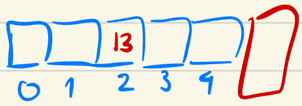
int [a=5]

array → [5, 10, 15, 20]

main (){
 ~
 int arr[5] = {} ← 0 (ค่าขยะ)
 arr[2] = 13
 * arr[5] = ระเบิด
}



sz: 1 2 3 4 (5) → O(n)

TORZY'\0'
0 1 2 3 4 5

string ✓
strlen → ✓ O(n)

```
string str
(1) * for (i=0 ; i< strlen(str) ; i++){      O(n × n) = n^2
        // do sth ← O(n)
    }
(2) for (i=0, n= strlen(str); i<n; i++){     O(n)
        //
    }
```

```
for (i:0 → ∞) {
    if(str[i] == '\0'){
        break
    }
    sz+=1
}
```

จุดสิ้นสุด < สิ่งที่เขาออกแบบ

TLE (Time Limit Exceed) → 10^8 คำสั่ง 1s

$n = 10^4$

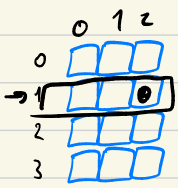
(1) → 10^8 → แตก TLE

(2) → 10^4 → ไม่ติด TLE

100 (1)/(2)
1000 (1)/(2)
10,000 (2)
>10,000 (2)

Array 2D

arr[ ]



arr[1][2]
แถว หลัก

Sort → เรียงจาก น้อยไปมาก

$n = 10^4$ $n = 10^6$

ค่าย 1 $O(n^2)$ ×

ค่าย 2 $O(n \log n)$ ✓ ✓

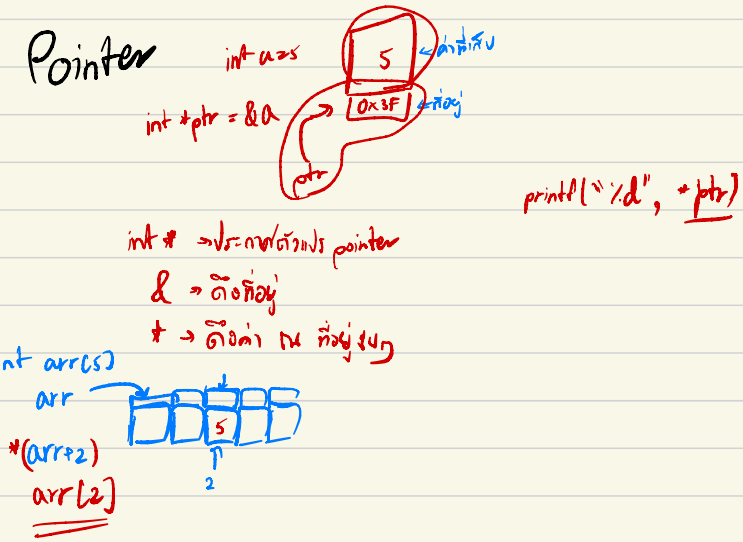
→ Bubble Sort

4 1 3 2
1 3 2 4
1 2 3 4

1 O
2 OO
3 OOO
n OO... OOO

pointer
function → recursive
struct

Pointer



Function

ชนิด ชื่อ (ค่าที่ส่งมา (Parameter)) {

return

}

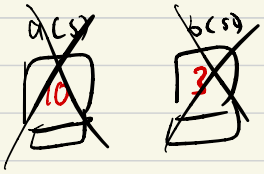
```
int add(int a, int b){
    return a+b
}
x = add(3,5)
```

```
void print(int a, int b){
    printf("%d %d\n", b, a);
    return;
}
```

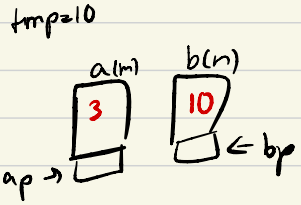
3,5 → 5,3

print(3,5) ← 5 3

```
void swap(int *ap, int *bp){
    int tmp = *ap
    *ap = *bp
    *bp = tmp
}
```




```
main
    a = 10
    b = 3
    swap(&a, &b)
```



factorial

sum 1...n

fibonacci

Recursive Function

factorial → n!

0039 → Permutation

f(n) = n!

f(3) = 3! → 3 × (2!) → 2 × 1!

f(3) = 3 × f(2)

f(2) = 2 × f(1)

f(n) := n × n-1 × n-2 × ... × 1

→ $f(n) \begin{cases} 1 ; n=0 \\ n \times f(n-1) ; n>0 \end{cases}$

```
int fac(int n){
    if(n==0) return 1
    return n x fac(n-1)
}
```

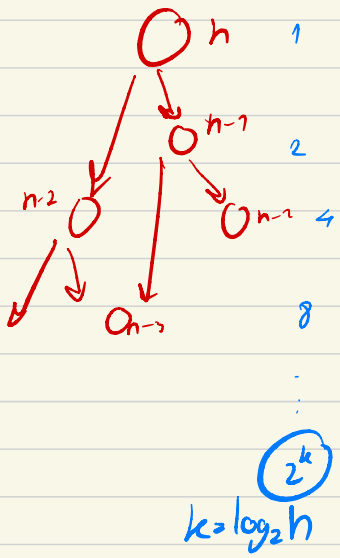
fibo(n) 1, 1, 2, 3, 5, 8 ...

nth 1 2 3 4 5

$fibo(n) \begin{cases} 1 ; n=1 \\ 1 ; n=2 \\ fibo(n-1)+fibo(n-2) ; n>2 \end{cases}$

```
int fibo(int n){
    if( n==1 || n==2) return 1
    return fibo(n-1)+fibo(n-2)
}
```

$O(2^n) < 30$




```
fibo[n+10] = {}
base case ; fibo[1] = fibo[2] = 1
    for(i=3 ; i≤n ; i++){
        fibo[i] = fibo[i-1] + fibo[i-2]
    }
```

O(n)

O(logn) Matrix exponentiation

gen-subset

recursive function

Bitmask

$S = \{1,2,3\}$

gen → { }
{1}
{2}
{3}
{1,2}
{1,3}
{2,3}
{1,2,3}

input : 3
output : _
1
2
3
1 ∪ 2
1 ∪ 3
2 ∪ 3
1 ∪ 2 ∪ 3

	3 2 1
_	000
1	001
2	010
3	100
1 ∪ 2	011
1 ∪ 3	101
2 ∪ 3	110
1 ∪ 2 ∪ 3	111

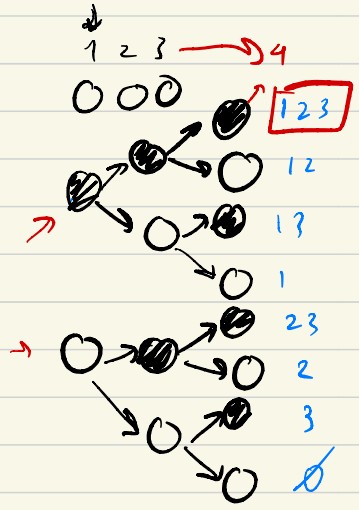
i-th

5 - 101

n

if((n & (1<<i)) != 0)
// i-th bit = 1

5 → 00 101
00 100 &2



1
→ recur(2) →

Trick → เก็บหรือไม่เก็บ

gen subset

```
int pick[5] = {};

void recur(int cur_lv, int lim){
    if(cur_lv == lim+1){
        for(i=1; i<=lim; i++){
            if(pick[i] == 1)
                printf("%d", i);
        }
        return
    }

    pick[cur_lv] = 1
    recur(cur_lv+1, lim)

    pick[cur_lv] = 0
    recur(cur_lv+1, lim)
}
```

Binary Search

1 ~~3 4~~ ~~10~~ (10) (15) (17) ~~30~~

find 15

$l = 0, r = 6, mid = \frac{(0+6)}{2} = 3$

$l = mid+1 = 4, r = 6, mid = \frac{(4+6)}{2} = 5$

$l = 4, r = mid-1 = 4, mid = \frac{(4+4)}{2} = 4$

n

$O(\log_2 n)$

Recursive

Iterative →

```
int arr[0...n-1]
l=0, r=n-1, ans=-1
while( l<=r){
    int mid=(l+r)/2
    if(arr[mid]==val){
        ans=mid;
        break
    else if (arr[mid]<val){
        l=mid+1
    } else {
        r=mid-1
    }
}
if(ans==-1) → Not found
else found at ans
```

2 /22 → (1) จงหาค่าที่น้อยที่สุดที่มากกว่า ~~17~~ 4

(2) จงหาค่าที่มากที่สุดที่น้อยกว่า

↳ A(9) = 4

A(22) = 17

$O(\log n)$

1 3 4 10 15 17 30

(1)
```
l=0, r=n-1, ans=-1
while (l<=r){
    int mid=(l+r)/2
    if( arr[mid] > X ){
        ans=arr[mid]
        r=mid-1
    }else{
        l=mid+1
    }
}
print(ans)
```

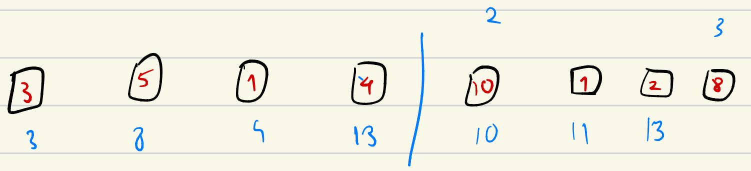
(2)
```
l=0, r=n-1, ans=-1
while (l<=r){
    int mid=(l+r)/2
    if( arr[mid] < x ){
        ans=arr[mid]
        l=mid+1
    }else{
        r=mid-1
    }
}
print(ans)
```

B search ของคำตอบ

จงหาเวลา/เงิน/ sth ที่น้อยที่สุดที่ทำให้เงื่อนไขบางอย่างเป็นจริง

มี ของ n ($1 \le n \le 10^5$) ชิ้น เรียงต่อกัน โดยของแต่ละชิ้นมีน้ำหนัก w_i ($1 \le i \le n$, $1 \le w_i \le 10^9$)

ต้องใส่กล่องตามลำดับ โดยใช้กล่องได้ไม่เกิน K ($1 \le K \le n$) ใบ จงแบ่งให้น้ำหนักที่มากสุดของกล่องให้น้อยที่สุด

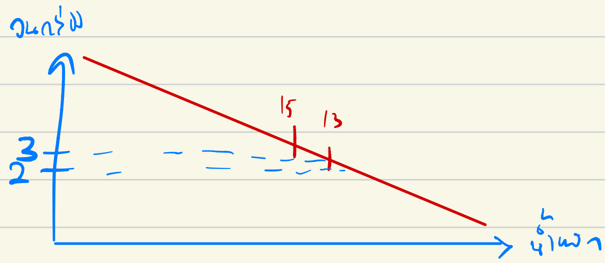


$K = 3$

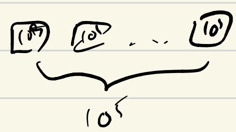
~~30~~ 30 ✓ 45 ✓

15 → >15 ✓

monotonic



$n = 10^5$, $w = 10^4$, $K = 1$



$= 10^{14}$

```
l=0, r=10^14, ans=-1
while (l<=r){            <- O(log2 n·w)
    long long mid=(l+r)/2   <- O(1)

    if( chk(mid) ){      <- O(n)
        ans=mid          <- O(1)
        r=mid-1          <- O(1)
    }else {
        l=mid+1          <- O(1)
    }
}
print (ans)
```

```
bool chk(int mid){
    sum=0, cnt_crate=1
    for(i=1->N){
        if(weight[i] >mid)
            return false
        if(sum+weight [i] > mid){
            cnt_crate ++
            sum =0
        }
        sum += weight[i]
    }
    return (cnt_crate <= K)  -> true
                             -> false
}
```

$O(n \log_2 n \cdot w)$

Sieve of Eratosthenes → หา prime number

prime number 10 → 2 3 9 5 ... 9 → 10 ไม่เป็น prime $O(n) \rightarrow O(\sqrt{n})$

```
for i in(2, n-1):
    if(x%.i ==0)
        not_prime = true
```

10 → 2 3 → 10 ไม่เป็น prime $O(\sqrt{n}) \rightarrow O(\sqrt{n})$

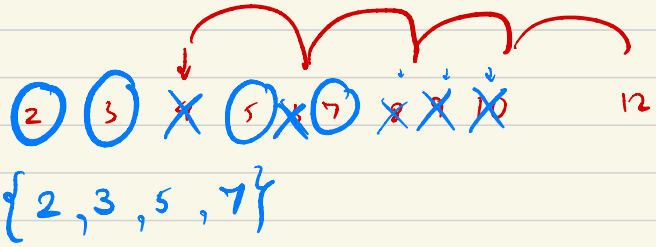
```
for(i=2, lim=sqrt(10)+1; i<=lim; i++){
    if(x%.i ==0)
        not_prime = true
}
```

9 → 3.3 $\sqrt{9}=3$

10 → 2.5 $\sqrt{10} \neq 3$

$O(n \log n)$ → หา prime ทุกตัว ตั้งแต่ [1, n]

{1 2 3 4 5 6 7 8 9 10} → {2,3,5,7} $O(10 \log 10)$



{2,3,5,7}

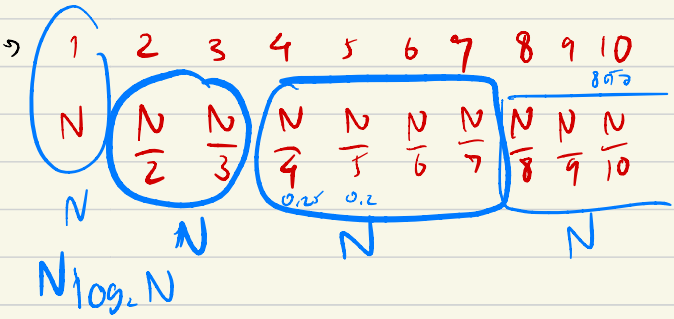
~~bool prime[N]~~ ↳ 1 byte = 8 bit → N byte

bitset<N> prime ↳ 1 bit, N bit ≈ $\frac{N}{8}$ byte

2 3 5 7 11 ...

$\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \frac{N}{7} + \frac{N}{11} + \dots$

```
void sieve(){
    for (i=2; i<=N; i++){
        prime[i] = true
    }
    for (i=2; i<=N; i++){
        if (prime[i] == true){
            for (j=i+i; j<=N; j+=i){
                prim[j] = false
            }
        }
    }
}
```



STL

sort, reverse

vector (dynamic array) - begin/end, rbegin/rend, front, back
[-1]
- push, pop
- size
- insert/erase
- vector<int>:: iterator it → auto it
- vector<int> (10,5) → "5" × 10 slot

queue - front
- push, pop
- size

stack - top
- push, pop
- size

pair/tuple - get<0>
- tie

set (AVL) - insert/erase
- begin/end, rbegin/rend
- size

Left heavy → $H_L - H_R > 1$
Right → 3,2,1
Left-Right → 3,1,2
Right heavy → $H_R - H_L > 1$
Left → 1,2,3
Right-Left → 1,3,2

pq (heap) (heap) - top
- push, pop
- size

insert left-to-right → heapify up
delete replace root with the latest element → heapify down

auto [ ]: map ← map/unordered_map (hash table)

Link list

STL → Standard Template Library

→ Function / Data Structure

→ อยู่ใน bits/stdc++.h

- sort, min, max, reverse
- Data Structure

vector (Dynamic array)

list (Linked List)

queue

stack

set (AVL tree)

map (AVL tree)

unordered_map (hash table)

priority_queue (heap)

pair, tuple → เก็บข้อมูลหลายอย่าง

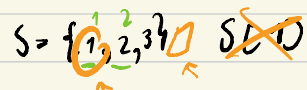
Set / Map

set

$S = \{1, 1, 2, 4\} = \{1, 2, 4\}$

- ไม่มีตัวซ้ำ
- เรียงค่าจากน้อยไปมาก

S.insert (3) → O(logn)
S.insert (1)
S.insert (2)



S.erase (3) → O(logn)

O(log n)
1, 2, ✗, 4, ✗, -3

O(n) → หาค่าที่น้อยที่สุดลำดับที่ k ; k ≤ n

print (S.size()) → O(1)

```
for (int &s : S){
    printf("%d\n", s)   // 1
    s = 5               // 2
}                       // 3
```

iterator → pointer ของ stl

```
for (set<int>::iterator it = S.begin(); it != S.end(); it++){ → ไปข้างหน้า
    print (*it)
}
```

S.rbegin(); it != S.rend(); it++ → ถอยไปหลัง

```
for(auto it = S.begin() ; it != S.end() ; it++){
    print(*it)
}
```

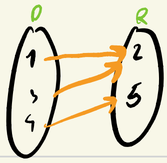
```
for ( auto it : S){      // it → int type
    print(it)  // 1, 2, 3
}
```

set<int> S = {1, 2, 3} หรือ set<int> S

insert (x) } O(logn)
erase (x)

size()
begin()
end()
rbegin() → reverse begin → *(S.rbegin()) = ค่าที่มากที่สุด
rend() } O(1)

Map



print (map[1]) → 2
print (map[3]) → 2
print (map[4]) → 5

mp = {(1, 2), (3, 2), (4, 5)}

(key: 1, value: 2)

```
map<string, int> fav;   //เก็บ จน. เด็กที่ชอบสัตว์ แต่ละชนิด

    fav["ants"] = 5          print (fav["ants"]) → 5
    fav["elephants"] = 2
    print(fav["ant"]) → 0

    // {("ants", 5), ("ant", 0), ("elephants", 2)}

for(auto &[key, val] : mp){
    print (key, val)
}
for (auto it = mp.begin(); it != mp.end(); it++){
    print(it->first) //ants
    print(it->second) //5
}
```

(*it).first → ("ants", 5)
(*it).second

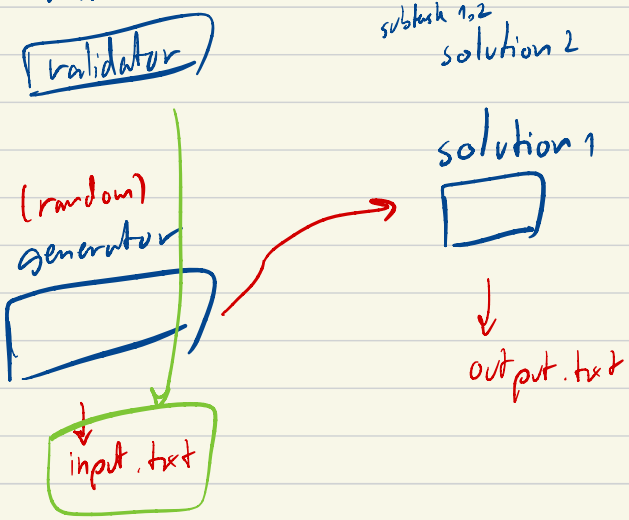
mp.erase("ants")

map <int, int> mp

- add key O(logn)
- erase key O(logn)
- size()
- begin()
- end()
- rbegin()
- rend()

size(), begin(), end(), rbegin(), rend() → O(1)

Gen Tmd

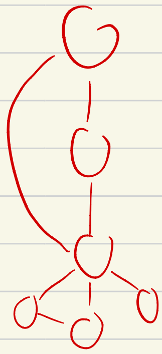


subtask 1 → $N \leq 100$
2 → $N \leq 10^4$
3 → $N \leq 10^5$

Parallel B search

weight
ใช้ dfs บน undirected G ↗ visit
ได้ directed G ↘ parent ใช้เหมือน tree

if(v==p)con
if(!vi[v]){
 dfs(v,u)
 low[u]=min(low[u],low[v])
}else{
 low[u]=min(low[u],disc[v])
}



Top down (Iterative)
Bottom UP (Recursive)

Push
Pull

DFS tree → Bridge
Arti → 2 cases
SCC? (ไม่ทัน)

Fen / Seg

Matrix Expo

QS
Knapsack
LIS
LCS
MCM
Partitioning (K-subarray)

Digit DP
Bitmask
DAG

1. What is dp? ✓
2. Recursive Vs Iterative ✓
3. Push Vs Pull ✓

Slime
Mixture

1. Stair ✓
2. Coin combination ↗ Double Counting ✓
3. Wine line (2,4,6,2,5) ✓
5. Frog ↗2 ก้าว ↗x ก้าว
4. Grid ✓
6. MCM

state space

	6	3	4	7	5	6
bfid	-1	-1	2	3	3	5

mxp	~~6~~ 3	4	~~7~~ 5	6
idx	~~1~~ 2	3	~~4~~ 5	4

DP 2D ✓
LCS ✓
LIS → 1151, cat, orchid
MCM / Range DP
Knapsack → max(dp(i-1, wei), dp(i-1, wei - w(i)) + val(i))
Coin Change ✓
Grid ✓
Island ✓