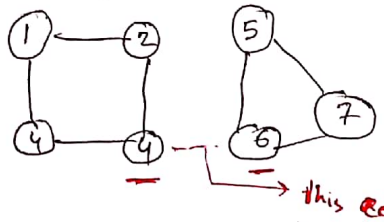
# Graph

G(V, E) ⇒ vertices and edges

(4)

Non-connected

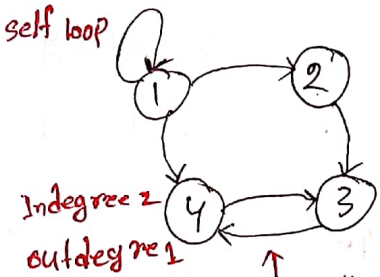



this edge are connected

2 components

(4, 6) → are called articulation point coz if 4 or 6 deleted then two or many seperate Graphs show

(1)

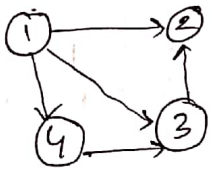



Directed Graph

Indegree 2
outdegree 1

↑ paralled Edges

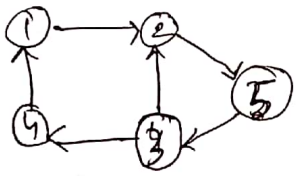
(2)




simple Digraph

(5)

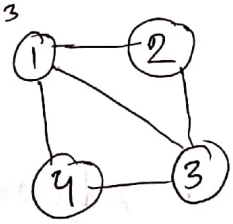
Strongly connected




coz if we start any vertecx we can reach all other vertecx

(3)

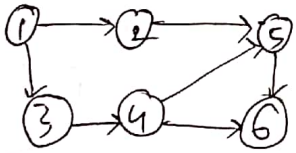
degree 3




Graph/Non-Directed Graph

(6)

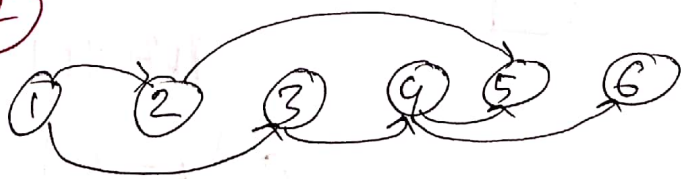
Directed Acyclic Graph




there is no cicle

(7)

topological Ordering when direction forword



Lecture 2

# Representing Graph

1. Adjacency matrix
2. Adjacency list
3. compact List

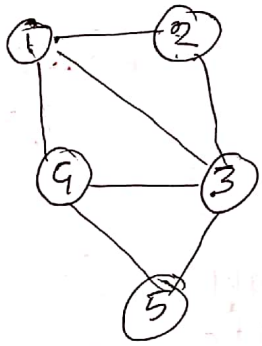
## 1. Adjacency matrix

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 |
| 3 | 1 | 1 | 0 | 1 | 1 |
| 4 | 1 | 0 | 1 | 0 | 1 |
| 5 | 0 | 0 | 1 | 1 | 0 |

5 x 5

$n * n = n^2$

$O(n^2)$




$G(V, E)$

$|V| = n = 5$

$|E| = e = 7$

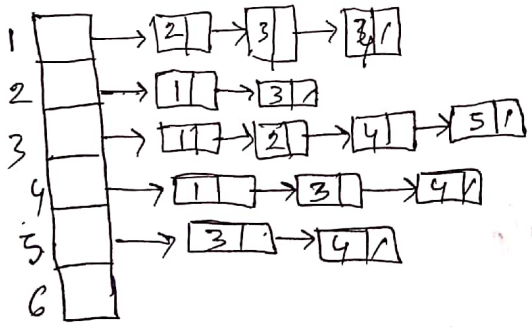
$(i, j)$ — if edge

$A[i][j] = 1$

if edge 1
not 0

## 2. Adjacency List

Array of link List



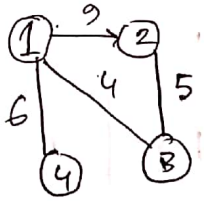

$|V| + 2|E|$

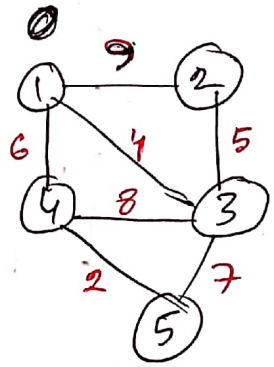
$n + 2e$

weight is given

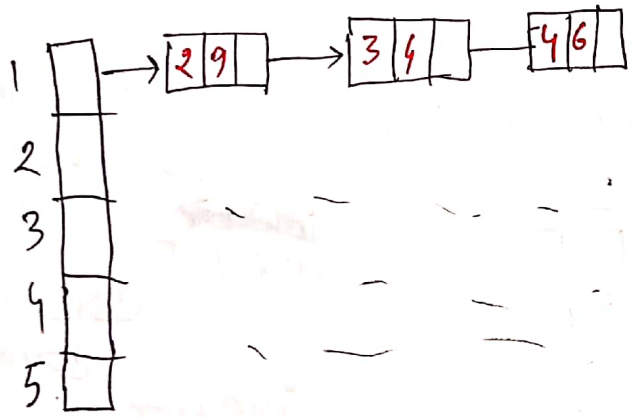
Cost Adjacency matrix

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | 9 | 4 | 6 | |
| 2 | 9 | | 5 | | |
| 3 | 4 | 5 | | 8 | 7 |
| 4 | 6 | | 8 | | 2 |
| 5 | | | 7 | 2 | |




cost of edges ☺




$|V| = n = 5$

$|e| = e = 7$

$\therefore |V| + 2|E| + 1$

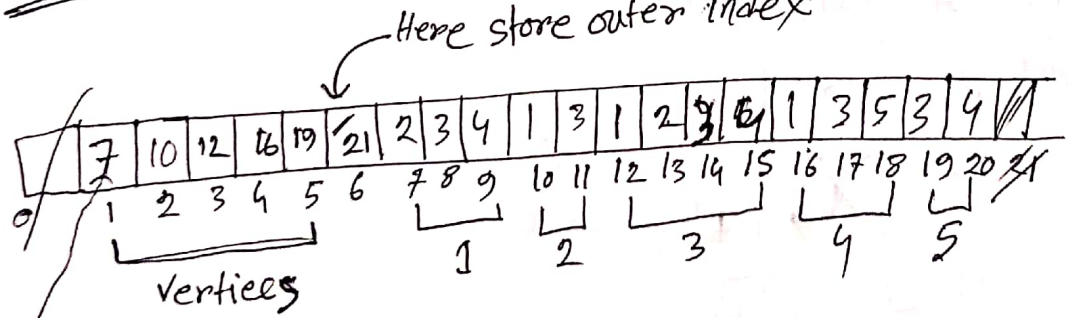
$\Rightarrow 5 + 14 + 1 = 20$

0 1 - - -

$20 + 1 = 21$

Space in Array.

## 3. Compact List Representation




7 - 10 → gives the starting point of vertices (1)

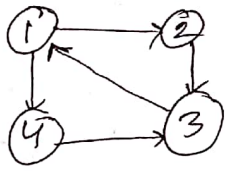
O(n) space

# Directed graph

## Lecture 3

### Adjacency matrix

| A | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 |



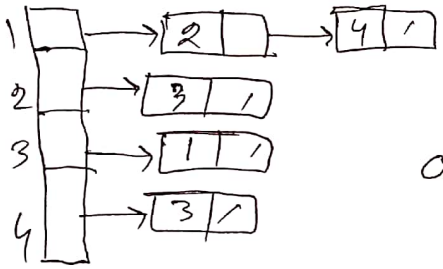

G(V, E)

|V| = 4

|E| = 5

4 × 4

### Adjacency List




↑ out degree

now check for what are go out edges

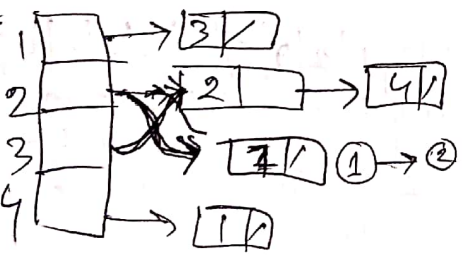
~~indegree~~ ↙ out degree

Collum check for come in edges find out

↙ indegree

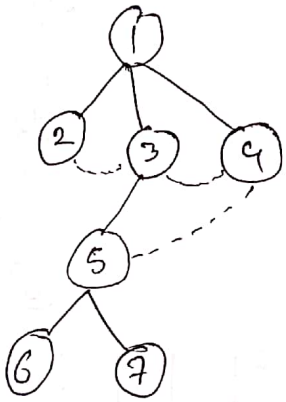
### Inverse Adjacency List

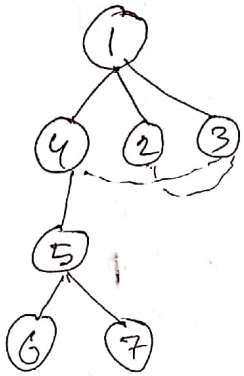
indegree

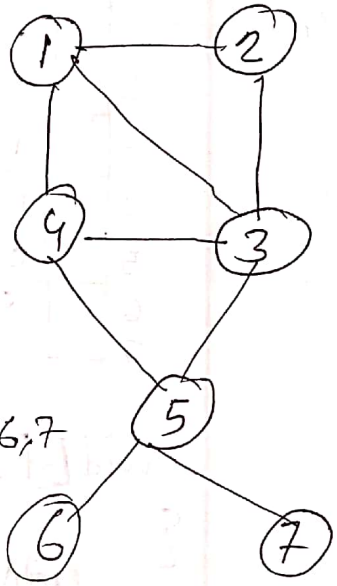


# Breath First search (BFS)

Lecture 4




BFS: 1,2,3,4,5,6,7 · BFS': 1,4,2,3,5,6,7

Many order possiable of BFS

if we select any vertices travers all connect vertices

traversal

similer to level order in tree

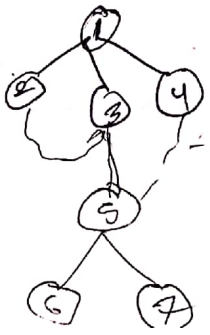
O(n) Analitical time

1. visiting
2. Exploring -

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

Queue: ~~1,2,3,4,5,6,7~~

take out and exploring




→ Dot Dot → cross edges

← Bfs speing tree

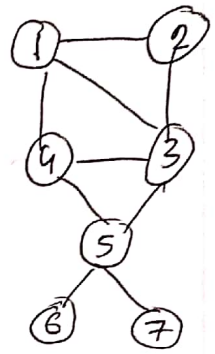
# BFS → Algo

## Lecture 5

(u) →

A =

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | |
| 1 | | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 4 | | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 5 | | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 7 | | 0 | 0 | 0 | 0 | 1 | 0 | 0 |




q | 1 | 2 | 3 | 4 | | | | |

visited | 1 | 1 | 1 | 1 | 1 | | | |
0 1 2 3 4 5 6 7

- Visited Array initialized with zero.

```
void BFS(int i)
{
    printf("%d ", i);
    visited[i] = 1;
    enqueue(q, i);
    while( ! isEmpty(q))
    {
        u = dequeue(q);
        for(v = 1; v <= n; v++)
        {
            if (A[u][v] == 1 && visited[v] == 0)
            { printf("%d ", v);
              visited[v] = 1;
            } enqueue(q, v);
        }
    }
}
```

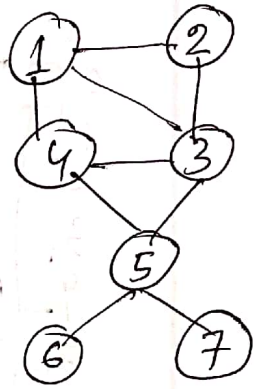
$O(n^v)$

# Depth First search

Lecture 6

1. visited
2. Exploring

DFS: 1, 2, 3, 5, 7, 6

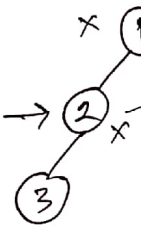



Step (1)

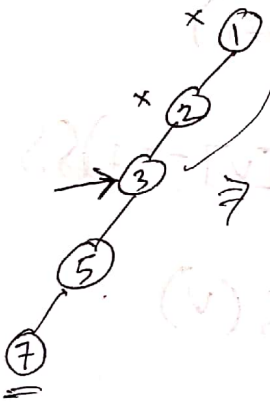


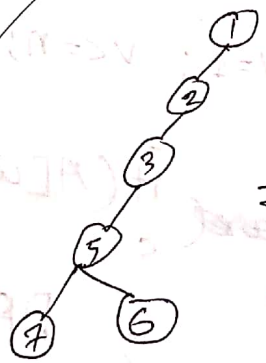
Dont explore All of 1 at a time Level to stack

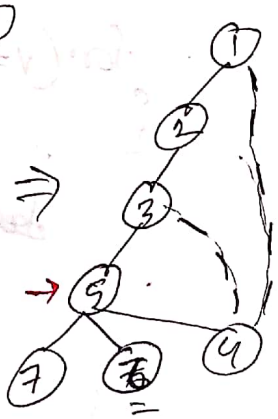
(2)




(3)




← Back edges

traversal

preorder tree traversal similar

O(n) Anlitical time

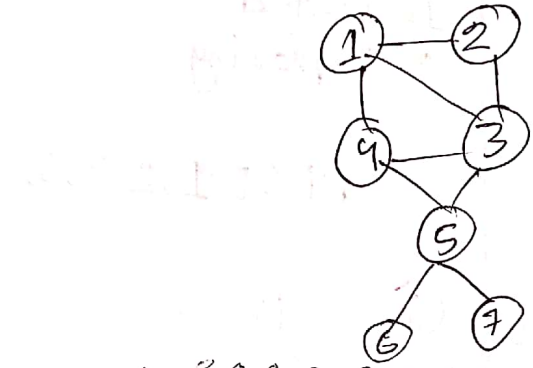
DFS speing tree.

There are many DFS result

# DFS → program

Lecture 7

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | |
| 2 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | |
| 3 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | |
| 4 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | |
| 5 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | |
| 6 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| 7 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | |




visited

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

```
void DFS(int u)
{
    if (visited[u] == 0)
    {
        printf("%d", u);
        visited[u] = 1;
        for(v=1; v<=n; v++)
        {
            if(A[u][v] == 1 && visited[v] == 0)
            {
                DFS(v)
            }
        }
    }
}
```

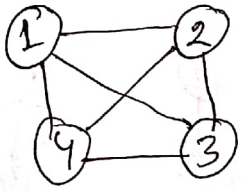
stack

01989047339

Lecture 9

# Spanning trees

1. Spanning tree
2. prims MST
3. Kruskals MST




spanning tree is subGraph as a graph All vertices But (n-1) edges

$G(V, E)$

$n = |V|$

$e = |E|$

$S \subset G$

$\Rightarrow S = (V', E')$

$|V'| = |V|$

$|E'| = |V| - 1$

$|V| = 4$

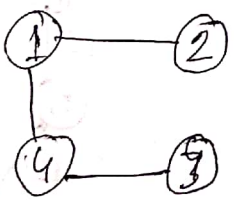
$|E| = 6$

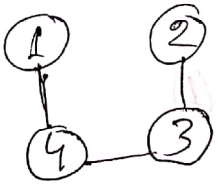
$|E| C_{|V|-1}$ → this many spaning tree possible have

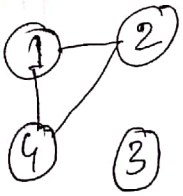
$|E|$ −cicle $C_{|V|-1}$

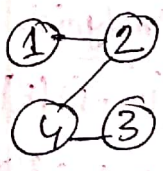
${}^6C_3 - 4 = 20 - 4 = 16$

} should not any cicly




×

Cicly

spanning tree

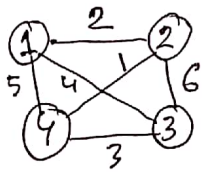
$\Rightarrow$ ${}^6C_3$ − cicly ✓ possible

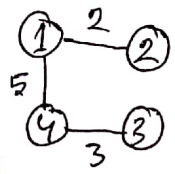
cost → MST

Lecture 10

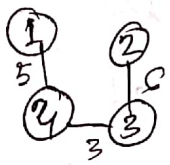
# prim's Algo

minimum spanning tree:

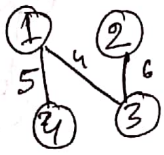


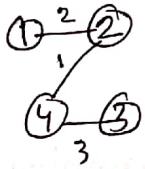

MST cost → 10




14




15




6

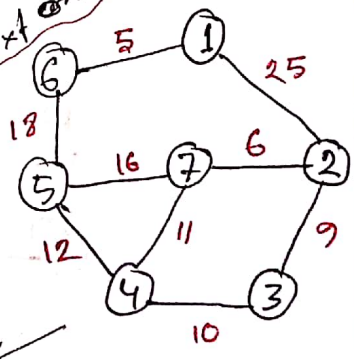
16 ⇒ spanning tree possiable ∴ have to try all of them and find out which gives minimum cost

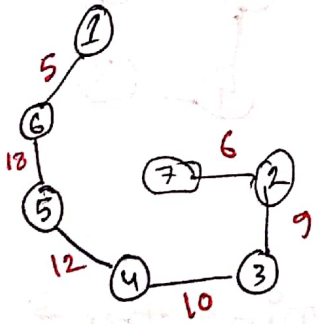
1. prims mst
2. kruskals mst

Lecture 10 — prim's minimum spanning tree

1. First find minimum cost from the Graph
2. Repeted steps conected minimum precious (vertices)

1 and 6
25 or 18 next one




5 + 18 + 12 + 10 + 9 + 6 = 60

** Graph is not conected then Not find out min cost of that

Analycies

$(|V|-1)|E|$

$ne = n*n$

$\Rightarrow O(n^2)$

using heap

$(|V|-1)\log|E|$

$\Rightarrow O(n \log n)$

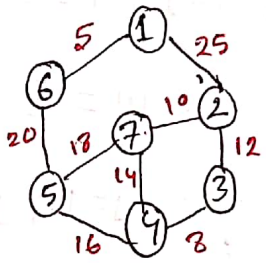
# prim's Algo ⇒ program

```
#define I 32767
int cost[8][8] = { ... };
int near[8] = {I,I,I,I,I,I,I,I};
int t[2][6]
void main()
{
    for(i=1; i<=n; i++)
    {
      for(j=i; j<=n; j++)
       {
          if (cost[i][j] < min)
          {
             min = cost[i][j]
             u=i , v=j;
          }
       }
    }
    t[0][0]=u;
    t[1][0]=v;
    near[u] = near[v] =0;

    for(i=1; i<=n; i++)
    {
      if (cost[i][u] < cost[i][v])
      {
          near[i]=u
      }
      else
          near[i] = v
    }
```

min cost

near[i] != 0

cost Adjacency Matrix




| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - | - | - |
| 1 | 1 | | 25 | | | | 5 | |
| 2 | 1 | 25 | | 12 | | | | 10 |
| 3 | 1 | | 12 | | 8 | | | |
| 4 | 1 | | | 8 | | 16 | | 14 |
| 5 | 1 | | | | 16 | | 20 | 18 |
| 6 | 1 | 5 | | | | 20 | | |
| 7 | 1 | | 10 | | | 14 | 18 | |

near

| | 0 | 1 | 6 | 65 | 60 | 0 | 65 |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

k=5 4

t

| 1 | 5 | 4 | 3 | 2 | 2 | | |
|---|---|---|---|---|---|---|---|
| 6 | 6 | 5 | 4 | 3 | 7 | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

↳ spanning store Here

① step: check lower or upper part of mat Find minum cost → 5

② Next Find conected of minum cost

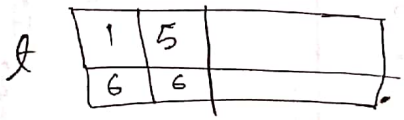
Repearting ~~cost~~ step

matrix


Find min cost

```
for( i=1; i< n-1; i++)
{
    min = I
    for(j=1; j<=n; j++)
    {
      if(near[j] !=0 && cost[j][near[j]] < min)
       {
          min = cost [j] [near[j]];
          k=j;
       } }
```

→ k 5

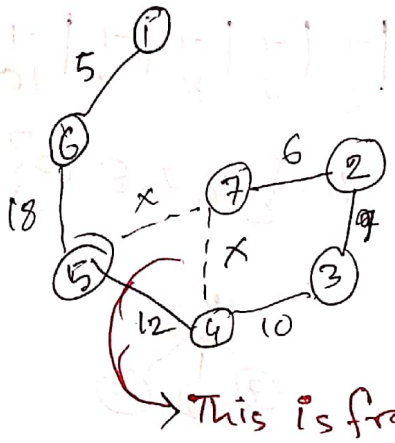


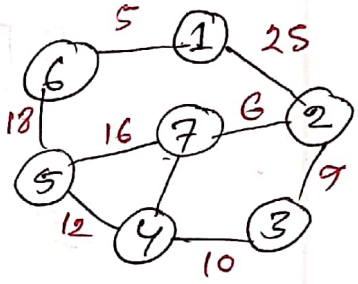

```
t[0][i] = k;
t[1][i] = near[k]
near[k] =0

for(j=1; j<=n; j++)
{ if (near[j] !=0 && cost[j][k] < cost[i][near[j]])
    {
        near[j] = k;
    }
}
print(t) →
```

# kruskal minimum.

Lecture 13

* kruskal say selected minimum always But not from a cicly.



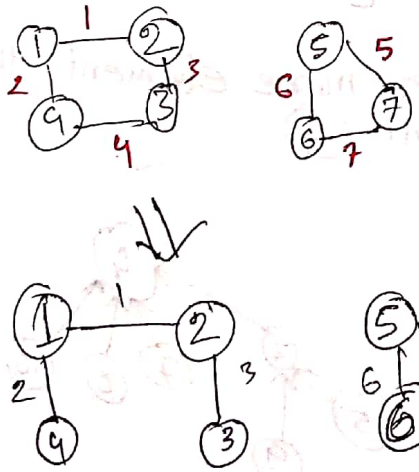




$(|V|-1)\cdot|E|$

We can take min heap so that time will be dredie ⇒ ve

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

Non-connect Gpaph · kruskal work




kruskal work on more component Gpaph ⇒ But prim's not

# Lecture 14

## Disjoint Subset

universal set $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩

S

| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Set $A = \{3, 5, 9\}$ $B = \{4, 7, 8, 10\}$ $A \cap B = \emptyset$


S

| -1 | -1 | -1 | -3 | 4 | 3 | -1 | 4 | 4 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

4 -7

Node Number in Negative Sign

* Union
* find

$A \cup B = \{3, 4, 5, 7, 8, 9, 10\}$

who ever more element that select as parent ④