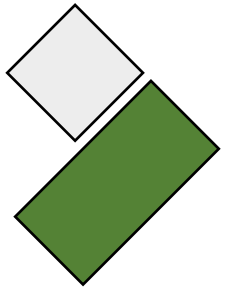


Counting

Contents

- The basics of counting
  - Product rule
  - Sum rule
  - Principle of inclusion–exclusion
  - Tree diagram
- Advanced techniques
- Divide – and –conquer algorithms

Introduction

***

- Choose a quiz password: *** with letters come from {a, b, c, d}
- How many possible passwords ?

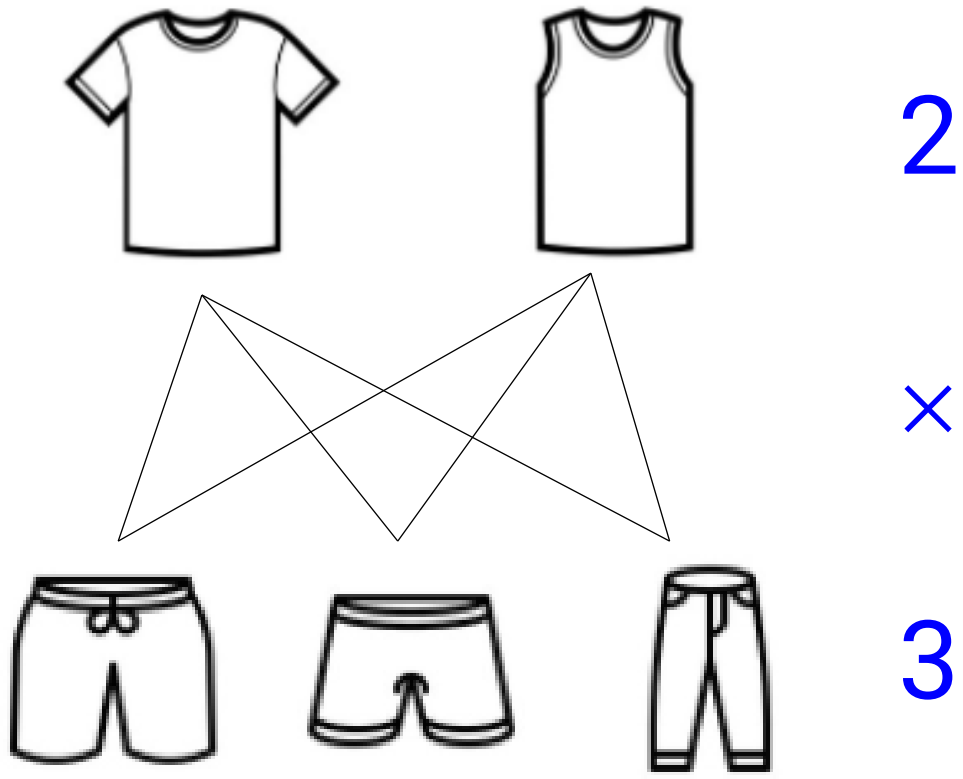
Basic Counting Principles – Product Rule

Product Rule.

Task = 1st task ➔ 2nd task

n_1 × n_2 ways

Suppose that a procedure can be broken down into a sequence of two tasks. If there are n_1 ways to do the first task and for each of these ways of doing the first task, there are n_2 ways to do the second task, then there are n_1n_2 ways to do the procedure.



Product rule – examples

Ex1. A new company with just two employees, Alex and Bob, rents a floor of a building with 12 offices.

How many ways are there to assign different offices to these two employees?

➔ 2 tasks: $12 \cdot 11 = 132$ ways

Ex2. How many different bit strings of length seven are there?

➔ 7 tasks: $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^7$ bit strings

Ex3. There are 32 microcomputers in a computer center. Each microcomputer has 24 ports. How many different ports to a microcomputer in the center are there?

➔ $32 \cdot 24 = 768$ ports

Ex. How many one-to-one functions are there from {○, □, ×, △} to {Long Pass, Shoot, Key Pass, Pass}?

$4 \cdot 3 \cdot 2 \cdot 1 = 24$ one-to-one functions

Product Rule – Example

- What is the value of k after the following code, where $n_1, n_2, \ldots, n_m$ are positive integers, has been executed?

```
k := 0
for i1 := 1 to n1
        for i2 := 1 to n2
        . . .
                for im := 1 to nm
                        k := k + 1
```

$\Rightarrow$ $k = n_1 n_2 \cdots n_m$

Product Rule – Example (RGB Colors)

- An RGB color value is specified with: rgb(red, green, blue).
- Each parameter (red, green, blue) defines the intensity of the color as an integer between 0 and 255.
- How many different RGB colors are there?

➔ $256 \cdot 256 \cdot 256 = 16{,}777{,}216$ colors

Basic Counting Principles – Product Rule

THE 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 the task.

Ex. Suppose that either a member of the mathematics faculty or a student who is a mathematics major is chosen as a representative to a university committee. How many different choices are there for this representative if there are 37 members of the mathematics faculty and 83 mathematics majors and no one is both a faculty member and a student?

➔ 37 + 83 = 120 possible ways to pick a representative

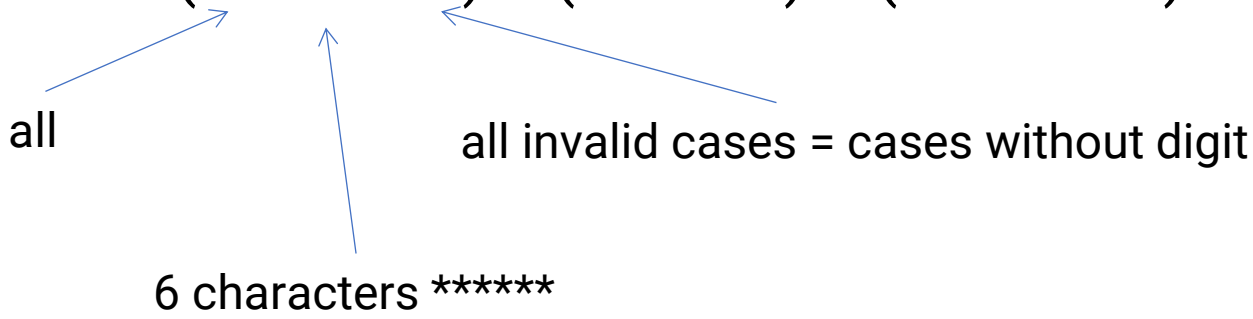
What is the value of k after the following code, where $n_1, n_2, \ldots, n_m$ are positive integers, has been executed?

```
k := 0
for i1 := 1 to n1
        k := k + 1
for i2 := 1 to n2
        k := k + 1
. . .
for im := 1 to nm
        k := k + 1
```

$\Rightarrow$ $k = n_1 + n_2 + \cdots + n_m$

Example – counting passwords

- Each user on a computer system has a password, which has properties:
  - six to eight characters long
  - character is an uppercase letter or a digit
  - contain at least one digit
- How many possible passwords are there?
- Result = $(36^6 - 26^6) + (36^7 - 26^7) + (36^8 - 26^8)$



Exercises

1/ If there are **5 multiple-choice** questions on an exam, each having four possible answers, how many different sequences of answers are there?

2/ In how many ways can a teacher seat 5 girls and 3 boys in a row seats if a boy must be seated in the first and a girl in the last seat?

Exercises

1/ How many positive divisors does 120 have?

2/ A = {1, 2, 3, 4, 5, 6}

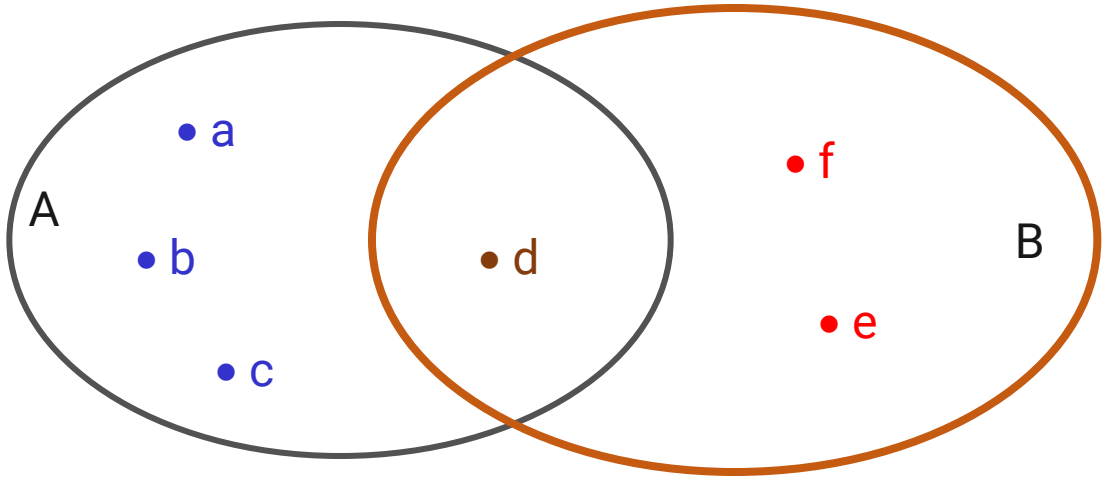
a. How many subsets of A can be constructed?

b. How many subsets of A that contain 1?

c. How many subsets neither contain 3 nor 4?

The principle of Inclusion–exclusion

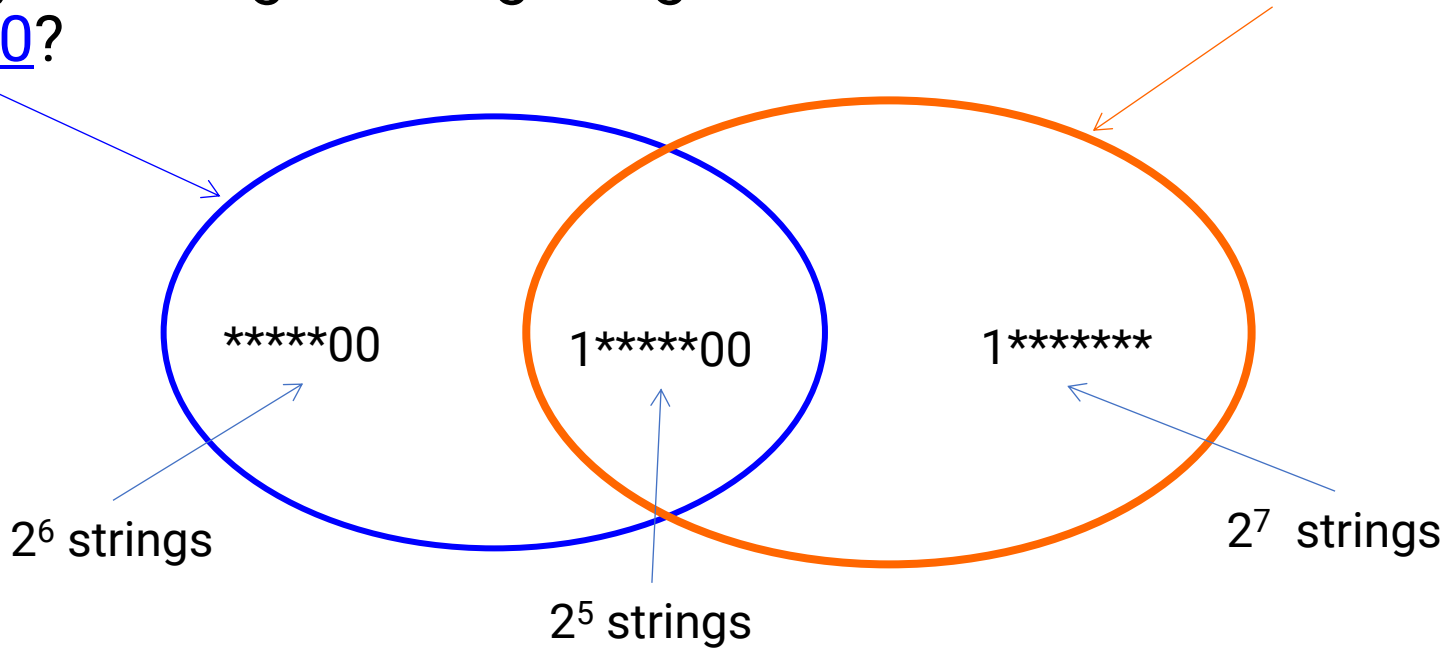
$$|A \cup B| = |A| + |B| - |A \cap B|$$



$$| A \cup B | = |A| + |B| - |A \cap B| = 4 + 3 - 1$$

The principle of Inclusion-exclusion

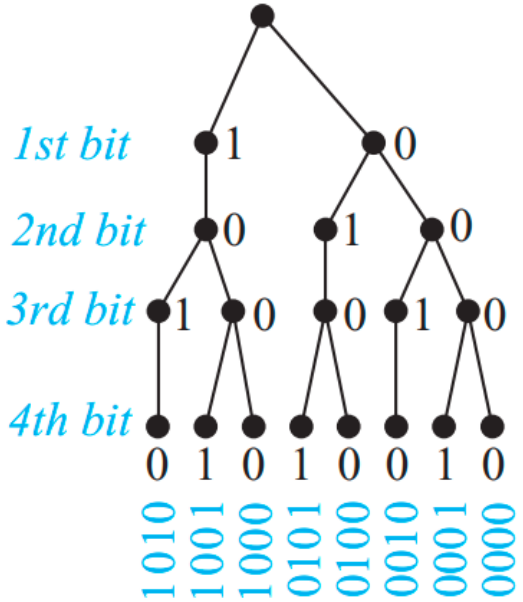
- **Ex.** How many bit strings of length eight either start with a 1 bit OR end with the two bits 00?



- Result = $2^7 + 2^6 - 2^5$

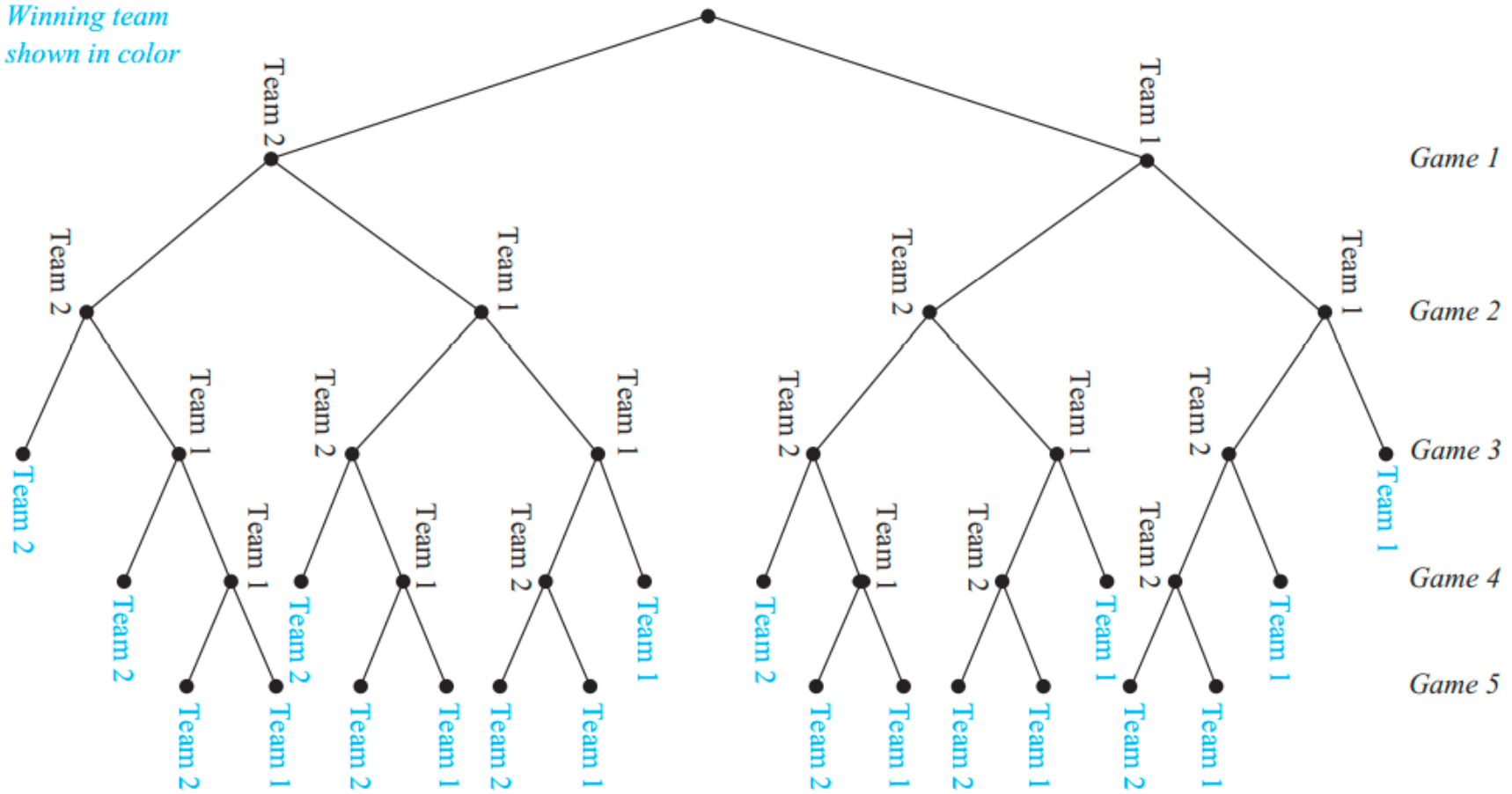
Tree Diagrams

Ex. How many bit strings of length four do not have two consecutive 1s?



Tree Diagrams

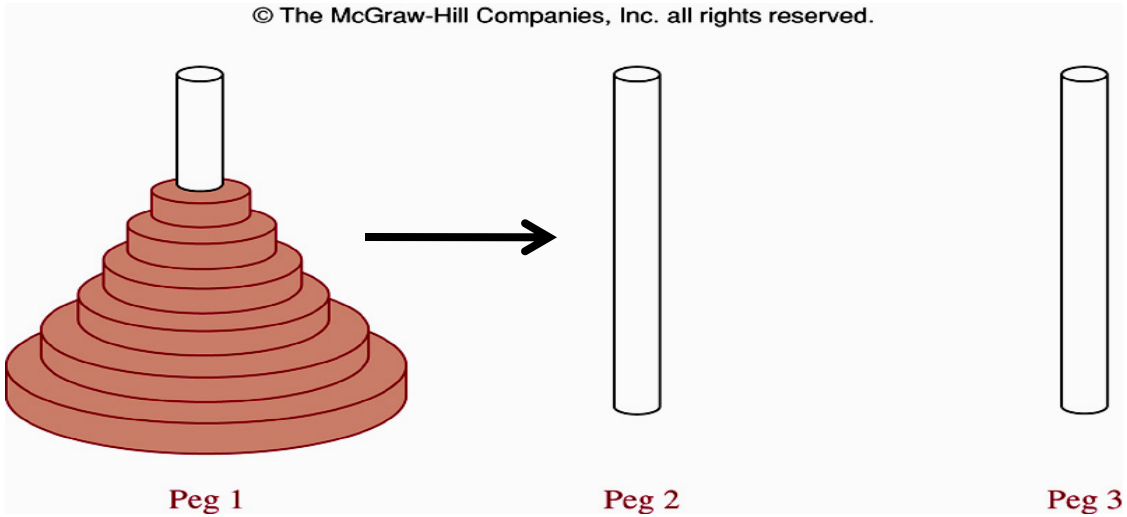
Ex. A playoff between two teams consists of at most five games. The first team that wins three games wins the playoff. In how many different ways can the playoff occur?



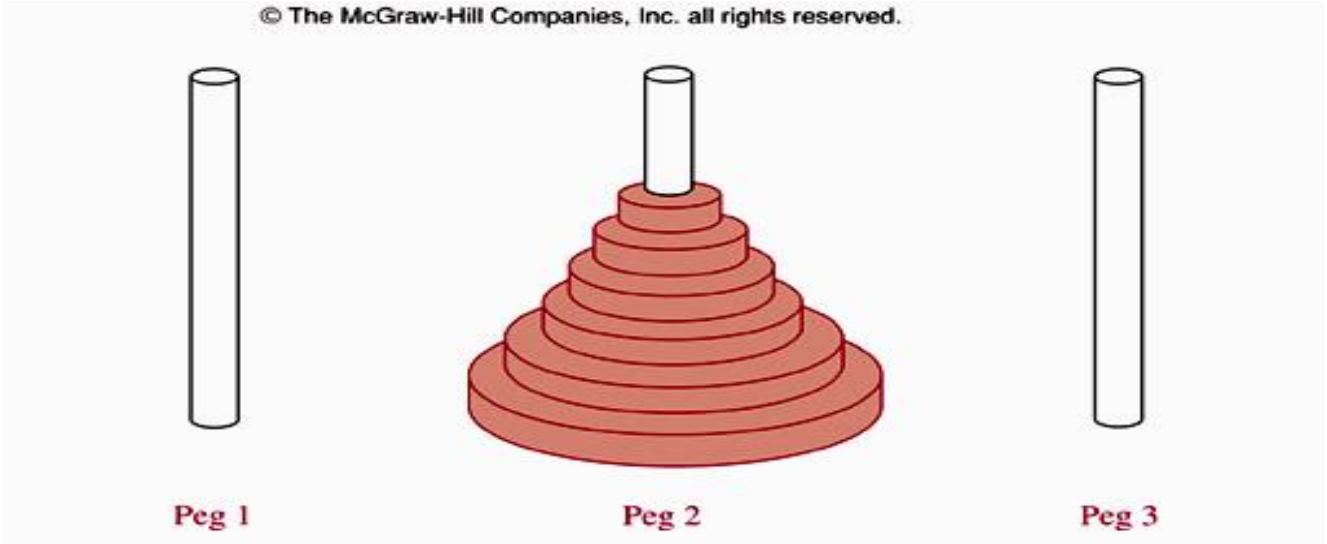
Advanced counting techniques

- P_0: initial deposit
- r: interest rate per year
- P_n: amount of money after n years
- $P_n = (1+r)P_{n-1}$
- $P_n = (1+r)^n P_0$

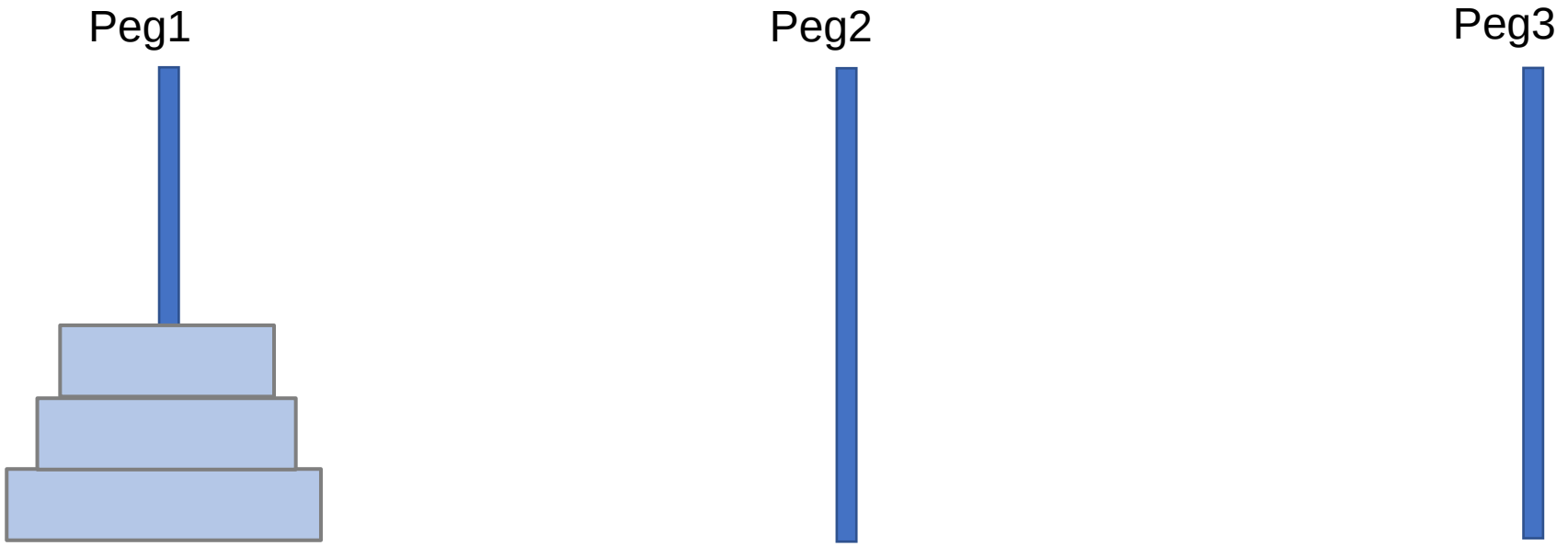
The Tower of Hanoi Problem



How many moves needed to solve the Tower of Hanoi problem with n disks?



The Tower of Hanoi – 3 disks



- 3 disks → 7 steps

The Tower of Hanoi problem

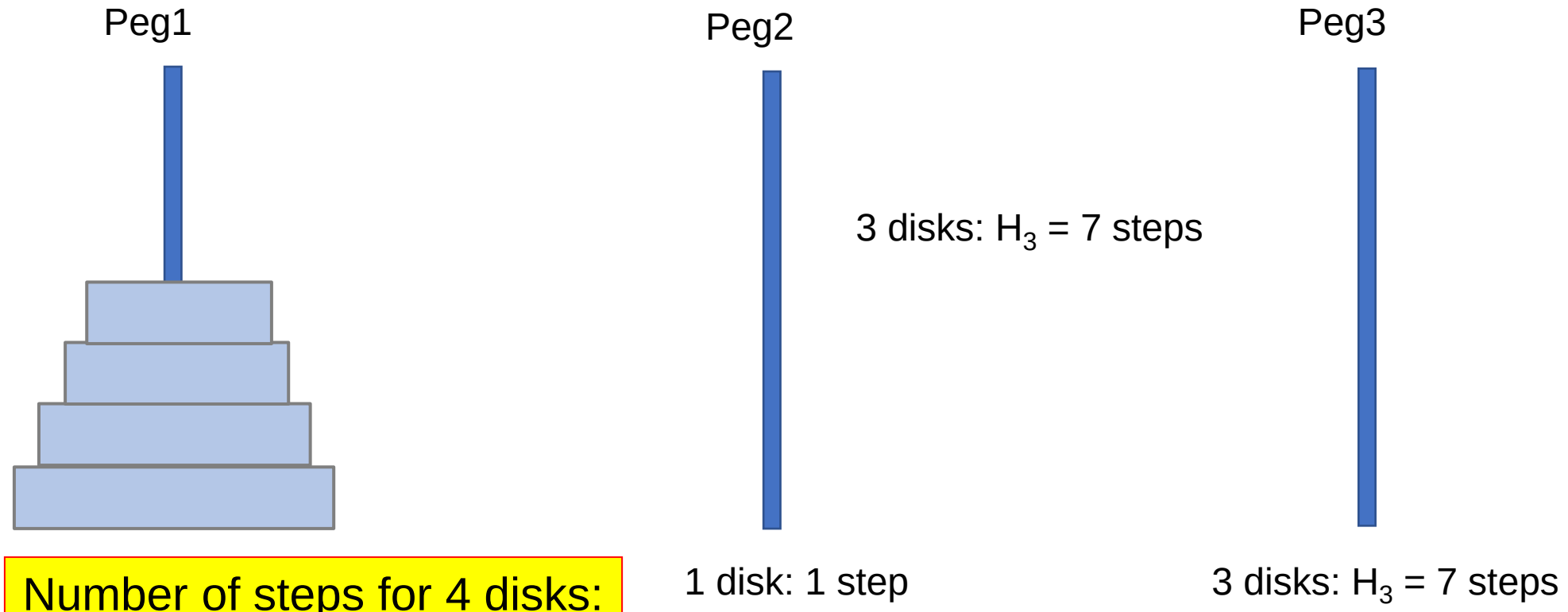
- n = 1 disk $\Rightarrow H_1$ = 1 step
- n = 2 disks:

Peg1		→		Peg3
Peg1	→	Peg2		
		Peg2	←	Peg3

$\Rightarrow H_2$ = 3 steps

- n = 3 disks $\Rightarrow H_3$ = 7 steps
- **n** disks $\Rightarrow$ **H_n** = ? // number of steps for **n** disks

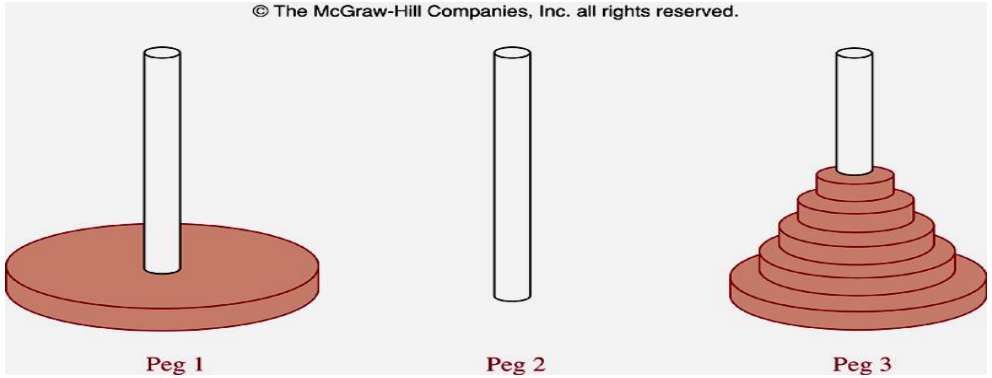
The Tower of Hanoi problem – How many steps for n = 4 and more?



Number of steps for 4 disks:

$$H_4 = 2H_3 + 1 = 15$$

The Tower of Hanoi problem – n disks



- $H_n = 2H_{n-1} + 1$ // recurrence relation
- $H_1 = 1$ // initial condition

$\Rightarrow H_n = 2^n - 1$, $n \geq 1$ // solution

- $n = 64 \rightarrow H_{64} = 2^{64}-1$

(=18 446 744 073 709 551 615)

1 move/sec → more than 500 billion years.

- Complexity $O(2^n)$

Ex. How many bit strings of length n that not have two consecutive 0s

- n : length of bit string
- a_n: number of such bit strings of length n

n	all 2^n bit strings of length n	a_n
1	0, 1	$a_1 = 2$
2	~~00~~, 01, 10, 11	$a_2 = 3$
3	~~000~~, ~~001~~, 010, 011, ~~100~~, 101, 110, 111	$a_3 = 5$
4	~~0000~~, ~~0001~~, ~~0010~~, ~~0011~~, ~~0100~~, **01**01, **01**10, **01**11, ~~1000~~, ~~1001~~, **1**010, **1**011, ~~1100~~, **1**101, **1**110, **1**111	$a_4 = 8$
…	…	…
n	a_{n-2}: number of such strings begin with 0 (in fact, 01) a_{n-1}: number of such strings begin with 1	$\mathbf{a_n = a_{n-2} + a_{n-1}}$

Recurrence Relations

- ***Recurrence relation:***
  - $H_n = 2H_{n-1} + 1$
  - $a_n = a_{n-1} + a_{n-2}$

- **Initial conditions:**
  - **$H_1=1$**
  - **$a_1=2$, $a_2=3$**

Recurrence Relations

- **Ex.** Determine whether $\{a_n\} = 3n$ is a ***solution*** of the recurrence relation

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

- $a_n = 3n \Rightarrow a_{n-1} = 3(n-1)$ and $a_{n-2} = 3(n-2)$
- The right-hand side = $2a_{n-1} - a_{n-2} = 2.3(n-1) - 3(n-2)$
- The right-hand side = $3n = a_n$
- Left-hand side = Right-hand side

$\Rightarrow a_n$ is a solution of the recurrence relation

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 relation:** $f(n) = af(n/b) + g(n)$
  - f(n): the number of operations required to solve the problem of size n
  - g(n): extra operations required to combine the solutions of the subproblems into a solution of the original problem.

Recurrence Relations for Binary Search

```
procedure  bsearch(x, i, j)
if  i > j
        return 0
else
        m= ⌊(i+j)/2⌋
        if x= a_m
                return m
        else if  x< a_m
                bsearch(x, i, m-1)
        else
                bsearch(x, m+1, j)
```

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

Recurrence Relations for Finding Maximum of a sequence

procedure *findmax*(i,j: integer ,a_i,a_{2+1},…,a_j: integers)
if i=j
    return (a_i)
else
    m:= $\lfloor$(i+j)/2$\rfloor$
    max1:= *findmax* (i,m,a_i,a_{i+1},…,a_m)

    max2:= *findmax* (m+1,j,a_{m+1},a_{m+2},…,a_j)
    if max1 > max2 **then**
        return max1
    else
        return max2

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

Recurrence Relations for Merge Sort

The merge sort algorithm splits a list to be sorted with n items, where n is even, into two lists with n/2 elements each, and uses fewer than n comparisons to merge the two sorted lists of n/2 items each into one sorted list. Consequently, the number of comparisons used by the merge sort to sort a list of n elements is less than M(n), where the function M(n) satisfies the divide-and-conquer recurrence relation

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

Fast Matrix Multiplication

- Invented by Volker Strassen in 1969, reduces the multiplication of two n × n matrices, when n is even, to seven multiplications of two (n/2) × (n/2) matrices and 15 additions of (n/2) × (n/2) matrices.
- Hence, if f(n) is the number of operations (multiplications and additions) used, it follows that

$$f(n) = 7f(n/2) + 15n^2/4$$

when n is even.

Theorem

MASTERTHEOREM (proof as Exercises 29–33)

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

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

Apply the Master theorem to some algorithms

- Binary search: f(n) = f(n/2) + 2 ➔ f(n) is O(logn)

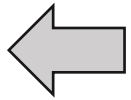
  (a = 1, b = 2, d = 0)

- Max: f(n) = 2f(n/2) + 1 ➔ f(n) is O(n)

  (a = 2 = b, d = 0)

- Merge sort: M(n) = 2M(n/2) + n ➔ M(n) is O(nlogn)

  (a = 2 = b, d = 1)



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

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.
- ➔ C(n,2)= n(n-1)/2= **$O(n^2)$**
- Michal Samos proposed an approach that is **O(nlogn)** only.
- Michal Samos's approach

(1) Sorting points in order of increasing x coordinates → O(nlog(n))

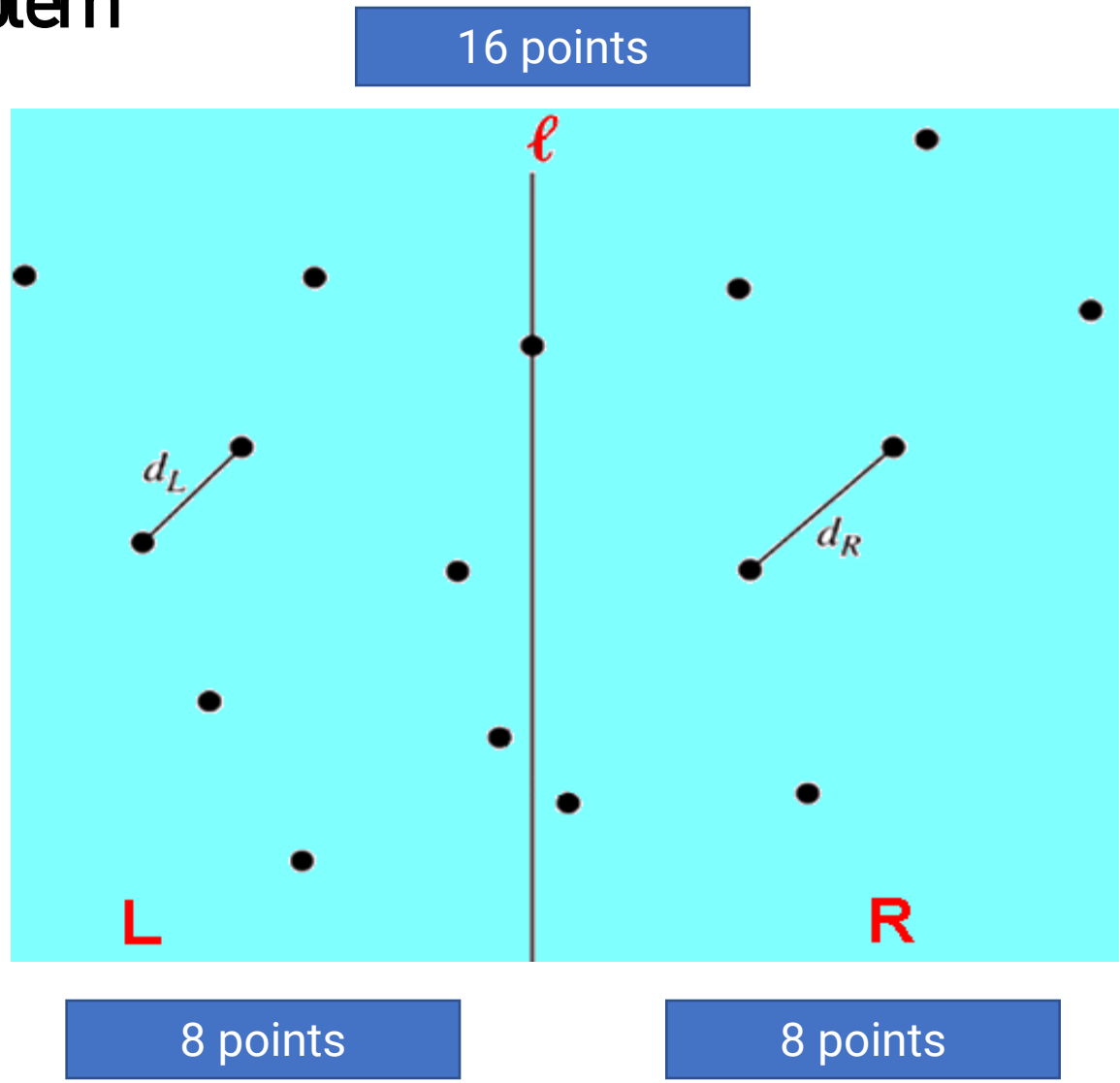
(2) Sorting points in order of increasing y coordinates → O(nlog(n))

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 be 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)

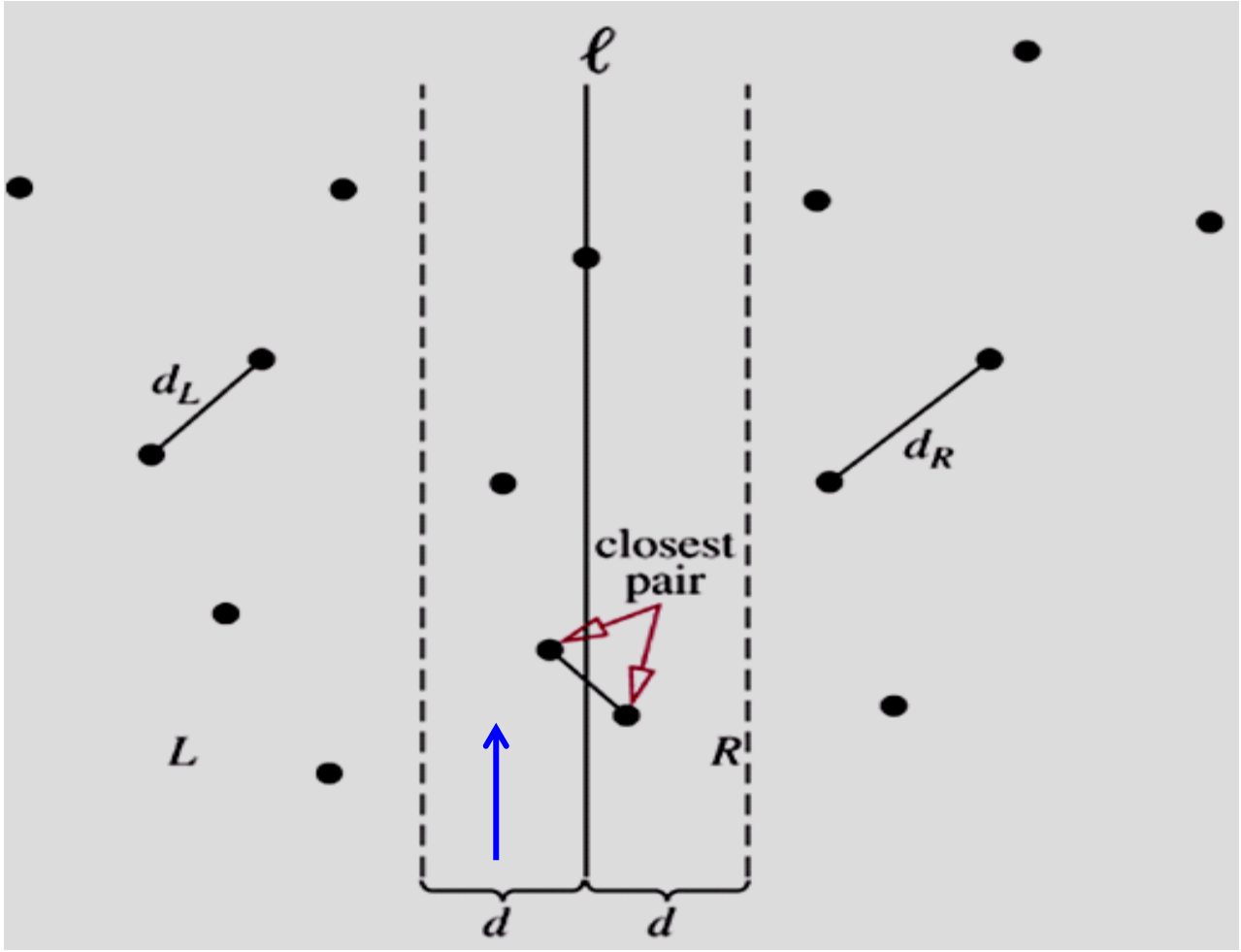
(5) Let $d=\min(d_L, d_R)$



An Demonstration: Closest-Pair Problem

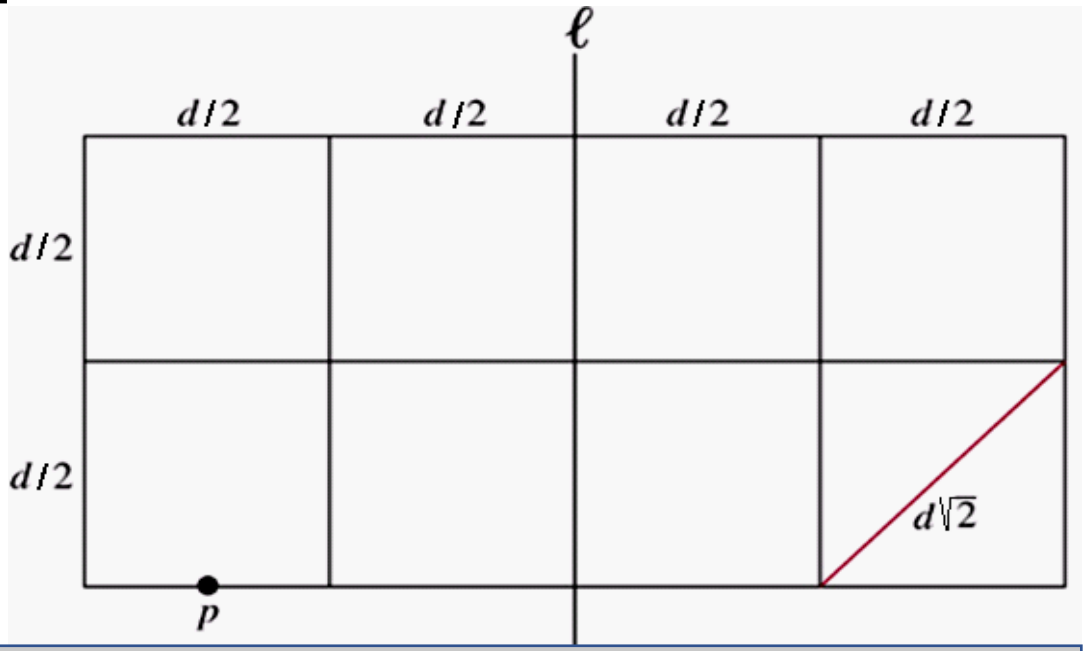
(6) Studying area [l-d, l+d]. This area may be contains the result.

(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 2dxd 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(nlogn)**

Summary

- The basics of counting
  - Product rule
  - Sum rule
  - Principle of inclusion–exclusion
  - Tree diagram
- Advanced techniques
- Divide – and –conquer algorithms

Assignment/Computer project

- Assignment: lms
- Computer project:
  - Given a sequence of positive integers, find the longest increasing and the longest decreasing subsequence of the sequence.
  - Given a positive integer n, list all bit strings of length n that do not have two consecutive 0s?

```
> no00(2)
[1] 0 1
[1] 1 0
[1] 1 1
```

```
> no00(3)
[1] 0 1 0
[1] 0 1 1
[1] 1 0 1
[1] 1 1 0
[1] 1 1 1
```

THANKS