thus, the particular solution is of the form:

$$n^m(p_t n^t + p_{t-1} n^{t-1} + \ldots + p_0)s^n$$
$$= n^2(p_2 n^2 + p_1 n^1 + p_0)3^n$$

If $F(n) = G(n) \times s^n$ where $G(n)$ is polynomial in n, and s is a constant and equal to characteristic root with multiplicities m, then particular solution $p(n)$ is of the form $\boldsymbol{n^m \times Q(n) \times s^n}$ where $Q(n)$ is of the form $G(n)$

Example 4

Solving the recurrence relation: $a_n = 6a_{n-1} - 9a_{n-2} + F(n)$

Characteristic equation: $r^2 - 6r + 9 = (r - 3)^2 = 0$

➔ Characteristic root with multiplicities 2: $r = 3$

Consider some cases $F(n) = G(n)s^n$ where $G(n)$ is polynomial of n:

- (3) Assume $F(n) = 3^n$
  - As: $F(n) = G(n)s^n = 1 \cdot 3^n$

    ➔ Degree of $G(n)$ is $t = 0$, $s = 3$ is characteristic root with multiplicities 2
  - Thus, the particular solution is of the form:

    $n^m(p_t n^t + p_{t-1} n^{t-1} + \ldots + p_0)s^n$

    $= n^2(p_0)3^n$
- (4) Assume $F(n) = n3^n$
  - As $F(n) = G(n)s^n = n3^n$

    ➔ Degree of $G(n)$ is $t = 1$, $s = 3$ is characteristic root with multiplicities 2
  - Thus, the particular solution is of the form:

    $n^m(p_t n^t + p_{t-1} n^{t-1} + \ldots + p_0)s^n$

    $= n^2(p_1 n^1 + p_0)3^n$

If $F(n) = G(n) \times s^n$ where $G(n)$ is polynomial in n, and s is a constant and equal to characteristic root with multiplicities m, then particular solution $p(n)$ is of the form $\boldsymbol{n^m \times Q(n) \times s^n}$ where $Q(n)$ is of the form $G(n)$

Example 5

Solving the recurrence relation:

$$a_n = a_{n\text{-}1} + a_{n\text{-}2} + 2^n \quad n \geq 2;$$
$$a_0 = 1,\ a_1 = 2,\ a_2 = 7.$$

Then, we have the method for solving linear nonhomogeneous recurrence relation:

1) Find the solution $h(n)$ of linear homogeneous part.
2) Find particular solution $p(n)$ of linear nonhomogeneous recurrence relation.
3) The solution of linear nonhomogeneous recurrence relation is of the form: $a_n = h(n) + p(n)$
4) Determine the constants from equations obtained by initial conditions

$$\underbrace{a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}}_{\text{Linear homogeneous recurrence relation}} + \underbrace{F(n)}_{\text{Nonhomogeneous term}}$$

Recurrence tree method

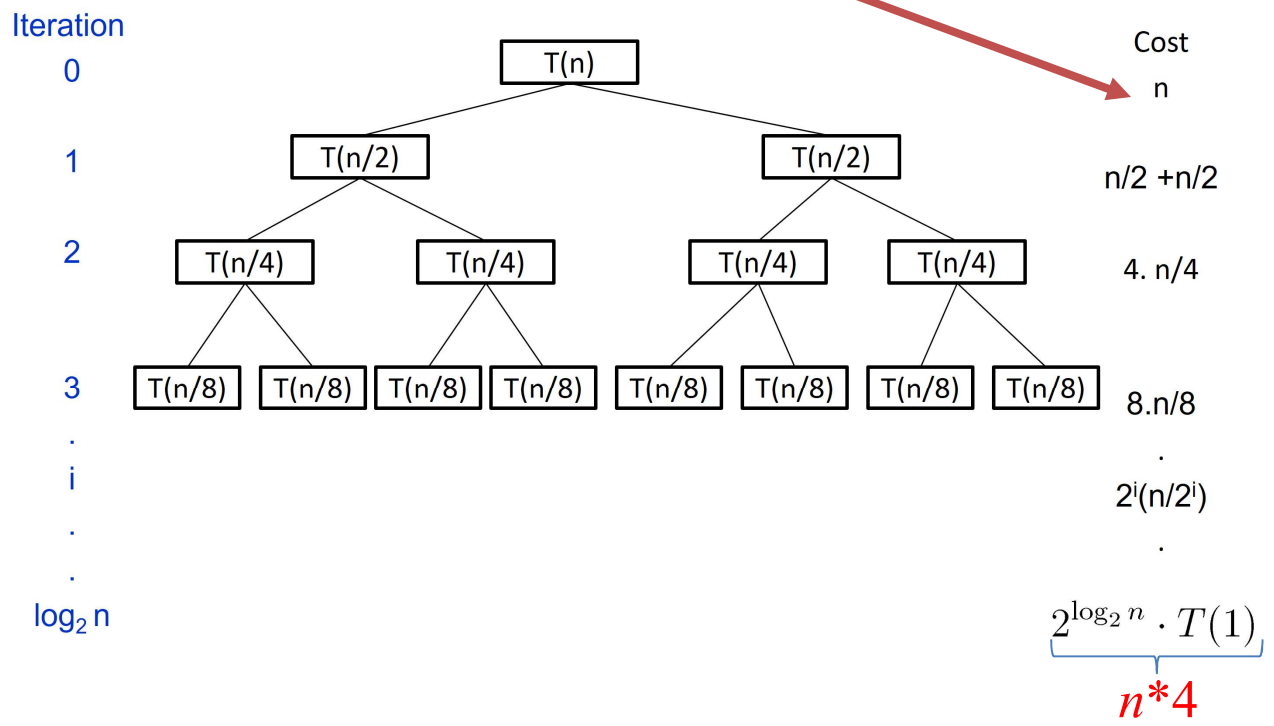
In this method, we draw a recurrence tree :

- Each node of the tree corresponds to an input data function. As going down the tree, the size of the input data will decrease.
- The last level of the tree corresponds to the smallest input data size.

We calculate the time taken by every level of tree, and finally, we sum the work done at all levels.

Example 1: Solving the recurrence relation:

$$T(n) = 2T(n/2) + n, \quad T(1) = 4$$



- The last level $\log_2 n$ has the value = $4n$, thus we have:

$$T(n) = 4n + \Sigma_{i=0}^{(\log_2 n - 1)} 2^i \frac{n}{2^i} = n(\log n) + 4n$$

Example 2: Performance of Recursive Binary Search

```
int binsearch(int low, int high, int A[], int key)
{
   if (low <= high)
   {
       mid = (low + high) / 2;
       if (A[mid]==key) return mid;
       else if (key < S[mid])
           return binsearch(low, mid-1, A, key);
       else
           return binsearch(mid+1, high, A, key);
    }
    else return -1;
}
```

➔ binsearch(0, n-1, A, key);

Let T(n): the number of times that binsearch is called in the worst case when array A has n elements

- $T(0) = 1$
- $T(1) = 2$
- $T(2) = T(1) + 1 = 3$
- $T(4) = T(2) + 1 = 4$
- $T(8) = T(4) + 1 = 4 + 1 = 5$
- $T(n) = T(n/2) + 1$

Recurrence relation:

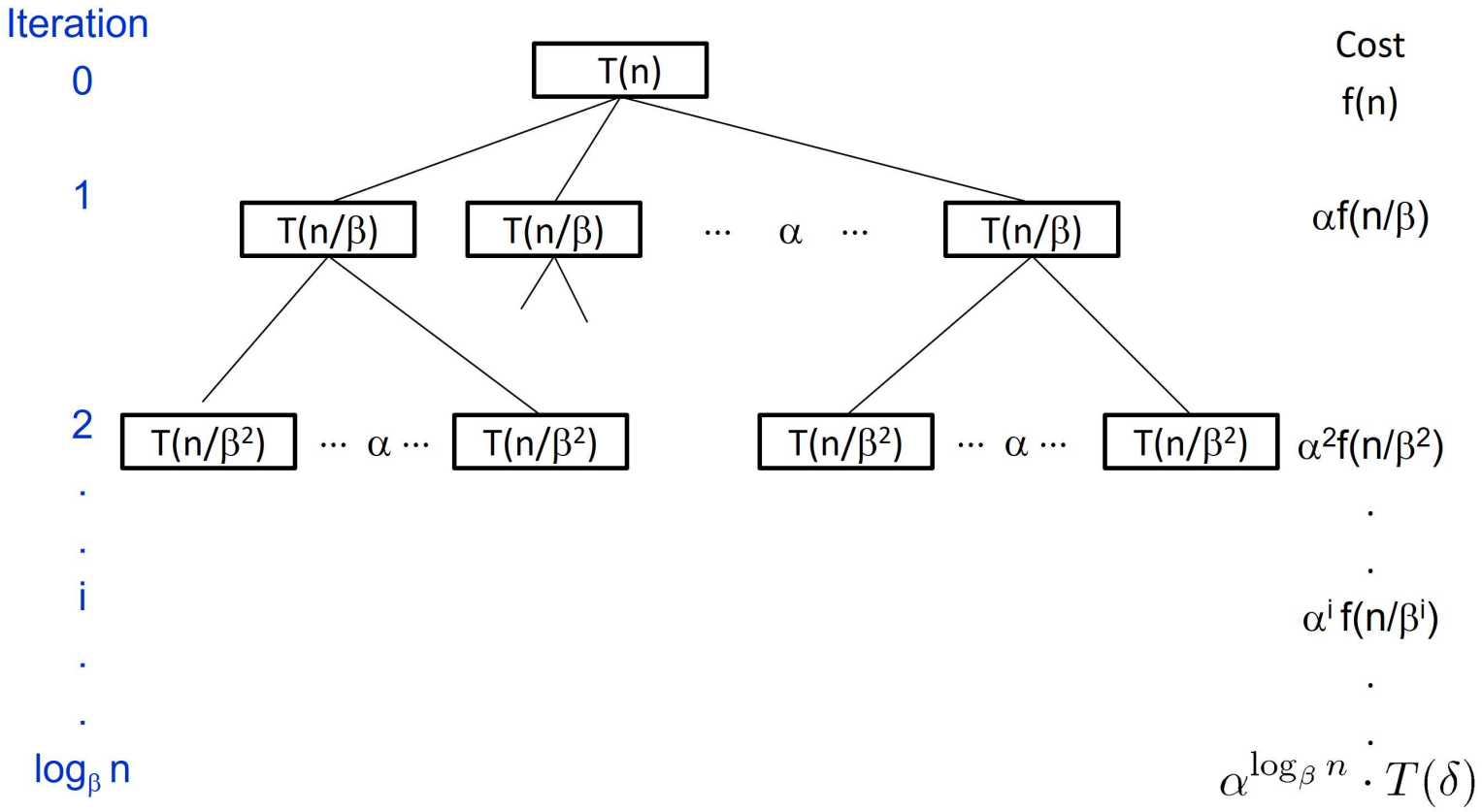
$T(n) = T(n/2) + 1$

$T(0) = 1$

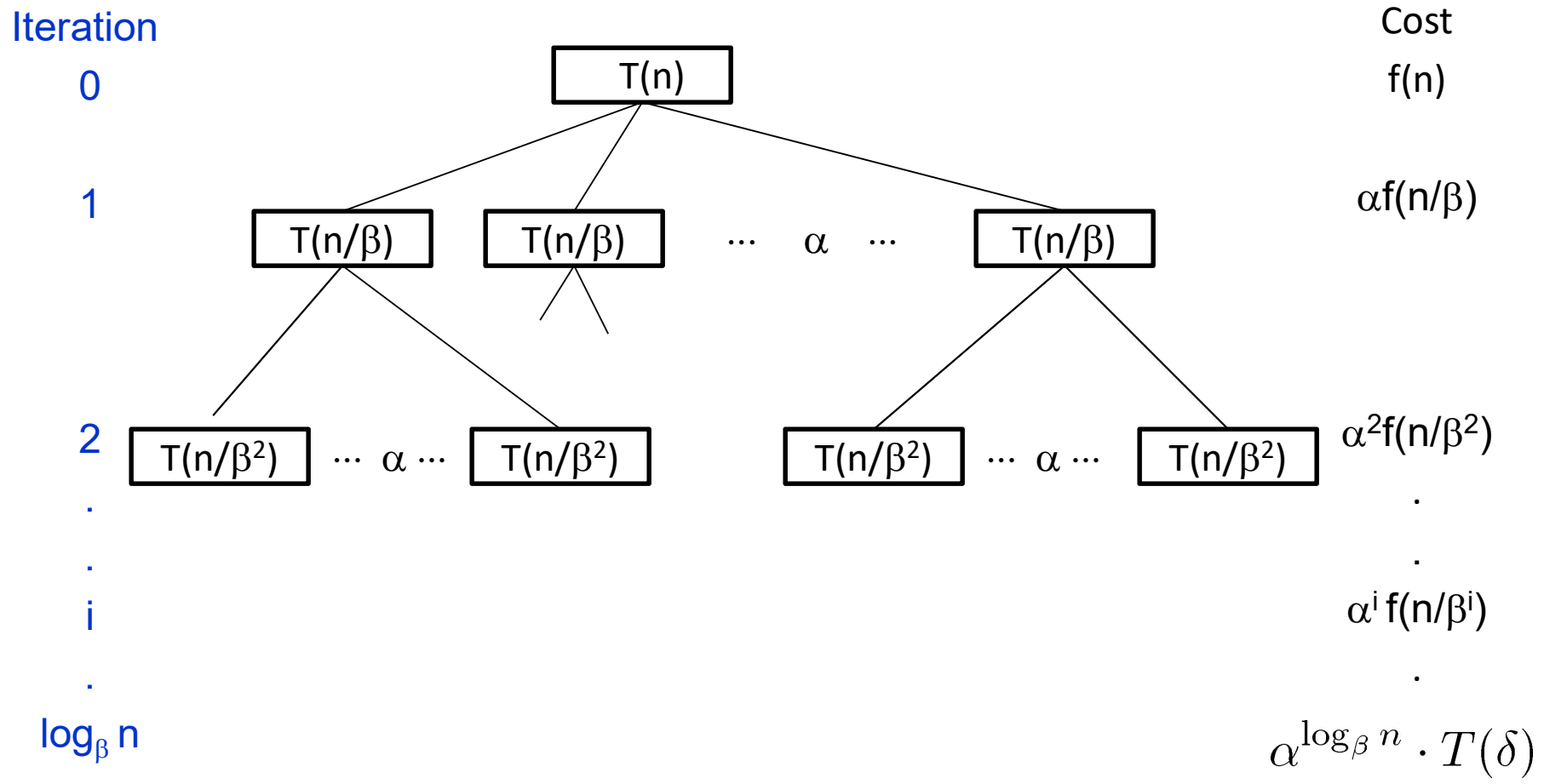
$T(1) = 2$

Example 3: Solving the recurrence relation

$T(n) = \alpha T(n/\beta) + f(n), \quad T(\delta) = c$



Example 3: Solving the recurrence relation $T(n) = \alpha T(n/\beta) + f(n), \quad T(\delta) = c$



- The value of T(n) is the sum of the cost of all level on the tree:

$$T(n) = \alpha^{log_\beta n} T(\delta) + \sum_{i=0}^{(log_\beta n)-1} \alpha^i f(\frac{n}{\beta^i})$$

2.5. Recurrence relations and generating functions

2.5.1. Recurrence relations

2.5.2. Generating functions

2.5.2. Generating functions

Generating functions are a tool to solve a wide variety of counting problems and recurrence relations.

Example 1:

How many ways to give 12 oranges for three children: A, B and C such that:
A gets at least four, and B and C gets at least two, but C gets no more than five

***a*, *b* , *c* are the number of oranges that A, B and C gets, respectively**

Find the number of integer solutions to

$$a + b + c = 12 \text{ where } a \geq 4, b \geq 2, 2 \leq c \leq 5$$

2.5.2. Generating functions

Example 1:

Find the number of integer solutions to
$a + b + c = 12$ where $a \geq 4, b \geq 2, 2 \leq c \leq 5$

Let

a: $a(x) = x^4 + x^5 + x^6 + x^7 + x^8$ (since $b + c \geq 4$ ➔ $a \leq 8$)

b : $b(x) = x^2 + x^3 + x^4 + x^5 + x^6$ (since $a + c \geq 6$ ➔ $b \leq 6$)

c : $c(x) = x^2 + x^3 + x^4 + x^5$ (since $2 \leq c \leq 5$)

n=12

=???

The coefficient of x^{12} in $g(x) = a(x) \times b(x) \times c(x)$

$=(x^4+ x^5+ x^6+ x^7+ x^8)\ (x^2+ x^3+x^4+ x^5+ x^6)(x^2+ x^3+x^4+ x^5)$

is the solution to the problem

Why???

$g(x)$ is called a *generating function*.

Assume x^a, x^b, x^c are terms derived from $(x^4+ x^5+ x^6+ x^7+ x^8)$, $(x^2+ x^3+x^4+ x^5+ x^6)$, $(x^2+ x^3+x^4+ x^5)$ respectively when developing the right-hand side ➔ $4 \leq a \leq 8$, $2 \leq b \leq 6$, $2 \leq c \leq 5$

When developing the right-hand side, these three terms give us the term x^n where $n = a + b + c$:

$g(x) = (x^4+ x^5+ x^6+ x^7+ x^8)\ (x^2+ x^3+x^4+ x^5+ x^6)(x^2+ x^3+x^4+ x^5)$

$= \ldots + Kx^n + \ldots$

➔ The coefficient of x^n in $f(x)$ is the number of positive integer solutions to

$a + b + c = n$ where $4 \leq a \leq 8$, $2 \leq b \leq 6$, $2 \leq c \leq 5$

2.5.2. Generating functions

Example 1:

Find the number of integer solutions to
$a + b + c = 12$ where $a \geq 4$, $b \geq 2$, $2 \leq c \leq 5$

Let

a: $a(x) = x^4 + x^5 + x^6 + x^7 + x^8$ (since $b + c \geq 4$ ➔ $a \leq 8$)

b : $b(x) = x^2 + x^3 + x^4 + x^5 + x^6$ (since $a + c \geq 6$ ➔ $b \leq 6$)

c : $c(x) = x^2 + x^3 + x^4 + x^5$ (since $2 \leq c \leq 5$)

The coefficient of x^{12} in $g(x) = a(x) \times b(x) \times c(x)$

$=(x^4+ x^5+ x^6+ x^7+ x^8)\ (x^2+ x^3+x^4+ x^5+ x^6)(x^2+ x^3+x^4+ x^5)$

which is 14, is the solution

$g(x)$ is called a *generating function*.

1. $x^4 * x^3 * x^5$
2. $x^4 * x^4 * x^4$
3. $x^4 * x^5 * x^3$
4. $x^4 * x^6 * x^2$
5. $x^5 * x^2 * x^5$
6. $x^5 * x^3 * x^4$
7. $x^5 * x^4 * x^3$
8. $x^5 * x^5 * x^2$
9. $x^6 * x^2 * x^4$
10. $x^6 * x^3 * x^3$
11. $x^6 * x^4 * x^2$
12. $x^7 * x^2 * x^3$
13. $x^7 * x^3 * x^2$
14. $x^8 * x^2 * x^2$

2.5.2. Generating functions

Example 2: Four kinds of jelly beans: Red, Green, White, Black. In how many ways can we select 24 jelly beans so that we have an even number of white beans and at least six black ones?

- White: $1+ x^2+ x^4+\ldots+ x^{16} + x^{18}$
- Black: $x^6+ x^7+\ldots+ x^{16}+ x^{18}$
- Red (green): $1+ x^1+ x^2+\ldots+ x^{16}+ x^{18}$

$g(x)=(1+ x^2+ x^4+\ldots+ x^{16}+ x^{18})(x^6+ x^7+\ldots+ x^{16}+ x^{18})\ (1+ x^1+ x^2+\ldots+ x^{16}+ x^{18})^2$

The coefficient of x^{24} is the solution.

Example 3: How many nonnegative integer solutions are there for

$$c_1+c_2+c_3+c_4=25?$$

- $g(x)=(1+ x^1+ x^2+\ldots+ x^{24}+ x^{25})^4$
- The coefficient of x^{25} is the solution

2.5.2. Generating functions

Definition: Let $a_0, a_1, a_2, \ldots$ be a sequence of real numbers. The function

$$g(x) = a_0 + a_1 x + \ldots + a_k x^k + \ldots = \sum_{i=0}^{\infty} a_i x^i$$

is called the *generating function* for the given sequence.

- A finite sequence

  $a_0, a_1, a_2, \ldots, a_n$

can be regarded as the infinite sequence

$a_0, a_1, a_2, \ldots, a_n, 0, 0, \ldots$ [set all terms higher than n to 0]

and its generating function

$$g(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

is a polynomial.

Example 4: For the binomial coefficients we already know that:

$$(x+y)^n = \sum_{k=0}^{n} C(n,k) x^k y^{n-k} \Rightarrow (x+1)^n = \sum_{k=0}^{n} C(n,k) x^k = g(x)$$

$(x+1)^n$ is the generating function for the sequence

$C(n, 0), C(n, 1), \ldots, C(n, k), \ldots, C(n, n), 0, 0, 0 \ldots$

2.5.2. Generating functions

Definition: Let $a_0, a_1, a_2, \ldots$ be a sequence of real numbers. The function

$$g(x) = a_0 + a_1 x + \ldots + a_k x^k + \ldots = \sum_{i=0}^{\infty} a_i x^i$$

is called the *generating function* for the given sequence.

Some useful generating function:

Ex1: $(1 - x^{n+1}) = (1 - x)(1 + x + x^2 + x^3 + \ldots + x^n)$.

So $(1-x^{n+1})/(1-x) = 1 + x + x^2 + x^3 + \ldots + x^n$,

➔ $(1-x^{n+1})/(1-x)$ is the generating function for the sequence:

1, 1, 1, …, 1, 0, 0, 0, … . (where the first n+1 terms are 1)

Ex2: From Ex1, If $n \to \infty$ and $|x| < 1$, then $1 = (1 - x)(1 + x + x^2 + x^3 + \ldots)$.

So, $1/(1-x) = 1 + x + x^2 + x^3 + \ldots$

➔ $1/(1-x)$ where $|x| < 1$ is the generating function for the sequence:

1, 1, 1, …, 1, … .

Ex3: From Ex2, set $x = ay$ ➔ $|ay| < 1$, then

$1/(1-x) = 1/(1-ay) = 1 + ay + (ay)^2 + (ay)^3 + \ldots$.

➔ $1/(1-ay)$ where $|ay| < 1$ is the generating function for the sequence $1, a, a^2, a^3, \ldots$

2.5.2. Generating functions

- If we have two generating functions $F(x)$ and $G(x)$, we define the sum and product as follows:

$$F(x)=\sum_{k=0}^{\infty}a_k x^k \qquad G(x)=\sum_{k=0}^{\infty}b_k x^k$$

$$F(x)+G(x)=\sum_{k=0}^{\infty}\left(a_k+b_k\right)x^k$$

$$F(x)G(x)=\sum_{k=0}^{\infty}\left(\sum_{j=0}^{k}a_j b_{k-j}\right)x^k$$

Match all terms with equal powers in x.

Ex4: Example why multiplying generating functions is useful:

$$\frac{1}{(1-x)^2}=\frac{1}{1-x}\frac{1}{1-x}=(1+x+x^2+x^3+\ldots)(1+x+x^2+x^3+\ldots) \quad \text{where } |x|<1$$

$$=\left[\sum_{k=0}^{\infty}(1)x^k\right]*\left[\sum_{k=0}^{\infty}(1)x^k\right]=\sum_{k=0}^{\infty}\left(\sum_{j=0}^{k}1\right)x^k=\sum_{k=0}^{\infty}(k+1)x^k$$

➔ $1/(1-x)^2$ is the generating function for the sequence 1,2,3,4,….

2.5.2. Generating functions

Another way to show that $1/(1-x)^2$ is the generating function of the sequence 1, 2,3, 4…:

$$\frac{1}{(1-x)} = 1 + x + x^2 + x^3 + \cdots$$

$$\rightarrow \frac{d}{dx}\left(\frac{1}{1-x}\right) = \frac{d}{dx}\left(1 + x + x^2 + x^3 + \cdots\right)$$

$$\rightarrow \frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + \cdots$$

→ $1/(1-x)^2$ is the generating function for the sequence 1,2,3,4,….

Ex5: Show that $x/(1-x)^2$ is the generating function for the sequence 0,1,2,3,….

We have

$$\frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + \cdots$$

$$\rightarrow \frac{x}{(1-x)^2} = x(1 + 2x + 3x^2 + \cdots)$$

$$= 0x^0 + x^1 + 2x^2 + 3x^3 + \ldots.$$

2.5.2. Generating functions

Ex5: Show that $x/(1-x)^2$ is the generating function for the sequence 0,1,2,3,….

We have

$$\frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + \cdots$$

$$\rightarrow \frac{x}{(1-x)^2} = x(1 + 2x + 3x^2 + \cdots)$$

$$= 0x^0 + x^1 + 2x^2 + 3x^3 + \ldots.$$

Ex6: Show that $(x+1)/(1-x)^3$ is the generating function for the sequence $1^2,2^2,3^2,4^2,\ldots$.

From Ex5 we have:

$$\frac{x}{(1-x)^2} = x + 2x^2 + 3x^3 + \cdots$$

$$\rightarrow \frac{d}{dx}\left(\frac{x}{(1-x)^2}\right) = \frac{d}{dx}(x + 2x^2 + 3x^3 + \cdots)$$

$$\rightarrow \frac{x+1}{(1-x)^3} = 1 + 2^2x + 3^2x^2 + \cdots$$

2.5.2. Generating functions

Ex6: Show that $(x+1)/(1-x)^3$ is the generating function for the sequence $1^2,2^2,3^2,4^2,\ldots$.

From Ex5 we have:

$$\frac{x}{(1-x)^2} = x + 2x^2 + 3x^3 + \cdots$$

$$\rightarrow \frac{d}{dx}\left(\frac{x}{(1-x)^2}\right) = \frac{d}{dx}\left(x + 2x^2 + 3x^3 + \cdots\right)$$

$$\rightarrow \frac{x+1}{(1-x)^3} = 1 + 2^2x + 3^2x^2 + \cdots$$

Ex7: Show that $x(x+1)/(1-x)^3$ is the generating function for the sequence 0^2, 1^2, 2^2, 3^2, 4^2, ….

2.5.2. Generating functions

Generating functions are a tool to solve a wide variety of counting problems and **recurrence relations**.

Definition: Let a_0, a_1, a_2, ... be a sequence of real numbers. The function

$$g(x) = a_0 + a_1 x + \ldots + a_k x^k + \ldots = \sum_{i=0}^{\infty} a_i x^i$$

is called the *generating function* for the given sequence.

Example 1: Solve the recurrence relation $a_n - 3a_{n-1} = n$, $n \geq 1$, $a_0 = 1$

Let $g(x) = \sum_{n=0}^{\infty} a_n x^n$ be the generating function for $a_0, a_1, \ldots, a_n$

$$\underbrace{\sum_{n=1}^{\infty} a_n x^n}_{[g(x) - a_0]} - 3 \underbrace{\sum_{n=1}^{\infty} a_{n-1} x^n}_{x \sum_{n=1}^{\infty} a_{n-1} x^{n-1}} = \underbrace{\sum_{n=1}^{\infty} n x^n}_{x + 2x^2 + 3x^3 + \cdots}$$

$$[g(x) - 1] - 3x g(x) = x + 2x^2 + 3x^3 + \cdots$$

Remember that (see Ex5) $\frac{x}{(1-x)^2} = x + 2x^2 + 3x^3 + \cdots$

2.5.2. Generating functions

Example 1: Solve the recurrence relation $a_n - 3a_{n-1} = n$, $n \geq 1$, $a_0 = 1$

$$g(x) - 1 - 3xg(x) = \frac{x}{(1-x)^2}$$

$$g(x) = \frac{1}{1-3x} + \frac{x}{(1-3x)(1-x)^2}$$

$$g(x) = \frac{1}{1-3x} + \frac{A}{(1-3x)} + \frac{B}{(1-x)} + \frac{C}{(1-x)^2}$$

$$\Rightarrow A = \frac{3}{4}; B = -\frac{1}{4}; C = -\frac{1}{2};$$

$$g(x) = \frac{1}{1-3x} + \frac{3/4}{(1-3x)} + \frac{-1/4}{(1-x)} + \frac{-1/2}{(1-x)^2}$$

$$g(x) = \frac{7/4}{(1-3x)} + \frac{-1/4}{(1-x)} + \frac{-1/2}{(1-x)^2}$$

$$g(x) = a_0 + a_1 x + \ldots + a_k x^k + \ldots = \sum_{i=0}^{\infty} a_i x^i$$

We find a_n by determining the coefficient of x^n in $g(x)$

➔ determining the coefficient of x^n in each of the three summands

2.5.2. Generating functions

Example 1: Solve the recurrence relation $a_n - 3a_{n-1} = n$, $n \geq 1$, $a_0 = 1$

$$g(x) = \frac{7/4}{(1-3x)} + \frac{-1/4}{(1-x)} + \frac{-1/2}{(1-x)^2}$$

We find a_n by determining the coefficient of x^n in each of the three summands

1. $(7/4)/(1-3x) = (7/4)\,[1/(1-3x)]$

   $= (7/4)\,[1 + 3x + (3x)^2 + (3x)^3 + \ldots]$ See Ex3.

➔ The coefficient of x^n is $(7/4)3^n$

2. $(-1/4)/(1-x) = (-1/4)\,[1/(1-x)]$

   $= (-1/4)\,[1 + x + (x)^2 + (x)^3 + \ldots]$ See Ex2.

➔ The coefficient of x^n is $(-1/4)$

3. $(-1/2)/(1-x)^2 = (-1/2)\,[1/(1-x)^2]$

   $= (-1/2)\,[1 + 2x + 3(x)^2 + 4(x)^3 + \ldots]$ See Ex4.

➔ The coefficient of x^n is $(-1/2)(n+1)$

Therefore, $a_n = (7/4)\,3^n - (1/2)n - 3/4$, $n \geq 0$

2.5.2. Generating functions

Example 2: Tower of Hanoi

Solve the recurrence relation $a_n = 2\,a_{n-1} + 1$, $n \geq 1$, $a_0 = 0$

Let $g(x) = \sum_{n=0}^{\infty} a_n x^n$ be the generating function for $a_0, a_1, \ldots, a_n$

$$\sum_{n=1}^{\infty} a_n x^n = 2\sum_{n=1}^{\infty} a_{n-1} x^n + \sum_{n=1}^{\infty} x^n$$

$$[g(x) - a_0] = 2x\sum_{n=1}^{\infty} a_{n-1} x^{n-1} + \left(\sum_{n=0}^{\infty} x^n\right) - x^0$$

$$[g(x) - 0] = 2xg(x) + (1 + x + x^2 + x^3 + \cdots)\text{-}1$$

$$(1 - 2x)g(x) = (1 + x + x^2 + x^3 + \cdots)\text{-}1$$

Remember that (see Ex2) $\frac{1}{(1-x)} = 1 + x + x^2 + x^3 + \cdots$

2.5.2. Generating functions

Example 2: Tower of Hanoi

Solve the recurrence relation $a_n = 2\,a_{n-1} + 1$, $n \geq 1$, $a_0 = 0$

$$(1-2x)g(x) = (1 + x + x^2 + x^3 + \cdots)\text{-}1$$

Remember that (see Ex2) $\frac{1}{(1-x)} = 1 + x + x^2 + x^3 + \cdots$

$$(1-2x)g(x) = \frac{1}{(1-x)}\text{-}1 = \frac{x}{1-x}$$

$$g(x) = \frac{x}{(1-2x)(1-x)}$$

$$g(x) = \frac{A}{(1-2x)} + \frac{B}{(1-x)} \quad \Rightarrow A = 1; B = -1$$

2.5.2. Generating functions

Example 2: Tower of Hanoi

Solve the recurrence relation $a_n = 2\,a_{n-1} + 1$, $n \geq 1$, $a_0 = 0$

$$g(x) = \frac{1}{(1-2x)} + \frac{(-1)}{(1-x)}$$

We find a_n by determining the coefficient of x^n in each of the two summands

1. $(1)/(1\text{-}2x) = (1)\,[1/(1\text{-}2x)]$
   $= (1)\,[1 + 2x + (2x)^2 + (2x)^3 + \ldots.]$

See Ex3.

➔ The coefficient of x^n is $(1)2^n$

2. $(\text{-}1)/(1\text{-}x) = (\text{-}1)\,[1/(1\text{-}x)]$
   $= (\text{-}1)\,[1 + x + (x)^2 + (x)^3 + \ldots.]$

See Ex2.

➔ The coefficient of x^n is $(\text{-}1)$

Therefore, $a_n = 2^n - 1$, $n \geq 0$

2.5.2. Generating functions

Example 3: Solve the recursive relation $a_{n+2}-5a_{n+1}+6a_n=2$, $n\geq 0$, $a_0=3$, $a_1=7$

Let $g(x)=\sum_{n=0}^{\infty} a_n x^n$ be the generating function for $a_0, a_1, \ldots, a_n$

$$\sum_{n=0}^{\infty} a_{n+2}x^{n+2} - 5\sum_{n=0}^{\infty} a_{n+1}x^{n+2} + 6\sum_{n=0}^{\infty} a_n x^{n+2} = 2\sum_{n=0}^{\infty} x^{n+2}$$

$$\underbrace{\sum_{n=0}^{\infty} a_{n+2}x^{n+2}}_{[g(x)-a_1x-a_0]} \underbrace{- 5x\sum_{n=0}^{\infty} a_{n+1}x^{n+1}}_{-5x[g(x)-a_0]} \underbrace{+ 6x^2\sum_{n=0}^{\infty} a_n x^n}_{+6x^2\,[g(x)]} = \underbrace{2\,x^2\sum_{n=0}^{\infty} x^n}_{=2x^2\frac{1}{1-x}}$$

$$\left[1-5x+6x^2\right]g(x) = 3-8x+\frac{2x^2}{1-x} = \frac{3-11x+10x^2}{1-x}$$

2.5.2. Generating functions

Example 3: Solve the recursive relation $a_{n+2}-5a_{n+1}+6a_n=2$, $n\geq 0$, $a_0=3$, $a_1=7$

Let $g(x)=\sum_{n=0}^{\infty} a_n x^n$ be the generating function for $a_0, a_1, \ldots, a_n$

$$\left[1-5x+6x^2\right]g(x)=3-8x+\frac{2x^2}{1-x}=\frac{3-11x+10x^2}{1-x}$$

$$g(x)=\frac{3-11x+10x^2}{(1-x)[1-5x+6x^2]}=\frac{(3-5x)(1-2x)}{(1-x)(1-3x)(1-2x)}=\frac{(3-5x)}{(1-x)(1-3x)}$$

$$g(x)=\frac{(3-5x)}{(1-x)(1-3x)}=\frac{A}{(1-x)}+\frac{B}{(1-3x)} \Rightarrow A=1; B=2$$

$$g(x)=\frac{1}{(1-x)}+\frac{2}{(1-3x)}=2\sum_{n=0}^{\infty}(3x)^n+\sum_{n=0}^{\infty}(x)^n$$

Therefore, $a_n=2(3^n)+1$, $n\geq 0$