CHAPTER 5 - COUNTING

- Choose a quiz password: ***

* can be chosen from {a, b, c, d}

- How many possible passwords ?

Product rule

Task = task 1 AND task 2

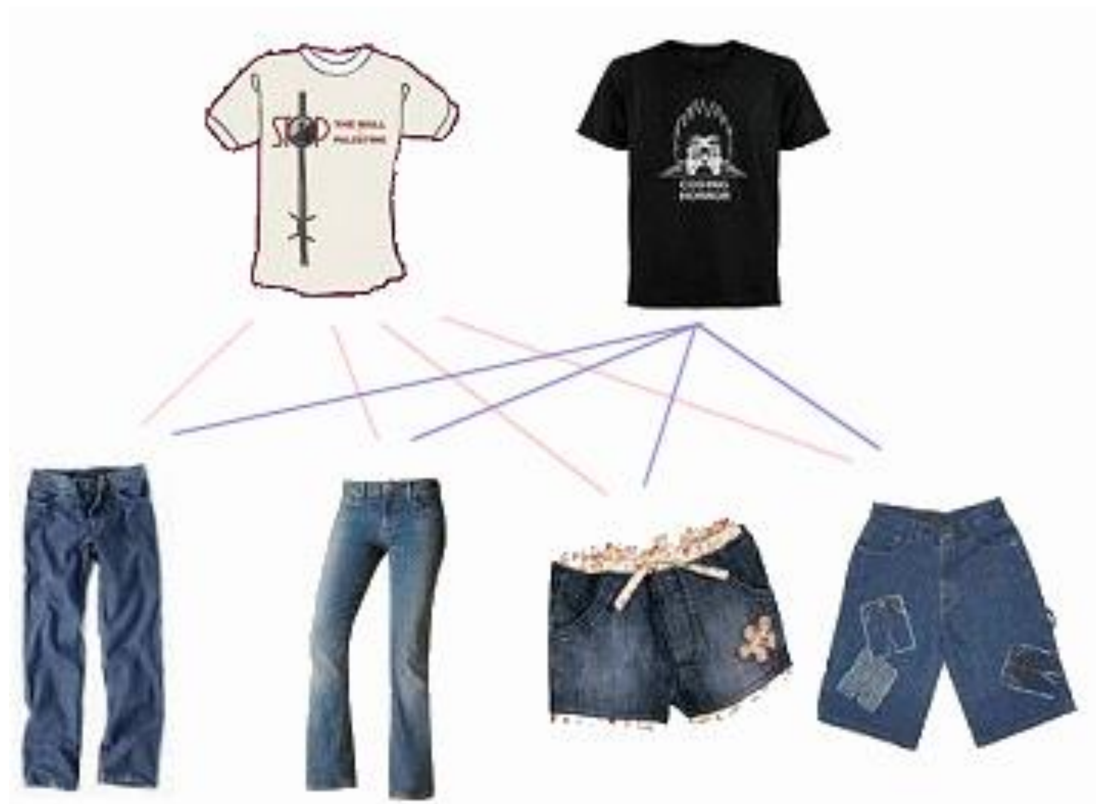
2

Task 1: choose one

×

Task 2: choose one

4



Product rule

THE PRODUCT RULE

Task = task1 → task 2 → task 3 → … → task k

task 1: n1 ways

task 2: n2 ways

…

Task k: nk ways

Product rule: n1.n2…nk ways to do the task

Product rule

Example 1. A new company with just two employees, A and B, 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. 11

Example 2. How many different bit strings of length seven are there?

➔ 7 tasks: 2.2.2.2.2.2.2 $=2^7$

How many functions are there from {a, b, c} to {1, 2, 3, 4, 5} ?

function Input ⟶ Output

A •   • 1

B •   • 2

C •   • 3

           • 4

           • 5

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 = $(36^6 - 26^6) + (36^7 - 26^7) + (36^8 - 26^8)$

all

all invalid cases = cases without digit

6 characters ******

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?

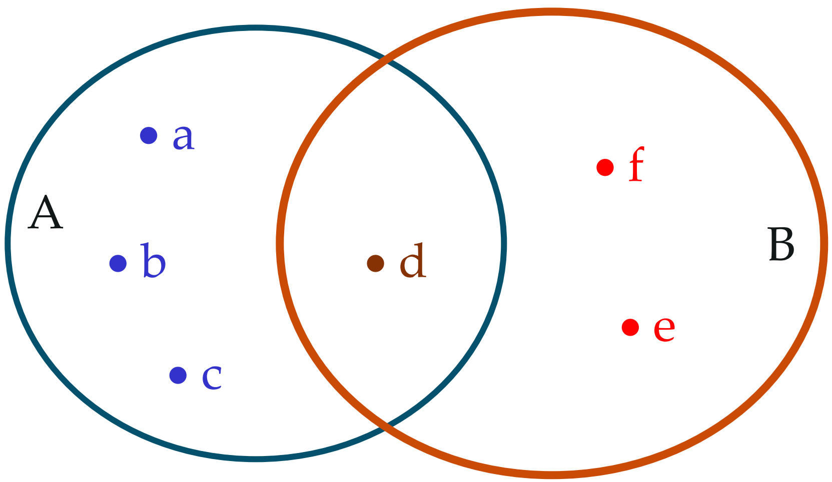
b. How many *subsets of A that* contain 1?

c. How many *subsets* neither contain 3 nor 4?

How many integers between 10 and 30 inclusive are divisible by 3 or 7?

The principle of Inclusion-exclusion

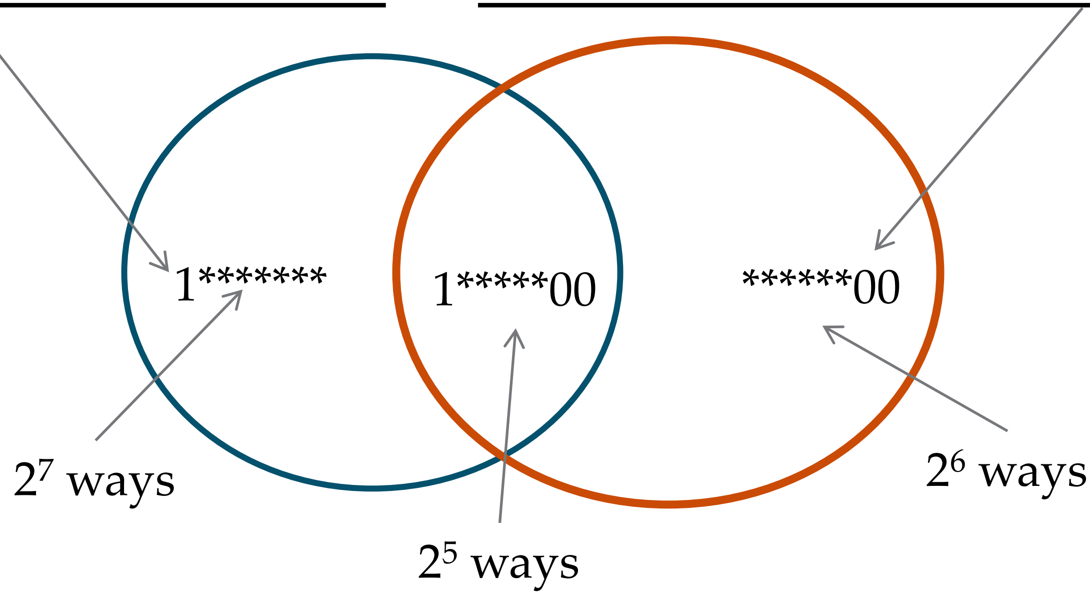
$$|A \cup B| = |A| + |B| - |A \cap B|$$



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

The principle of Inclusion-exclusion

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



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

Advanced counting techniques

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

Advanced counting techniques

Readings:

- **Tower of Hanoi problem:**

Q1. How many moves if $n = 5$?

Q2. How to move?

- **Solution of a *recurrence relation*:**

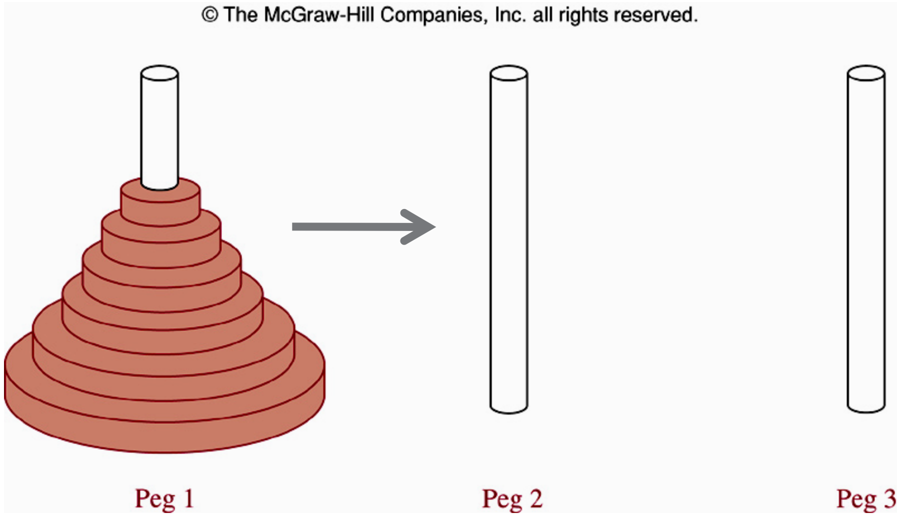
Q3. is $a_n = 3n$ a ***solution*** of $a_n = 2a_{n-1} - a_{n-2}$**?**

- **Divide and conquer algorithm:**

Q4. How to estimate complexity of a divide and conquer algorithm?

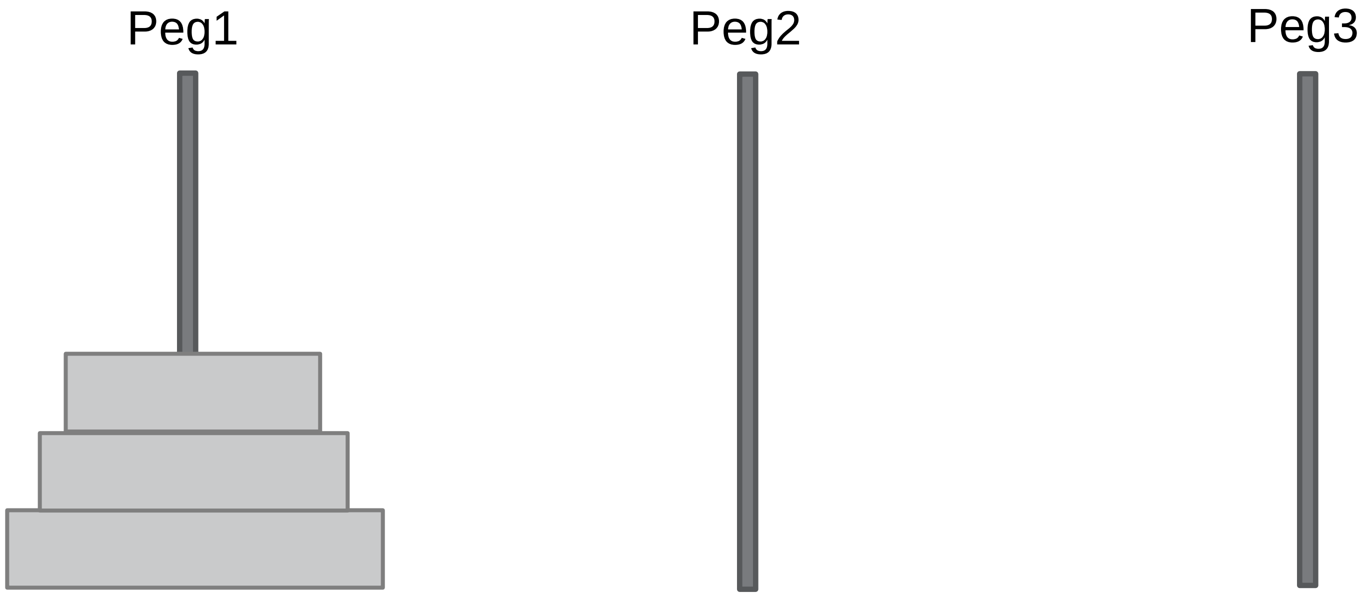
Q5. If $f(n) = 2f(n/2) + 1 \rightarrow f(n)$ is $O($**?**$)$

The Tower of Hanoi Problem- Ex5, page 452



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

Tower of Hanoi – 3 disks



- 3 disks → 7 steps

Tower of Hanoi problem

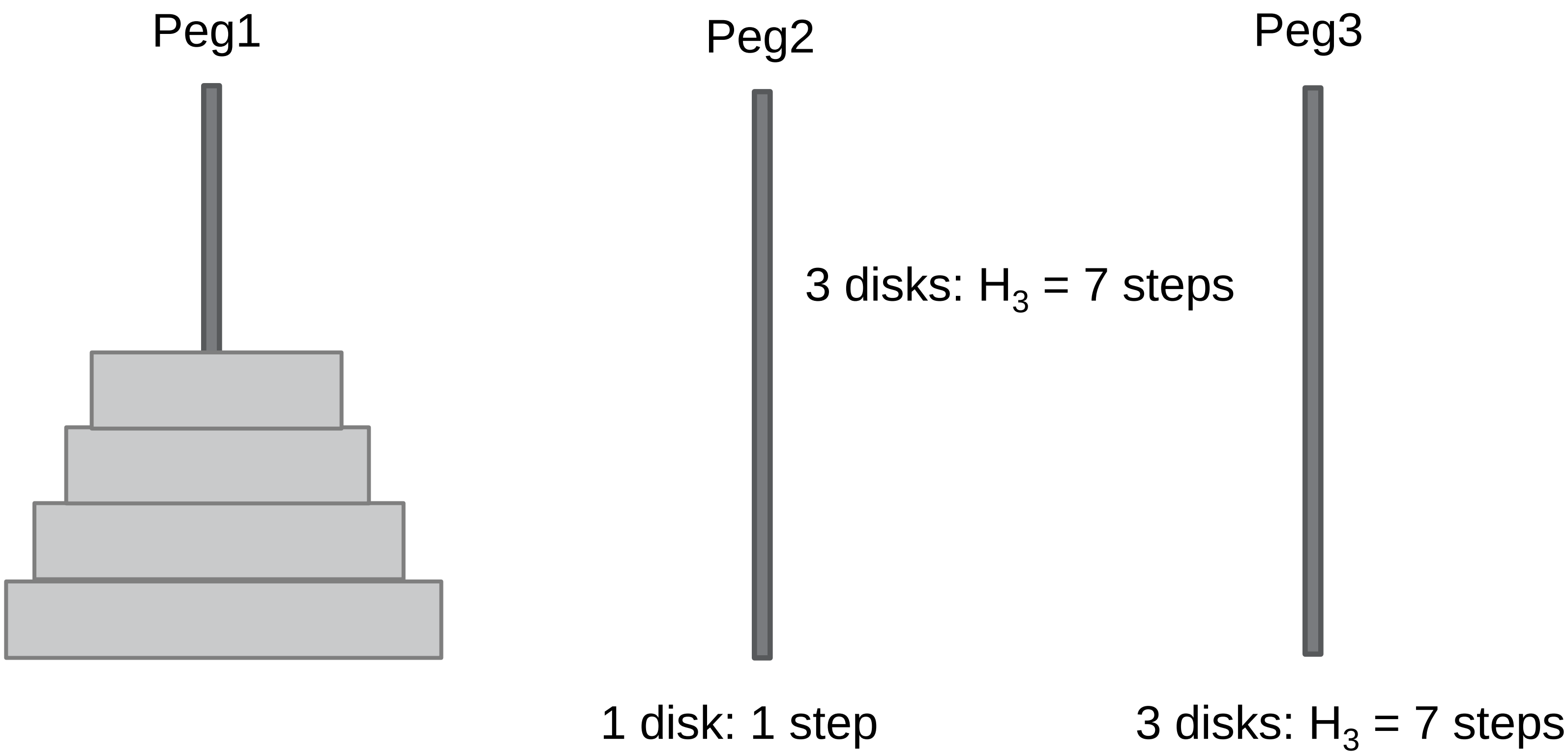
- n = 1 disk $\Rightarrow H_1 = 1$ step
- n = 2 disks:

Peg1		$\rightarrow$		Peg3
Peg1	$\rightarrow$	Peg2		
		Peg2	$\leftarrow$	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

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



Number of steps for 4 disks:

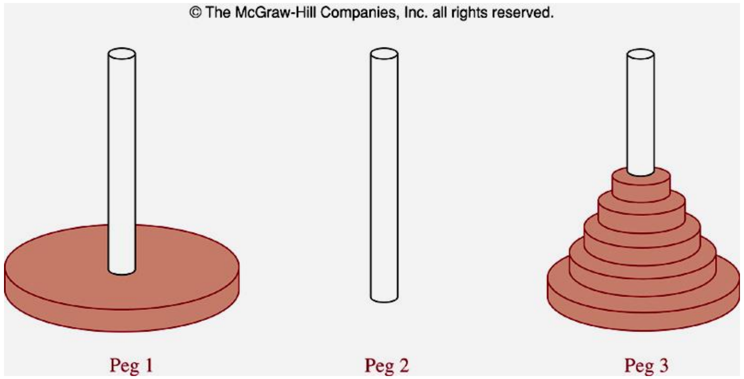
$$H_4 = 2H_3 + 1 = 15$$

Tower of Hanoi problem with n disks

- $\mathbf{H_n = 2H_{n-1} + 1}$ // recurrence relation
- $H_1 = 1$ // initial condition
- $\Rightarrow$ $\mathbf{H_n = 2^n - 1}$, $n \geq 1$ // 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 $\mathbf{O(2^n)}$



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~~, 0101, 0110, 0111, ~~1000~~, ~~1001~~, 1010, 1011, ~~1100~~, 1101, 1110, 1111	$a_4 = 8$
…	…	…
n	a_{n-2}: number of such strings begin with 0 (in fact, 01) a_{n-1}: number of such strings begin with 1	$a_n = a_{n-2} + a_{n-1}$ (Fibonacci)

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

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

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

- $a_n = 3n \Rightarrow a_{n-1} = 3(n-1)$ 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.

Recurrence Relations for Binary Search

procedure *bsearch*(x, i, j)
if i > j
    return 0
else
    m= $\lfloor (i+j)/2 \rfloor$
    if x= a_m
        return m
    else if x< a_m
        bsearch(x, i, m-1)
    else
        bsearch(x, m+1, j)

need **f(n)** comparisons

need **f(n/2)** comparisons

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
        return max1
    else
        return max2

need **f(n)** comparisons

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

need **f(n/2)** comparisons

need **f(n/2)** comparisons

Theorem. […]

$f(n) = af(n/b) + c \Rightarrow f(n)\ is \begin{cases} O(n^{\log_b a}) & \text{if } a>1 \\ O(\log n) & \text{if } a=1 \end{cases}$

a=1

b=2

c=2

Ex.

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

$\Rightarrow$ f(n) is O(logn)

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)
```

need **f(n)** comparisons

need **f(n/2)** comparisons

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

O(logn) time complexity

Theorem. […]

$f(n) = af(n/b) + c \Rightarrow f(n)\ is \begin{cases} O(n^{\log_b a}) & \text{if } a>1 \\ O(\log n) & \text{if } a=1 \end{cases}$

Recurrence Relations for Finding Maximum of a sequence

What is big-O time complexity of the *findmax* algorithm?

procedure *findmax*(i,j: integer ,$a_i, a_{2+1}, \ldots, a_j$: integers)
if i=j
    return (a_i)
else
    m:= $\lfloor (i+j)/2 \rfloor$
    max1:= *findmax* (i,m,$a_i, a_{i+1}, \ldots, a_m$)
    max2:= *findmax* (m+1,j,$a_{m+1}, a_{m+2}, \ldots, a_j$)
    if max1 > max2
        return max1
    else
        return max2

need **f(n)** comparisons

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

need **f(n/2)** comparisons

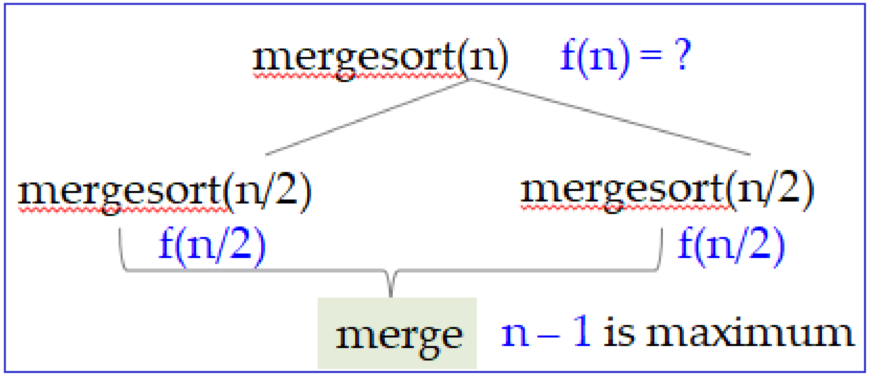
need **f(n/2)** comparisons

Complexity of merge sort

Master Theorem. [...]

f(n) = af(n/b) + cn^d

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



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

a = b^d: f(n) is O(n^{1}logn)

A 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))

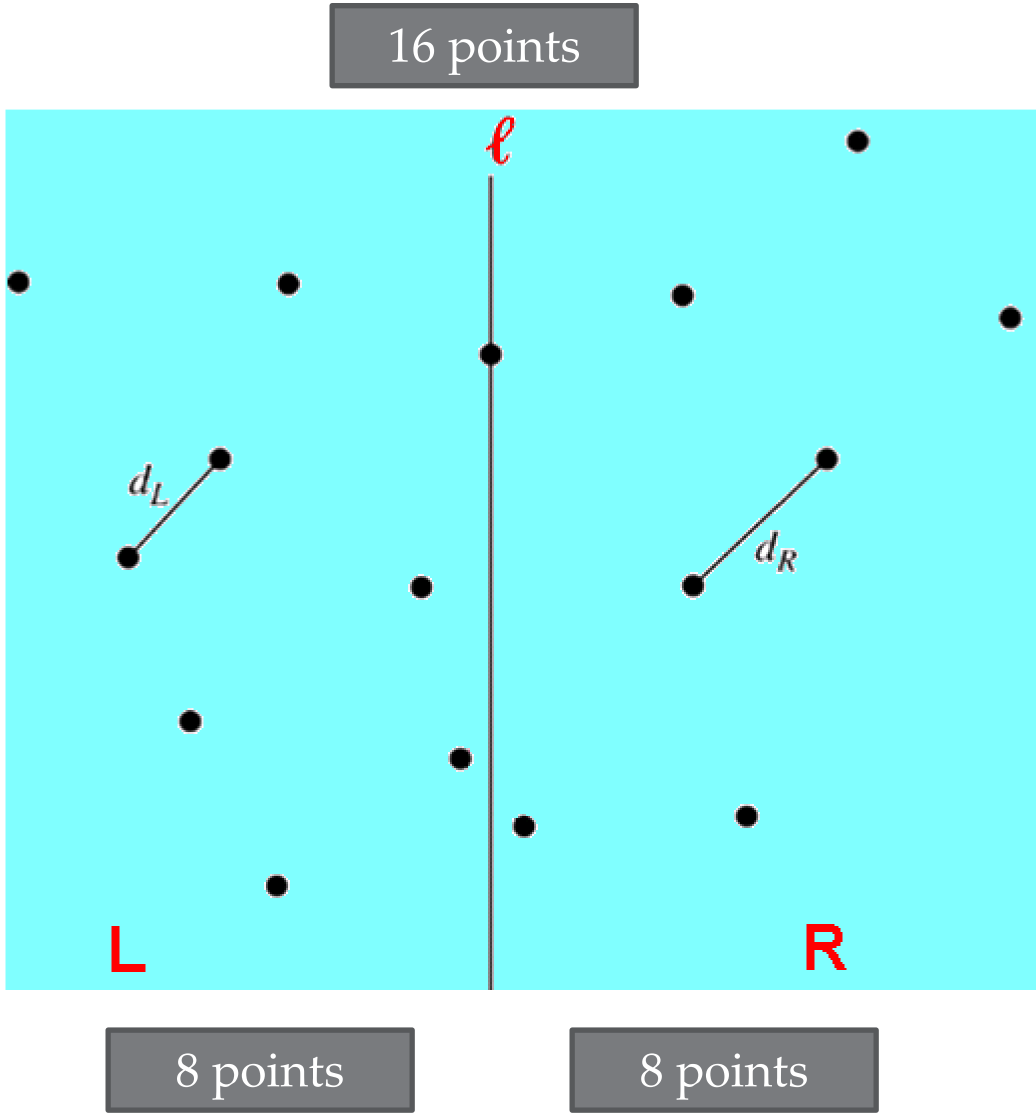
(2) Sorting points in order of increasing y coordinates → O(nlog(n))

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 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)

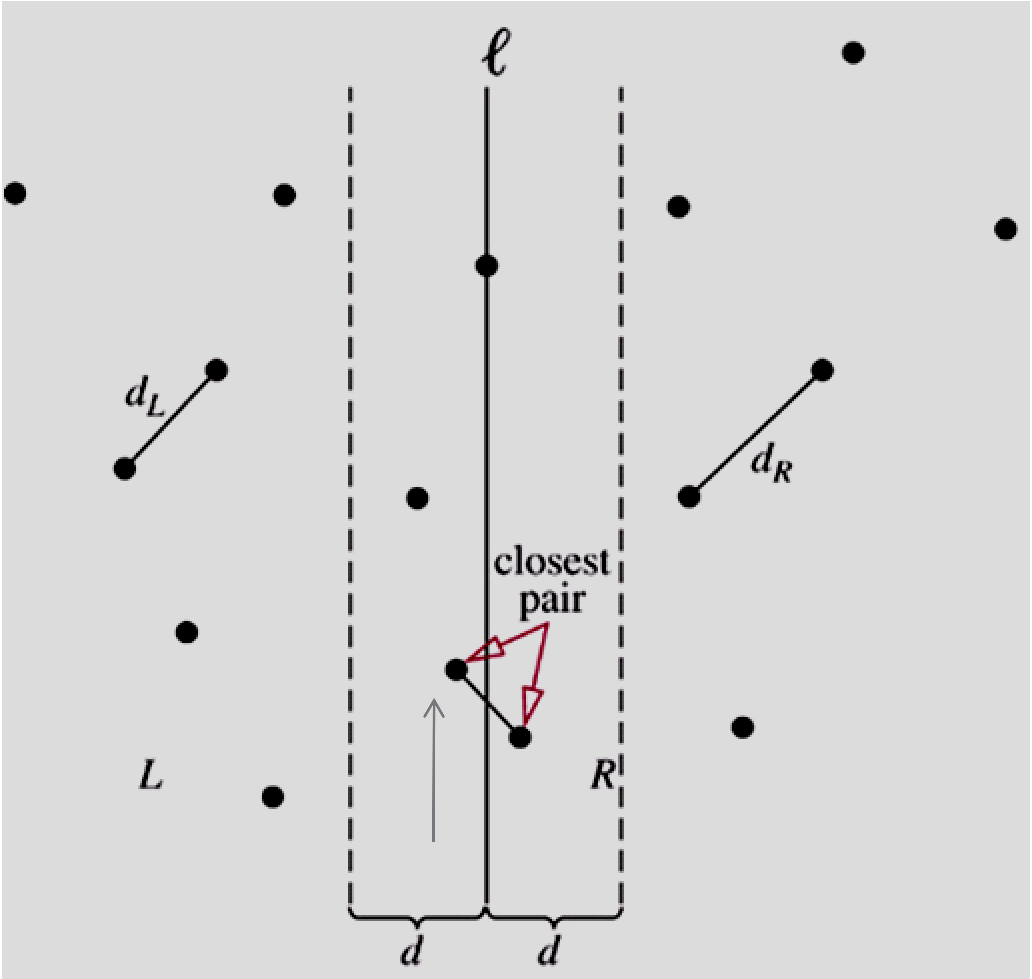
(5) Let $d=\min(d_L, d_R)$



An Demonstration: Closest-Pair Problem

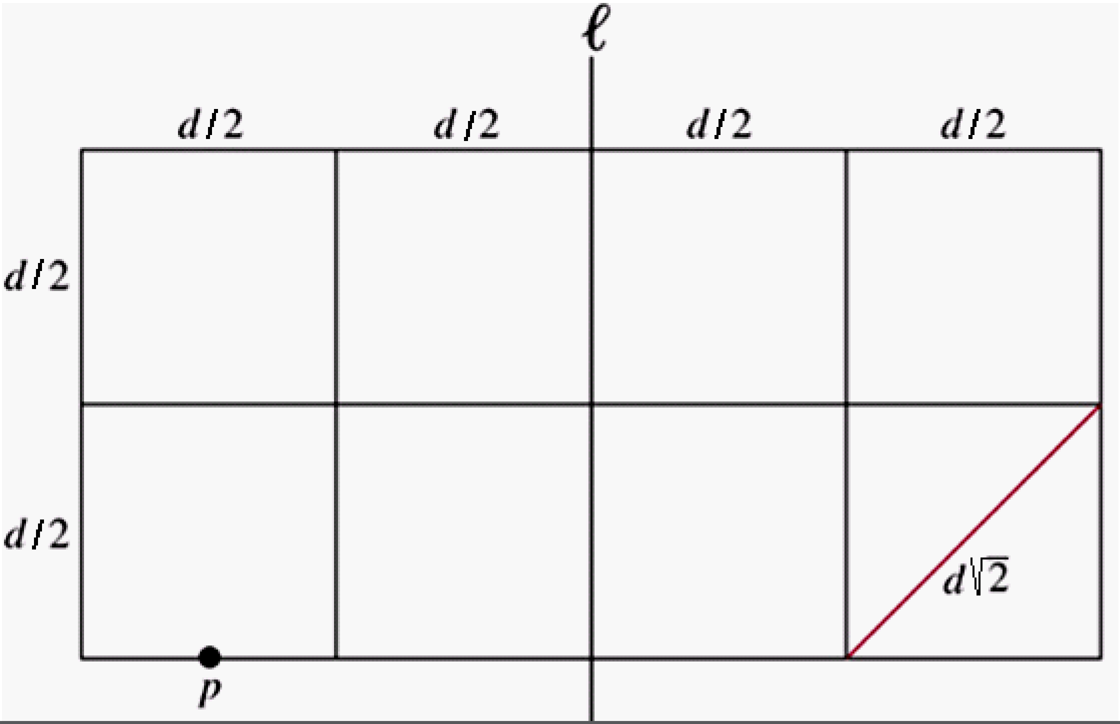
(6) Studying area [l-d,l+d]. This area may be contains the result.

(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 compare 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

- **Basic** counting rules:
    - The Product rule
    - The Sum rule
- **Inclusion-exclusion** principle
- Advanced counting technique: **recurrence relations**
    - **Tower of Hanoi**
    - **Count bit strings of length n that satisfy some properties**
- **Complexity of Divide and conquer** algorithms
    - Finding max, binary search in a recursive version
    - Merge sort

- **THANKS**