

Chapter 5 & 7

Counting

Objectives

- Some questions we can face:
 - “How can we determine the complexity of an algorithm?”
 - “How many cases possible are there to arrange n objects?”
 - “How can we generate input data from given initial data?”
 - A password consists of six characters. How many such password with some criteria?
 - How many bit strings of length n that do not have two consecutive 0s?

Contents

- The basics of Counting
- Recurrence relations
- Divide – and – Conquer Algorithms and recurrence relations

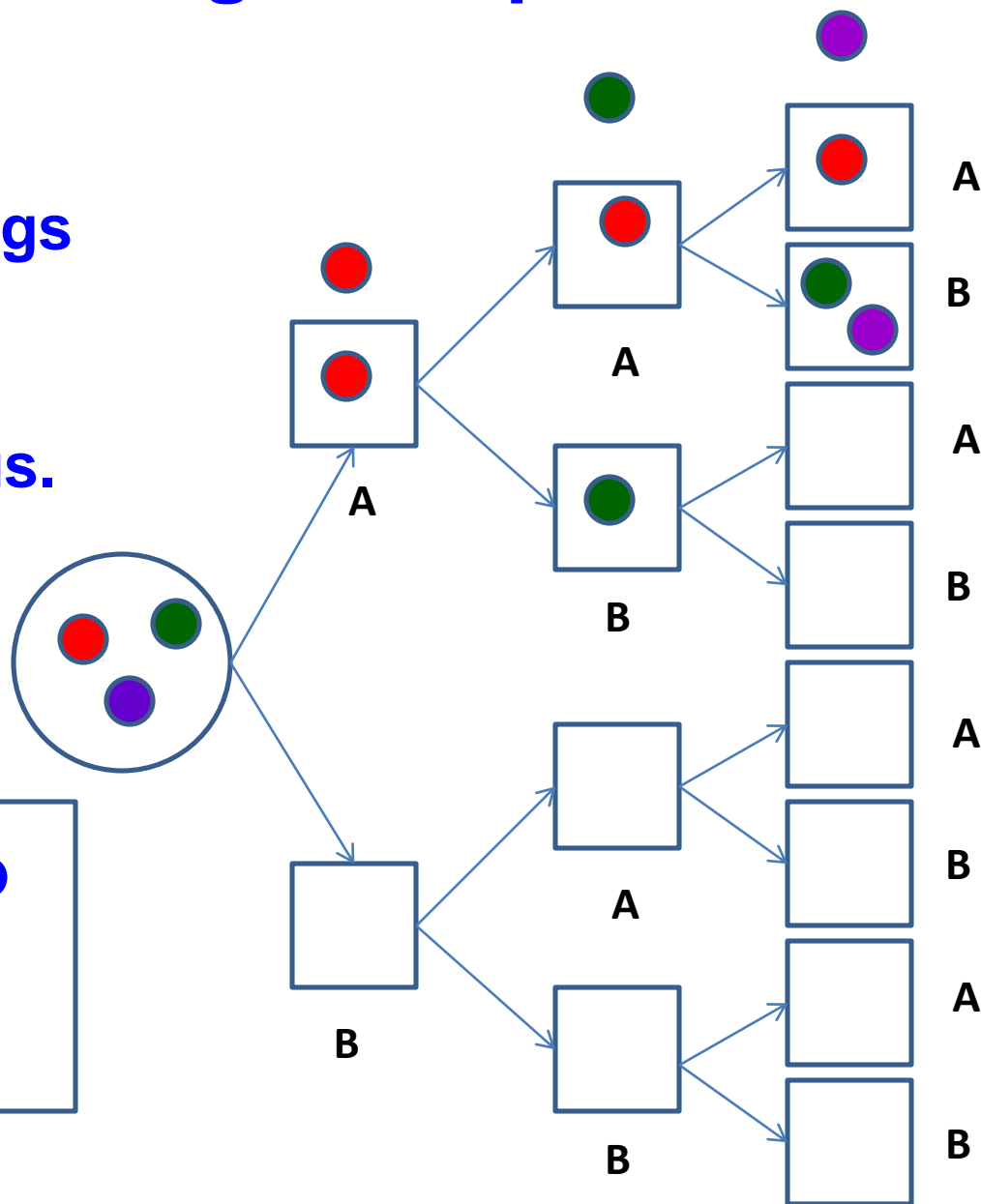
5.1- The Basics of Counting

- Basic Counting Principles : Product rule, sum rule, The Inclusion-Exclusion Principle
- Tree Diagrams

Basic Counting Principle

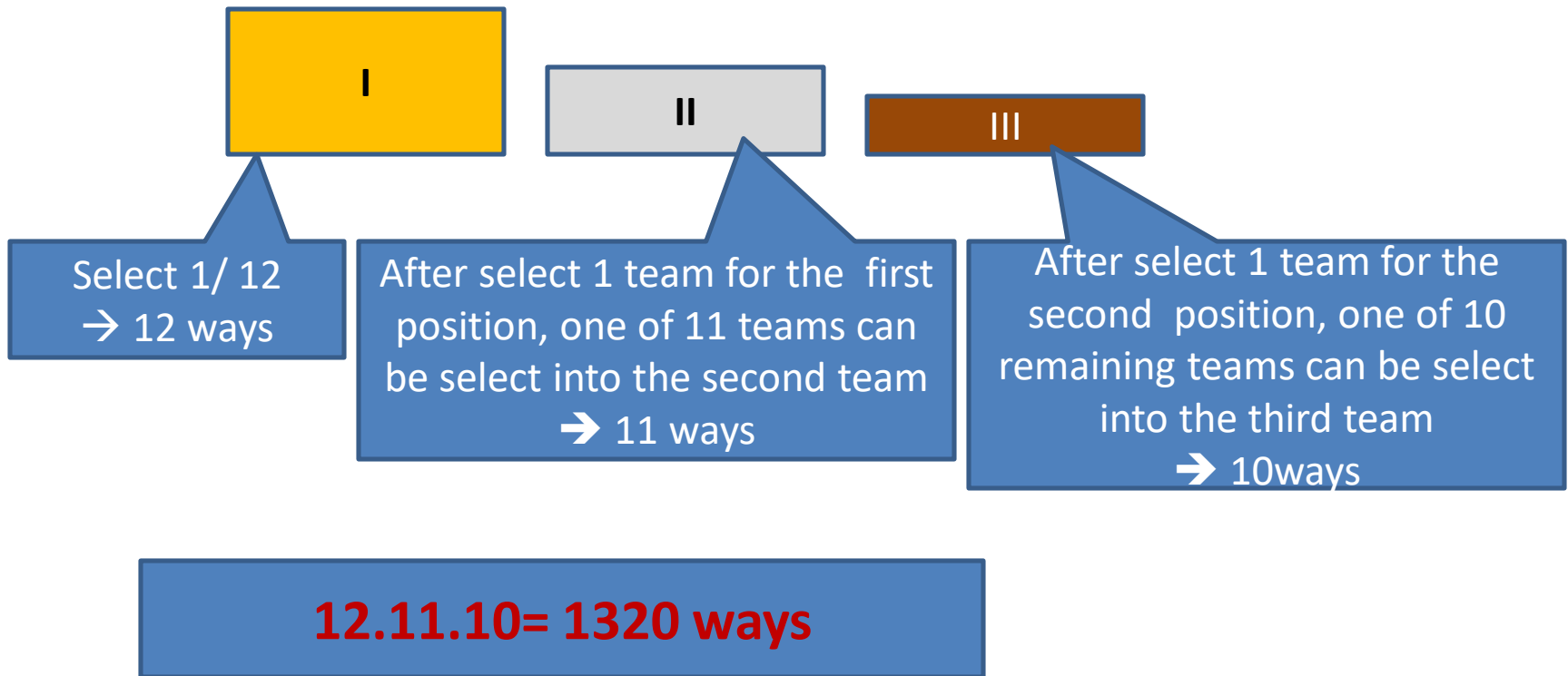
- Ex: Distribute **3** distinct balls to **2** bags
- We must do **3** times, each time we have **2** ways to select a bags.
- $2 \cdot 2 \cdot 2 = 2^3$ ways

There are r^n ways to distribute n distinct objects to r bags



Basic Counting Principle

- 12 teams. How many ways to select 3 teams for the first, second and third teams



Basic Counting Principle: Product rule

- Suppose that a procedure can be broken down into a sequence of two tasks. If there is n_1 ways to do the first task, there is n_2 ways to do the second task, then there are $n_1 n_2$ ways to do the procedure.
 - **Example 1:** 2 persons, 12 rooms. How many ways are there to assign different room to these two person?
12 ways can be applied to assign the first
11 ways can be applied to assign the second after the first assigned.
→ $12 \cdot 11 = 132$ ways.
- Examples 3, 4, 5, 6, 7, 8, 9: Refer to pages 336, 337

Basic Counting Principle: Sum rule

- If a task can be done **either** in one of n_1 ways **or** in one of n_2 ways, **where** none of the set of n_1 ways is the same as any of the set of n_2 ways, then there are $n_1 + n_2$ ways to do this task.
- **Example 11:** There are 37 faculties, 83 students. A person either a faculty or student can be choose to attend a committee. How many ways are there to select such person?

$$37 + 83 = 120 \text{ ways}$$

Examples 12, 13: Refer to page 339

Basic Counting Principles...

- **Example 14:** Name of a variable in BASIC language, an case insensitive language, is a string of 1 or 2 characters in which the first must be a character, the second may be an alphanumeric characters, and must be different from 5 two-character keywords . How many different variable names are there?

26 characters (a..z), 10 digits (0..9)

Name with 1 character: 26

Name with 2 character: $26 \cdot 36 - 5 = 931$

➔ There are 957 different names

Basic Counting Principles...

- **Example 16 – page 341:**Counting Internet Address
- Class A address: largest networks
- Class B address: medium-sized networks
- Class C address: smallest networks.

IPv4

© The McGraw-Hill Companies, Inc. all rights reserved.

Bit Number	0	1	2	3	4	8	16	24	31	
Class A	0	netid				hostid				
Class B	1	0	netid				hostid			
Class C	1	1	0	netid				hostid		
Class D	1	1	1	0	Multicast Address					
Class E	1	1	1	1	0	Address				

Basic Counting Principles:

The Inclusion-Exclusion Principle

Nguyên lý thêm vào rồi bỏ ra

- Suppose that a task can be done in n_1 or n_2 ways, but that some ways in the set of n_1 ways are the same as some ways in the set of n_2 ways.
- n_s : number of ways that are the same in two sets of ways.
- Total ways $n = n_1 + n_2 - n_s$

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

Example 17: How many 8-bit strings either start with 1 or end with 00?

$$2^7 + 2^6 - 2^5 = 160 - \text{Refer to page 342}$$

Basic Counting Principles: The Inclusion-Exclusion Principle

Example 18:

350 applications, in which

A1: 220 people majored in IT

A2: 147 people majored in business

A3: 51 people majored both in IT and business

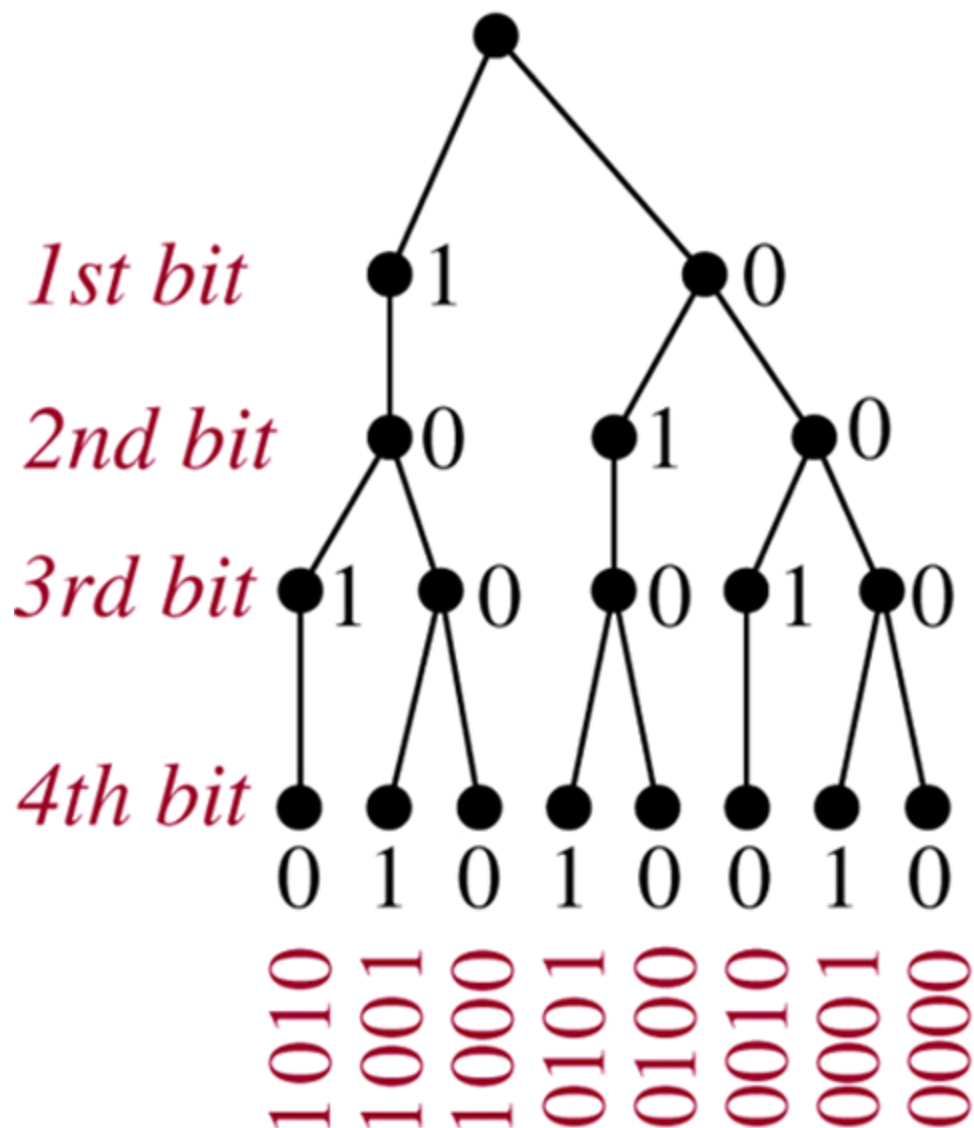
How many applicants majored **neither** in IT **nor** business?

Number of applicants who majored **either** IT **or** business =
 $(|A1| + |A2| - |A3|) = 220 + 147 - 51 = 316$

Result: $350 - 316 = 34$

Tree Diagrams

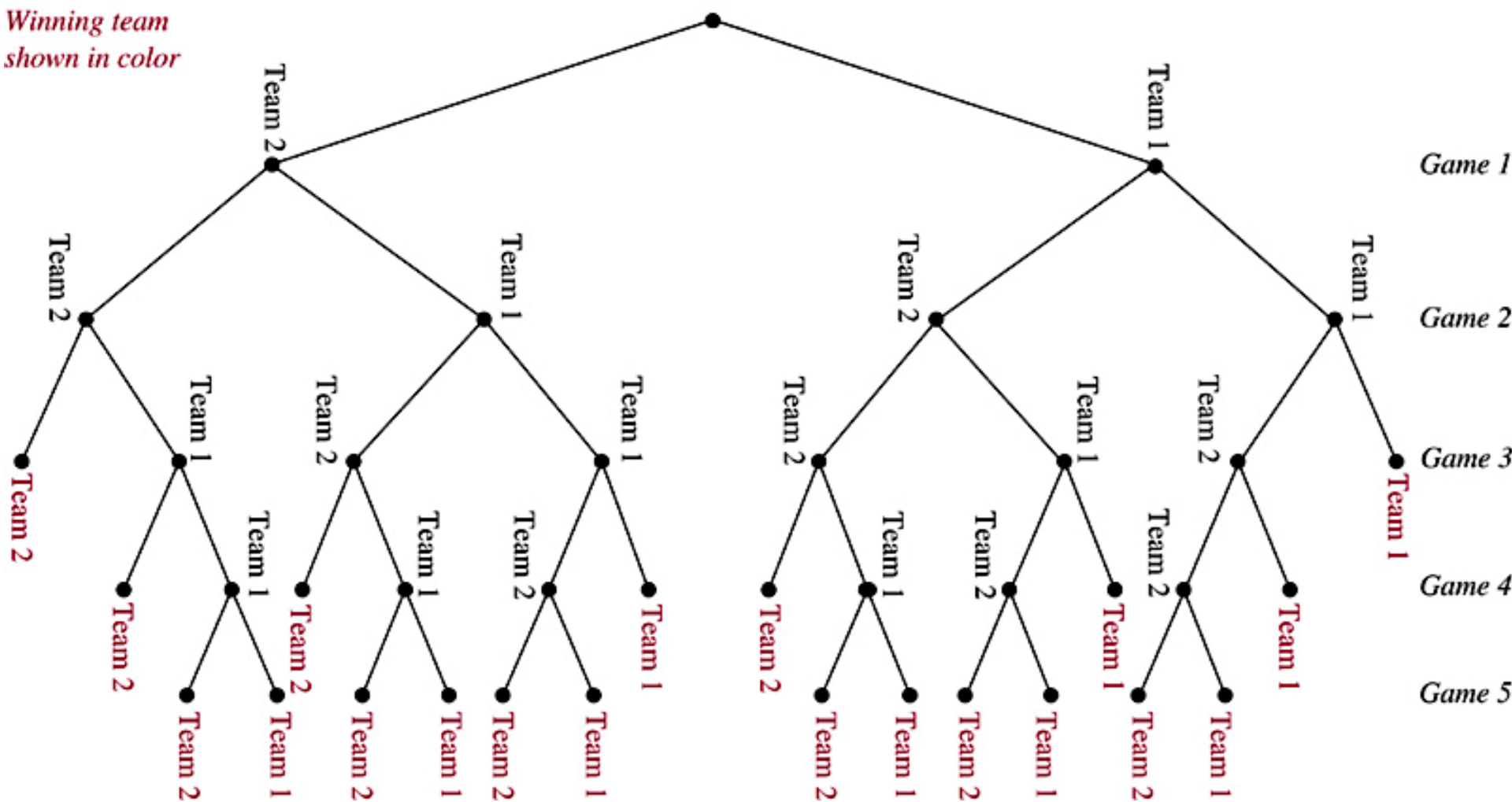
- Counting problem can be solved using **Tree Diagram**.
- Example:** How many 4-bit strings do not have two consecutive 1s?



Example 20: A playoff between two teams consists of at most 5 games. The first team that wins 3 games wins the playoff. In how many different ways can the playoff occur?

© The McGraw-Hill Companies, Inc. all rights reserved.

Winning team
shown in color



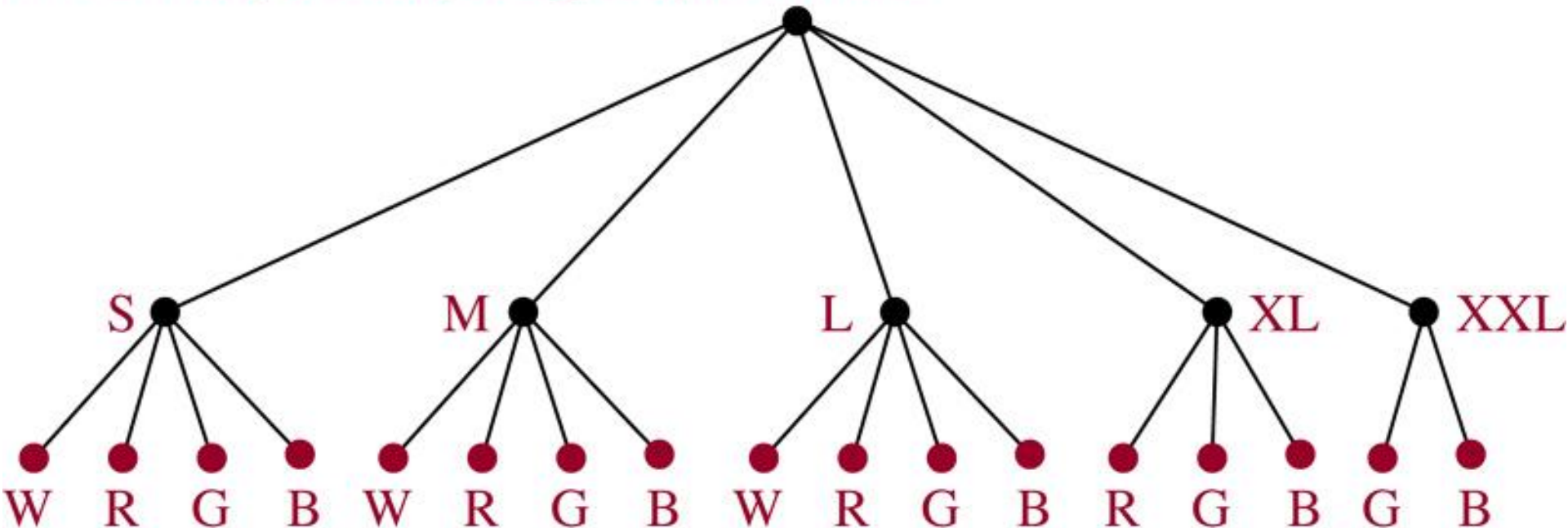
Example 22: T-Shirts come in 5 sizes: S, M, L, XL, XXL.

Each size comes in 4 colors, W, R, G, B excepts for XL, which comes only in R, G, B, and XXL, which comes only in G and B.

- How many different shirts a souvenir shop have to stock to have at least one of each available size and color of the T-shirt?

© The McGraw-Hill Companies, Inc. all rights reserved.

W = white, R = red, G = green, B = black



7.1. Recurrence Relations

- **Definition 1:** Recurrence relation on the sequence $\{a_n\}$:
 a_n is expressed in terms of one or more of previous items for all n with $n \geq n_0$
- $\{a_n\}$ is called a **solution** of recurrence relation.
- **Example 1:** $a_0=3, a_1=5 \quad a_n=a_{n-1} - a_{n-2}, n>1$
- **Example 2:** Determine whether $\{a_n\} = 3n, n \geq 0$ is a solution of the recurrence relation $a_n= 2a_{n-1} - a_{n-2}, n \geq 2$?
- **Initial conditions:** Terms that **precede the first** item where the recurrence takes effect.

Modeling with Recurrence Relations

- Problem \leftarrow Modeling with recurrence relation
- Example 3 (page 451): Compound interest

Problem – Lãi gộp:

P_0 : Initial deposit

r : interest rate per year

P_n : amount at the n^{th} year












$$P_n = P_{n-1} + rP_{n-1}$$

$$\Rightarrow P_n = (1+r)^n P_0$$

Fibonacci numbers of Leonardo Pisano

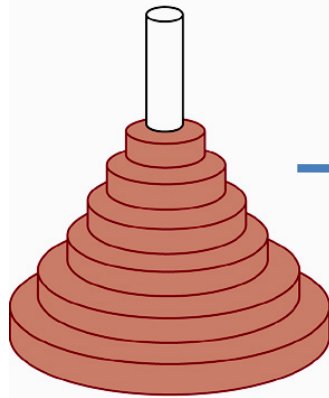
- Example 4: : $f_1=f_2=1;$ $f_n= f_{n-2} + f_{n-1}$

© The McGraw-Hill Companies, Inc. all rights reserved.

Reproducing pairs (at least two months old)	Young pairs (less than two months old)	Month	Reproducing pairs	Young pairs	Total pairs
		1	0	1	1
		2	0	1	1
		3	1	1	2
		4	1	2	3
		5	2	3	5
		6	3	5	8
					

The Tower of Hanoi Problem- Ex5, page 452

© The McGraw-Hill Companies, Inc. all rights reserved.



Peg 1



Peg 2



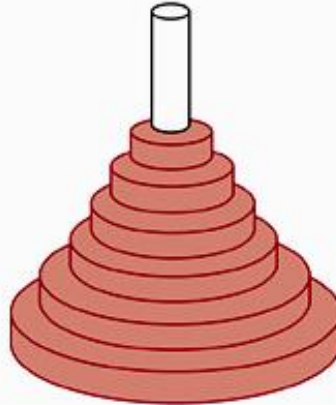
Peg 3

How many steps this problem is solved if there is n disks on the peg 1?

© The McGraw-Hill Companies, Inc. all rights reserved.



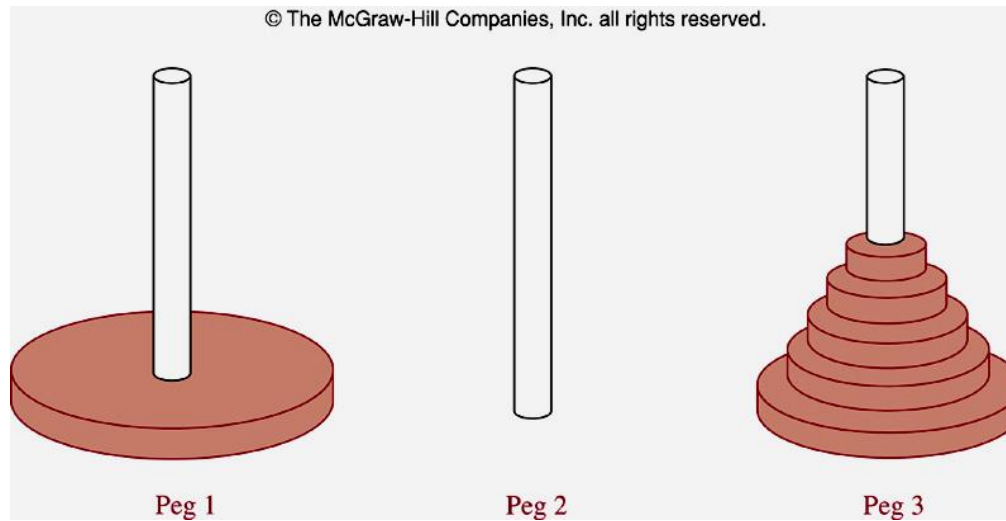
Peg 1



Peg 2



Peg 3



Let H_n is the number of moves needed to solve the Tower of Hanoi problem. We will set up a recurrence relation for the sequence H_n .

-Begin with n disk on peg 1.

-We transfer the top $n-1$ disk from peg1 to peg3 $\rightarrow H_{n-1}$ moves .

-We transfer the largest disk from peg1 to peg 2 $\rightarrow 1$ move

-We transfer $n-1$ disk from peg3 to peg 2

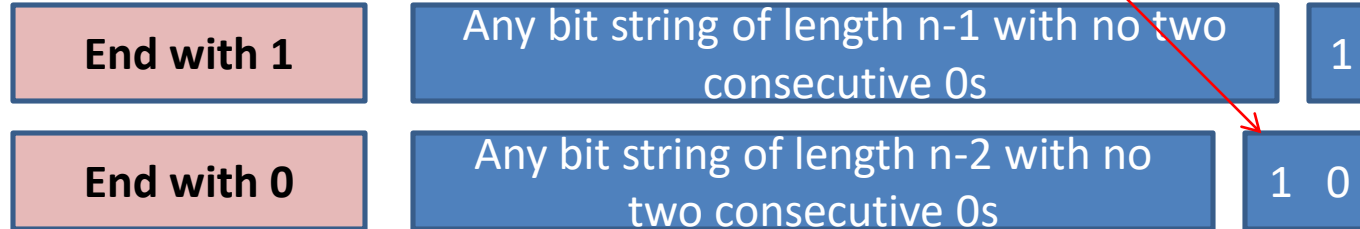
$$\rightarrow H_n = 2H_{n-1} + 1$$

$$H_1 = 1 \rightarrow H_n = 2^n - 1 \quad (\text{See page 453})$$

$n=64 \rightarrow 2^{64} - 1 = 18\,446\,744\,073\,709\,551\,615$. With 1 move/sec $\rightarrow 500$ billion years.

Find recurrence relation and give initial conditions for the number of bit strings of length n that **do not have two consecutive 0s**. How many such bit strings are there of length five.

- a_n : number of bit string of length n that do not have two consecutive 0s.
- One bit string can terminate with bit 1 or 0
- Format of a bit string that **do not have two consecutive 0s**:



→ $a_n = a_{n-1} + a_{n-2}$ for $n \geq 3$ { Fibonacci sequence }

→ $a_1 = 2$ { two string "0" and "1" }

→ $a_2 = 3$ { "10" , "01" , "11" }

With $n=5$

→ $a_3 = a_2 + a_1 = 5$

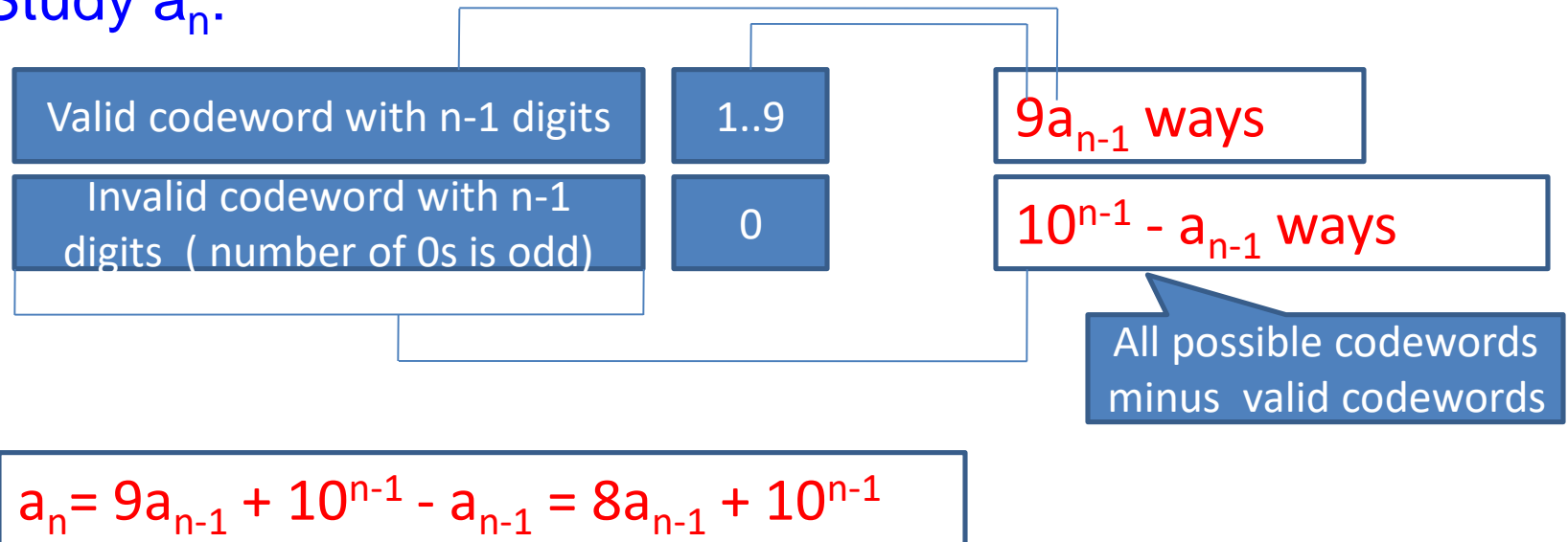
→ $a_4 = a_3 + a_2 = 5 + 3 = 8$

→ $a_5 = a_4 + a_3 = 8 + 5 = 13$

Counting valid codewords

valid codeword: String of decimal digits that contains an even number of 0 digits.

- a_n : number of valid n -digit codewords
- Find a recurrence relation for a_n
- $a_1=9$ // “1”, “2”, ..., “9” → “0” is not used
- Study a_n :



7.3. Divide-and-Conquer Algorithms and recurrence Relations

- **Divide:** Dividing a problem into one or more instances of the same problem of smaller size
- **Conquer:** Using the solutions of the smaller problems to find a solution of the original problem, perhaps with some additional work.

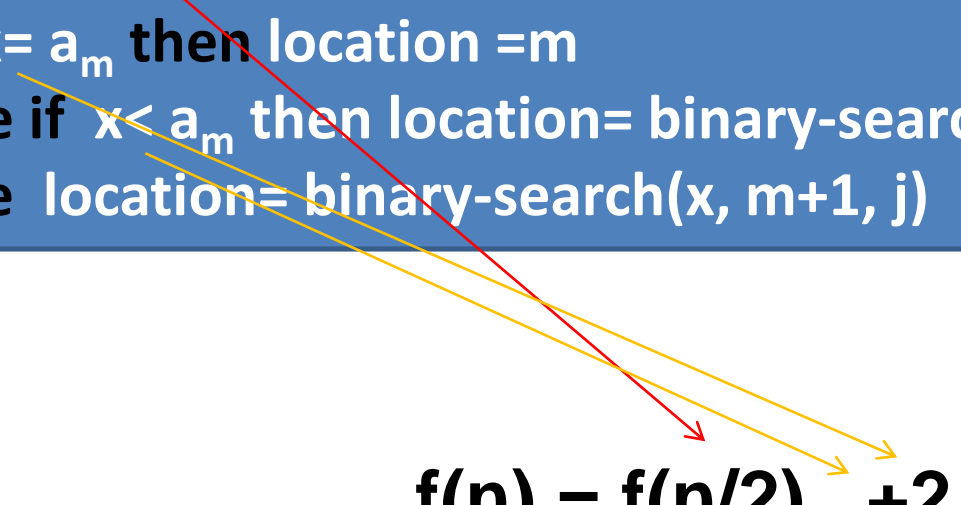
Divide-and-Conquer Recurrence Relations

- n : size of the original problem
- n/b : size of the sub-problem
- $f(n)$: number of operation required to solve the original problem.
- $\rightarrow f(n/b)$: number of operation required to solve a sub-problem.
- $g(n)$: overhead for additional work of the step **conquer**.
- Recurrence relation: **$f(n) = af(n/b) + g(n)$**

Recurrence Relations for Binary Search

```

procedure binary-search(x, i, j)
if i>j then location=0
m=  $\lfloor (i+j)/2 \rfloor$ 
if x= am then location =m
else if x< am then location= binary-search(x, i, m-1)
else location= binary-search(x, m+1, j)
    
```

$$f(n) = f(n/2) + 2$$


Recurrence Relations for Finding Maximum of a sequence

```

procedure max(i,j: integer ,ai,a2+1,...,aj: integers)
if i=j then
  begin
    max:= ai
  end
else
  begin
    m= ⌊(i+j)/2⌋
    max1= max (i,m,ai,ai+1,...,am)
    max2= max (m+1,j,am+1,am+2,...,aj)
    if max1>max2 then max:= max1
    else max:=max2
  
```

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

Theorem 1

- Let f be an **increasing function** that satisfies the recurrence relation $f(n) = af(n/b) + c$

whenever n is divisible by b , where $a \geq 1$, b is an integer and greater than 1, and c is a positive real number. **Then**

$$f(n) \text{ is } \begin{cases} O(n^{\log_b a}) & \text{if } a > 1 \\ O(\log n) & \text{if } a = 1 \end{cases}$$

Furthermore, when $n = b^k$, where k is a positive integer,

$$f(n) = C_1 n^{\log_b a} + C_2$$

Where $C_1 = f(1) + c/(a-1)$ and $C_2 = -c/(a-1)$

Proof: page 477

Using Theorem 1

- Example 6: $f(n) = 5f(n/2) + 3$, $f(1)=7$. Find $f(2^k)$, k is a positive integer. Estimate $f(n)$ if f is increasing function.

Using theorem 1: $a=5$, $b=2$, $c=3$, $n=2^k$

$$C_1 = f(1) + c/(a-1) = 7 + 3/(5-1) = 7 + 3/4 = 31/4$$

$$C_2 = -c/(a-1) = -3/(5-1) = -3/4$$

$$n^{\log a} = 2^{k \log a} = a^k \quad \{ \log a \equiv \log_2 a \}$$

$$f(n) = f(2^k) = C_1 a^k + C_2 = a^k \cdot 31/4 - 3/4$$

f is increasing function, $a > 1 \rightarrow f(n) = O(n^{\log a}) = O(n^{\log 5})$

- Estimate the number of comparisons used by a binary search

$$f(n) = f(n/2) + 2, a=2 \rightarrow f(n) = O(\log n) \text{ // theorem 1}$$

- Estimate the number of comparisons to locate the maximum element in a sequence

$$f(n) = 2f(n/2) + 1, a=2 \rightarrow f(n) = O(n^{\log a}) = O(n) \text{ // theorem 1}$$

Theorem 2: Master Theorem

Let f be an **increasing function** that satisfies the recurrence relation $f(n) = af(n/b) + cn^d$

Whenever $n = b^k$, where k is a positive integer, $a \geq 1$, b is an integer greater than 1, and c and d are real numbers with c positive and d nonnegative. Then

$$f(n) \text{ is } \begin{cases} O(n^d) \text{ if } a < b^d \\ O(n^d \log n) \text{ if } a = b^d \\ O(n^{\log_b a}) \text{ if } a > b^d \end{cases}$$

An Demonstration: Closest-Pair Problem

- n points in the plane. How to determine the closest-pair of points?
- (1) Determine the distance of every pair of points.
- (2) Determine the pair of points that have minimum distance.
- $\rightarrow C(n,2) = n(n-1)/2 = O(n^2)$
- Michal Samos proposed an approach that is $O(n \log n)$ only.
- Michal Samos's approach
 - (1) Sorting points in order of increasing x coordinates $\rightarrow O(n \log n)$
 - (2) Sorting points in order of increasing y coordinates $\rightarrow O(n \log n)$

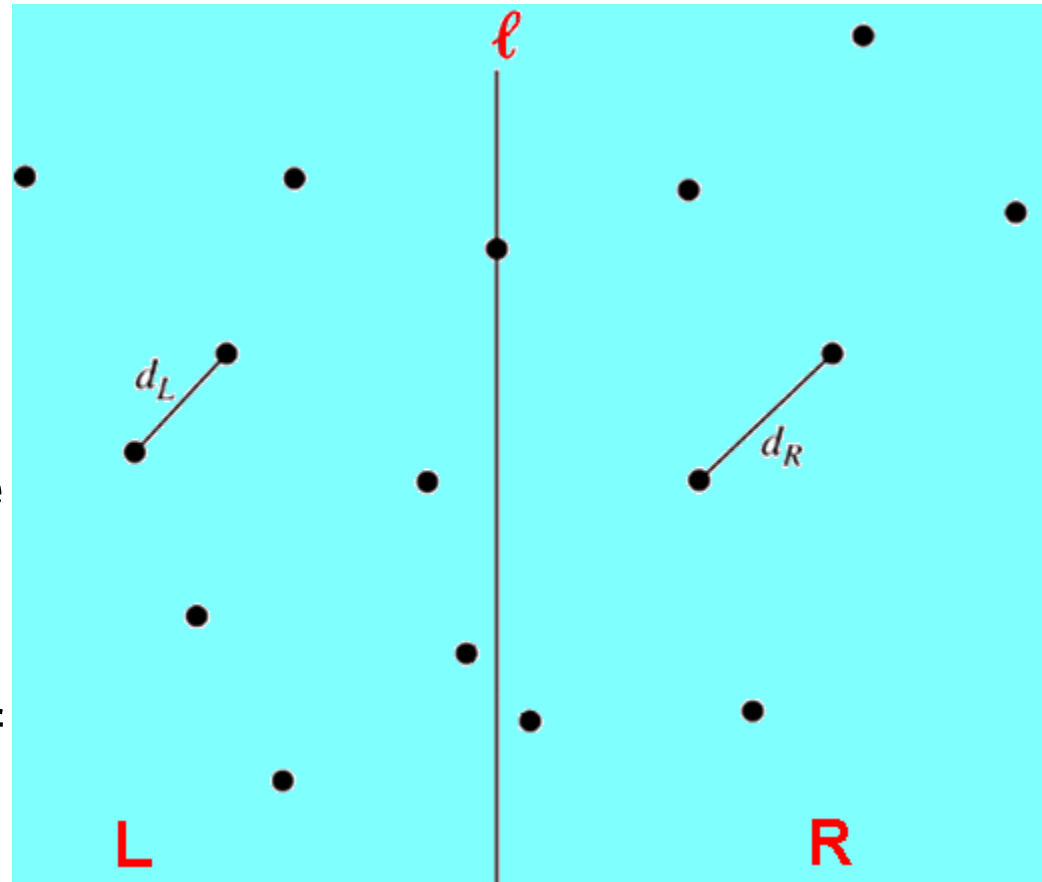
An Demonstration: Closest-Pair Problem

(3) Using recursive approach to divide the problem into 2 subproblem with $n/2$ points (left and right points based on x coordinates). Let ℓ is the line that partitions two subproblems. If there is any point on this dividing line, we decide these points among the two parts if necessary)

(4) Finding out closest-pair of points in two side (d_L , d_R)

(5) Let $d = \min(d_L, d_R)$

16 points

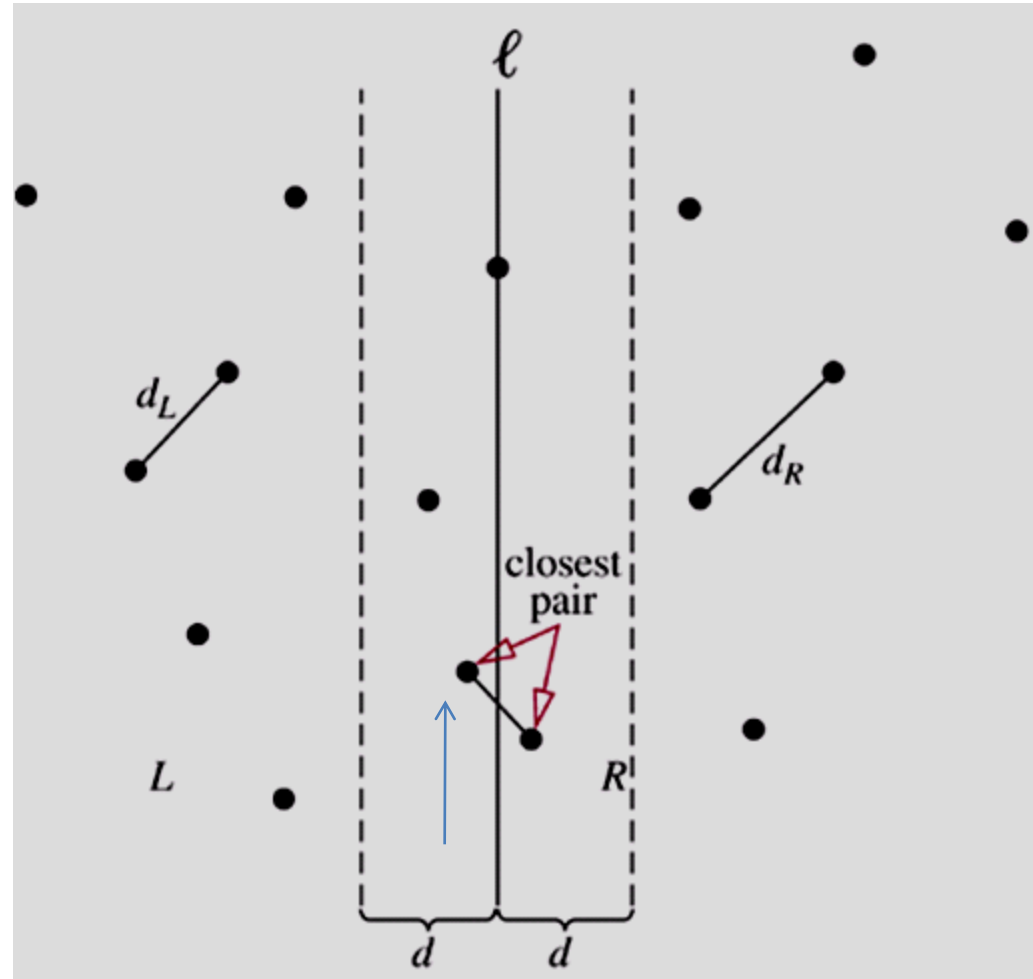


8 points

8 points

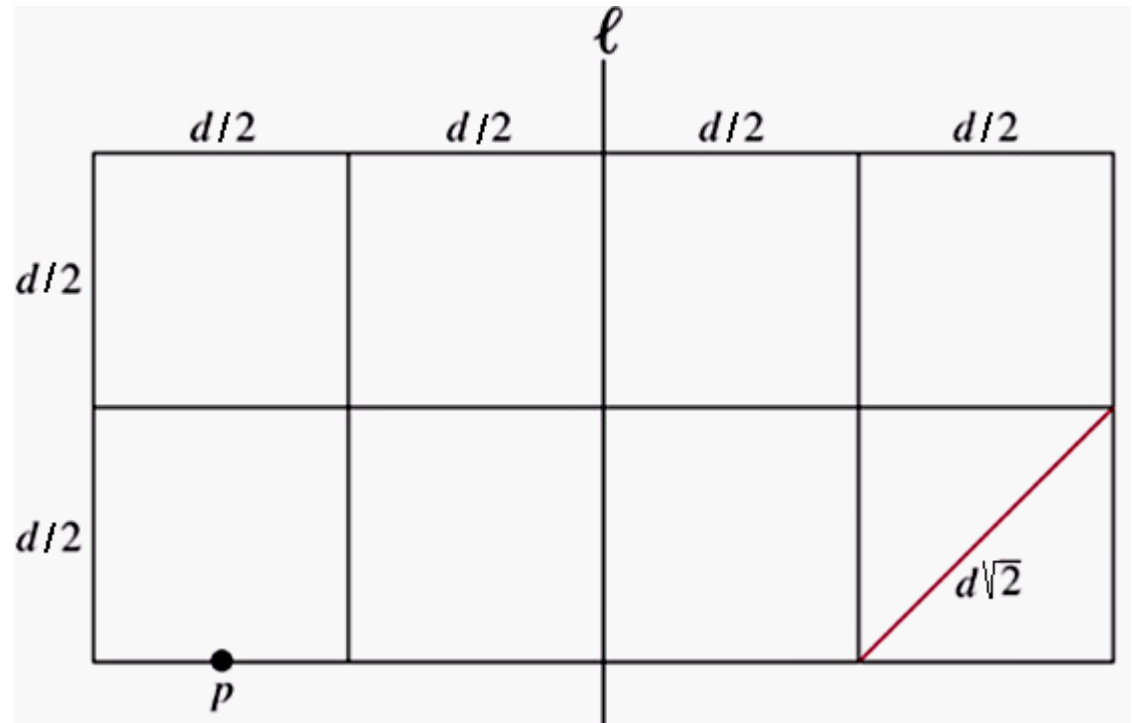
An Demonstration: Closest-Pair Problem

- (6) Studying area $[l-d, l+d]$.
This area may be contains the result.
- (7) Because we sorted points by their y coordinate. We examine for points p in the strip of width $2d$ that has the line l as its center with upward direction.



An Demonstration: Closest-Pair Problem

Total number of points in the strip does not exceed n and there are at most **8 points**, including p , can lie in or on the $2d \times d$ rectangle.



- A point will be computed with **7** others.
- At most **$7n$** distances need to be compared with d to find the minimum distance between points.
- The increasing function $f(n)$ satisfies the recurrence relation :

$$f(n) = 2f(n/2) + 7n$$
- By the Master Theorem, it follows that $f(n)$ is **$O(n \log n)$**

Summary

- **Product rule:** Suppose that a procedure can be broken down into a sequence of two tasks. If there is n_1 ways to do the first task, there is n_2 ways to do the second task, then there are $n_1 n_2$ ways to do the procedure.
- **Sum rule:** If a task can be done either in one of n_1 ways or in one of n_2 ways, where none of the set of n_1 ways is the same as any of the set of n_2 ways, then there are $n_1 + n_2$ ways to do this task.

Summary

- **Inclusion-Exclusion Principle:** Suppose that a task can be done in n_1 or n_2 ways, but that some ways in the set of n_1 ways are the same as some ways in the set of n_2 ways.

Summary

- **Recurrence relation** on the sequence $\{ a_n \}$ is an expression in which a_n is expressed in terms of one or more of previous items for all n with $n \geq n_0$. $\{a_n\}$ is called a **solution** of recurrence relation.
- **Initial conditions:** Terms that **precede the first item where the recurrence takes effect.**
- **Modeling with recurrence relation:** Finding out an recurrence relation for input of a problem.

Summary

- **Divide:** Dividing a problem into one or more instances of the same problem of smaller size
- **Conquer:** Using the solutions of the smaller problems to find a solution of the original problem, perhaps with some additional work
- Recurrence relation: $f(n) = af(n/b) + g(n)$

Summary – Theorem 1

- Let f be an **increasing function** that satisfies the recurrence relation $f(n) = af(n/b) + c$ whenever n is divisible by b , where $a \geq 1$, b is an integer and greater than 1, and c is a positive real number. **Then**

$$f(n) \text{ is } \begin{cases} O(n^{\log_b a}) & \text{if } a > 1 \\ O(\log n) & \text{if } a = 1 \end{cases}$$

Furthermore, when $n = b^k$, where k is a positive integer,

$$f(n) = C_1 n^{\log_b a} + C_2$$

Where $C_1 = f(1) + c/(a-1)$ and $C_2 = -c/(a-1)$

Proof: page 477

Summary – Theorem 2- Master Theorem

Let f be an **increasing function** that satisfies the recurrence relation $f(n) = af(n/b) + cn^d$

Whenever $n = b^k$, where k is a positive integer, $a \geq 1$, b is an integer greater than 1, and c and d are real numbers with c positive and d nonnegative. Then

$$f(n) \text{ is } \begin{cases} O(n^d) \text{ if } a < b^d \\ O(n^d \log n) \text{ if } a = b^d \\ O(n^{\log_b a}) \text{ if } a > b^d \end{cases}$$

- **THANKS**