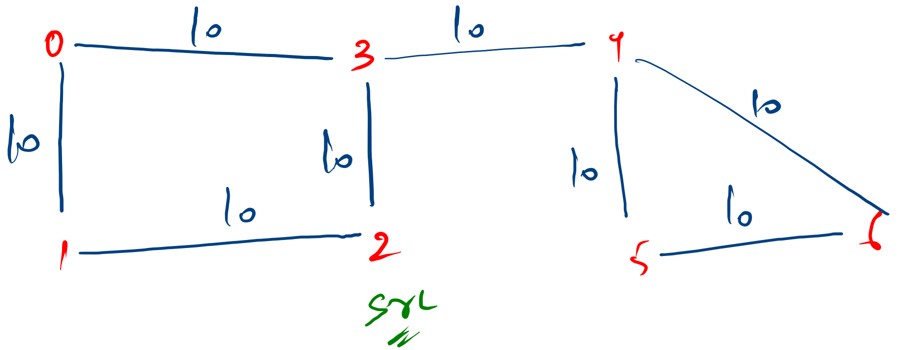
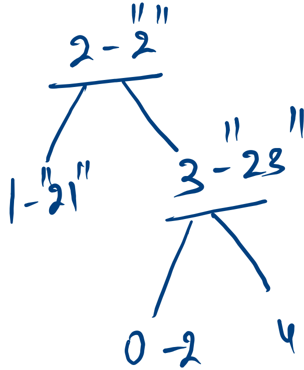
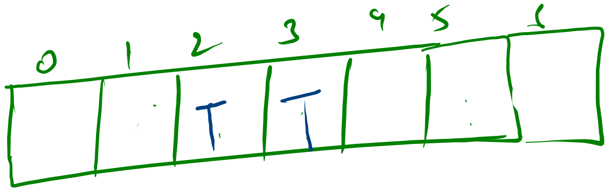
DFS →

recursion → Iteration

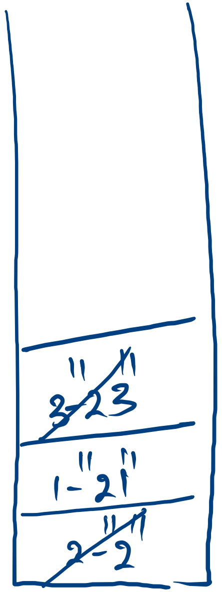




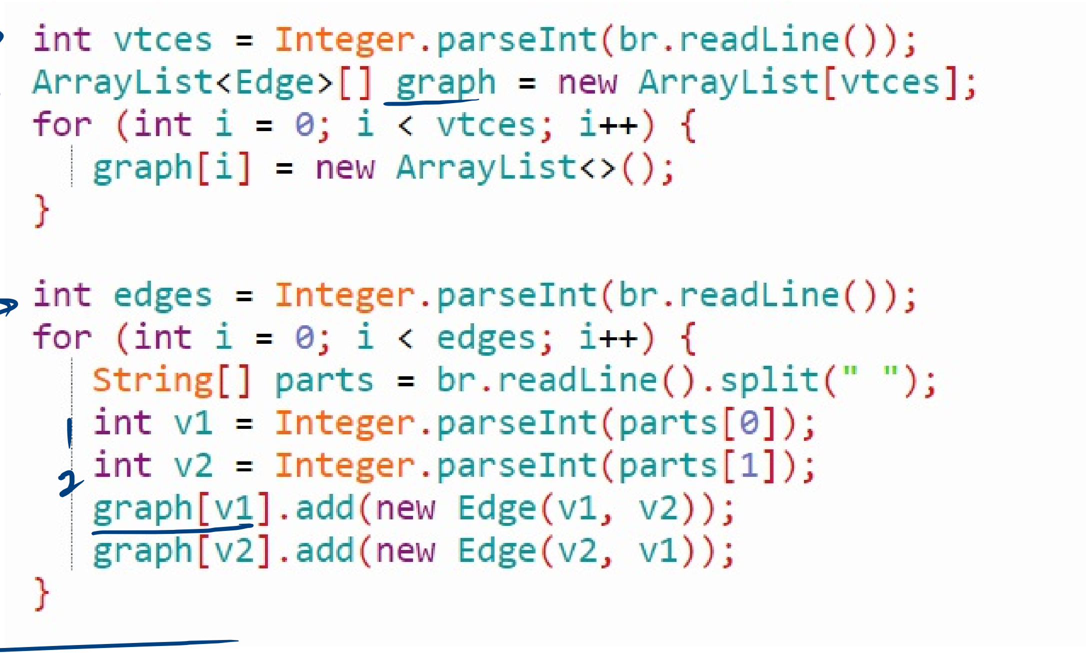
7
8
0 1 10
1 2 10
2 3 10
0 3 10
3 4 10
4 5 10
5 6 10
4 6 10
2 → src



✓ 2@2
✓ 3@23
4@234
6@2346
5@23465
0@230
1@2301

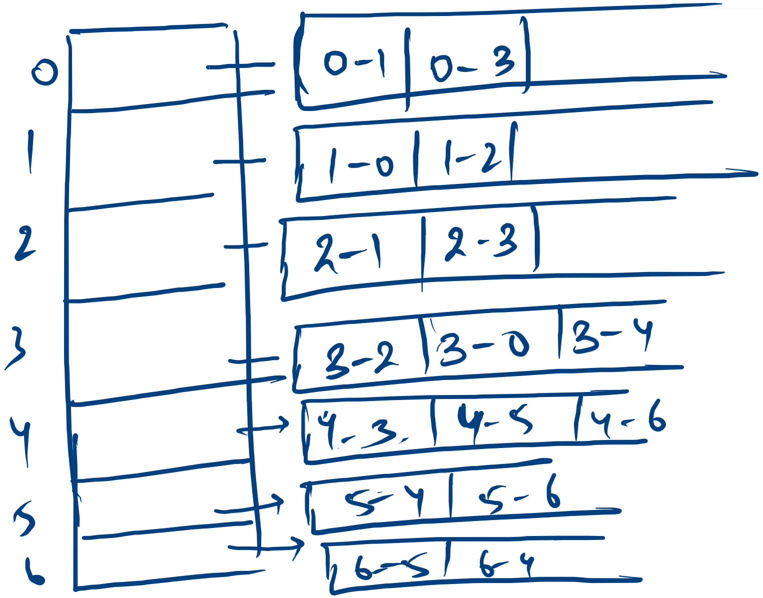


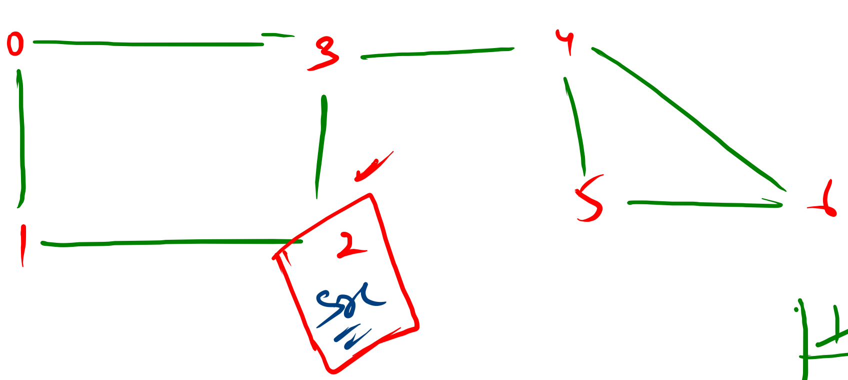
0 1 10
1 2 10
2 3 10
0 3 10
3 4 10
4 5 10
5 6 10
4 6 10

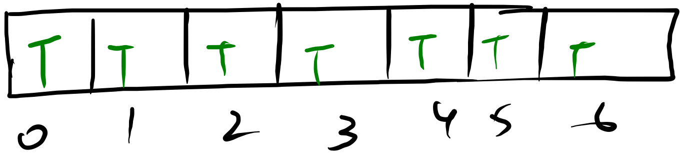


```
int vtces = Integer.parseInt(br.readLine());
ArrayList<Edge>[] graph = new ArrayList[vtces];
for (int i = 0; i < vtces; i++) {
    graph[i] = new ArrayList<>();
}

int edges = Integer.parseInt(br.readLine());
for (int i = 0; i < edges; i++) {
    String[] parts = br.readLine().split(" ");
    int v1 = Integer.parseInt(parts[0]);
    int v2 = Integer.parseInt(parts[1]);
    graph[v1].add(new Edge(v1, v2));
    graph[v2].add(new Edge(v2, v1));
}
```

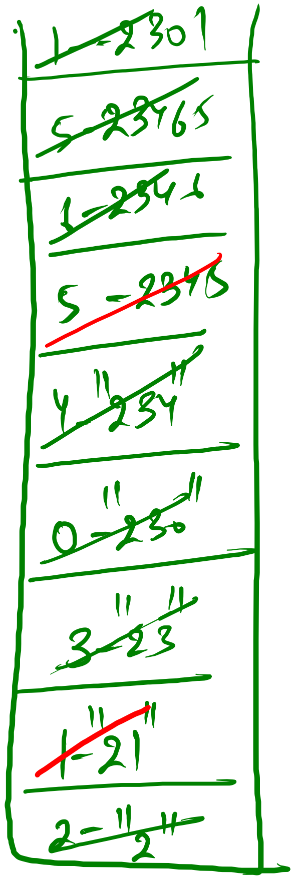


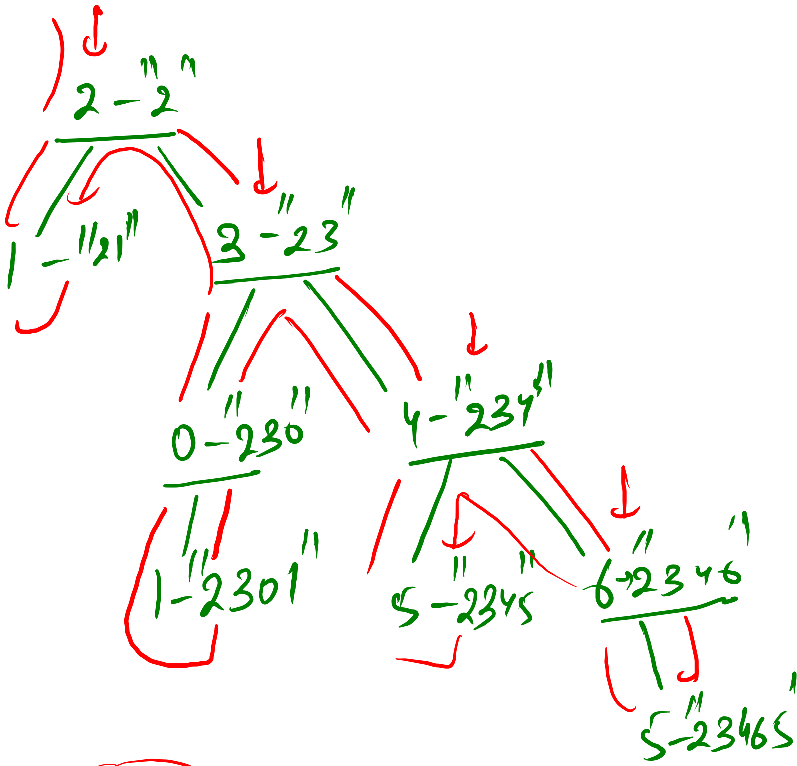




0	0-1	0-3	
1	1-0	1-2	
2	2-1	2-3	
3	3-2	3-0	3-4
4	4-3	4-5	4-6
5	5-4	5-6	
6	6-5	6-4	

2@2
3@23
4@234
6@2346
5 @23465
0@230
1@2301



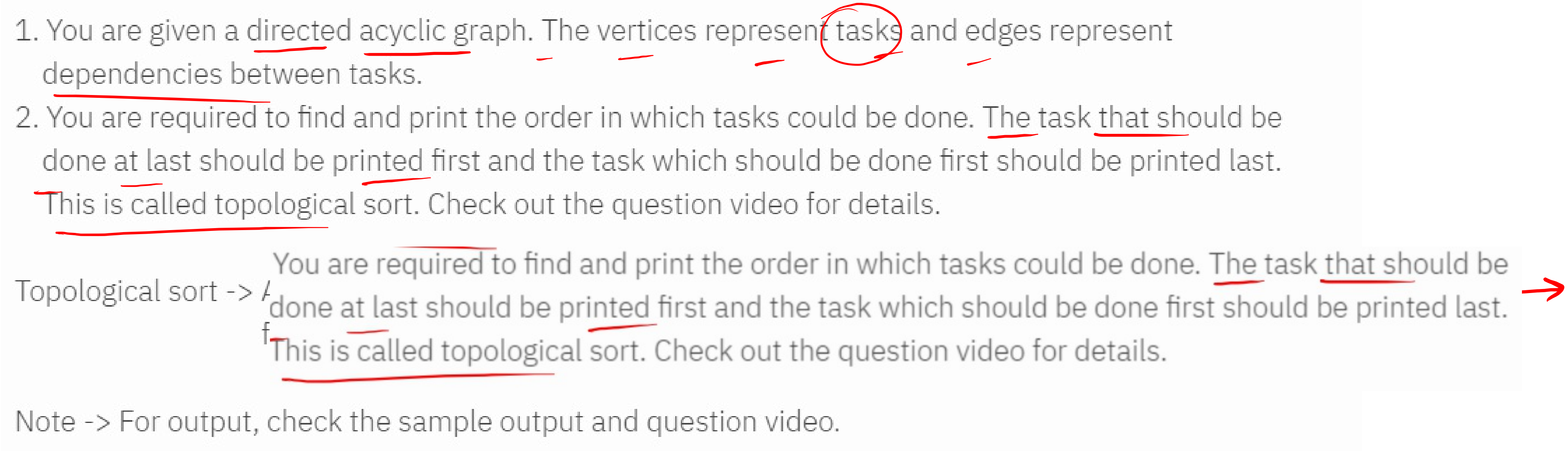


Pair
vtx
psf

BFS
remove

DFS
stack

remove
print
add



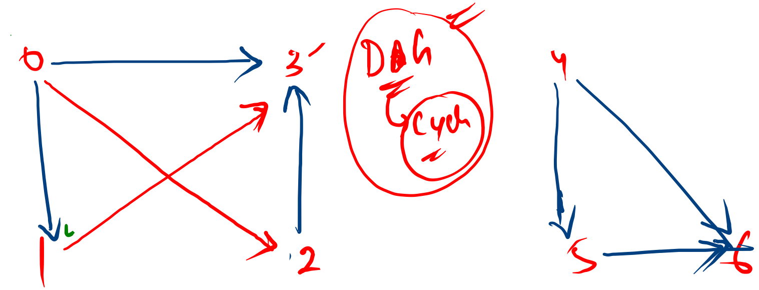
1. You are given a directed acyclic graph. The vertices represent tasks and edges represent dependencies between tasks.
2. You are required to find and print the order in which tasks could be done. The task that should be done at last should be printed first and the task which should be done first should be printed last. This is called topological sort. Check out the question video for details.

Topological sort -> /

You are required to find and print the order in which tasks could be done. The task that should be done at last should be printed first and the task which should be done first should be printed last. This is called topological sort. Check out the question video for details.

f

Note -> For output, check the sample output and question video.

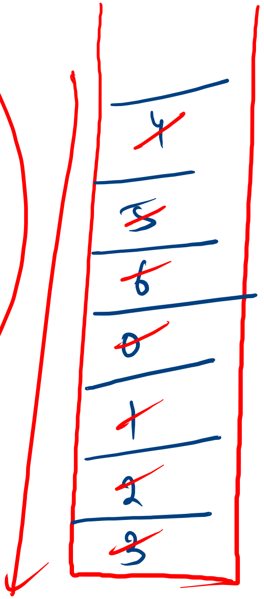


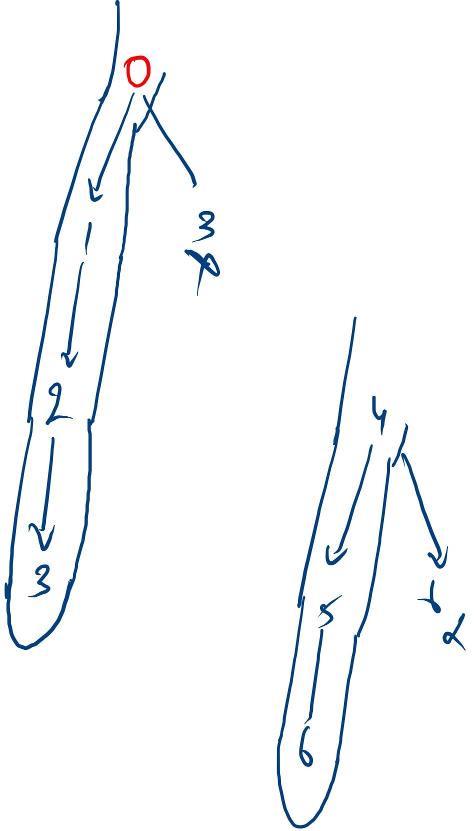
0 2 1 3

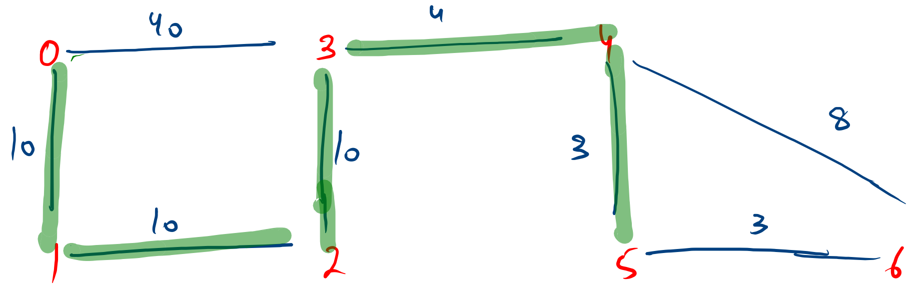
0 1 2 3

0	1	2	3	4	5	6
~~f~~	~~f~~	~~f~~	~~f~~	~~f~~	~~f~~	~~f~~
T	T	T	T	T	T	T

4 5 6 0 1 2 3







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

1-0@10

2-1@10

3-2@10

4-3@4

5-4@3

vtx - pvtx - wt

~~0-(-1)-0~~

~~1-0-10~~

3-0-40

~~2-1-10~~

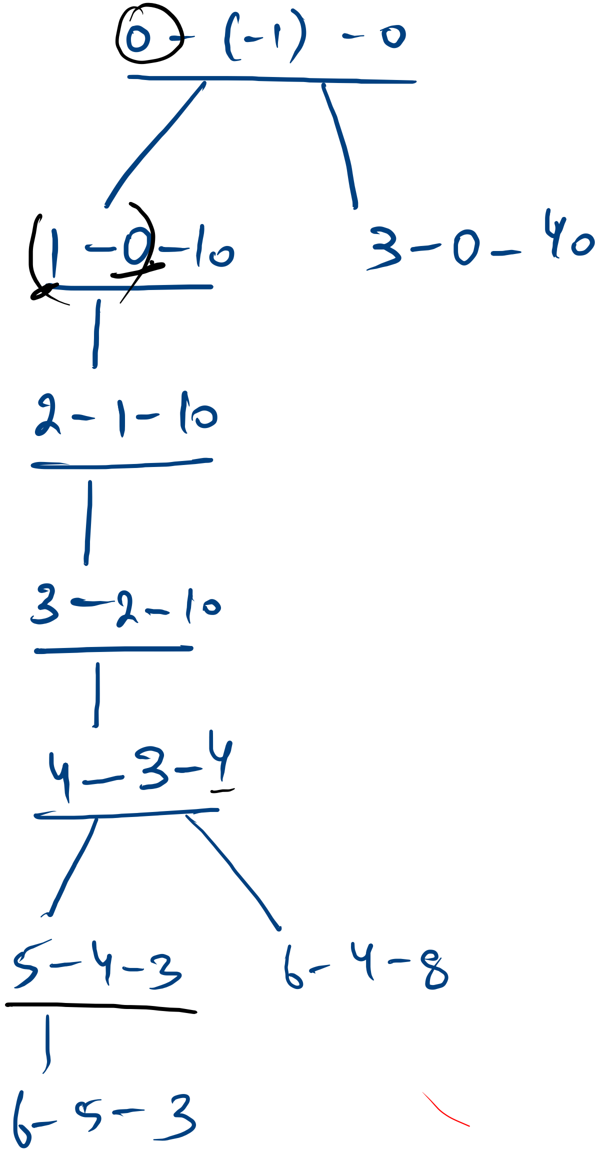
~~3-2-10~~

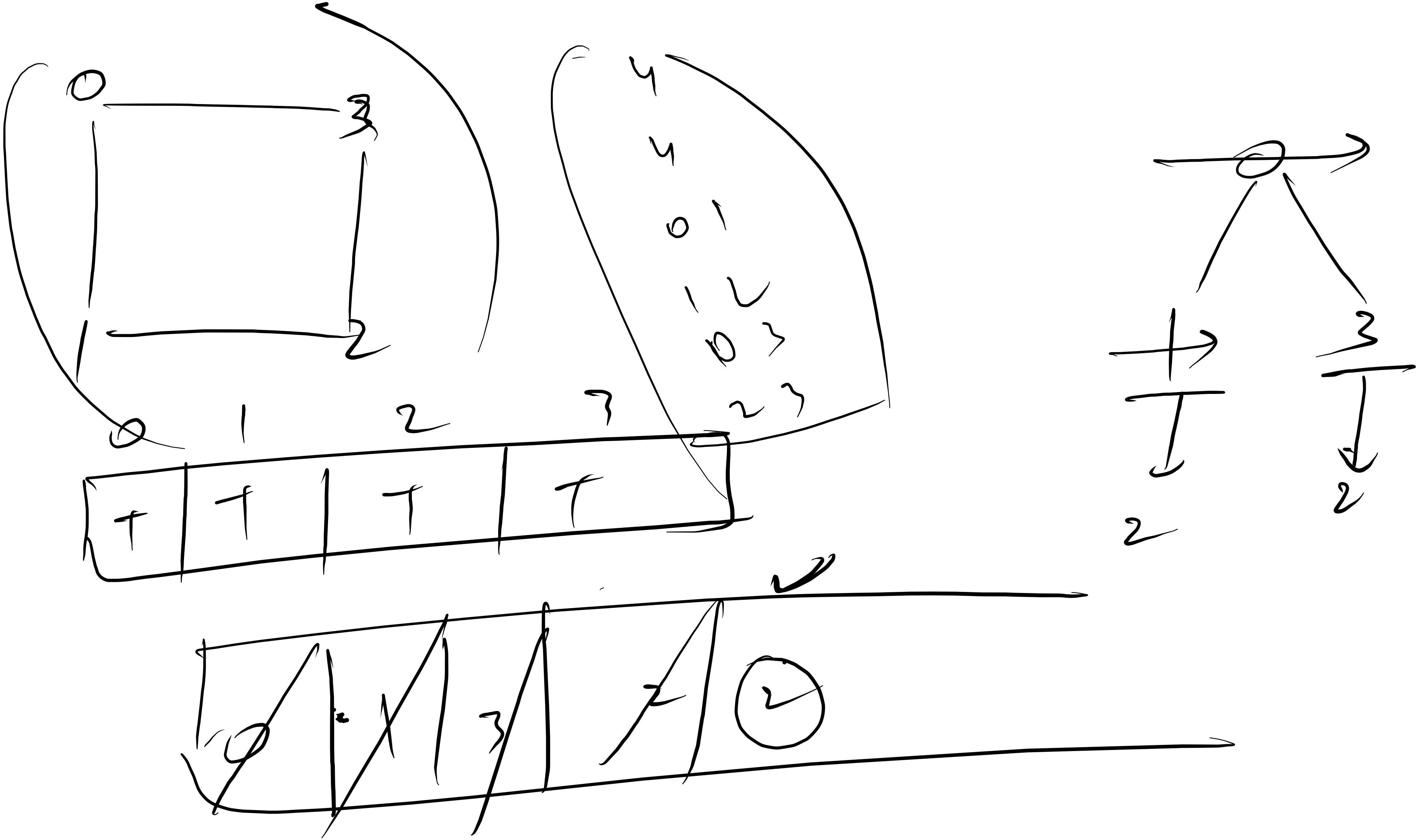
~~4-3-4~~

~~5-4-3~~

6-4-8

6-5-3

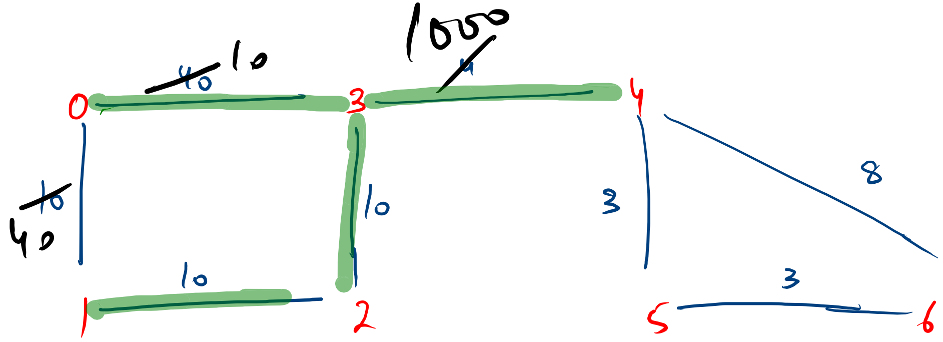




4 4
0 1
1 2
0 3
2 3

0	1	2	3
T	T	T	T

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



Priority Queue → min

Same

Using Heap

~~0-(-1)-0~~ ~~1-2-10~~

~~1-0-40~~ 5-4-3

~~3-0-10~~ 6-4-8

~~2-3-10~~

~~4-3-1000~~

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