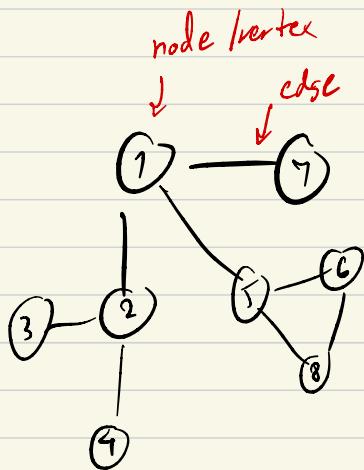


Graph

→ BFS, DFS
SSSP, Dijkstra

$G_2(V, E)$



ก ສົວ ດັບດົນຂອງ V ໃຊ້ E

V ສົວ ພົມຂອງ node (ຄູນຫາ) $V = \{1, 2, 3, 4, \dots, 8\}$

E ສົວ ພົມຂອງ Edge (ສິນເຊື່ອ) $E = \{(1, 7), (1, 2), (1, 5), (5, 6), (5, 8), (6, 8), (2, 3), (2, 4)\}$

Edge → ຂະກິດການ (Directed)

$1 \rightarrow 2$ 1 ອຸນ 2 ຖົບ
2 ອຸນ 1 ນຳໃກ້

→ ຂະກິດການ (Undirected)

$1 - 2$ 1 ອຸນ 2 ບົບ
2 ອຸນ 1 ບົບ

Indirect
Graph

Degree ສົວ ຈຳເລັກຂອງກີດຈອດ ທີ່
 $\star \text{Edegree} = 2|E|$

$\begin{aligned} \deg(1) &= 4 \\ \deg(4) &= 2 \\ \deg(3) &= 1 \end{aligned}$

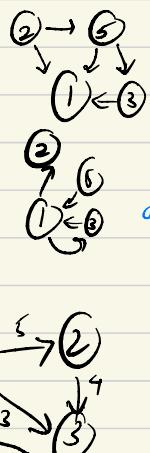
Directed
Graph

In-degree ສົວ ຈຳເລັກກີດຈອດ

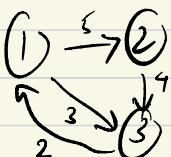
$\begin{aligned} \text{indeg}(1) &= 3 \\ \text{indeg}(3) &= 1 \end{aligned}$

Out-degree ສົວ ຈຳເລັກກີດຈອດ

$\begin{aligned} \text{outdeg}(1) &= 2 \\ \text{outdeg}(2) &= 0 \end{aligned}$



Weight ສົວມັນຄວາມ



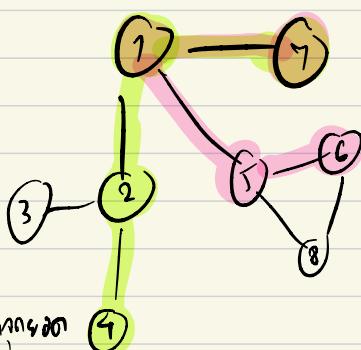
Path ສົວກາດີນ

$4 \rightarrow 2 \rightarrow 1 \rightarrow 7 \rightarrow 1 \rightarrow 5 \rightarrow 6$

Path ສົວລົດທອງດຽວ

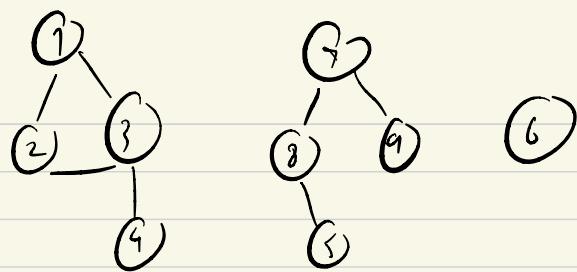
$P = (4, 2, 1, 7, 1, 5, 6)$

- Simple Path ສົວກາດີນທີ່ໄວ້ທົກຈອດ
 $P = (4, 2, 1, 7)$



↪ #Component = 1

Component

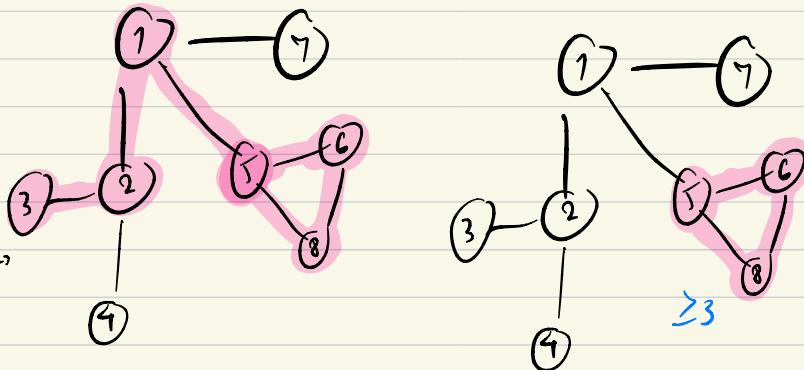


#Component = 3

(5, 8, 6, 5, 1, 2, 3, 2, 1, 5)
Cycle/Loop កំពុង សម្រាប់
តែកំពុង path ទៅរវាងនាមភាព

Simple loop

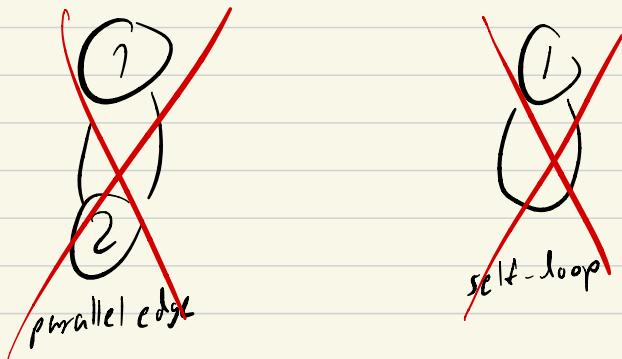
សំណើ cycle ក្នុង node នៃនៅ node ផ្លូវ
(node នៅលើលីបី)



Self loop



Simple Graph - graph នៃវិធាន parallel edge និង self-loop



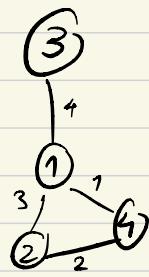
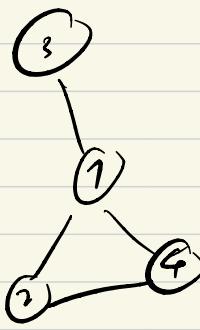
โครงสร้าง Graph

- Adjacency matrix

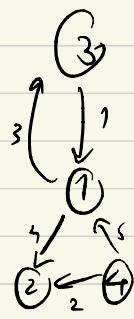
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 1 \\ 3 & 0 & 0 & 2 \\ 4 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \end{bmatrix}$$

$$G[1][1] = 0$$

$$G[1][2] = 1$$



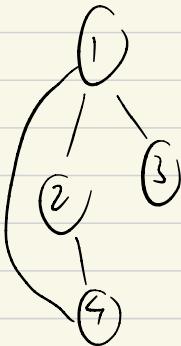
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 4 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 5 & 2 & 0 & 0 \end{bmatrix}$$



$$V[E] \rightarrow \begin{bmatrix} 4 & 5 \\ 1 & 3 & 3 \\ 3 & 1 & 1 \\ 1 & 2 & 4 \\ \vdots \end{bmatrix}$$

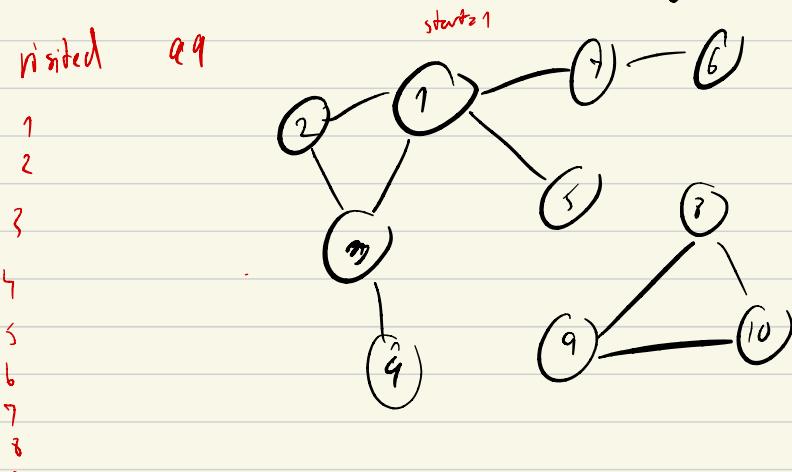
- Adjacency list (Vector)

1	→ 2, 3	vector[1] = {2, 3}
2	→ 1, 4	vector[2] = {1, 4}
3	→ 1	vector[3] = {1}
4	→ 2, 1	vector[4] = {2, 1}



Graph Traversal → BFS / O(V+E) → DFS / O(V+E)

BFS (Breadth -First Search) - queue



```
bool visited[N] = {0};  
void bfs(int start) {
```

queue<int> qq
qq.push(start)

```

while (!qq.empty()) {
    int v = qq.front();
    qq.pop();
    visited[v] = true;
    for (int w: G[v]) {
        if (!visited[w]) {
            qq.push(w);
            visited[w] = true;
        }
    }
}

```

3

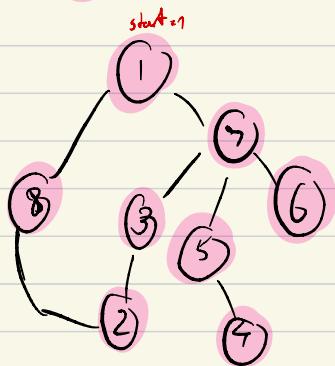
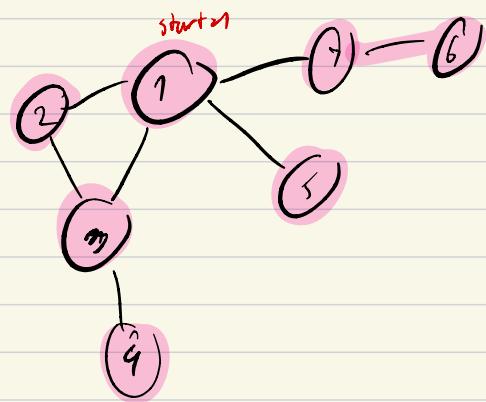
int main();

for ($v \in V$, $v \notin \text{visited}[v]$)
 do $\text{fs}(v)$

2
1

$O(V+E)$

DFS (Depth-First Search) → recursion



void dfs(int v){

if (!vis[v]) return;

vis[v] = true;

for (v : G[w])

dfs(v);

}

int main () {

for (u=1; u<=V; u++)

dfs(u);

}

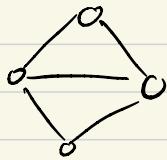
$\times 3 / 2$
 $a \rightarrow \frac{x_3}{2}$
 $b \rightarrow \frac{x_3}{2}$

$a=11 \quad b=15$

$11 \xrightarrow{1/2} 5 \xrightarrow{x_3} 15 = 2$
 $11 \xrightarrow{2} 33 \xrightarrow{1/2} 16 \xrightarrow{x_3} 2 = 15$

$11 \xrightarrow{x_3} 33 \xrightarrow{\text{cont}=1} 5 \xrightarrow{a,b \leq h} *O(k)$

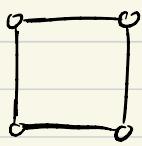
Euler path



ເຄີຍຄວນຖ່າງ edge ຕາງໆນີ້ edge

deg ດີນເທົ່າ 2 ເກົ່ານັ້ນ ທີ່ໄວ້ຈະເປັດ

Euler circuit



ເຄີຍຄວນຖ່າງ edge ຕາງໆນີ້ edge ໂລກກວ່າມທີ່ດູ

deg ນັ້ນ node ເປົ້າງວັນຍຸດ

Tree

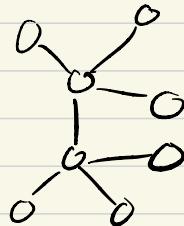
special case von graph

3 ວິວ 1. ຂະໜາ n-1 edge

2. ປົກສະເໝັດ

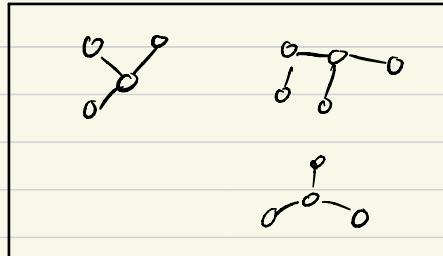
3. ຜົບປົວຕົວ (Connected component)

- ຖືກອີງ (u, v) ຢ່າງ graph ຈະ 1 path ໃຫ້ເກີນ



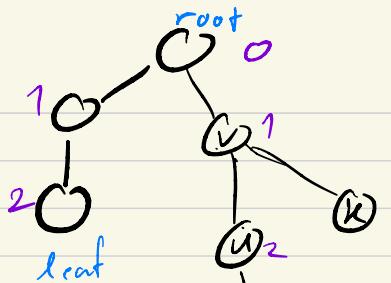
Forest

graph ສີແລ້ວ tree ລວມກັນ



Terminology

root - ດັບຕົວຕົວຕົວ



leaf - node ຖ້າມ

parent - node (v ບໍ່ u parent v $\sim u$)

ancestor - node ນີ້ node u $\sim u$

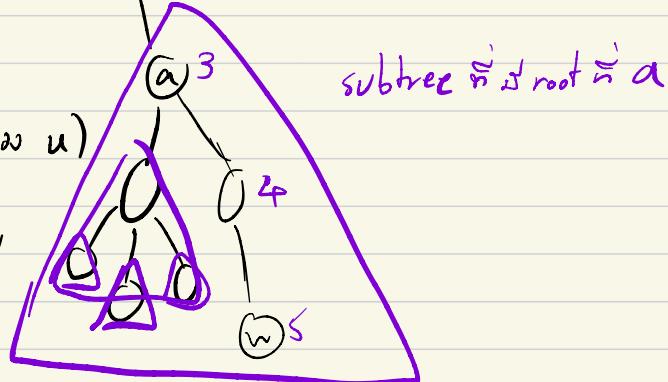
siblings - node ສະກິດ parent ຢັ້ງ

child - ລຸ່ມ (u ບໍ່ v child u $\sim v$)

descendant - node ນີ້ node $\exists v$ v

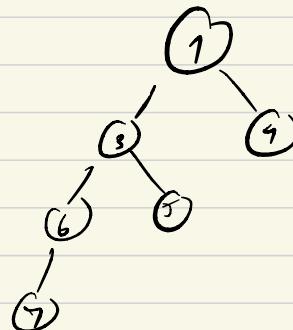
depth - ອາກສິກ

subtree - ຕັ້ງໄຟລະຍຸ



Binary Tree

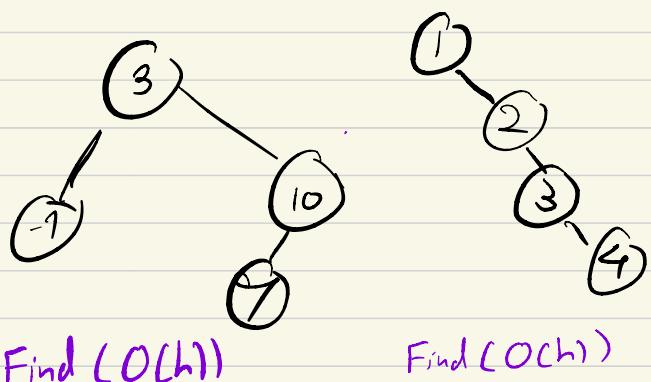
node ຫຼື node ຂອງ ≤ 2 nodes



Binary Search Tree

node ພາວກໍາ $<$ node w

node w $<$ node ອົບອາ



Balanced Search Tree

Self-Balanced BST (AVL Tree)

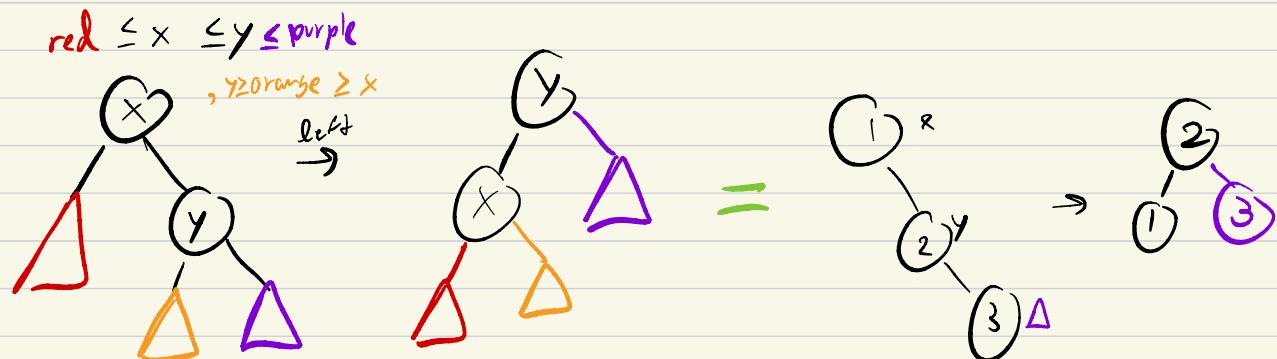
» set, map ^{9.4 STL}

- Min Find, Add, Delete ϑ ($O(\log h)$)

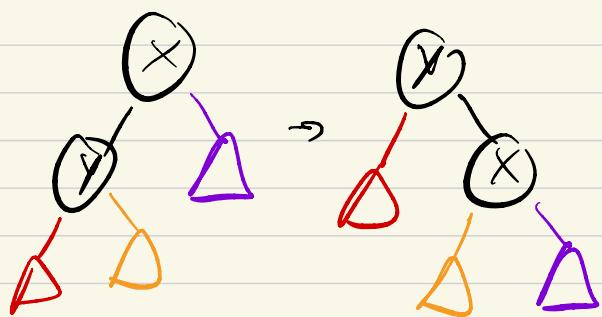
Balance: $\text{abs}(\text{dep}(LC) - \text{dep}(RC)) \leq 1$

↳ Operations: Left Rotation, Right-left Rotation \diamond
Right Rotation, Left-right Rotation \times

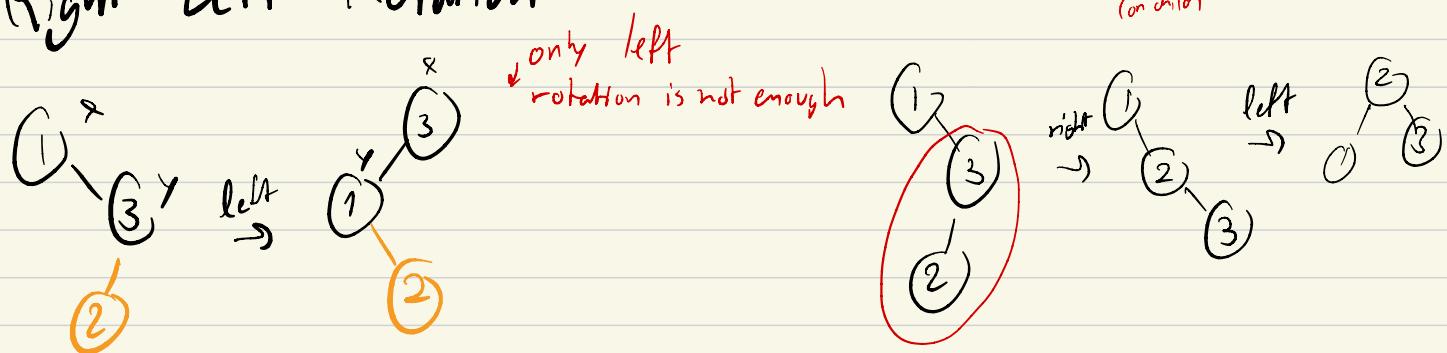
Left Rotation



Right Rotation



Right-Left Rotation



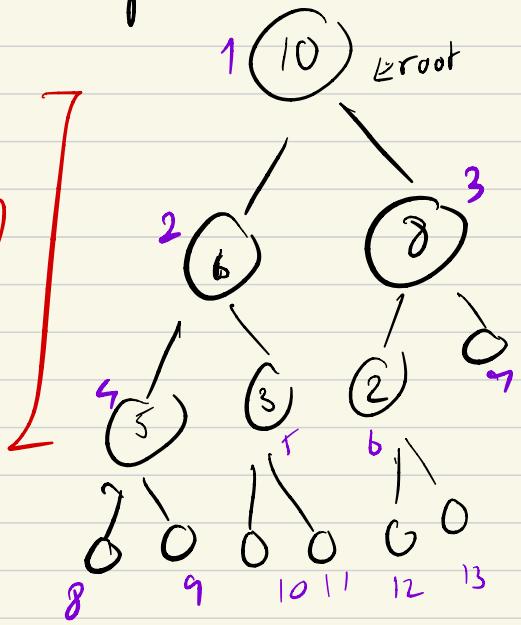
Heap (Priority Queue)

- min, max

- BT

- parent is in min/max child

Max-heap



7, 2, 3, 5, 4, 6

Insert $O(h) = O(\log n)$

heapify up

Top $O(1)$

Pop $O(h) = O(\log n)$

replace root with last element

and heapify down

7, 5, 2, 3, 4

biggest \rightarrow biggest

Bubble $\rightarrow O(n^2) \times$

Mergesort PQ \rightarrow Multiset Bubble

Merge $\rightarrow O(n \log n)$

PQ (Heap) $\rightarrow O(n \log n)$

Multiset (AVL) $\rightarrow O(n \log n)$

Flood Fill (BFS)/DFS

G			P	
G	G	NP	P	P
G	B	B	NP	
R		B		
R	R			

1	2	3	
G	1	2	3
12	23	3	
1	4	3	
R	1	2	

$O(R \cdot C)$

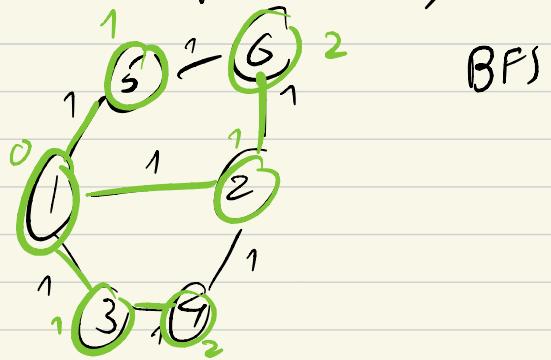
$O(k \cdot R \cdot C)$

Shortest Path

Dijkstra

Floyd Warshall

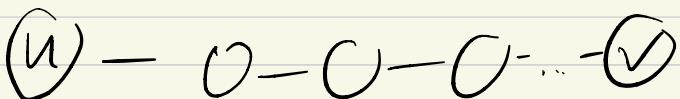
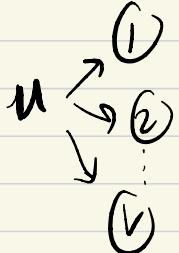
shortest path where every edge's weight is the same



Dijkstra $O(E \log E)$

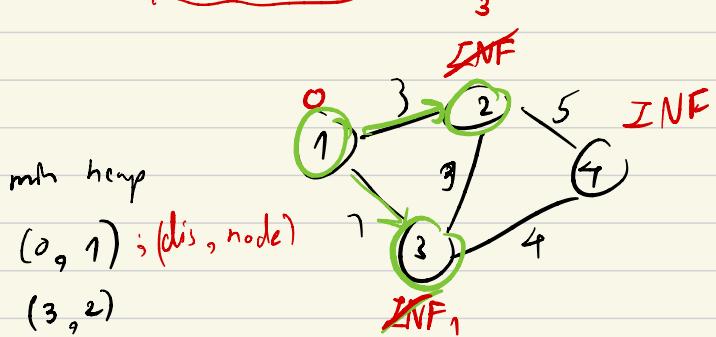
- min shortest path min $n^2 \log n$ node vgraph

- shortest path contains at most $V-1$ edges



Bellman Ford / SPFA

- negative cycle can crash the algo

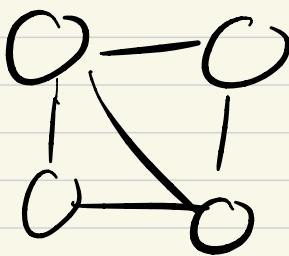


```

pq.push ({0, 1})
while (!pq.empty()) {
    auto [u, total_dis] = pq.top(); pq.pop();
    if (dis[u] >= total_dis) continue;
    for (auto [v, w]: g[u]) {
        if (dis[u] + w < dis[v]) {
            dis[v] = dis[u] + w
            pq.push ({dis[v], v});
        }
    }
}

```

Dense graph



$$E = \frac{V(V-1)}{2} \approx V^2$$

$$O(V^2 \log V)$$

for (V : row) {

 for ($i \geq V$) {

$\text{graph}[i][\text{row}] = \text{graph}[\text{row}][i] = \text{dis}[i]$

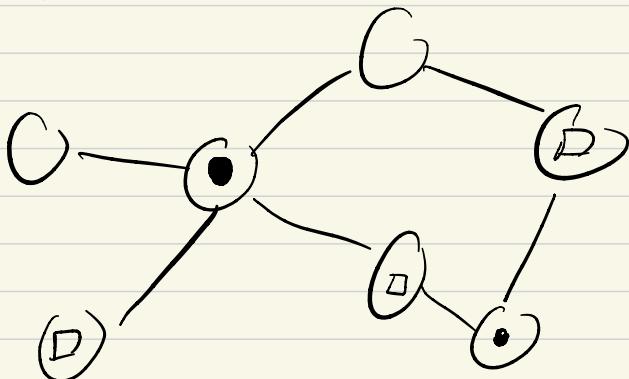
 }

 for ($v, w : \text{list}[u]$) {

$\text{dis}[v] = \text{dis}[w] = \infty$

 }

Multisource



Multi state

in shortest path on $1 \cup V$ ຕາມນັກງານ ໂດຍ

+ ດີເລີກວິທີກົດປົກກົດ ແລ້ວກົດປົກກົດ ເພື່ອເປັນ state

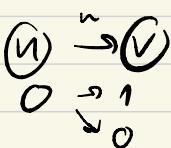
+ ປົກກົດປົກກົດ

$\text{dis}[u](0) := \text{SP}$ ຈາກ u ໃນ $1 \cup V$ ແລ້ວກົດປົກກົດ

$\text{dis}[u](1) := \text{SP}$ ຈາກ $1 \cup V$ ໃນ $1 \cup V$ ແລ້ວກົດປົກກົດ

$$\text{dis}[u](0) \xrightarrow{\text{dis}[v](0) + w} \text{dis}[v](0)$$

$$\text{dis}[u](1) \xrightarrow{\text{dis}[v](1) + w} \text{dis}[v](1)$$

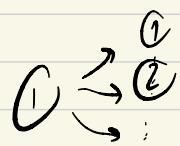


```

auto it = G[u].begin →
for (auto [v, w] : G[u]) {
    if (dis[w][0] + w < dis[v][0])
        relax
    if (dis[w][1] + w < dis[v][1])
        relax
    if (dis[w][0] < dis[v][1])
        relax
}

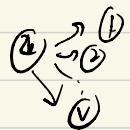
```

Floyd Warshall (All pair shortest path)

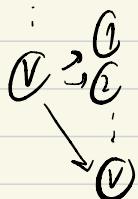


- in shortest path 3 uses node

- $O(V^3)$; $V \geq 1000$ min



- Adjacency Matrix



in V, E

for ($v : 1 \rightarrow V$) $G[v][v] = 0$

for ($l : 1 \rightarrow E$) {

int v, v, w

or v, v, w

$G[u][v] = \min(G[u][v], w)$

$G[v][w] = \min(G[v][w], w)$

$$G[u][v] = \begin{cases} 0; u = v \\ \text{INF}; u \neq v \end{cases}$$

}

for ($h : 1 \rightarrow V$)

for ($i : 1 \rightarrow V$) {

for ($j : 1 \rightarrow V$) {

$G[i][j] = \min(G[i][j], G[i][h] + G[h][j])$

}

}

ຖរណី SSSP

បង្កើតនៃវិធាន់

ព័ត៌មានអារគ្គការដែលមានការសម្រេចចាប់ពី $S \in V$ ទៅកំណត់លើកដែលត្រូវការពិនិត្យ k

S, E នឹងត្រូវបានការពិនិត្យ

$Q \leq 10^5$ ជាពីរ

$1 \leq k \leq V$

$V, E \leq 10^5$

$S \rightarrow$ រូបភាព

$E \rightarrow$ រូបភាព

dis_1 និង dis_2 នឹង S និង E

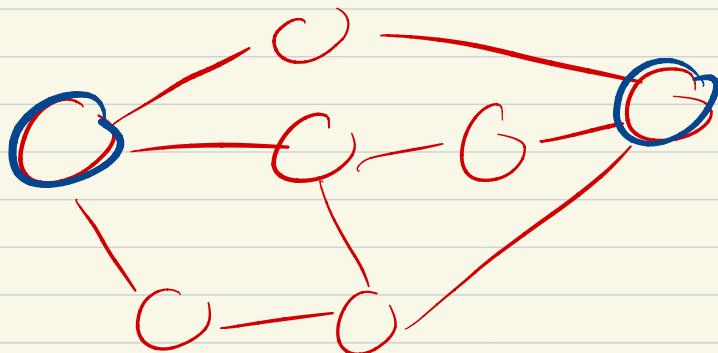
while ($Q \neq \emptyset$) {

 int h ;

 cout $\ll dis_1[h] + dis_2[h]$

} $\overbrace{S \mapsto h}$ $\overbrace{h \leftarrow E}$

$S \mapsto E$



បញ្ជាក់ថា edge នេះ

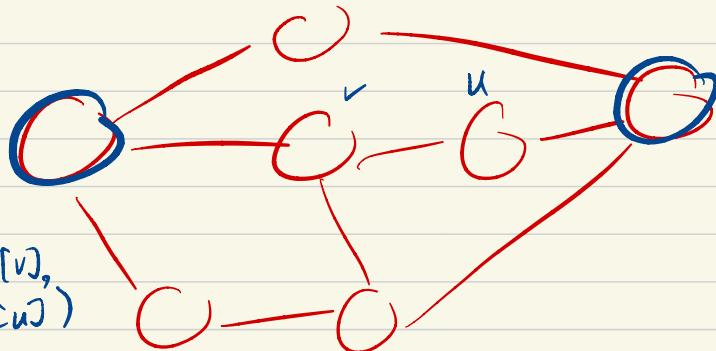
run dijkstra 2 នៃ S និង E

while ($Q \neq \emptyset$)

 int u, v ;

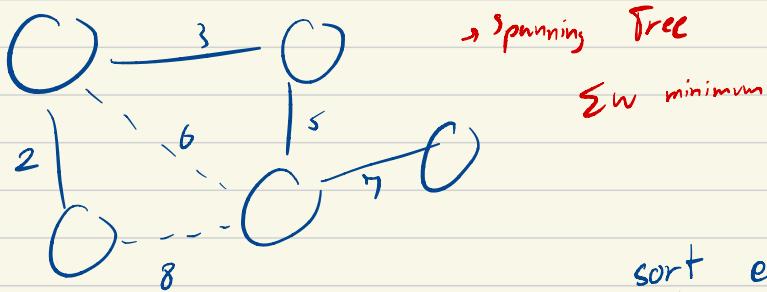
 cout $\ll w + \min(dis_1[u] + dis_2[v],$
 $dis_1[v] + dis_2[u])$

}



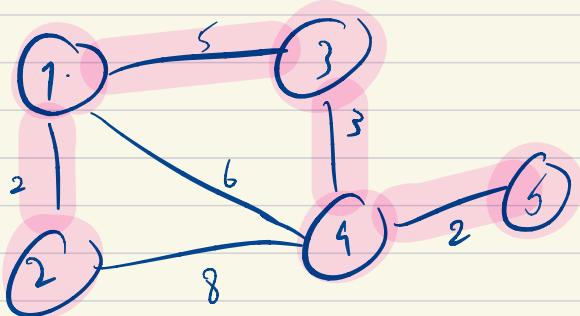
MST (Minimum Spanning Tree) \rightarrow (Kruskal) $\&$ DSU
 Maximum \rightarrow (Prim)

Spanning Tree - Tree in a Graph



$O(E^2)$
 $O(E(V+E))$

sort edge by weight $\rightarrow E$
 If u, v in different component "BFS"
 $O(V+E)$



1. Union of V components
2. If merging two components \Rightarrow component $\cup 1$
3. Union of $V-1$ times

$$2+5+3+2 = 12$$

CODE

```
for(i : 1 to V) par[i] = i // important
```

sort edge

```
for (auto [w, u, v] : edges) {
    if (merge(u, v)) {
        ans += w
    }
}
cout < ans
```

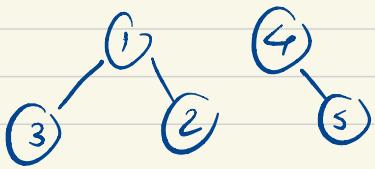
DSU (Disjoint Set Union)

2. $\xrightarrow{m \in O(h)} O(1) \rightarrow O(\log n)$
 $\xrightarrow{\text{merge (Union)}} O(\log n)$

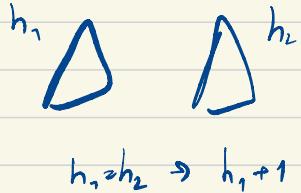
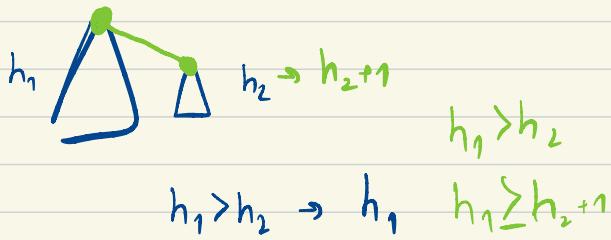
$n \log n \rightarrow \log n$

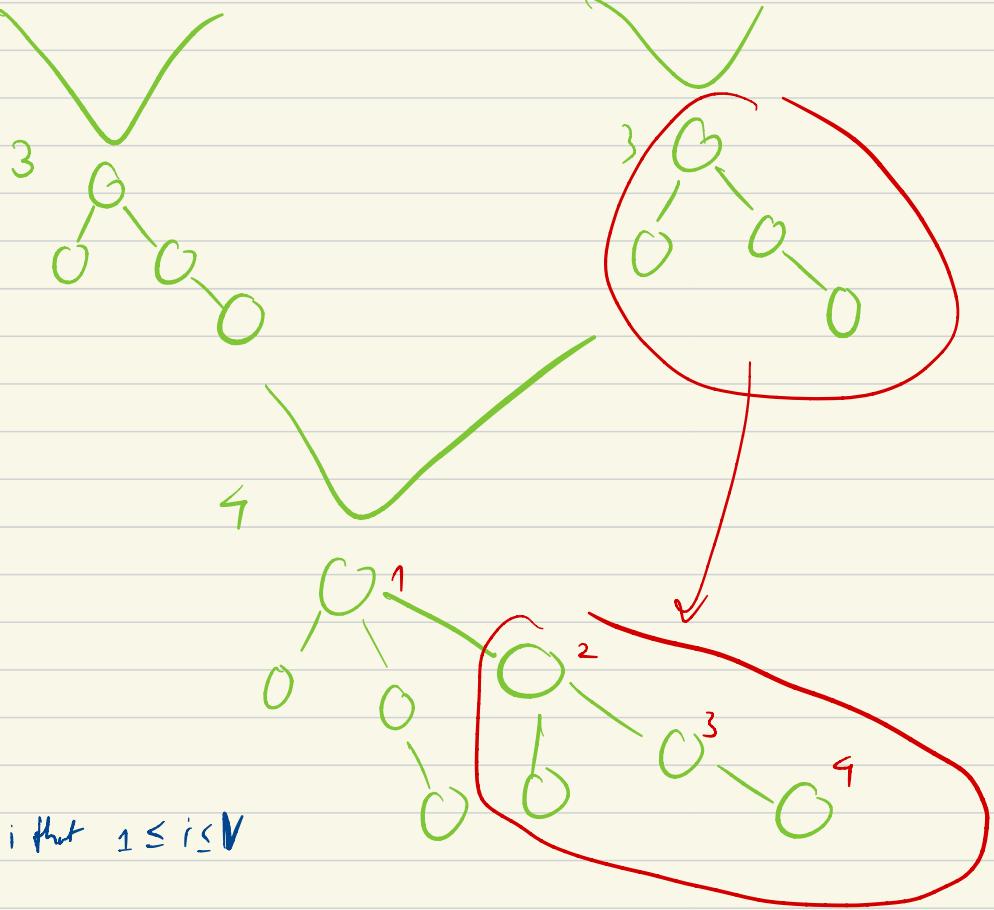
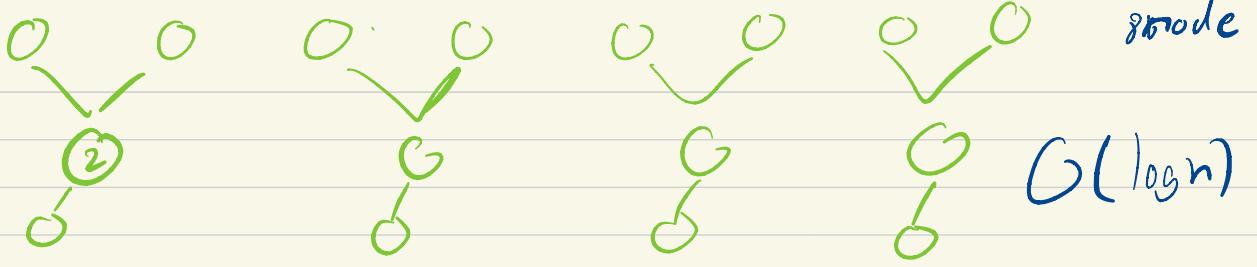
① ② ③ ④ ⑤

1. $O(\log n)$ Union by Rank



Merge (Union)
 1. u, v և u, v ինքնահանություն $h_u, h_v \rightarrow O(1)$
 2. Եթե $h_u > h_v$ ապա h_u դառնում է (Merge) $\rightarrow O(1)$





```
int find_parent (int u){
    if (pa[u] == u) return find_parent (pa[u])
    return u
```

```
bool unite (int u, int v){
    int hu = find_parent (u)
    int hv = find_parent (v)
```

```
if (hu != hv){
    if (lv(hu) > lv(hv)){

```

```
        pa(hv) = hu
```

```
    } else if (lv(hu) < lv(hv)){

```

```
        pa(hu) = hv
```

```
    } else {

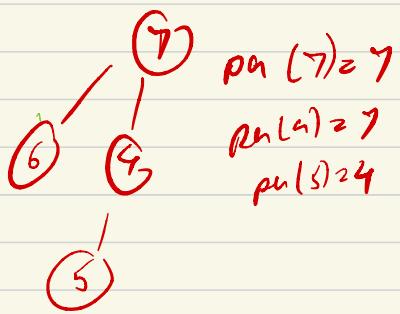
```

```
        pa(hv) = hu
```

```
        lv(hu) ++
    }
```

```
    return true
}
```

```
else
    return false
}
```

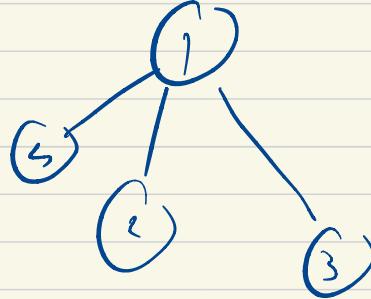


2. Path Compression $O(\alpha(n)) \approx O(1)$
 inverse ackermann of n
 Amortized (amortization)

```
int find_parent (int u) {
    if (pa[u] != u) return pa[u] = find_parent (pa[u])
    return u
}
```

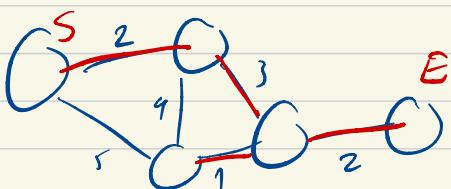
```
bool merge (int u, int v) {
    int hu = find_parent (u)
    int hv = find_parent (v)

    if (hv != hu) {
        pa[hv] = hu
        return true
    }
    return false
}
```



ຈົກສູ່ DSU / MST

1. ຈົກສູ່ນາງຂອງ (max weight MST ດັວນໄດ້)

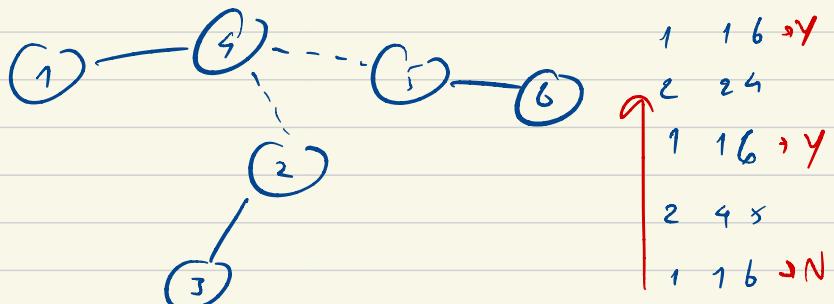


2. ຕົວ edge ການເຊີ້ນໄລຍະ ປິຈຳ node

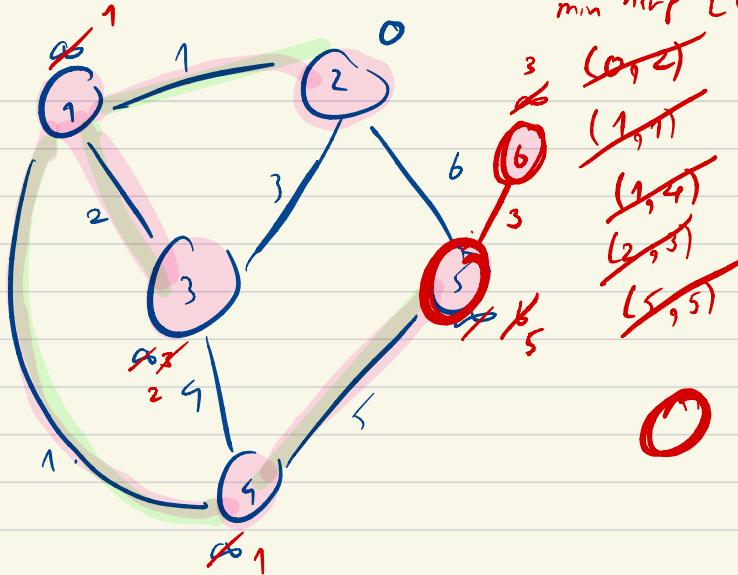
2² 1. ວິທີກົດຕືກໄລຍະ
 2. ດັວນ edge ສະເໜີ່ງ n, v ໂດຍມີກົດຕືກໃນ edge ປີ

* ດັວນຈະກົດຕືກໄລຍະ

ອີກ DSU ຕົວ edge ລາຍລະອຽດ



Prim



$wei[u] :=$ ក្រោមគេងនៃ នុយសំខាន់ដែលត្រូវបានរៀបចំ

min heap [(wei, node)]

bfs / dfs
list / mat
mst
sssp

vector<int> wei(V+5, INF)

wei[0] = 0

pq = min heap

while (!pq.empty()) {

auto [v, w] = pq.top();

pq.pop

if (!view[w]) continue;

view[w] = true;

for (auto [v, w] : G[w]) {

if (w < wei[v] and !view[w]) {

wei[v] = w;

pq.push (wei[v], v);

}

$$mst = \sum_{i=1}^V wei[i]$$

Topological Order \rightarrow DP

մասնակիւն
մասնակիւն
պարզ
հաջորդական
լուծութեան

① ② ③ ④ ⑤

① \rightarrow ②

1, 2, 5, 3, 4

⑤

1, 5, 3, 2, 4

③ \rightarrow ⑦

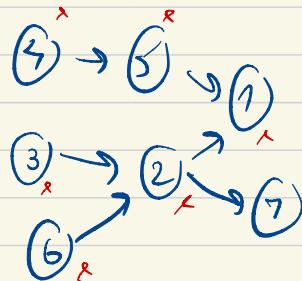
① \rightarrow ② \rightarrow topo Do Not Exist
DFS սկզբ 1 BFS սկզբ

2. \rightarrow Iterative (Queue)

\hookrightarrow Recursive (DFS)

DFS

```
stack<int> topo_order
void topo(int u) {
    if (vis[u]) return
    vis[u] = true
    for (v: G[u]) {
        topo(v)
    }
    topo_order.push(u)
```



4, 3, 5, 6, 2, 7, 1

Queue

```

for(u:adjV){
    if(indeg[u]==0)
        qq.push(u)
}

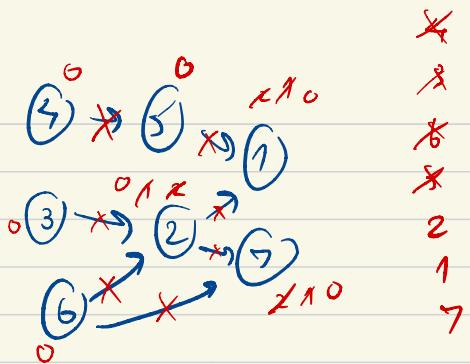
while(!qq.empty()){
    int v = qq.front(); qq.pop()
    topo_order.push_back(v)
}

```

```

for(v: G[u]){
    indeg[v]--
    if(indeg[v]==0){
        qq.push(v)
    }
}

```



topo order : 4, 3, 6, 5, 2, 1, 7

