Big O notation

for : 1 → N
  // do something

รับค่า N N = 100 ↙ 100รอบ O(n)
for N รอบ
  รับค่า → a[i] ~ 10^8

$n < 10^8$ O(n)

$O(n^2)$

for : 1 → N :
  for : 1 → N :

1: N
2: N
⋮
N: N

$O(N^2)$

$n < 10^6$
$q < 10^6$

Quicksum

i :	1	2	3	4	5	6	7
a[i]:	1	2	6	7	8	9	13
qs[i]:	1	3	9	16	24	33	46

3→5

1→5
1→2

qs_i = sum 1 → i

$sum_{l \to r}$
$= qs_r - qs_{l-1}$

$sum_{1\to5} - sum_{1\to2}$
$= sum_{3\to5}$

O(n) ⇒ O(1)

N
a_1 a_2 a_3 ... a_n
Q
l_1 r_1
l_2 r_2
⋮
l_Q r_Q

$n < 10^6$
$q < 10^6$

Complexity Brute-force	using quicksum
O(n)	O(n)
	qs init: O(n)
O(n), O(n), ⋮, O(n) } q รอบ	O(1), O(1), ⋮, O(1) } q รอบ = O(q)
O(n q)	O(n+q)
$\sim 10^6 \cdot 10^6 = 10^{12}$ ops.	$\sim 10^6 + 10^6 = 2 \cdot 10^6$ ops.

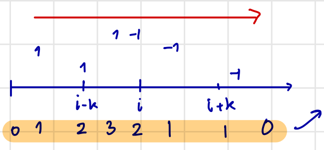
Reverse Quicksum (Sweepline Algorithm)

x[i-k] += 1
x[i+k+1] -= 1

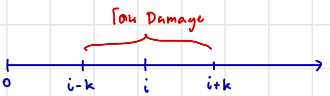
O(2) = O(1)
⇒ O(n)

Bomb

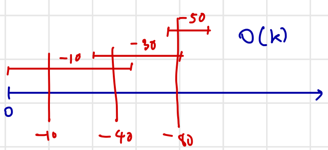
ตั้งระเบิดที่ i ระเบิด พลัง = k



O(n)
O(2n)



O(nk)



O(k)

Merge Sort ⇒ inversion count

std::sort → in cpp

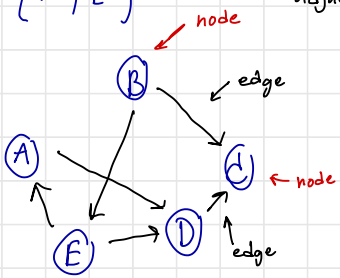
$i > j$ $a_i < a_j$

i	1	2	3	4
a_i	1	3	2	4

3 2

그래프 (graph)

$G = \{ N, E \}$



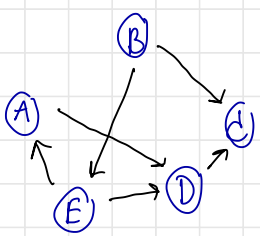
adjacency list

graph[A] = { D }
graph[B] = { C, E }
C = { }
D = { C }
E = { A, D }

w → for in graph[w]

graph traverral dfs, bfs

dfs (depth - first search)

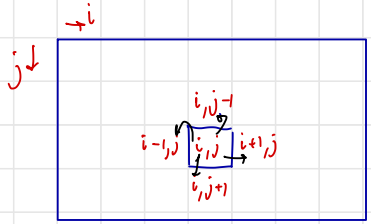


dfs(A)

```
bool visited[N] = {false}

void dfs(int w) {
    if (visited[w]) return;
    visited[w] = true;
    for (int v : g[w]) {
        dfs(v);
    }
}
```

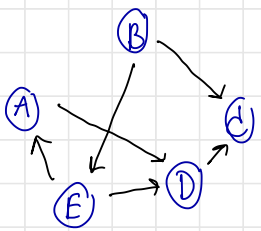
flood fill




```
void flood_fill(int i, int j){

flood_fill(i-1, j)
 — n — (i+1, j)
 — n — (i, j+1)
 — n — (i, j-1)
```

bfs (breath-first search)



dfs ⟷ bfs

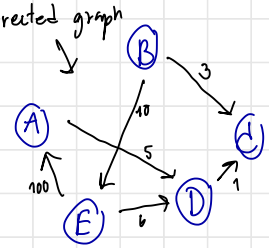
```
queue<int> q;
while(not q.empty()){

}
```

MST (Minimum Spanning Tree)

directed graph

weight = ความยาวถนน

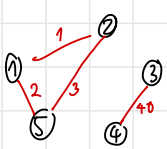


g[A] = {(D,5)}
g[B] = {(C,3), (E,10)}
⋮
vector<pair<int,int>> g[maxN];

เลือกเส้นที่ไม่เชื่อมกันทุก node
โดยที่ sum(w_i) น้อยที่สุด

undirected graph Kruskal's algorithm #

① เอา Edge ทั้งหมดมา sort ตาม w
② เอา Edge ที่จำเป็นใส่กราฟ

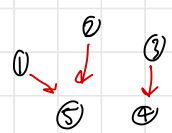


DSU (Disjoint-Set Union)

merge(1, 2)
merge(1, 2)

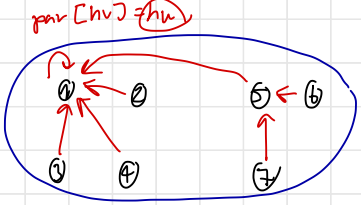
par[i] = i
par[1] = 1

fi(int w
return



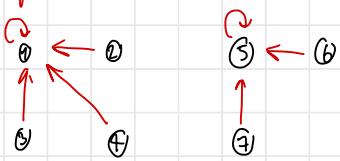
fi(1) = 5
fi(2) = 5

```cpp
bool merge(int u, int v){
    int hu = fi(u);
    int hv = fi(v);
    if(hu == hv) return false;
    par[hv] = hu;
    return true;
}
```




```cpp
int fi(int u){
    if(par[u] == u) return u;
    return par[u] = fi(par[u]);
}
```

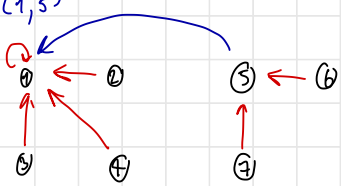
path compression → ทุกจุดที่ fi ยิงผ่าน จะชี้ไป representative node



★ Union by size

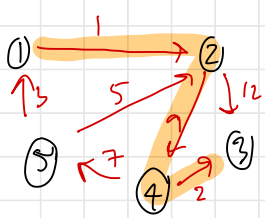
$O(\log^* n) \sim O(1)$

merge (1,5)



fi(2) = 1

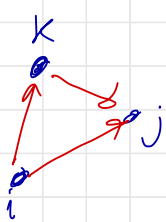
APSP (All-Pairs Shortest Path) → Floyd-Warshall Algorithm



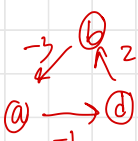
SP 1→3 = ?
1+9+2
= 12

long long dist[N][N];
dist[i][j] เก็บ SP on i→j

set i→j: dist[i][j] = ∞ (2e18) ★
dist[i][i] = 0 ★

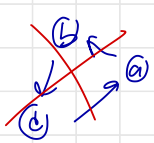


Negative cycle



วน loop

DAG graph = OK
(Directed Acyclic Graph) ★



★ for k: 1→N {
  for i: 1→N {
    for j→N {
      dist[i][j] = min (dist[i][j]
        , dist[i][k] + dist[k][j]

  }
}
}

Building Roads 2 test case Analysis

in:

4 2

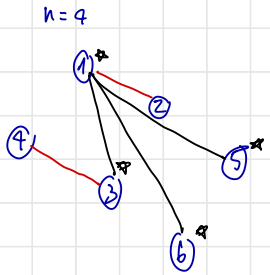
1 2

3 4

out.

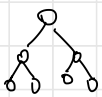
1

2 3



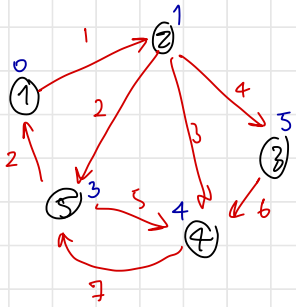
SSSP (Single Source Shortest Path) $O(V + E\log_2 V)$

Heap



priority_queue <int>

- push(u) O(log V)
- pop() O(log V)
- top() O(1)



	dist[N]
1:	0
2:	1
3:	
4:	
5:	3