GRAPHS → Non-linear

Linear Data structure

Arrays

Linked list

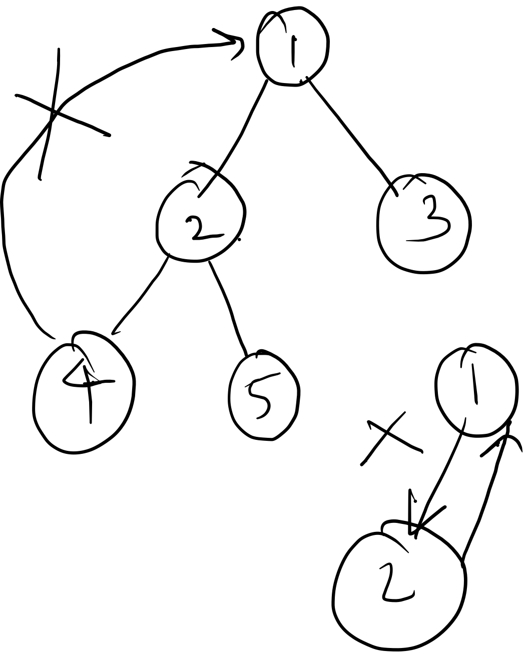
Non-linear Data structure

Trees

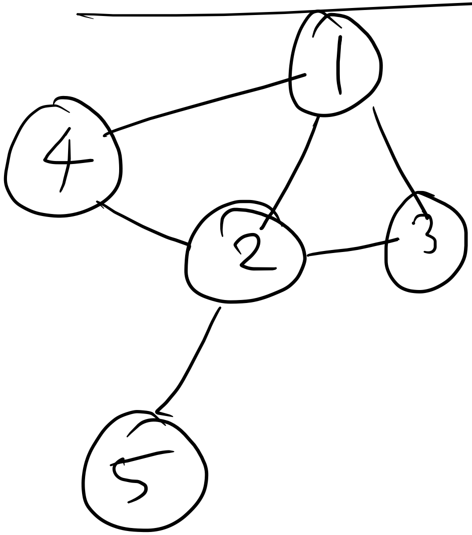
→ Non-linear

Trees vs Graph

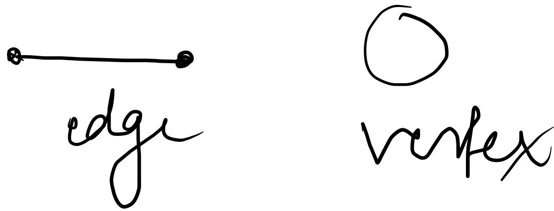
herrarchical connections

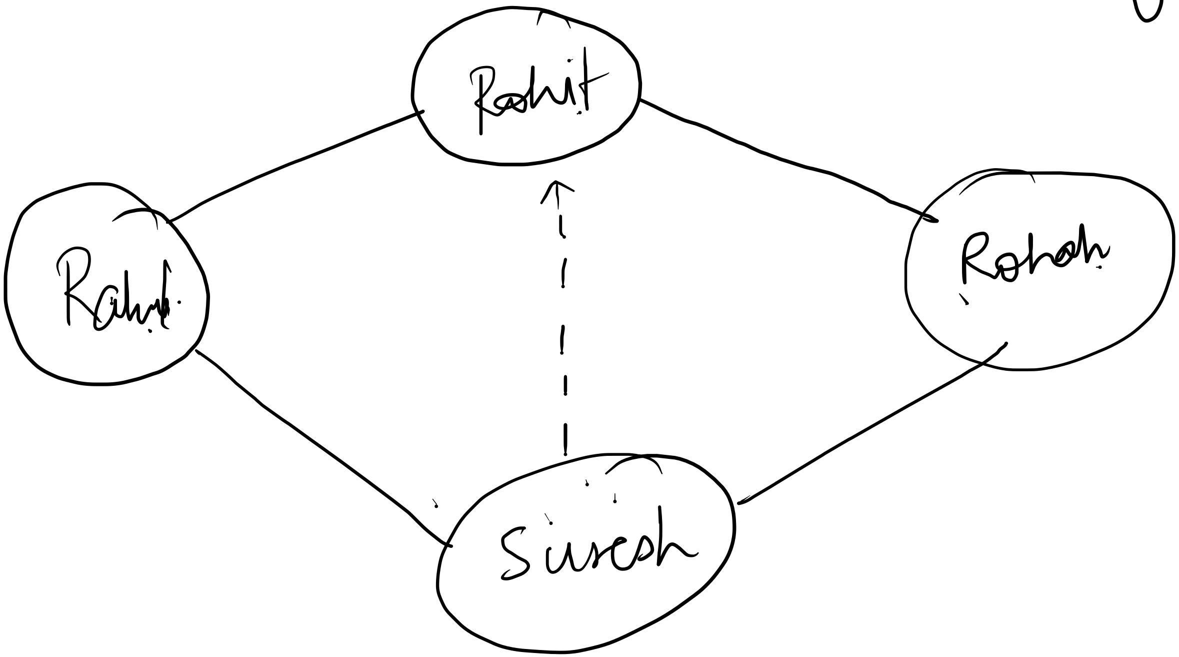


No rules for connection



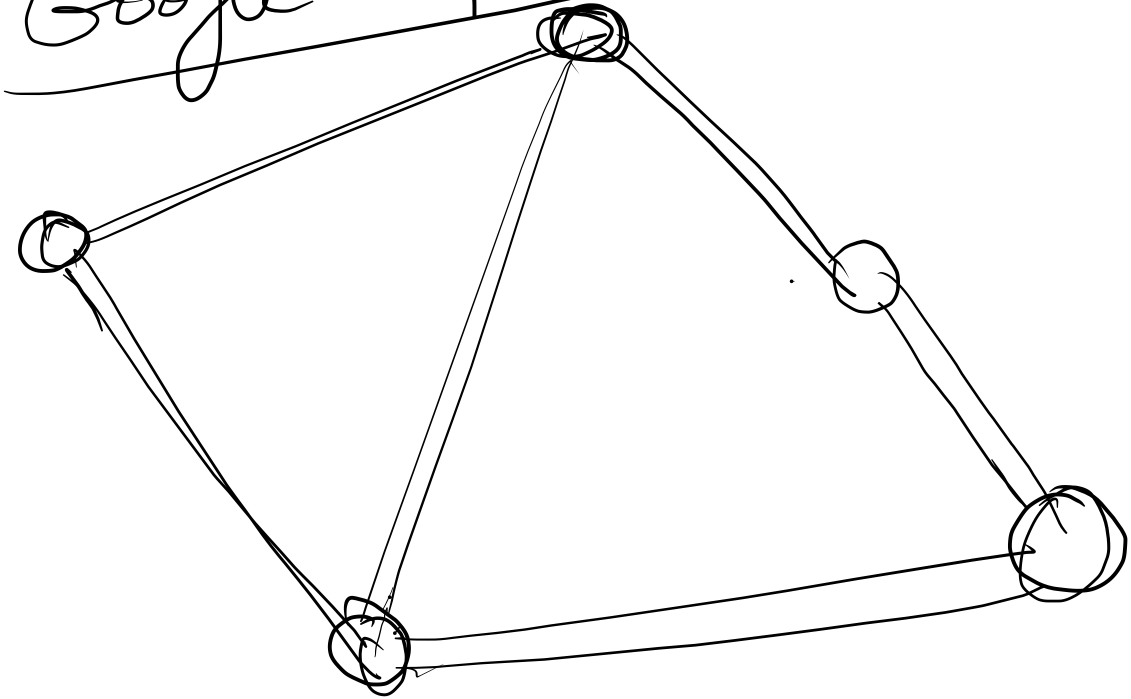
GRAPHS





GRAPHS

Google maps

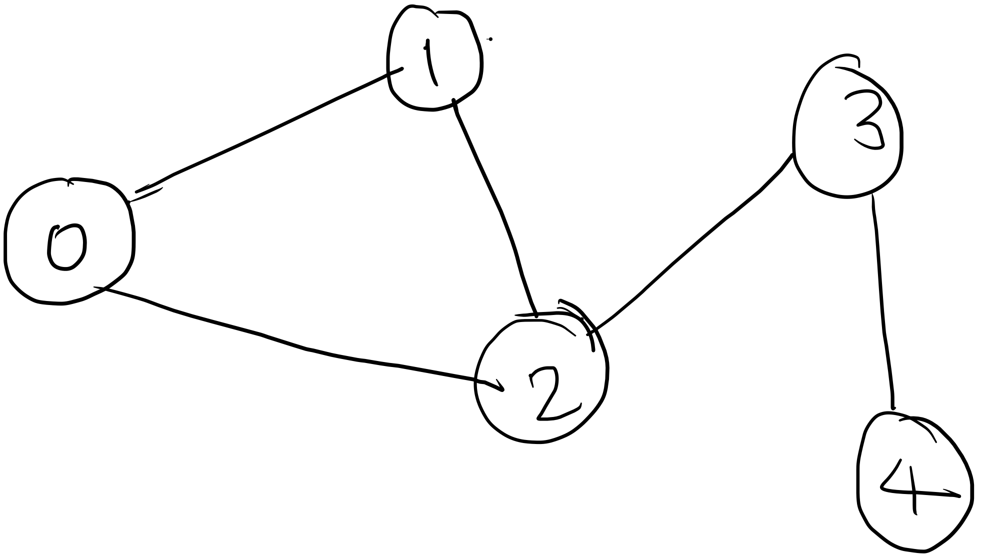


Types

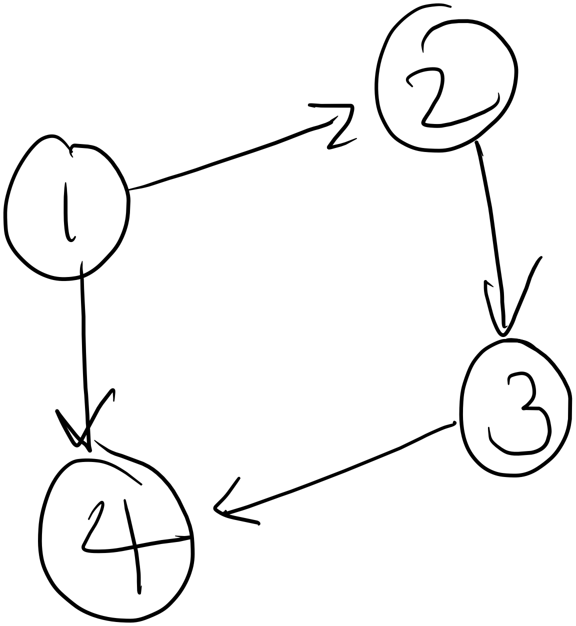
① Undirected

② Directed

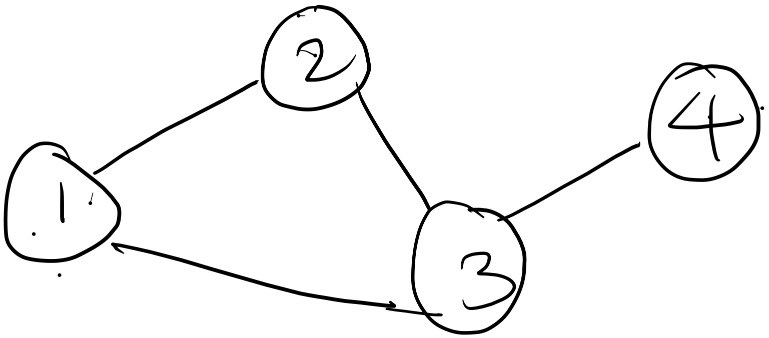
Undirceted Graph

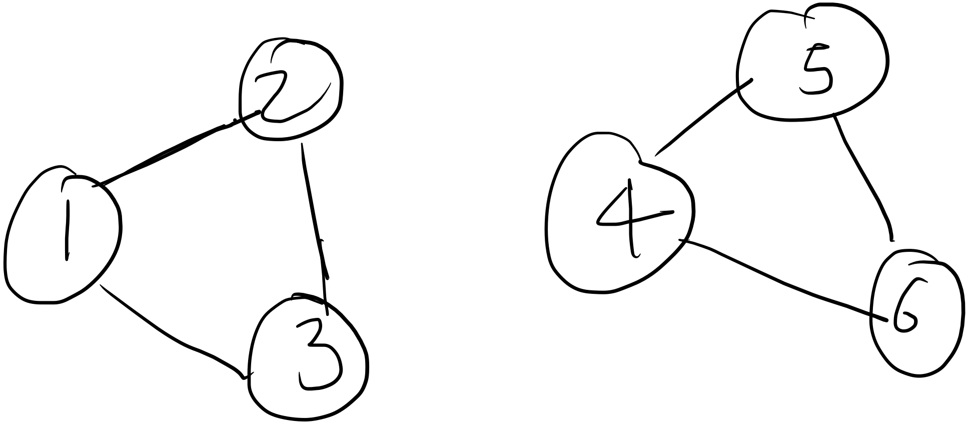


Directed Graph



Connected



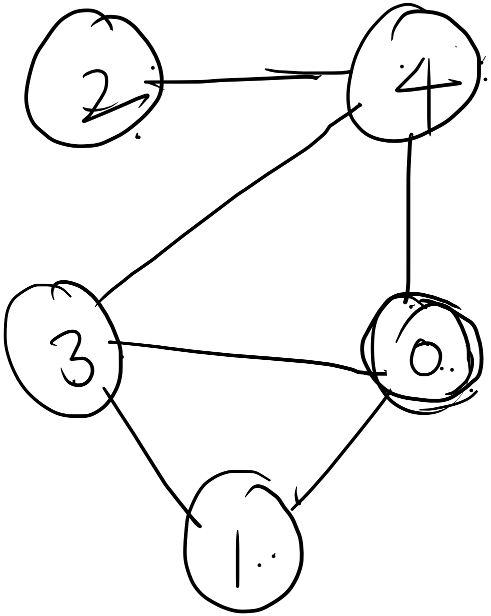


Non - connected

Graph Representation

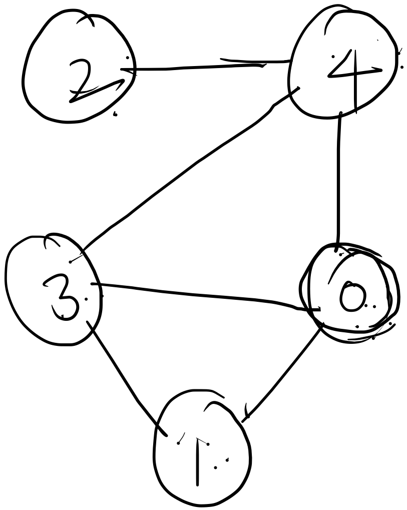
1. Adjacency matrix
2. Adjancency list

① Adjacency Matrix



m	0	1	2	3	4
0	0	1	0	1	1
1	1	0	0	1	0
2	0	0	0	0	1
3	1	1	0	0	1
4	1	0	1	1	0

② Adjacuncy list



0 → {1, 3, 4}

1 → {0, 3}

2 → {4}

3 → {0, 1, 4}

4 → {0, 2, 3}

$$\left\{ \underset{0}{\underline{\{1,3,4\}}}, \underset{1}{\underline{\{0,3\}}}, \underset{2}{\underline{\{4\}}}, \underset{3}{\underline{\{0,1,4\}}}, \underset{4}{\underline{\{0,2,3\}}} \right\}$$

adj. list

Traversal of Graphs

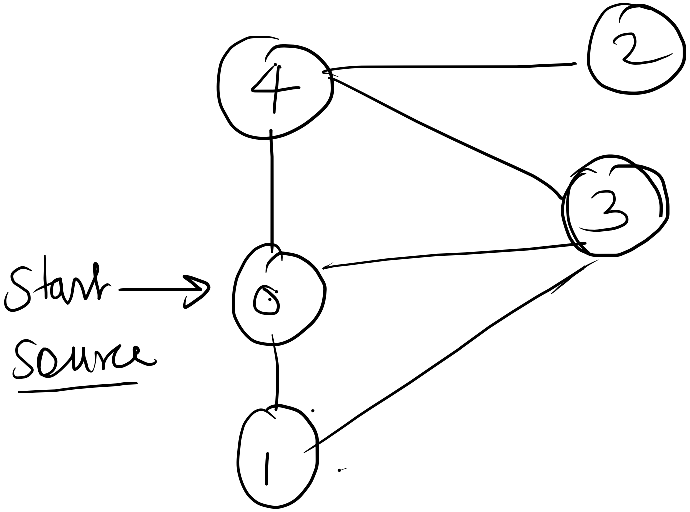
① DFS (Depth first search)

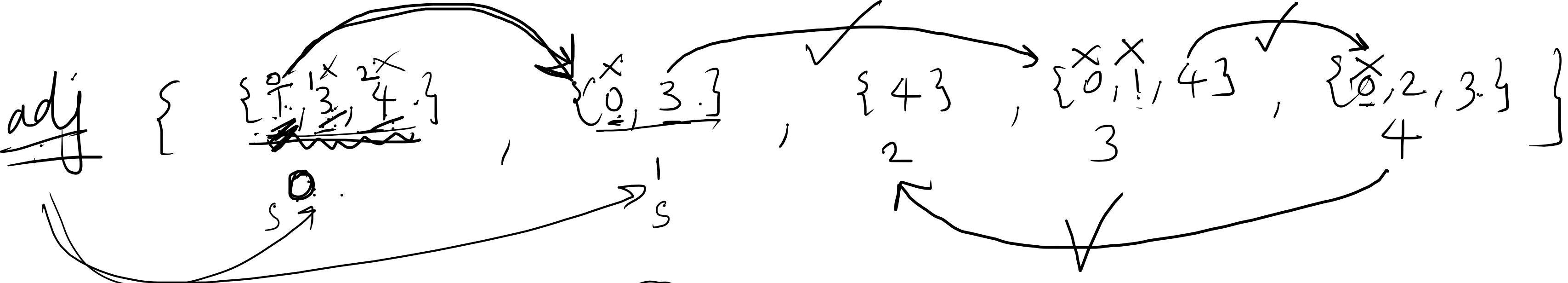
② BFS (Breadth first search)

① DFS of Graph

visited

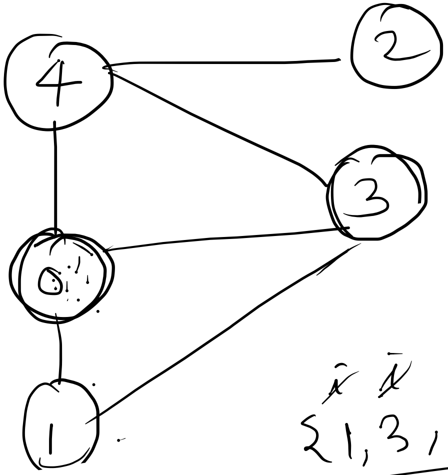
0	1	2	3	4





{ 4 }

i=0



{1, 3, 4}
0 1 2

visited				
T	T	T	T	T
0	1	2	3	4

ans = { 0, 1, 3, 4, 2 }

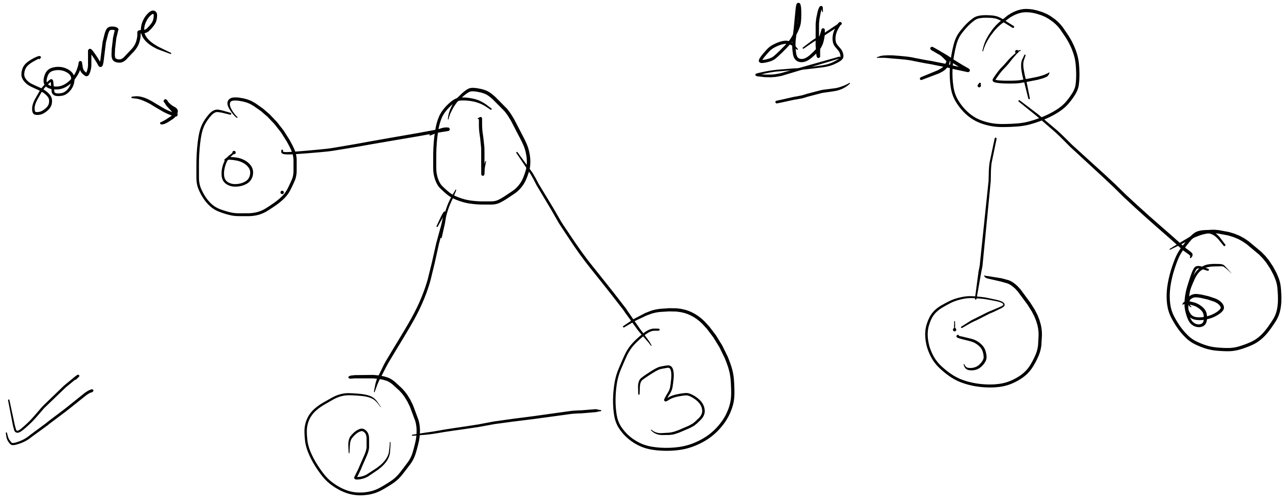
adj.get(s).get(i)

```
for( i =0 ; i< L.size() ; i++)
{

}
```

adj.get(s).size()

adj { {1,3,4}

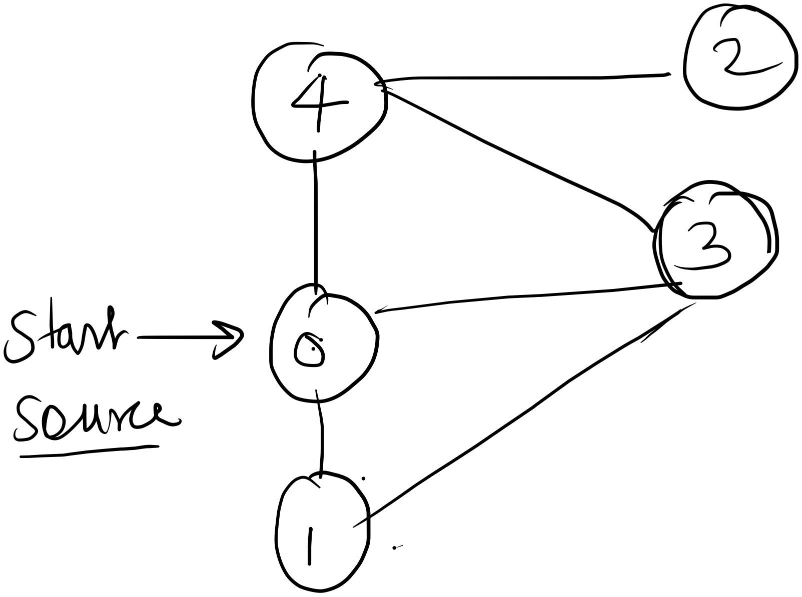
S }



for

T	T	T	T	F	F	F
0	1	2	3	4	5	6

BFS



adj. [{1, 3, 4} , {0, 3} , {4} , {0, 1, 4} , {0, 2, 3}]

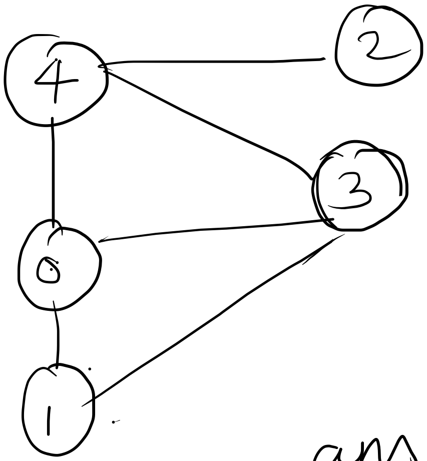
0 1 2 3 4

v

adj.get(v)

Start →

source



q

~~0~~	~~1~~	~~3~~	~~4~~	~~0~~	~~3~~	~~0~~	~~1~~	~~4~~	~~0~~	2	3

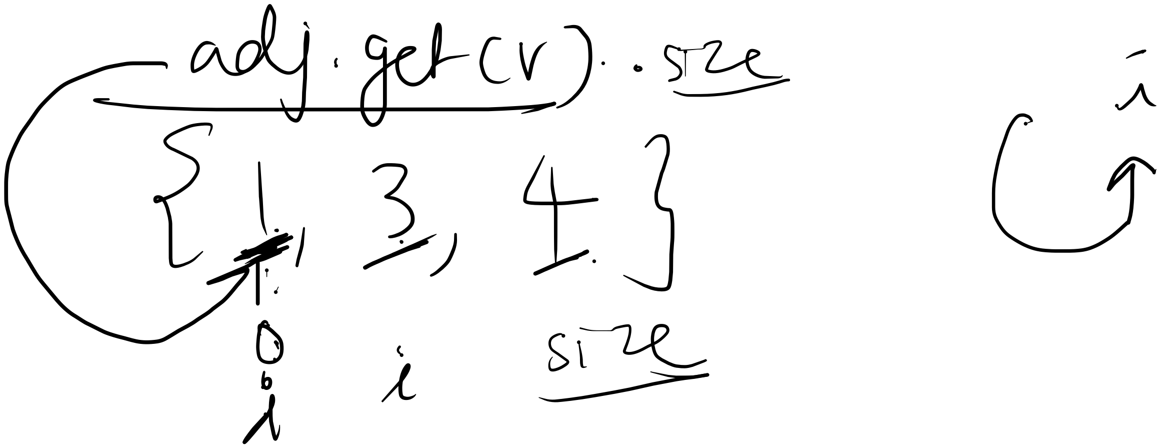
~~4~~

vis

0	1	2	3	4
T	T	T	T	T

ans = { 0, 1, 3, 4, 2 }

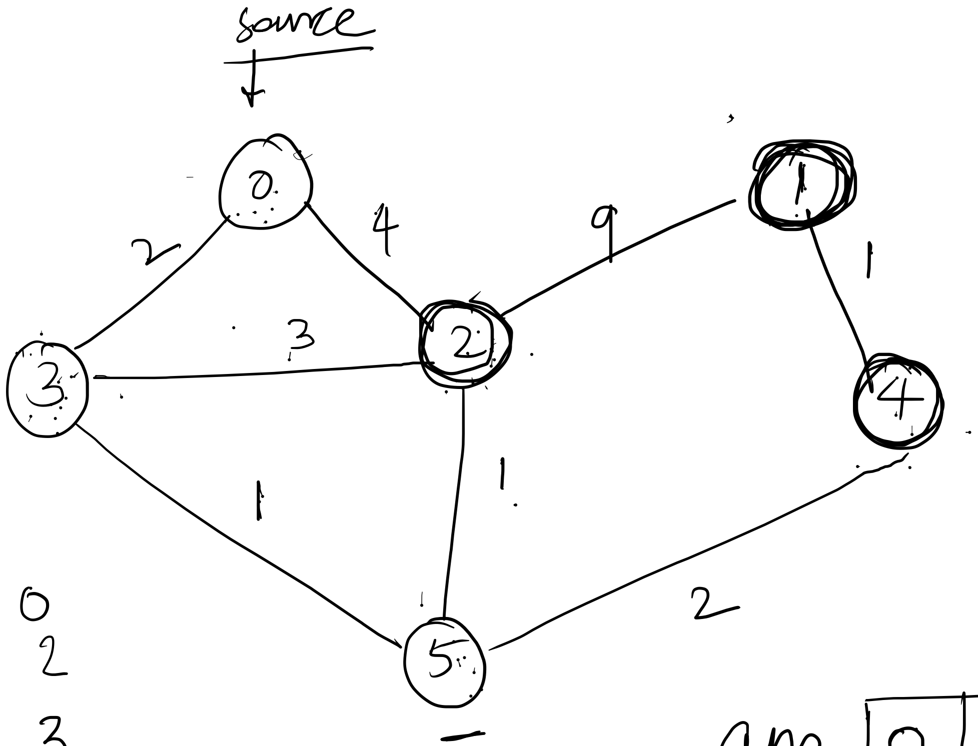
```
while( !q .isEmpty())
{
        int v =    remove

        if ( vis[v] == false )
        {
              vis[v] = T
              ans.add(v)

           Loop → adj.get(v)
                     add all connecteds
                     in q

        }
}
```



adj.get(v).get(i)

Dijkstra's Algorithm

shortest path from source



Vis

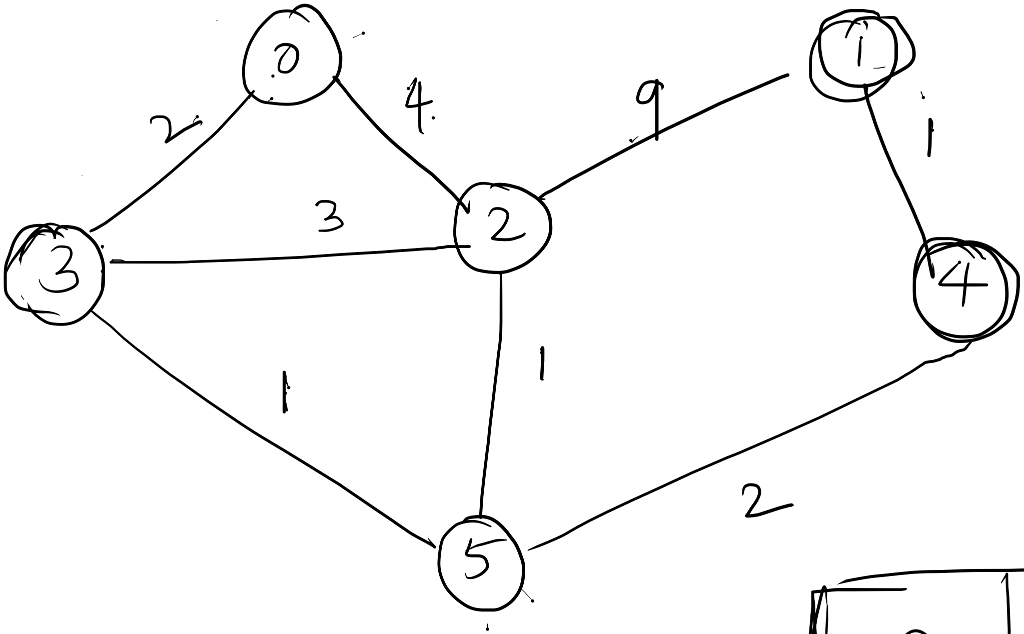
0	0
3	2
5	3
2	4
4	5
1	6

pq

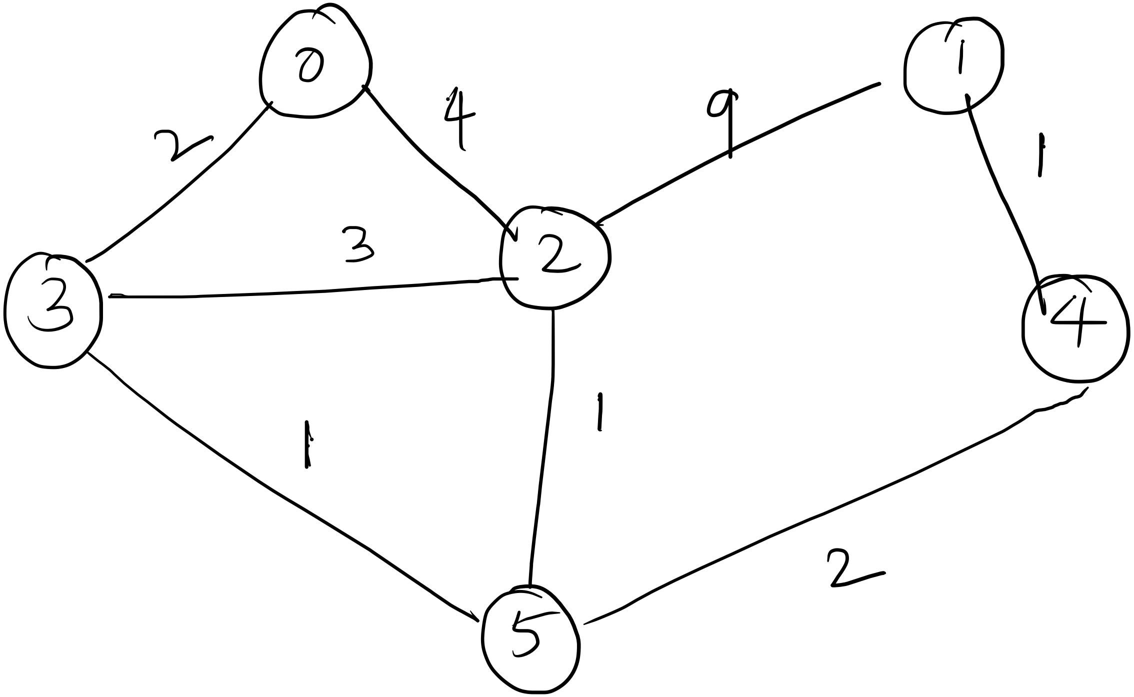
v	d
~~0~~	~~0~~
~~3~~	~~2~~
~~2~~	~~4~~
~~5~~	~~3~~
~~4~~	~~5~~
~~1~~	~~6~~

ans

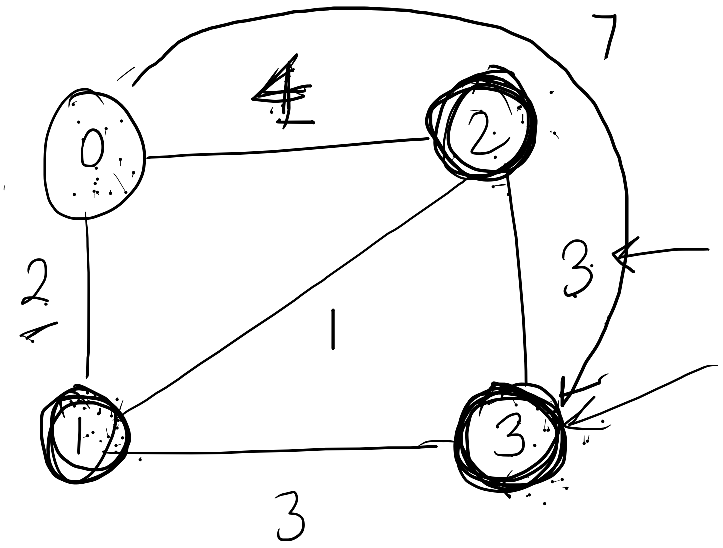
0	6 ~~13~~	4	2	5	3
0	1	2	3	4	5



	✓	✓	✓	✓	
0	6	4	2	5	3
0	1	2	3	4	5


0
1
2
3
4
5
2
4
9
1
3
1
1
2

✓ 0	0
✓ 1	2
✓ 2	3
3	5



wt

vi

ans[vertex] + wt

pq

v	dist
0	0
1	2
2	3
3	5

ans[vi] ≤ 5

$4+3 < 5$

$7 < 5$

	↓	↓		
ans	0	2	3	5
	0	1	2	3

4

a

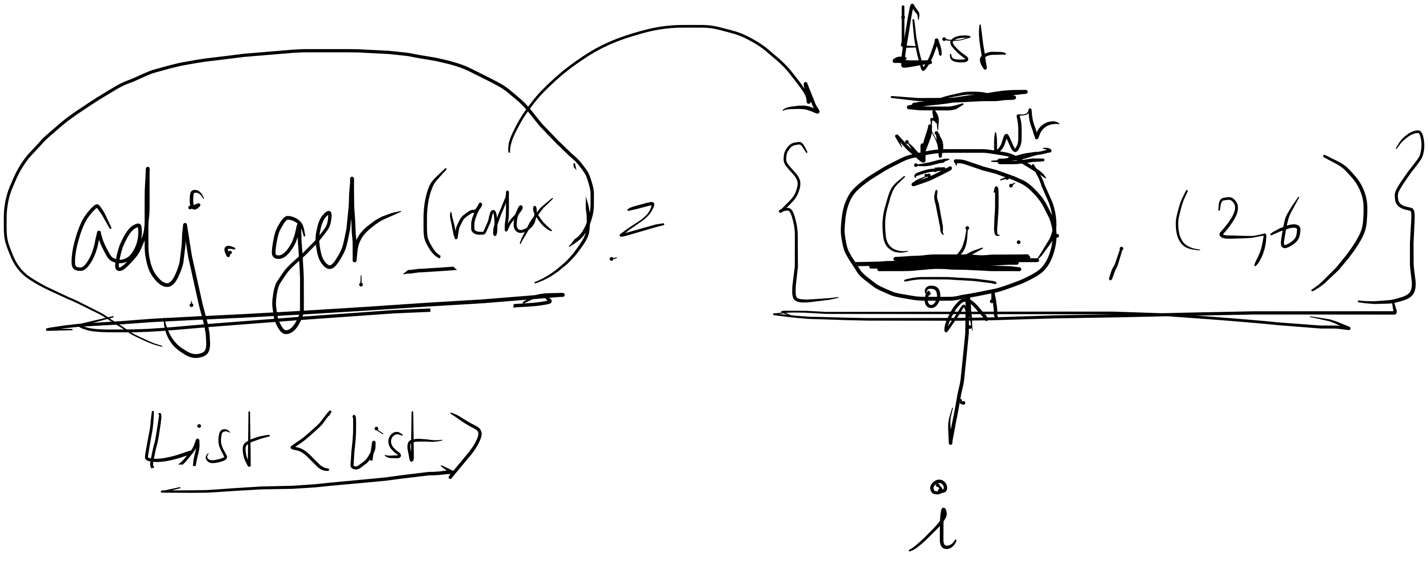
0	1
5	3

b

0	1
2	4

a[1] = 3

b[1] = 4



i = (1, 1)
    0 1

vi = i.get(0)

wt = i.get(1)