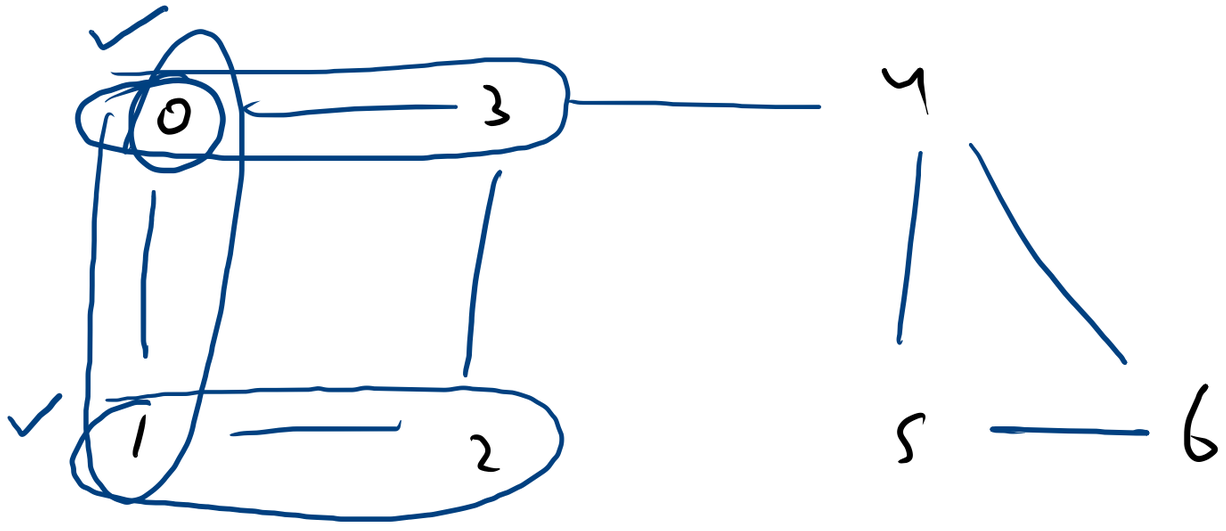


BFS

pair {

int vj A B

int group 1, -1

v	0	1	3	2	4
g	1	-1	-1	1	1

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

0

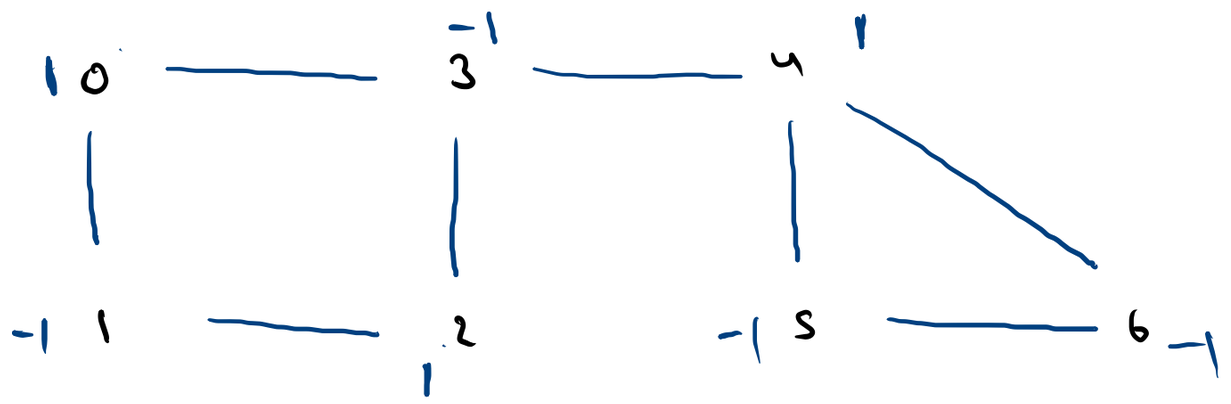
1

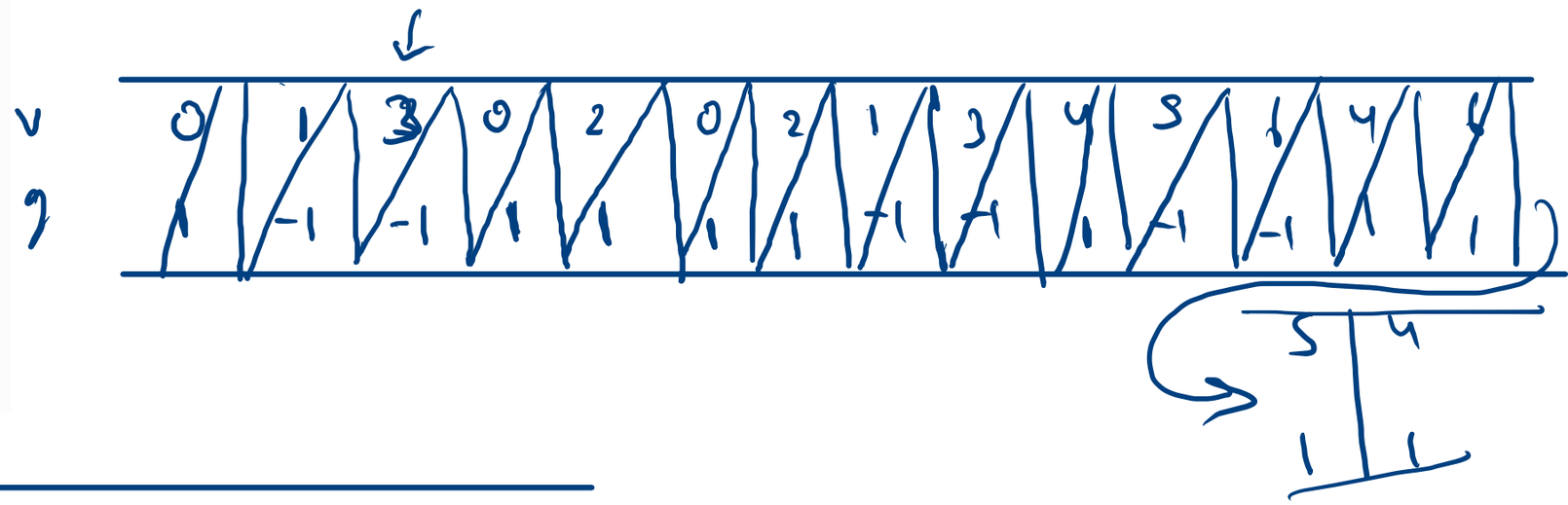
3

1

-1

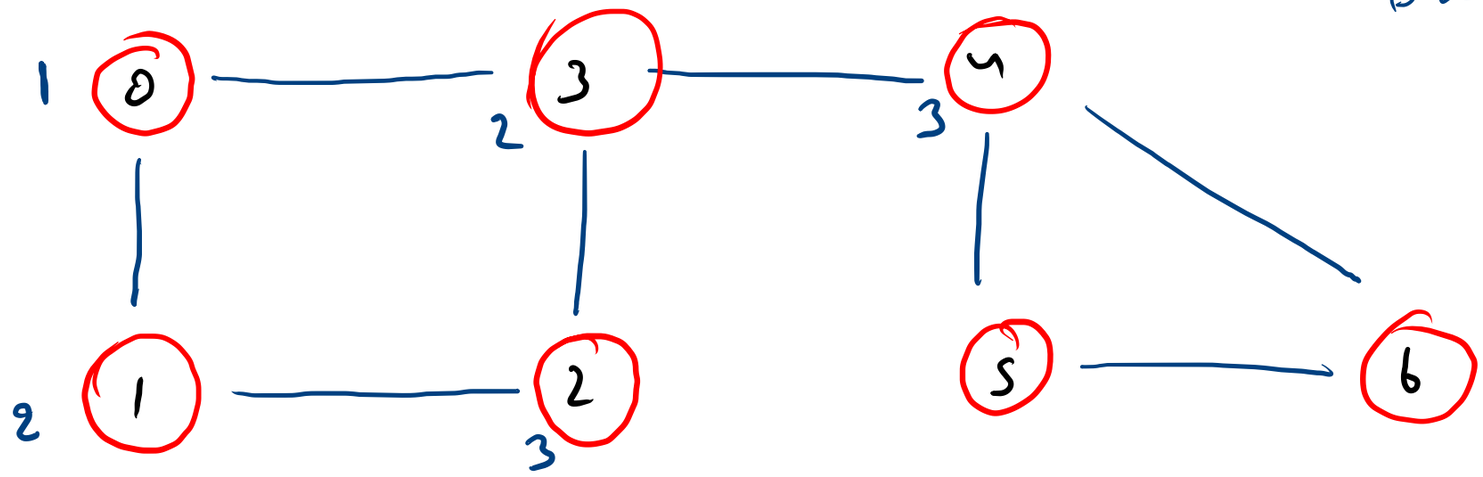
```
Queue<Pair> q = new LinkedList<>();

q.add(new Pair(src, 1));

while(q.size() > 0){
    Pair p = q.remove();
    if(group[p.v] != 0){
        if(group[p.v] != p.g){
            return false;
        }
        continue;
    }else{
        group[p.v] = p.g;
    }
    for(Edge e: graph[p.v]){
        q.add(new Pair(e.nbr, p.g*-1));
    }
}

return true;
```



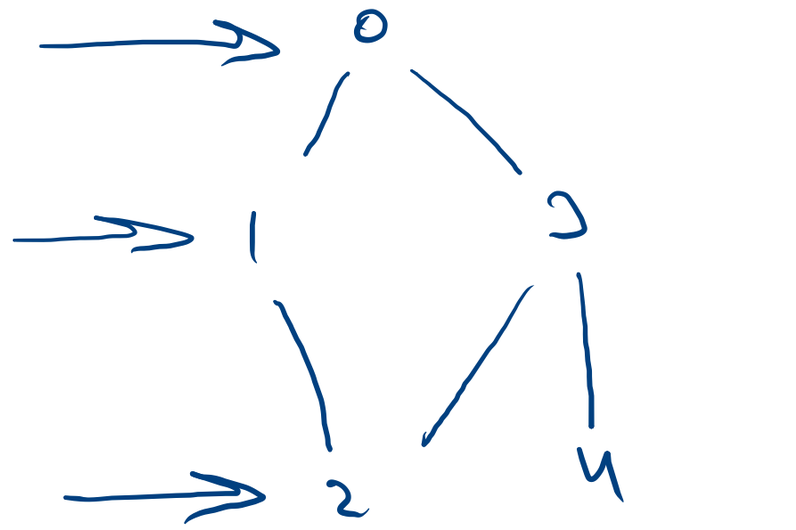


group

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



BFS



src = 0

t = 2 3

3 5

1	=	0
2	=	1, 3
3	=	2 4
4	=	5, 6

Q

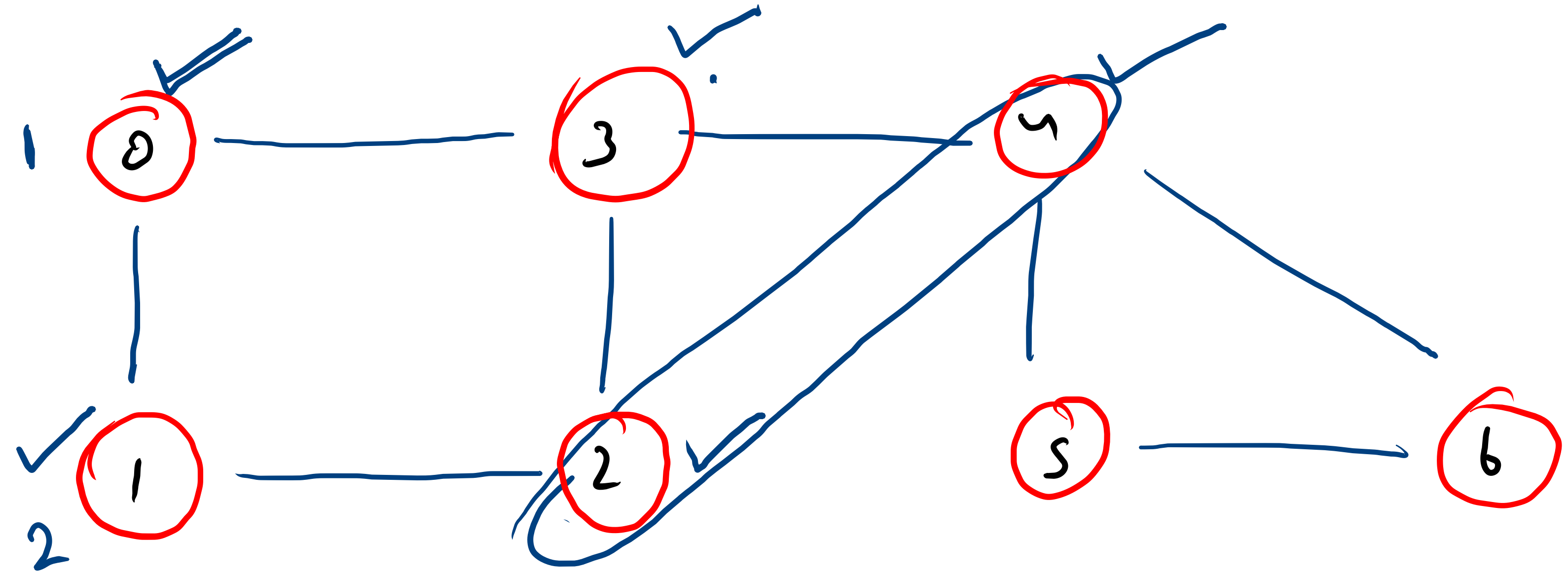
add

remove

2 rem 2

8 7

src



t = 3

pair {

int v;

int t;

remove

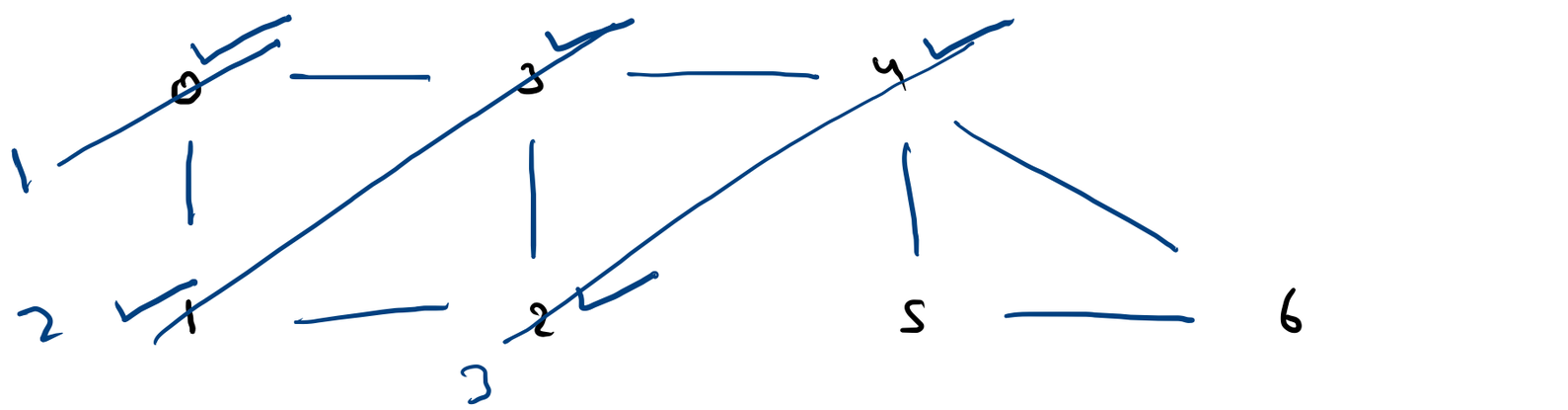
visit

work

child

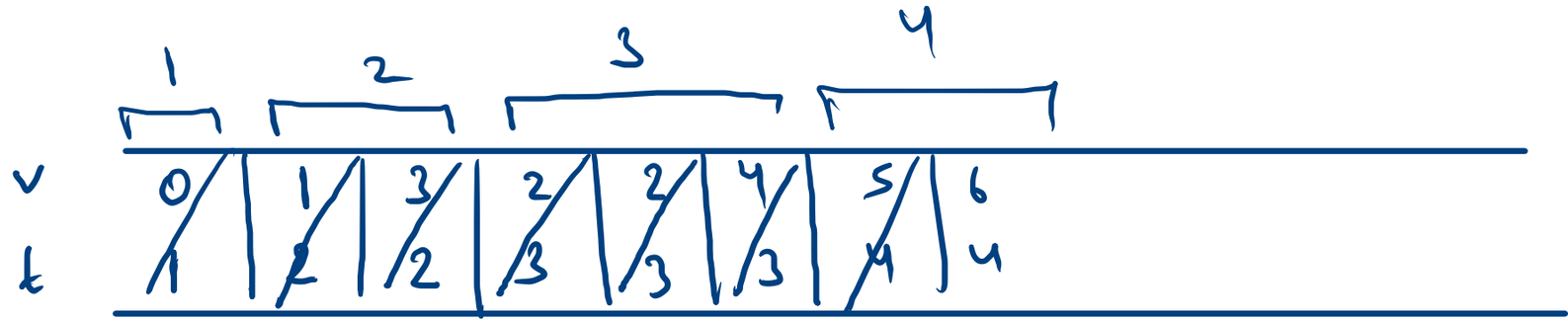
1 2 3 4 5

```java
Queue<Pair> q = new LinkedList<>();
q.add(new Pair(src, 1));
int count =0 ;

while(q.size() > 0){
    Pair p = q.remove();
    if(p.t > t)break;
    if(visited[p.v]){
        continue;
    }else{
        visited[p.v] = true;
    }
    count++;
    for(Edge e: graph[p.v]){
        if(visited[e.nbr] == false){
            q.add(new Pair(e.nbr, p.t+1));
        }
    }
}

System.out.println(count);
```

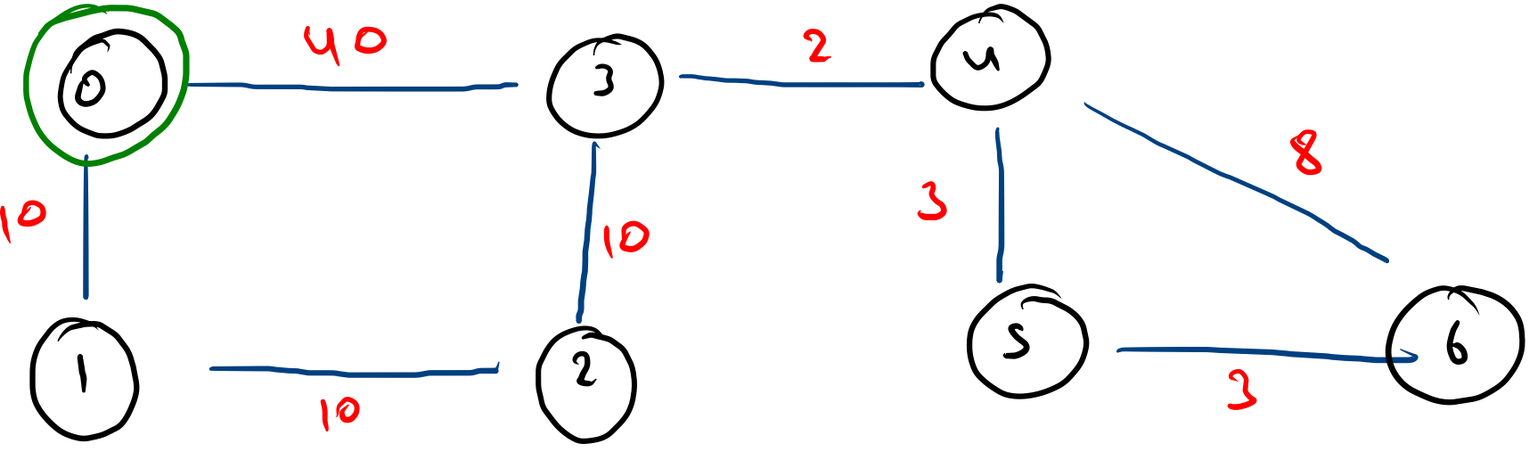


src = 0

t = 3



count 0 1 2 3 4 5



v	path	wt
0 — 1	01	10
0 — 2	012	20
0 — 3	0123	30
0 — 4	01234	32
0 — 5	01234	35
0 — 6	01234	38

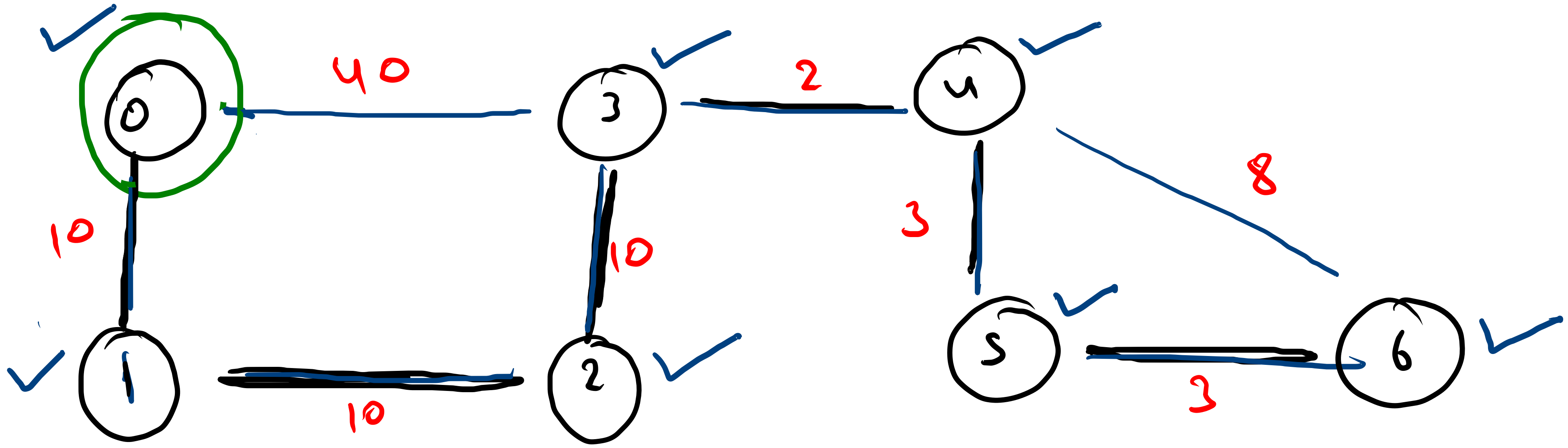
min edges 03

min wt 0123

0	1	2	3	...
0	10	20	40	

1 skip

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



Pair {
v
path
wt
comparable wt
}

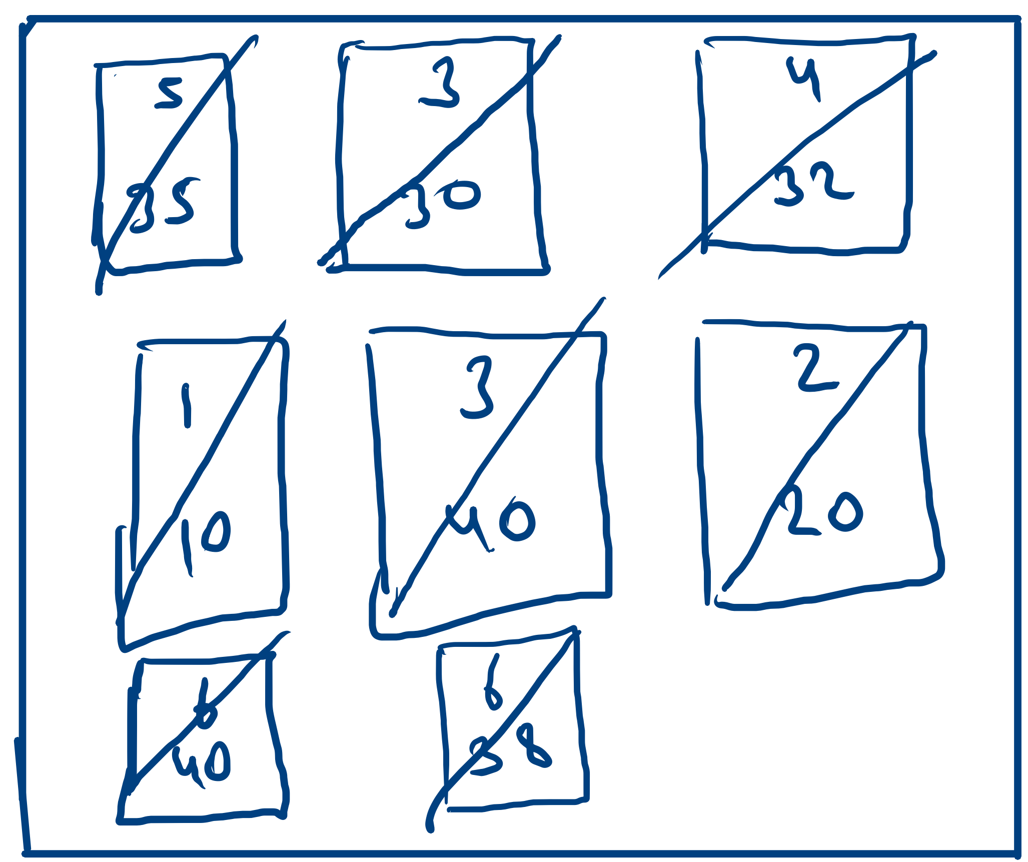
r
visited
work
child

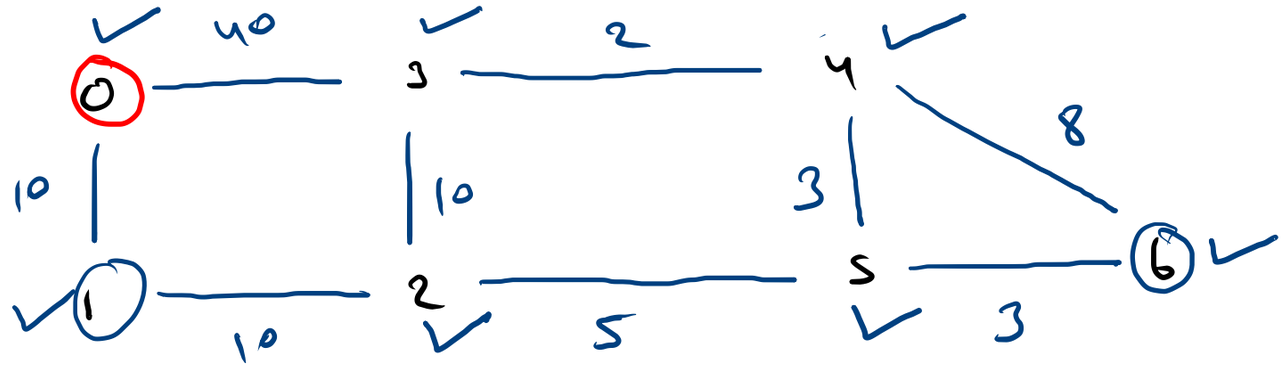
0-0, 0
0-1, 10
0-2, 20
0-3, 30
0-4, 32
0-5, 35
0-6, 38

min wt

v 0
wt 10

PQ






```
PriorityQueue<Pair> pq = new PriorityQueue<>();
boolean visited[] = new boolean[vtces];
pq.add(new Pair(src, src+"", 0));

while(pq.size() > 0){
    Pair p = pq.remove();
    if(visited[p.v]){
        continue;
    }
    visited[p.v] = true;

    System.out.println(p.v+" via "+p.path+" @ "+p.wt);

    for(Edge e: graph[p.v]){
        if(visited[e.nbr] == false){
            pq.add(new Pair(e.nbr, p.path+e.nbr, p.wt+e.wt));
        }
    }
}
```

0 1 0+10

0 3 0+40

0, "0", 0

1, 01, 10

3, 03, 40

2, 012, 20

3, 0123, 30

5, 0125, 25

4, 01254, 28

6, 01256, 28

4, 012564, 36

3, 012543, 30

0 via 0 @ 0

1 01 10

2 012 20

5 0125 25

6 01256 28

4 01254 28

3, 0123, 30

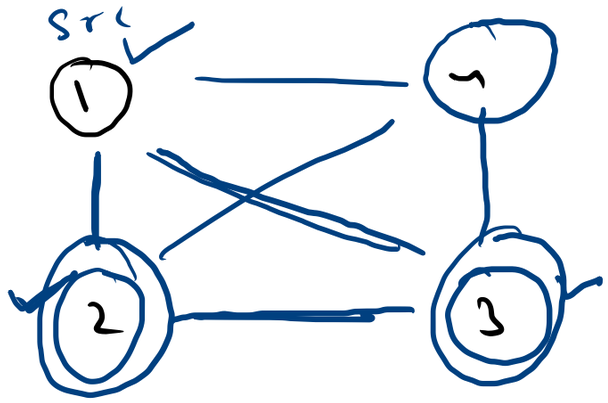
$log_2(n)V^2$ worst case

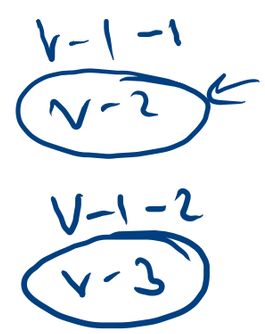
```java
PriorityQueue<Pair> pq = new PriorityQueue<>();
boolean visited[] = new boolean[vtces];
pq.add(new Pair(src, src+"", 0));

while(pq.size() > 0){
    Pair p = pq.remove();
    if(visited[p.v]){
        continue;
    }
    visited[p.v] = true;

    System.out.println(p.v+" via "+p.path+" @ "+p.wt);

    for(Edge e: graph[p.v]){
        if(visited[e.nbr] == false){
            pq.add(new Pair(e.nbr, p.path+e.nbr, p.wt+e.wt));
        }
    }
}
```

V

$E = V(V-1)$

$\simeq V^2$





V times

pq = V-1, V-2, V-3 ....

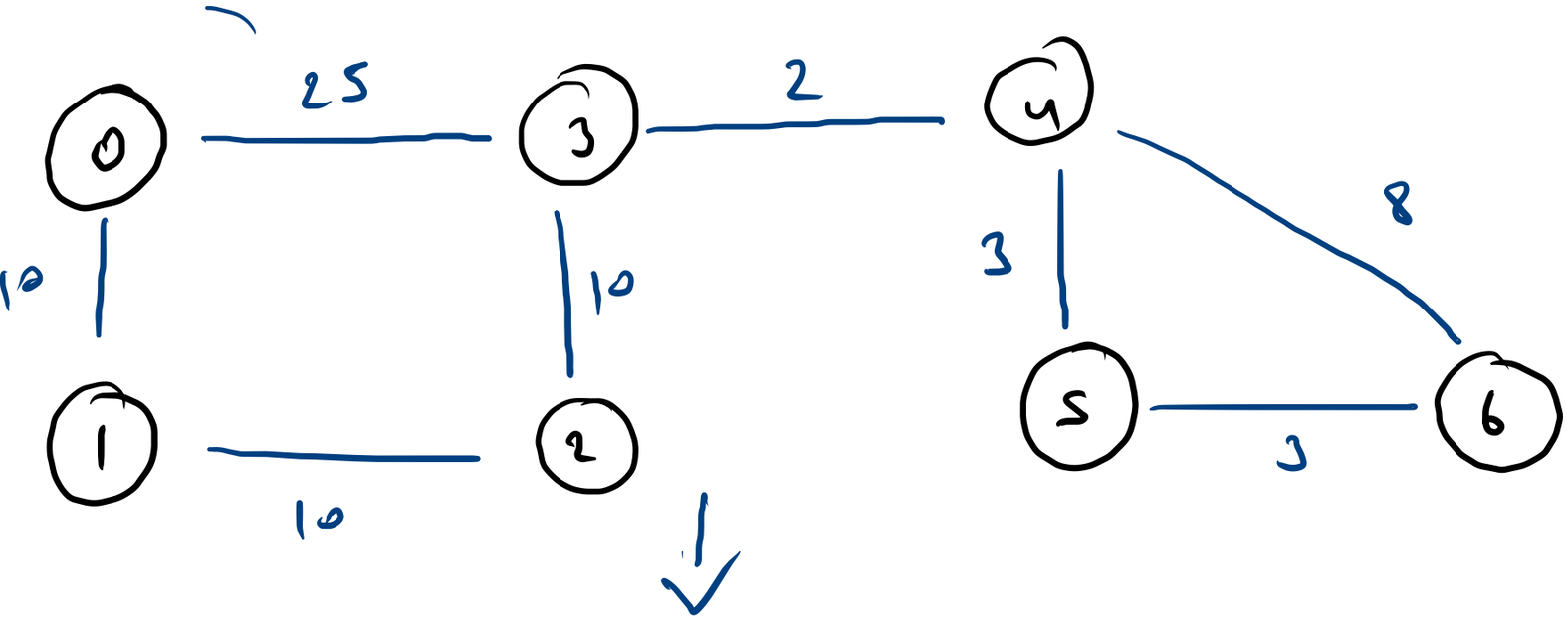
V-1, V-2, V-3 ..... V-V

V+ V-1, 5 4 3 2 1 0

$\frac{V(V+1)}{2} = V^2$

$O(V^2 log_2(V^2)) \Rightarrow O(E log_2(V^2))$

$O^2 E \log V$



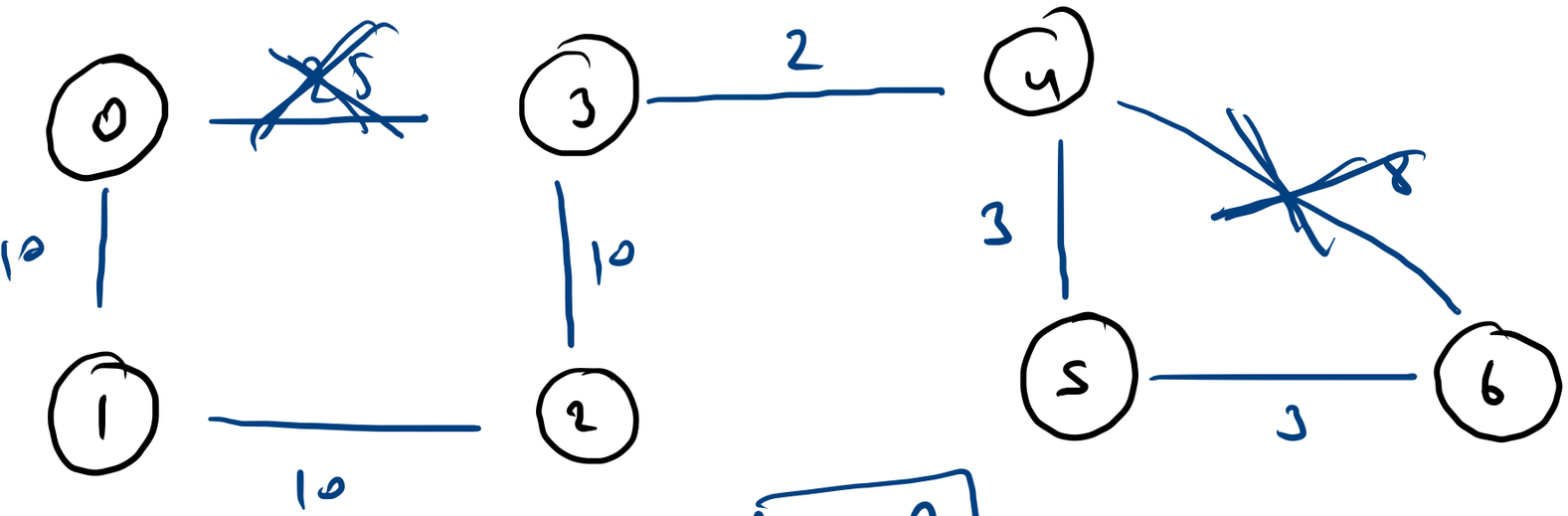
spanning tree

V connects
min. numb~ E

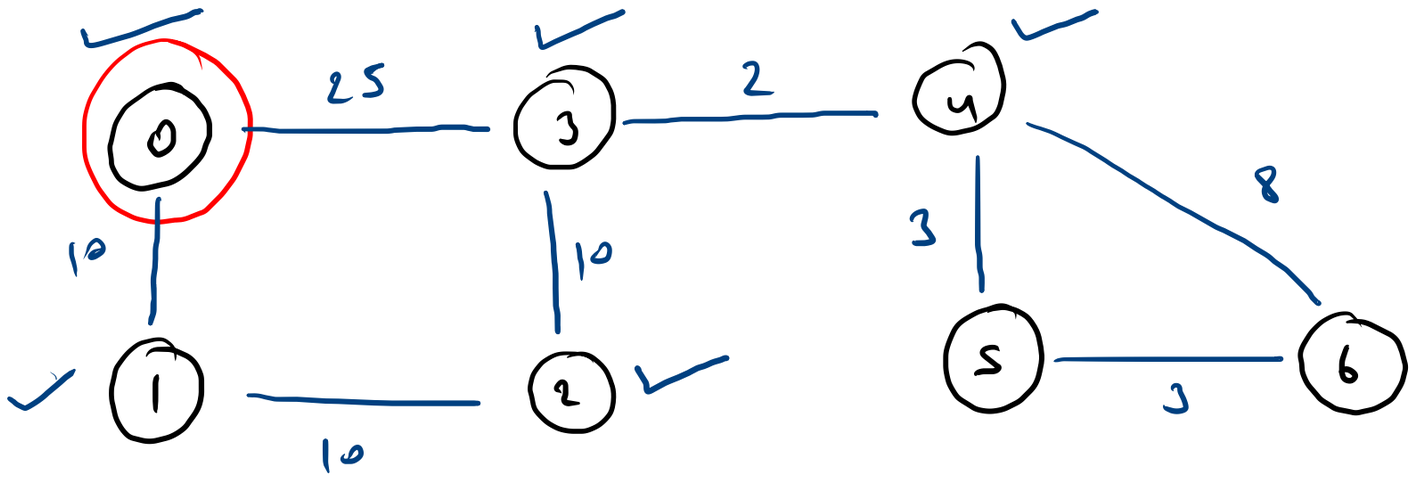
V
E → V-1

| Σ wt |

minimum



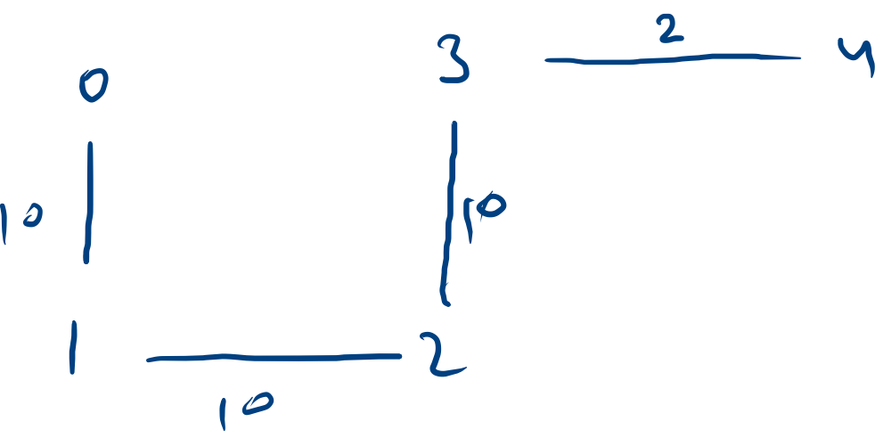
V = 7
E = 6



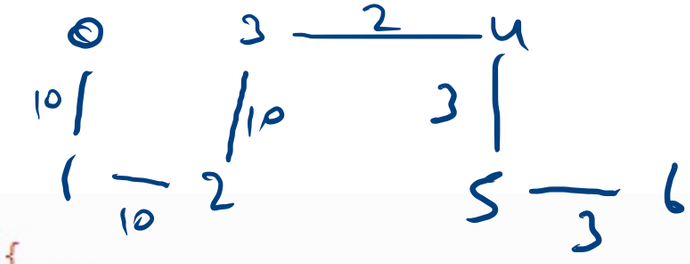
Prim's algo

Spanning tree

min
max



~~0, 0~~	~~4, 2~~
~~1, 10~~	5, 3
3, 25	6, 8
~~2, 10~~	
~~3, 10~~	



```
while(pq.size() > 0){
    Pair p = pq.remove();

    if(visited[p.v]){
        continue;
    }
    visited[p.v] = true;

    if(p.u != -1)
        System.out.println("["+p.v+"-"+p.u+"@"+p.wt+"]");

    for(Edge e: graph[p.v]){
        if(visited[e.nbr] == false){
            pq.add(new Pair(e.nbr, p.v, e.wt));
        }
    }

}
```

src
parent
wt