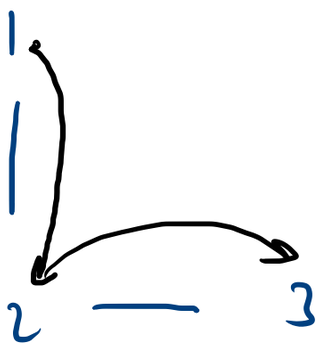
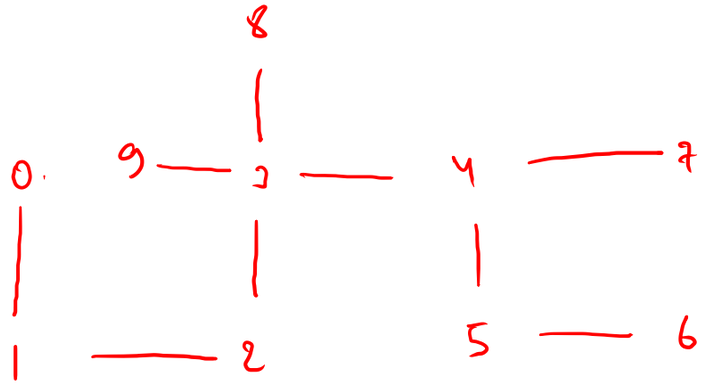
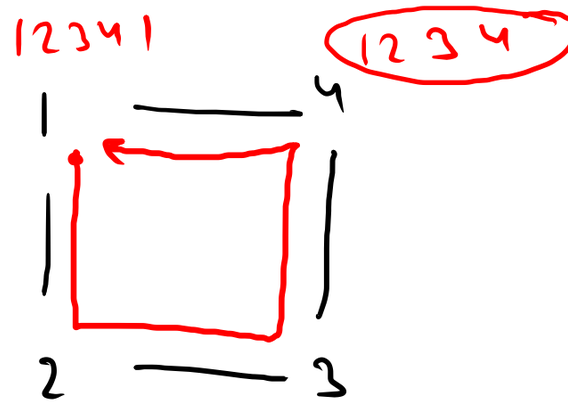
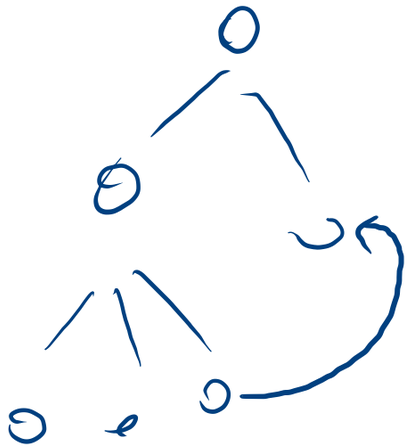
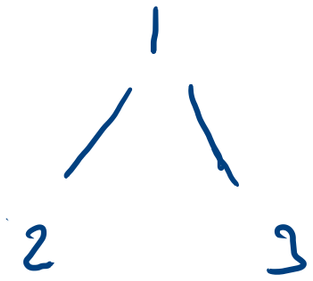


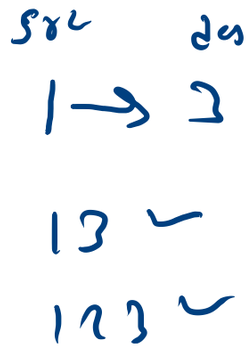


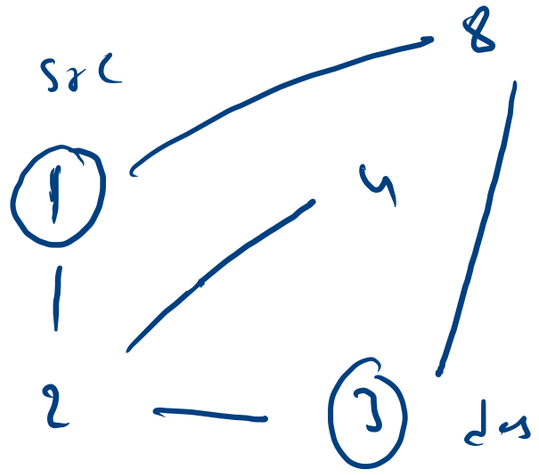


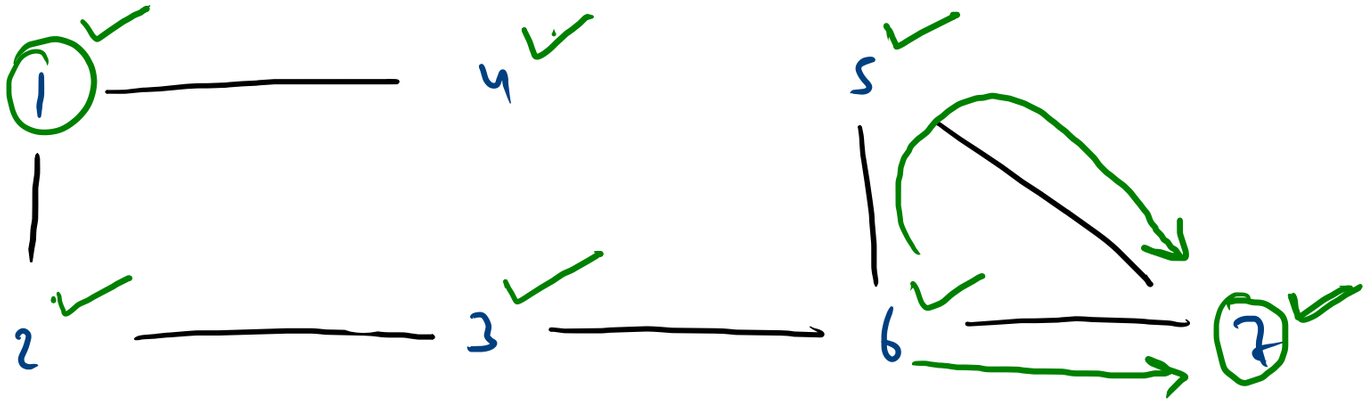


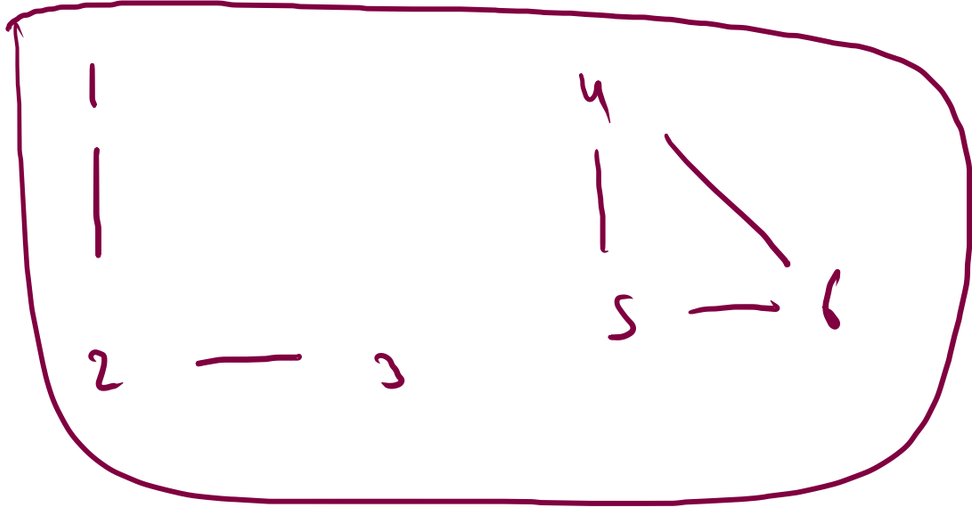
0	1	10
1	2	10
2	3	10
3	4	10
4	5	10
5	6	10



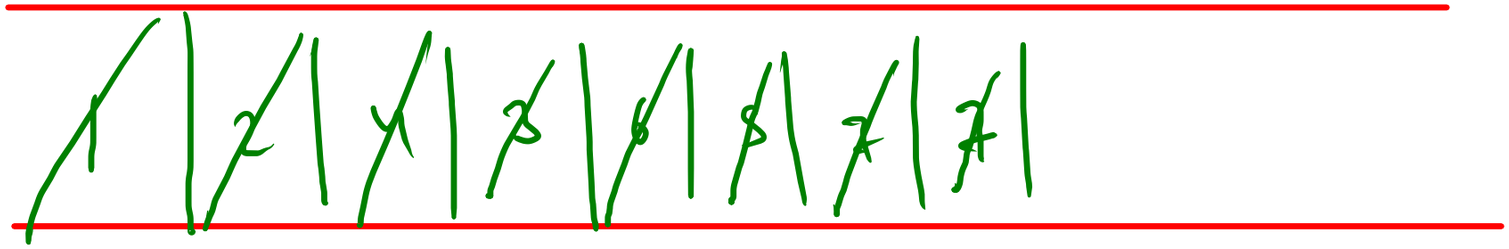




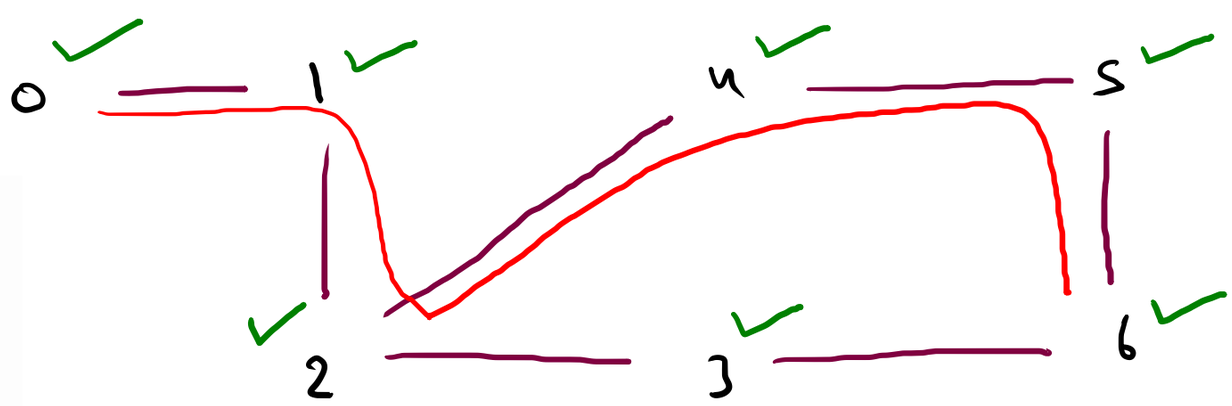


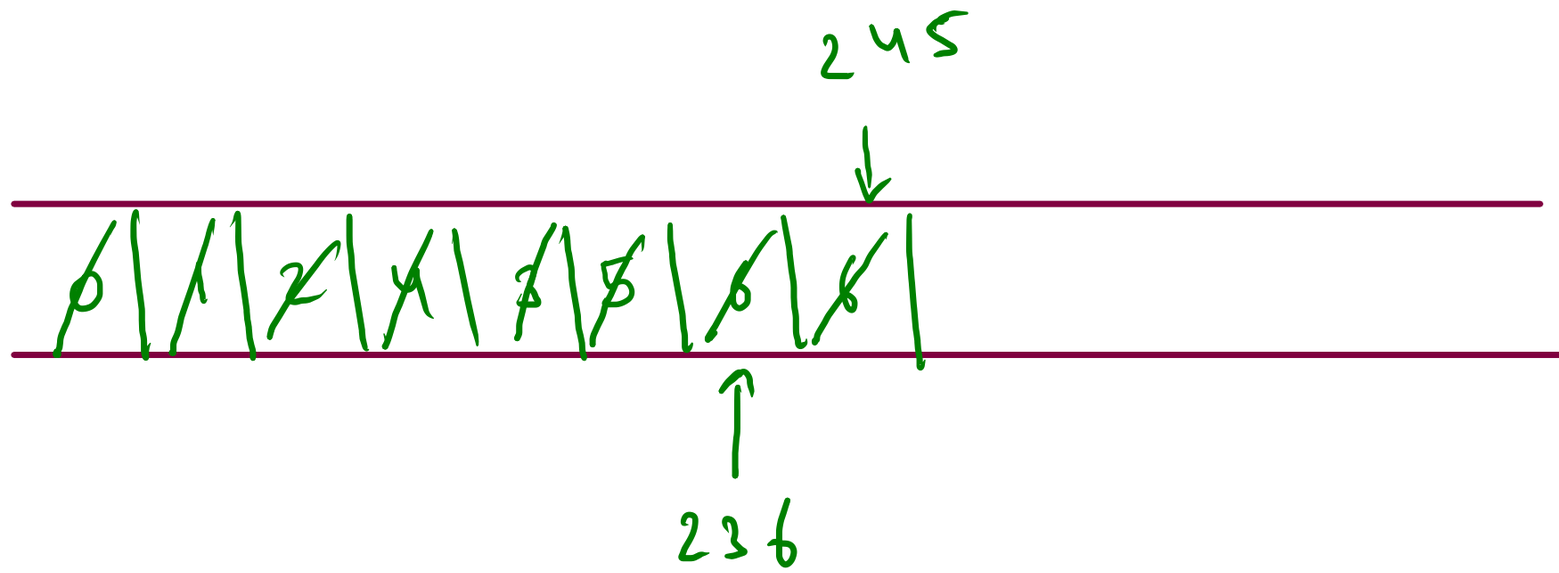


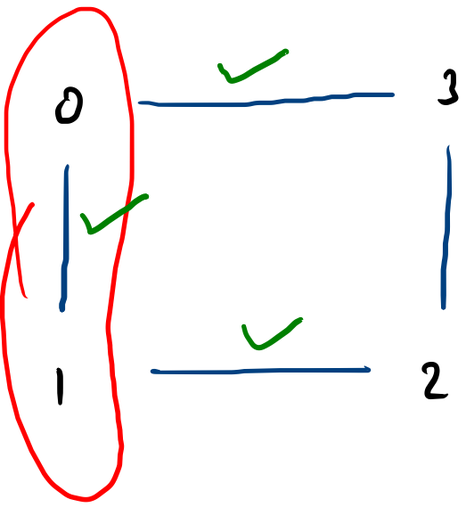
γ m all chil

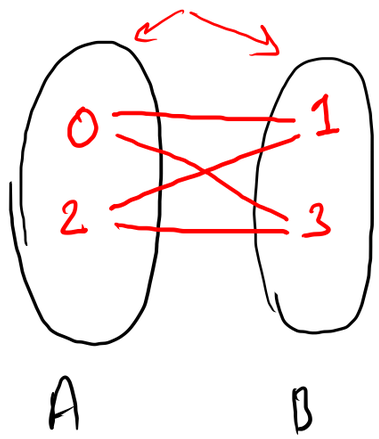


```java
Queue<Integer> q = new ArrayDeque<>();

q.add(src);


while(q.size() >0){
        int e = q.remove();

        if(visited[e]){
            return true;
        }
        visited[e] = true;

        for(Edge edge: graph[e]){
            if(visited[edge.nbr] == false){
                q.add(edge.nbr);
            }
        }
}

return false;
```



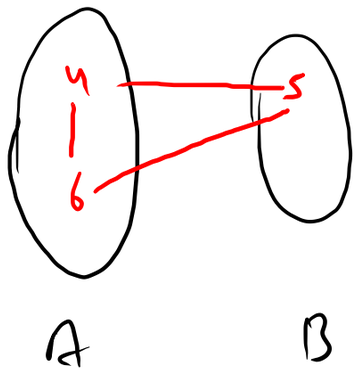


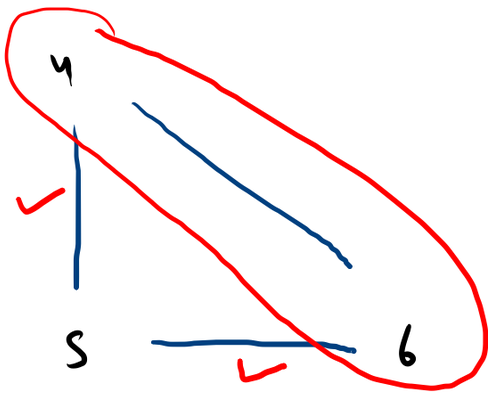




True ✓

False



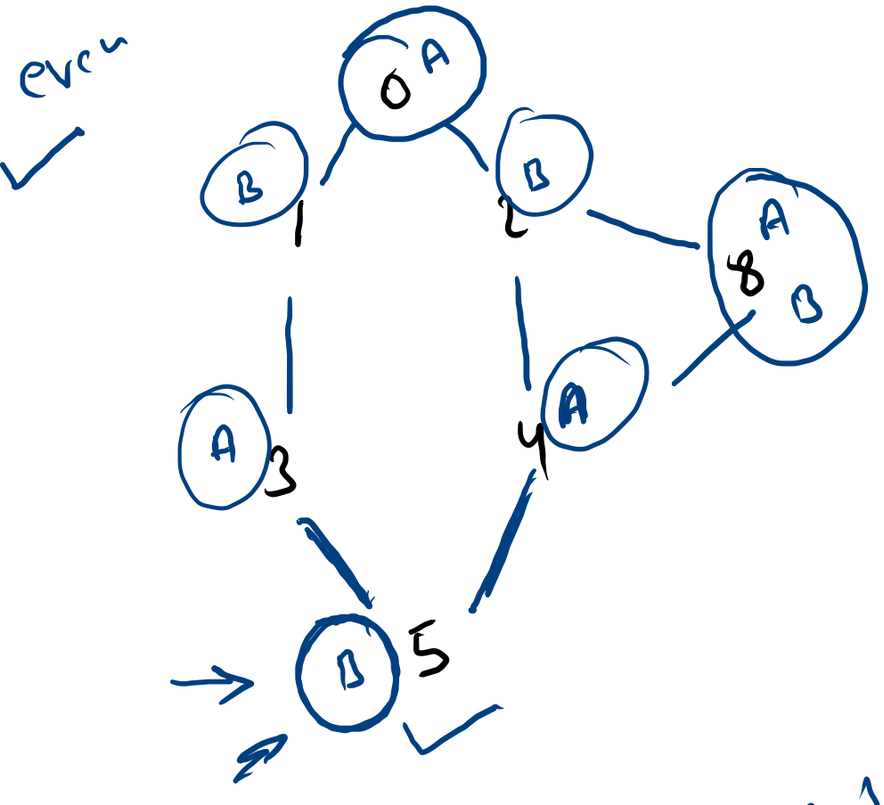


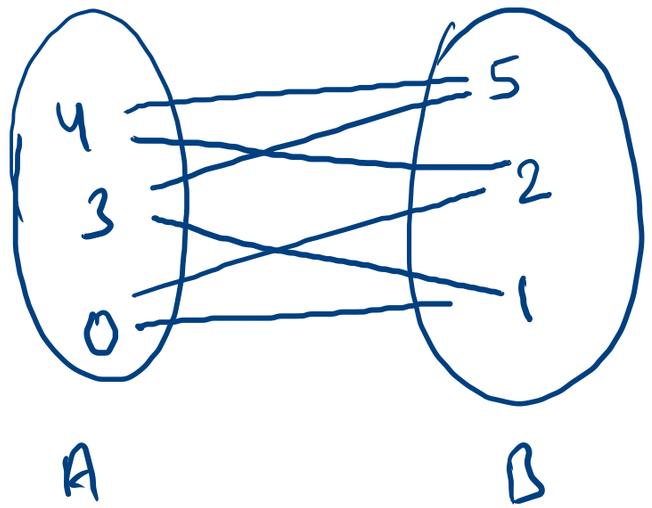
4 — 5
|
6

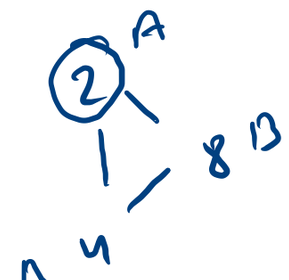
False ✓

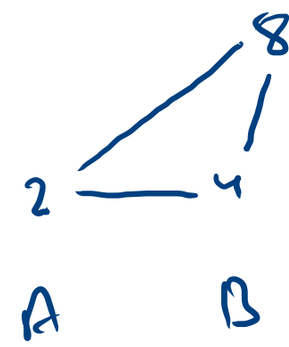
4 — 6
|
5

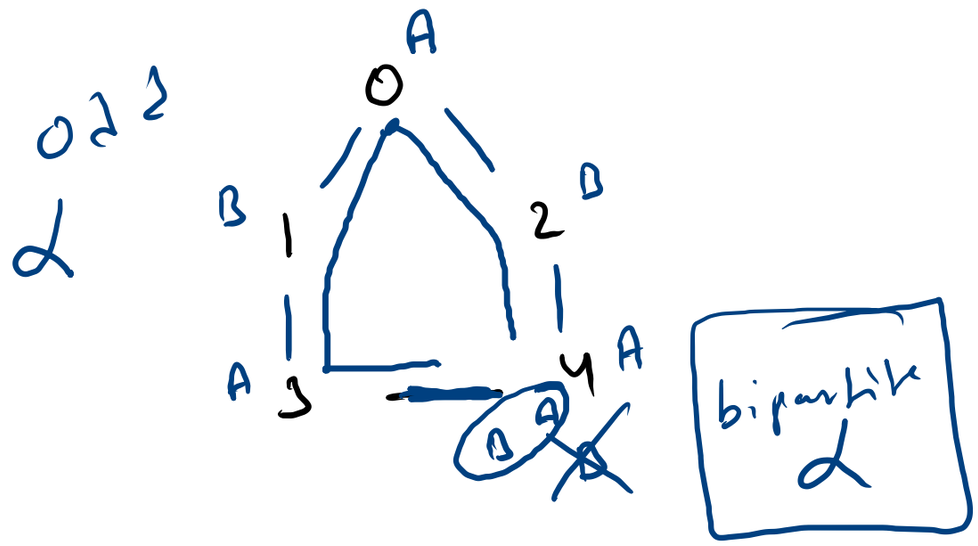
4 — 5
|
6

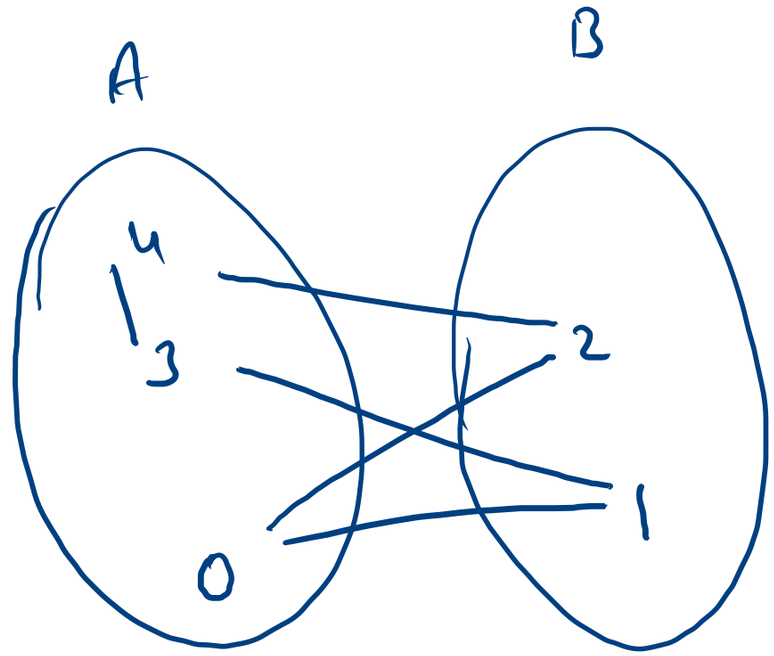

evet
0 A
B 1
2 B
A 3
4 A
8 A B
B 5

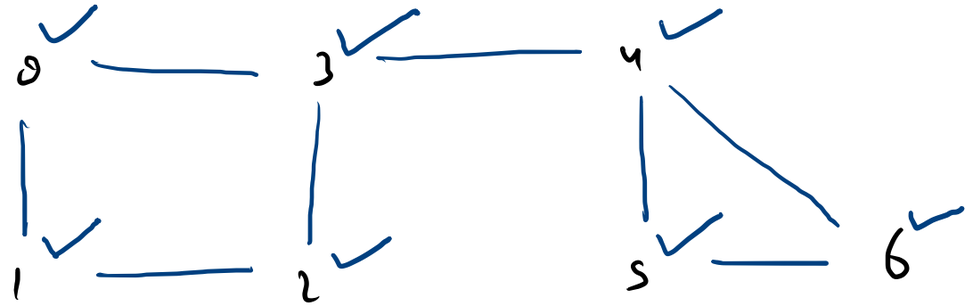

4
3
0
5
2
1
A
B

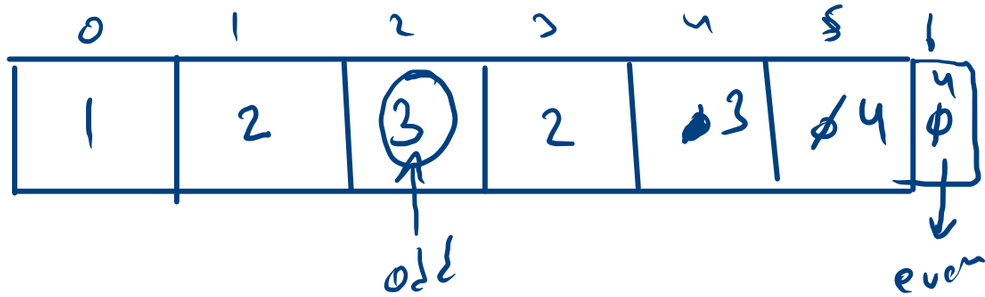

2 A
8 B
B 4


8
2
4
A
B

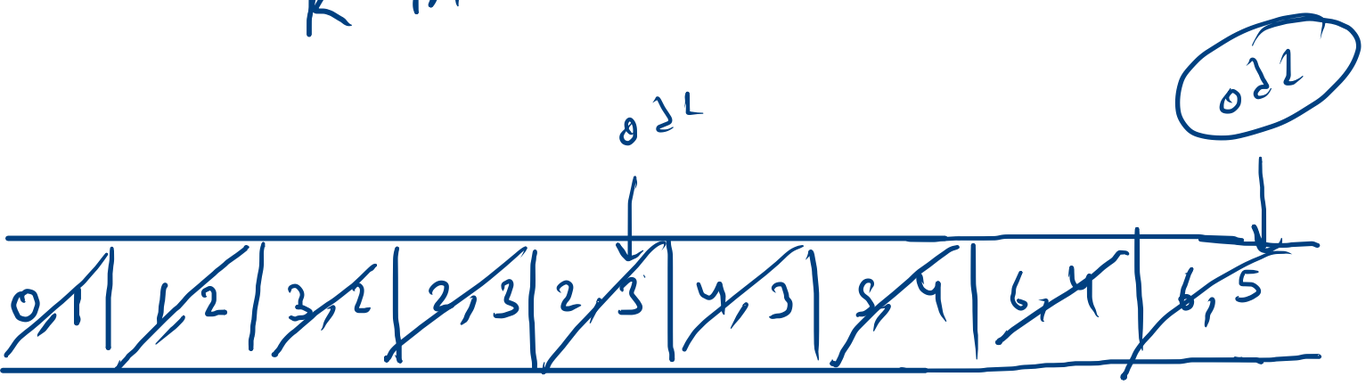

0 2 2
A
0
B 1
2 B
A 3
4 A
1 2
bipartik
L

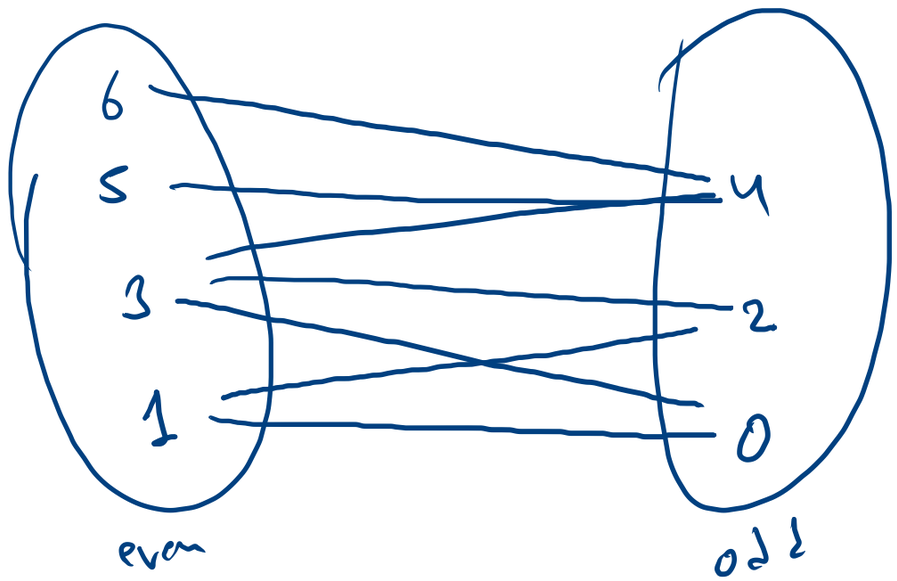

A
B
4
3
2
1
0


0
3
4
1
2
5
6

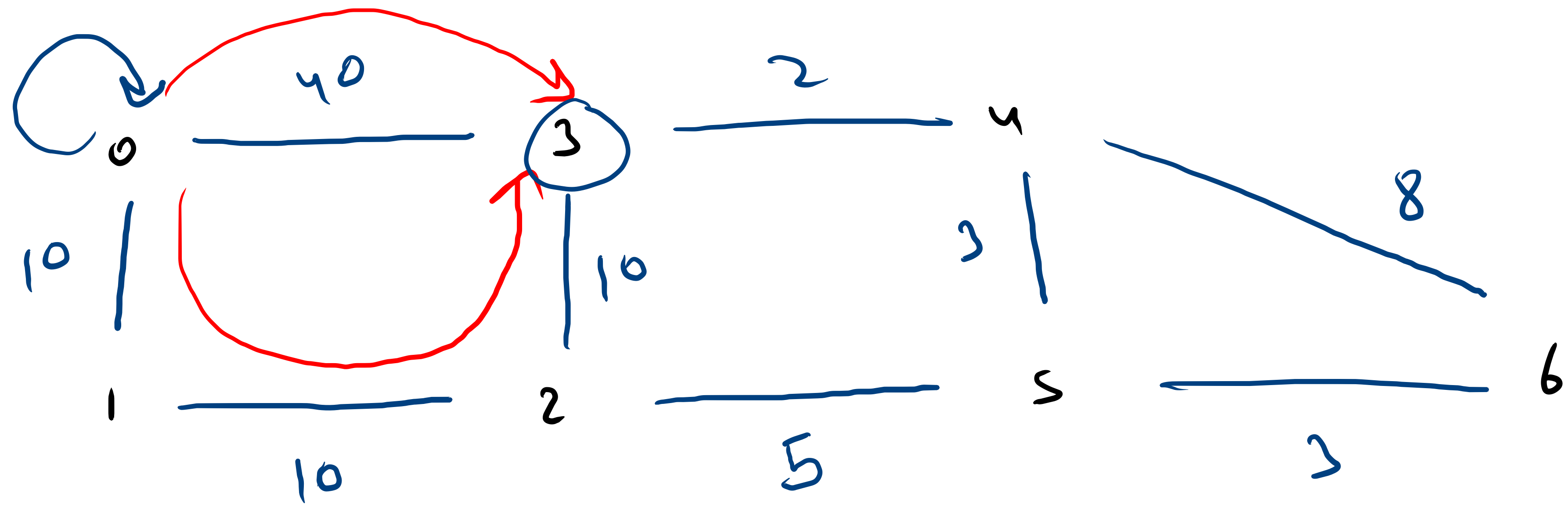

0 1 2 3 4 5 6
1
2
3
2
3
4
4
odd
even

R M


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


6
5
3
1
4
2
0
even
odd

edges

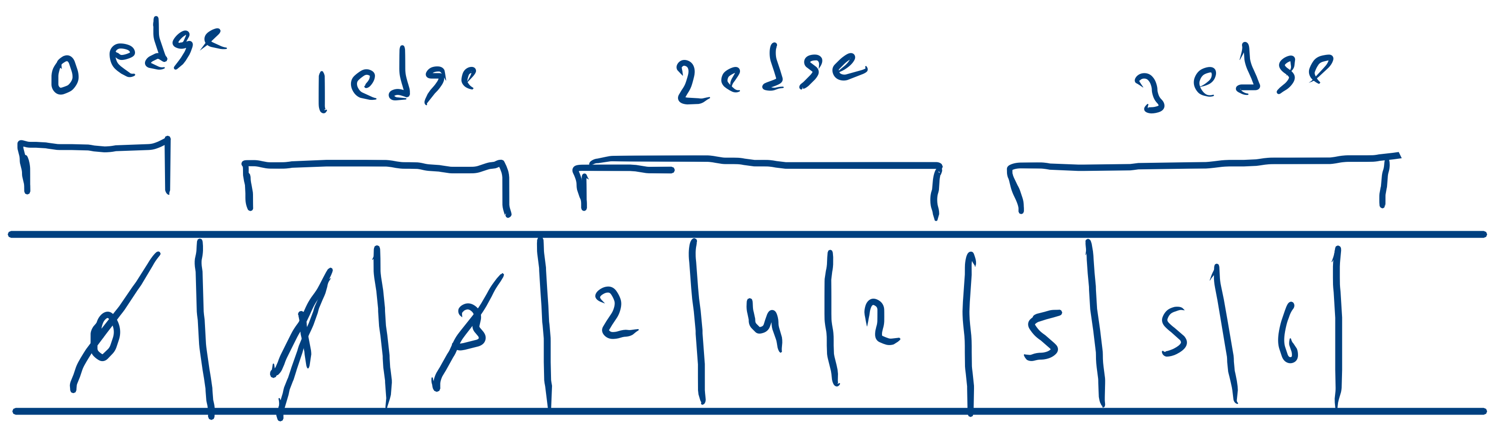


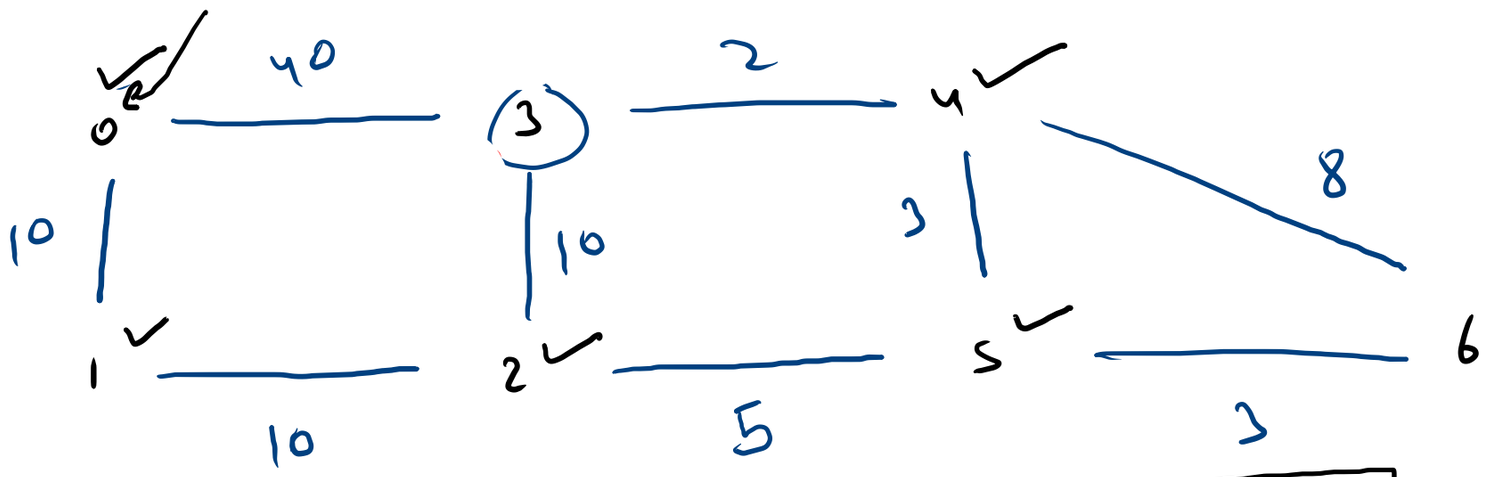
3 via 0123 (a) 30

5 via 0125 (a) 25

Queue (in 1)

45
55
25





DYKSTRA

```
PriorityQueue<Pair> q = new PriorityQueue<>();
q.add(new Pair(src, ""+src, 0));

boolean visited[] = new boolean[vtces];
while(q.size() > 0){
    Pair p = q.remove();
    if(visited[p.src] == true)continue;
    visited[p.src]=true;
    System.out.println(p.src+" via "+p.path+" @ "+p.wt);
    for(Edge e: graph[p.src]){
        if(visited[e.nbr] == false){
            q.add(new Pair(e.nbr, p.path+e.nbr, p.wt+e.wt));
        }
    }
}
```

des	cost
0	0
1	10
2	20

src path wt

4, "01254", 28

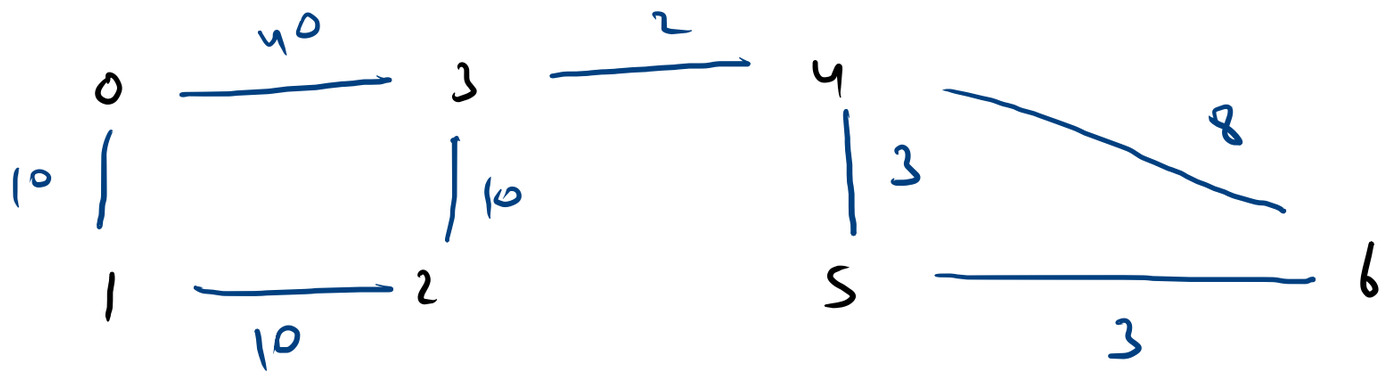
6, "01256", 28

3, "03", 40

6, "012546", 36

3, "0123", 30

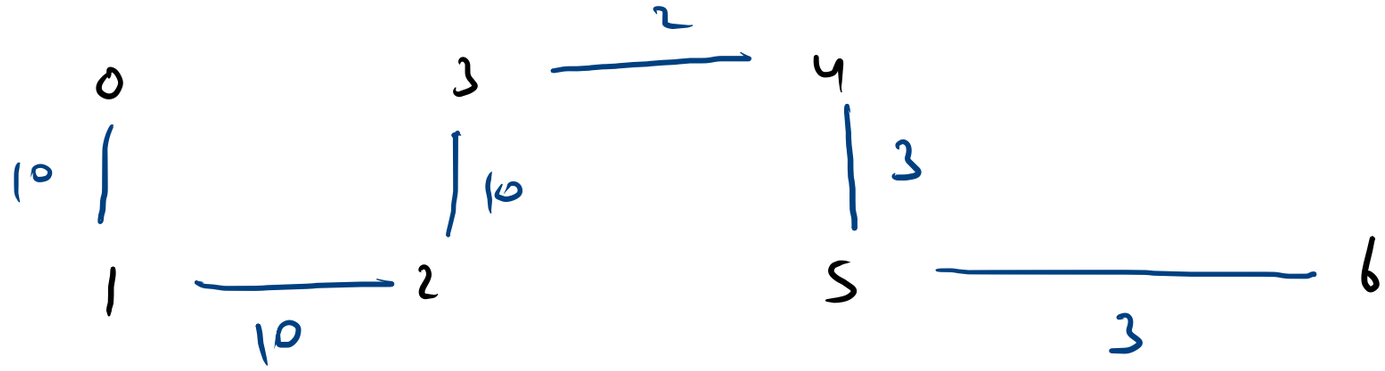
3, "012543", 30



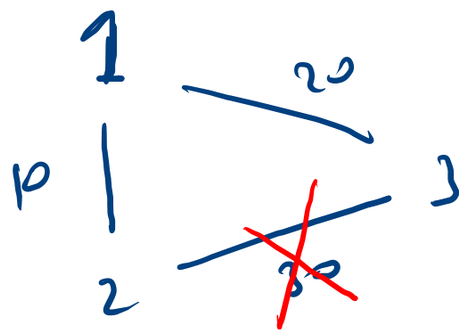
u to v wt

0 — 1 @ 10

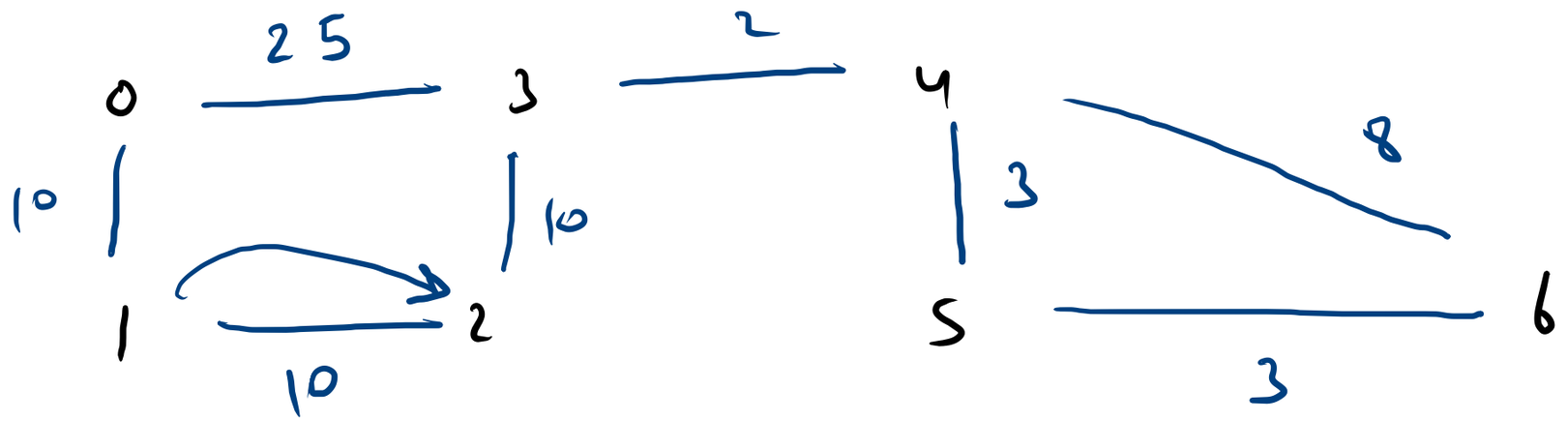
1 — 2 @ 10



Total = 30 unit



Dyic



sn → Dest

sn cost

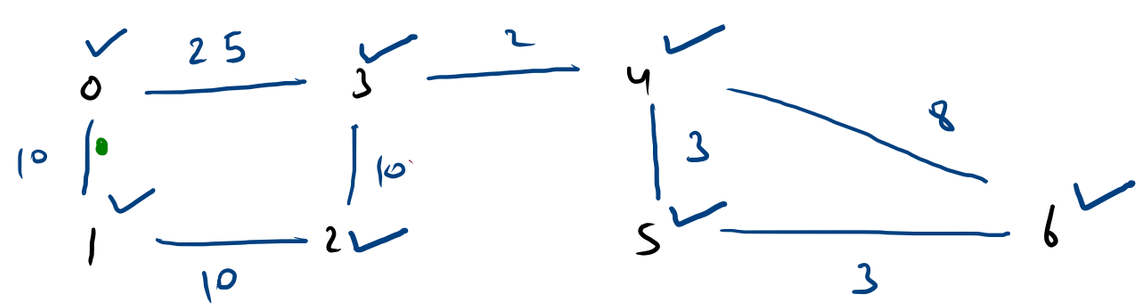
0 , 0

1 , 10

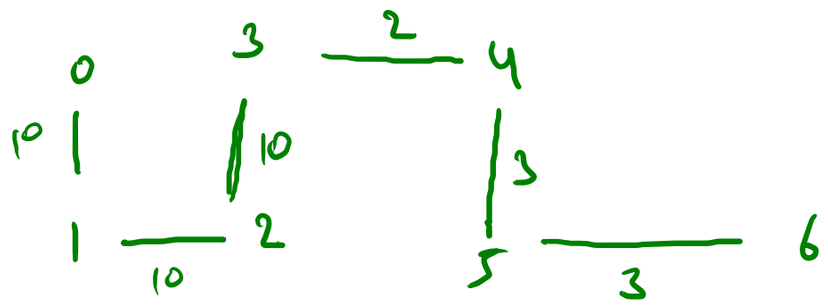
2 , 10

3 , 10

min. spanning tree

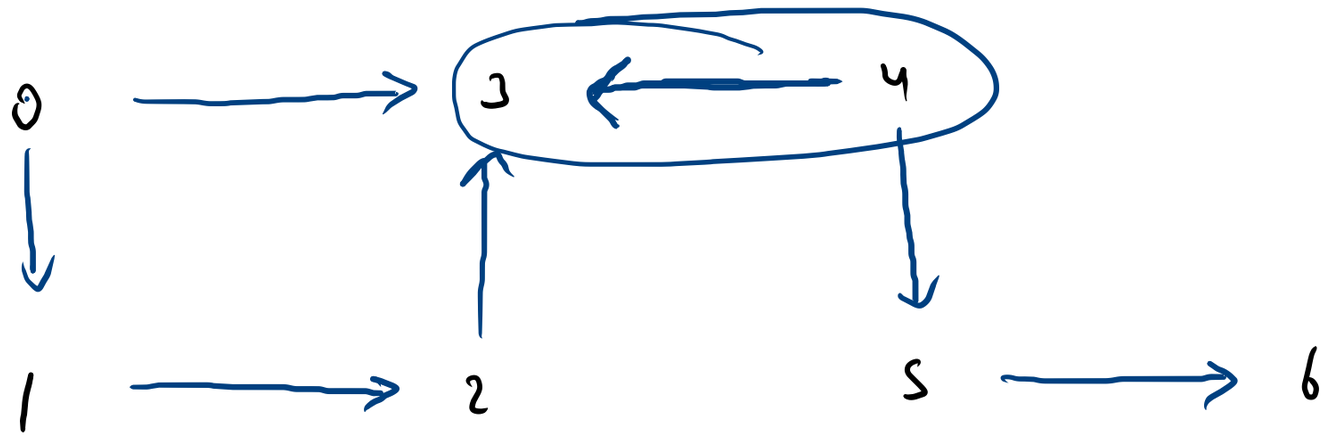


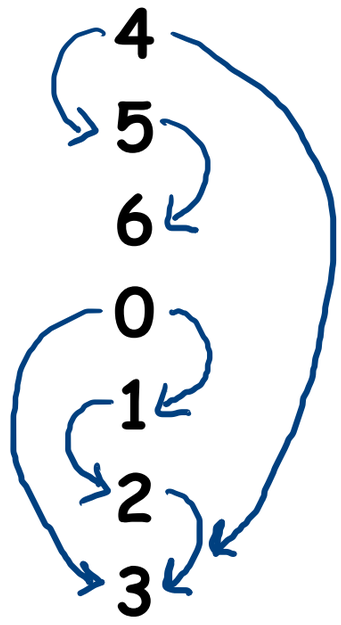
src	par	wt
3	0	25



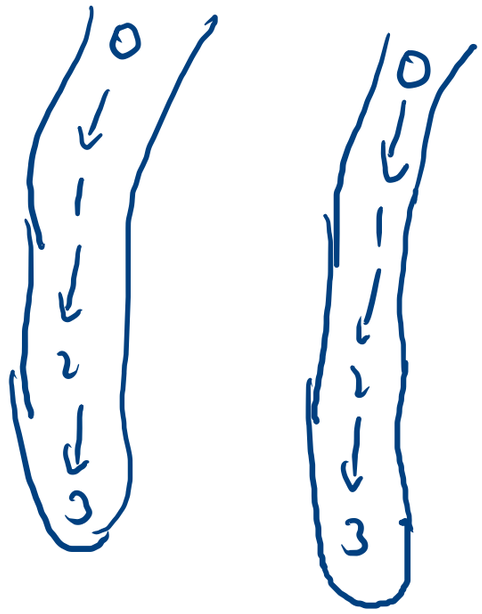
Prims Also

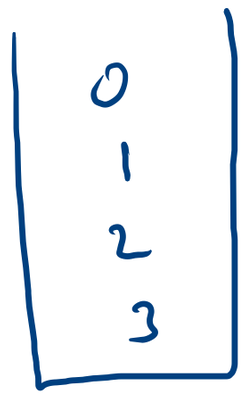
```
PriorityQueue<Pair> q = new PriorityQueue<>();
q.add(new Pair(0, -1, 0));
boolean visited[] = new boolean[vtces];
while(q.size() > 0){
    Pair p = q.remove();
    if(visited[p.src] == true)continue;
    visited[p.src]=true;
    System.out.println("["+p.src+"-"+p.par+" @ "+p.wt+"]");
    for(Edge e: graph[p.src]){
        if(visited[e.nbr] == false){
            q.add(new Pair(e.nbr, e.src, e.wt));
        }
    }
}
```

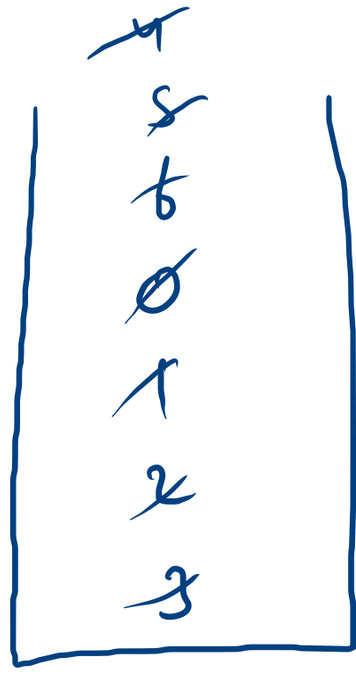




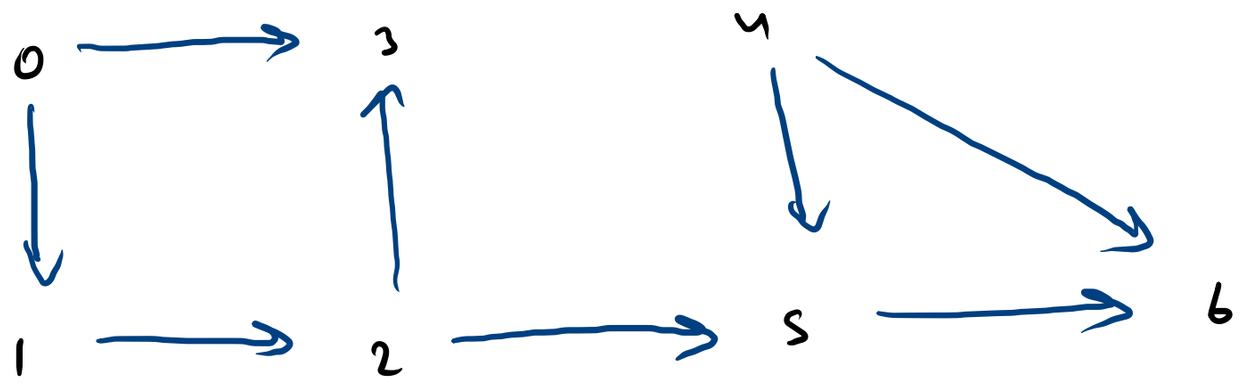
[0, 1, 2, 3]



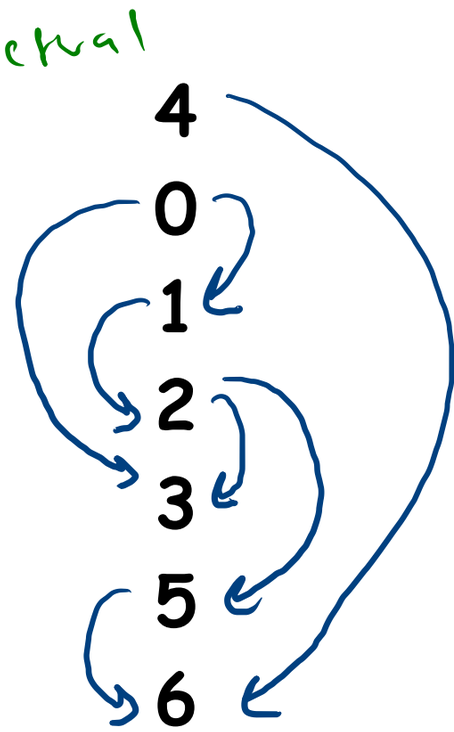


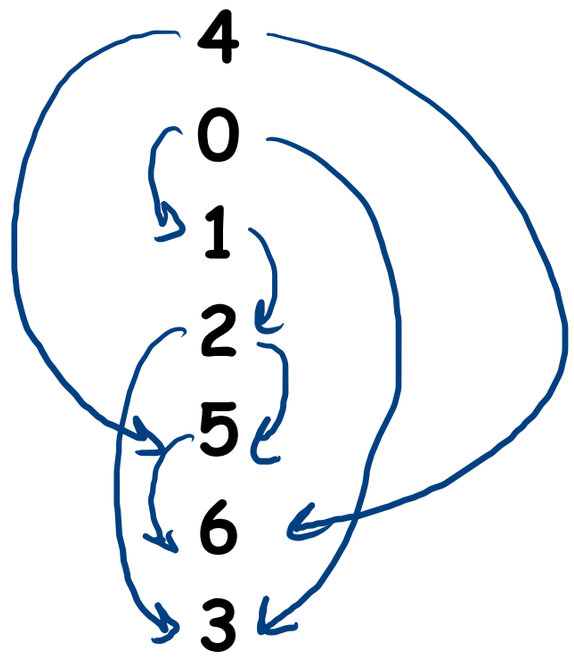


7 8
0 1
1 2
2 3
0 3
4 5
5 6
4 6
2 5

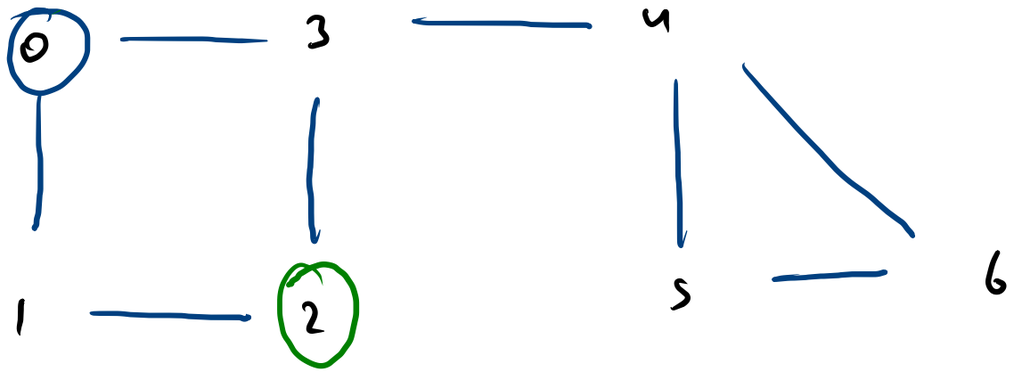


actual

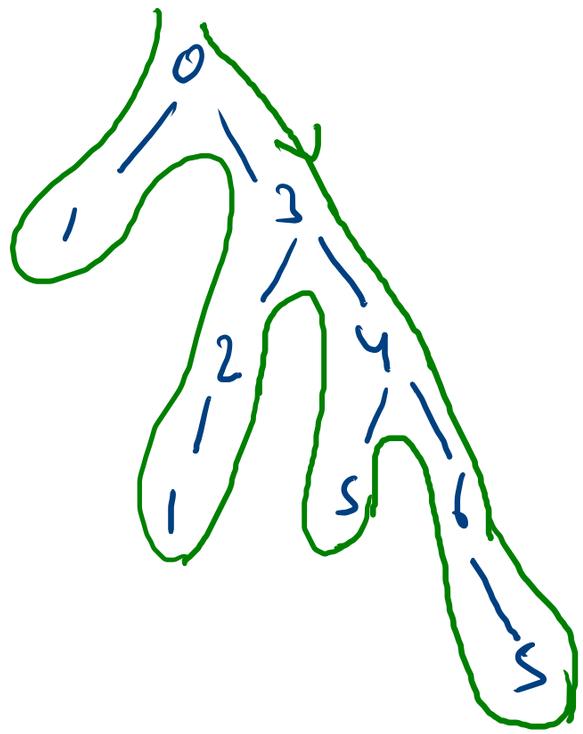




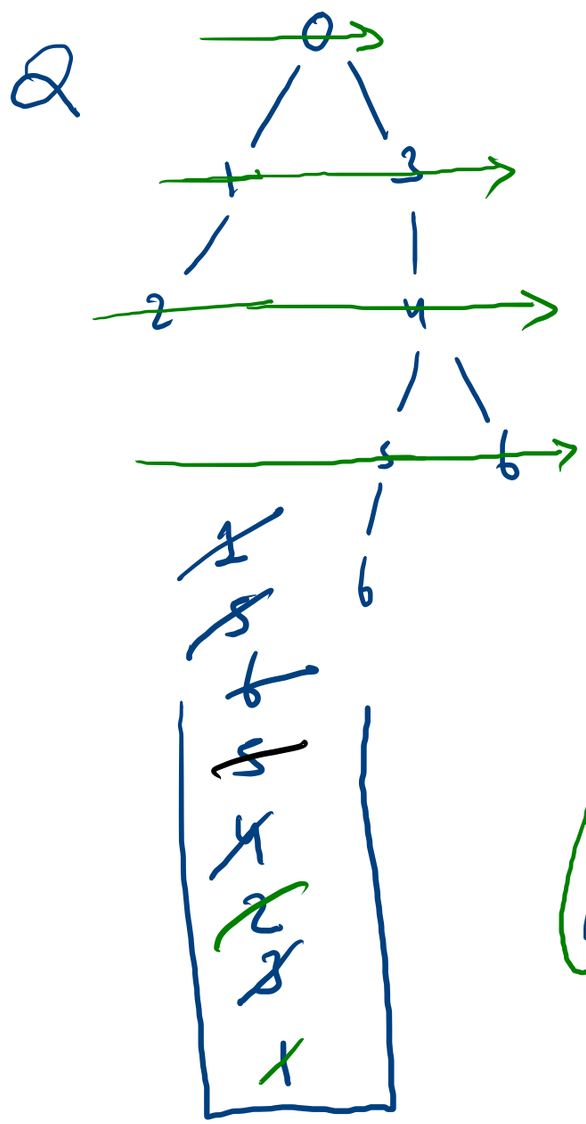
expected

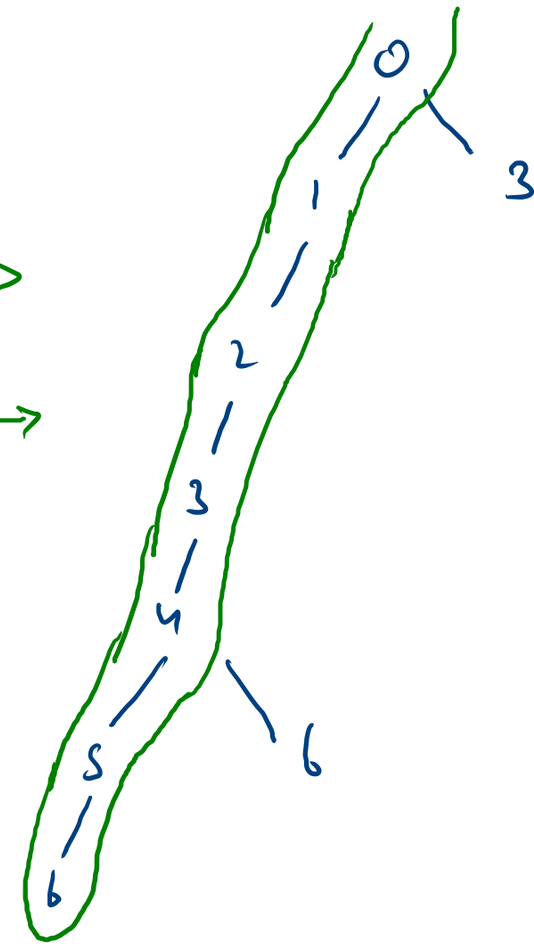


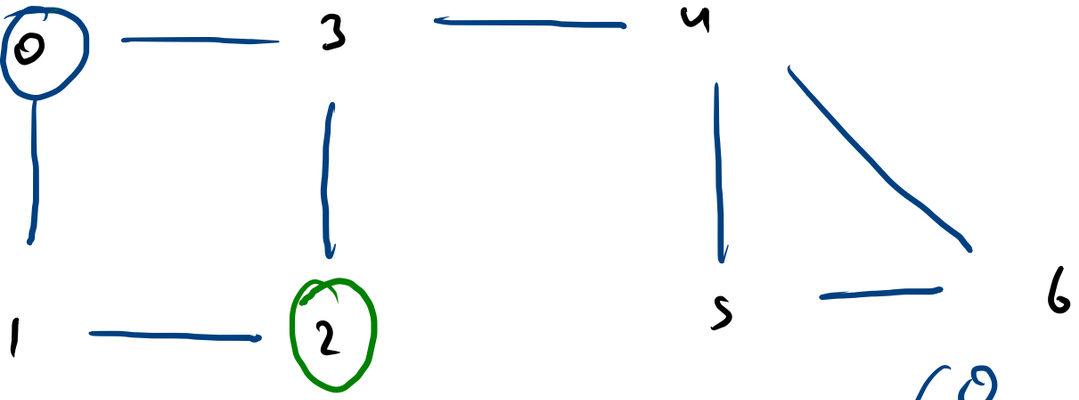
γ m ω a



reverse print order

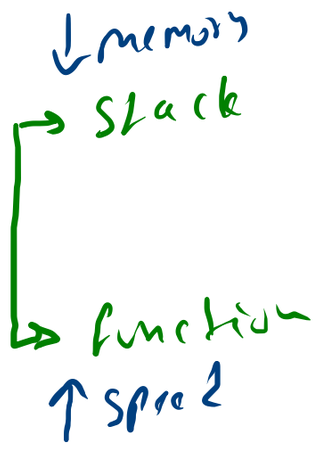


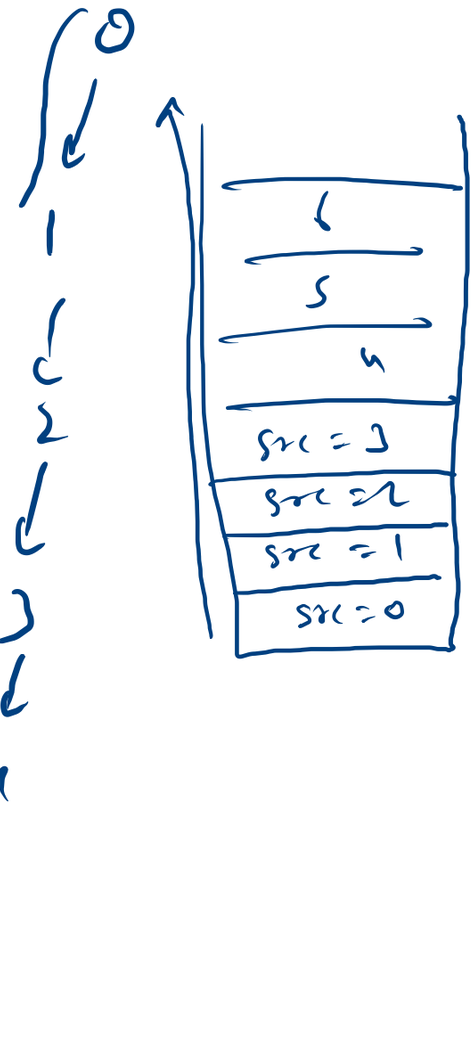


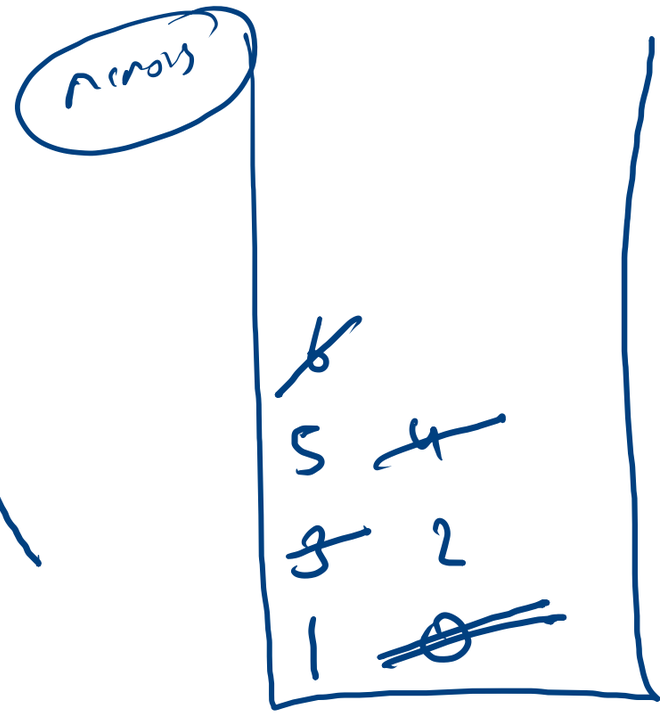


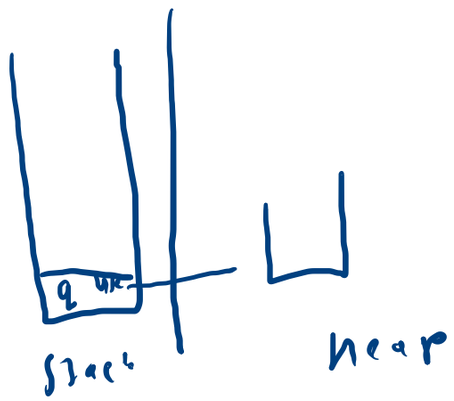
Iterative DFS

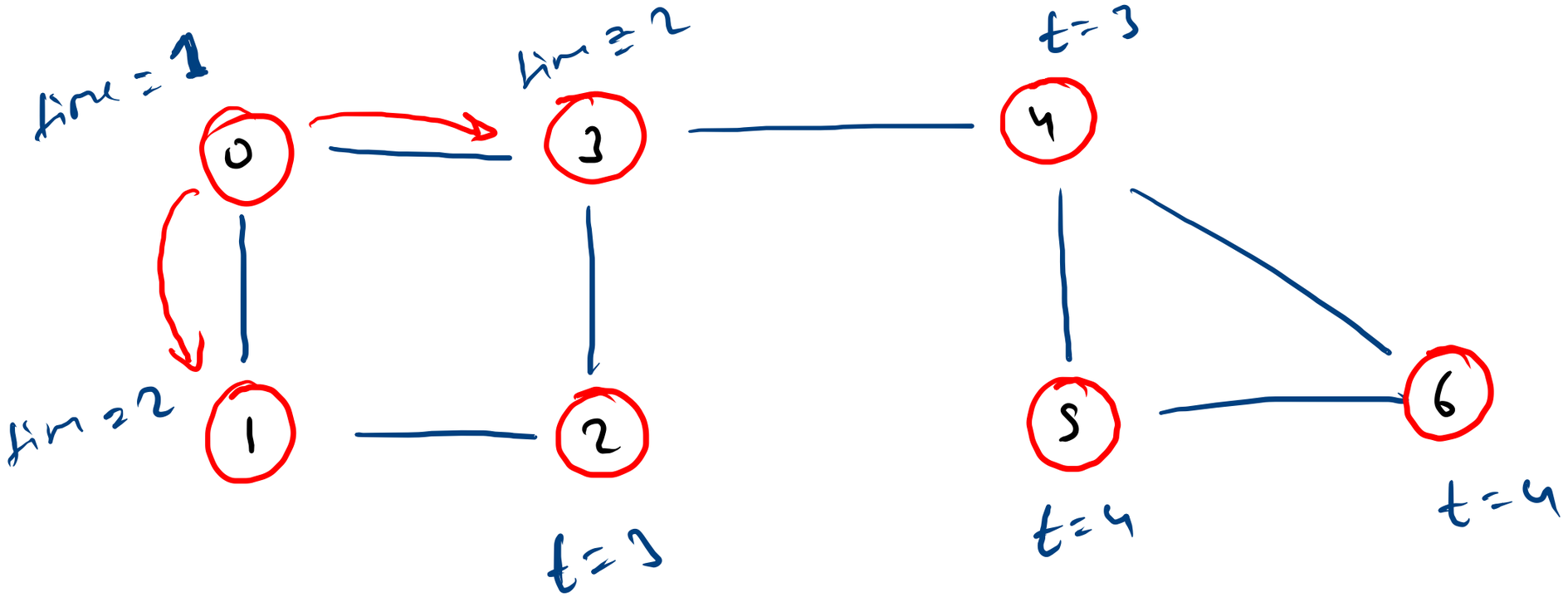
Recursive











sz(-> 0

t: 3 -> 5

t = 4 -> 7

t = 5 -> 7

7 8

0 1 10

1 2 10

2 3 10

0 3 10

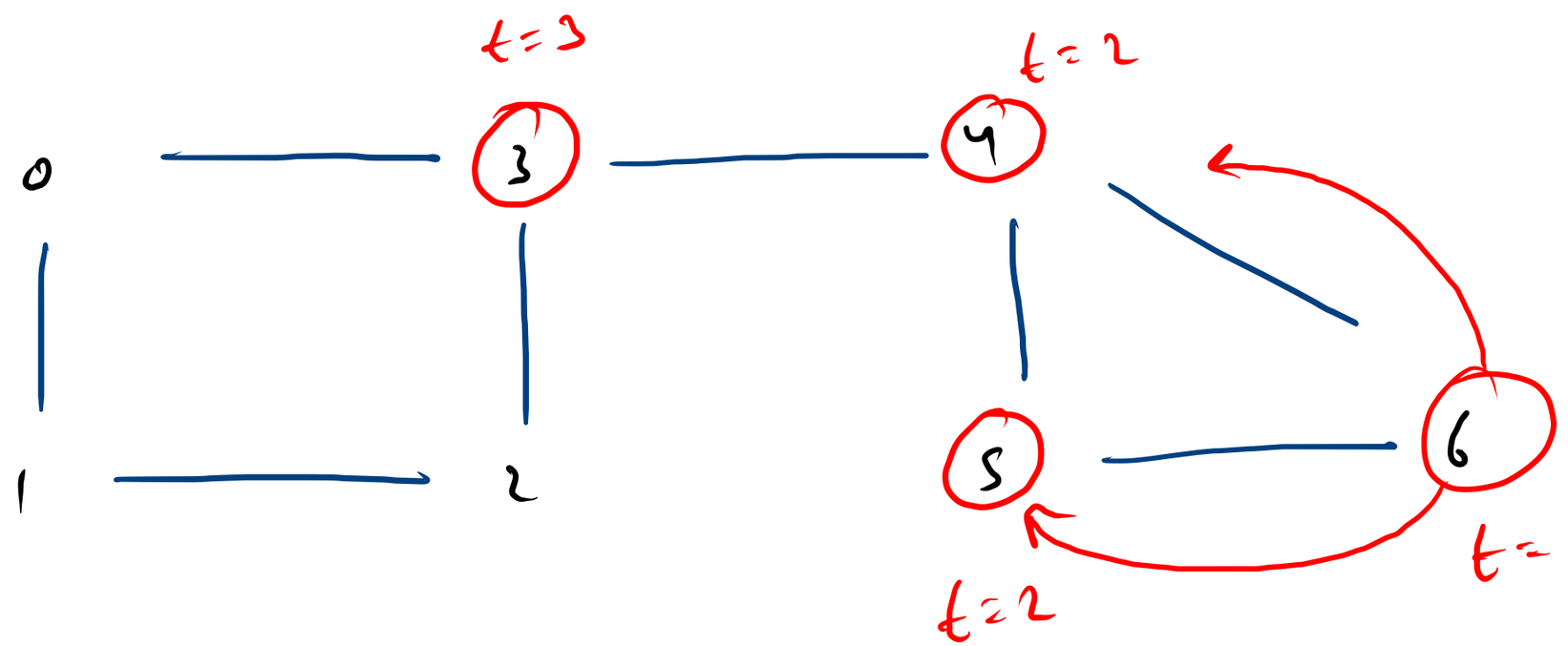
3 4 10

4 5 10

5 6 10

4 6 10

6 3



src = 6

t = 3

ans = 4