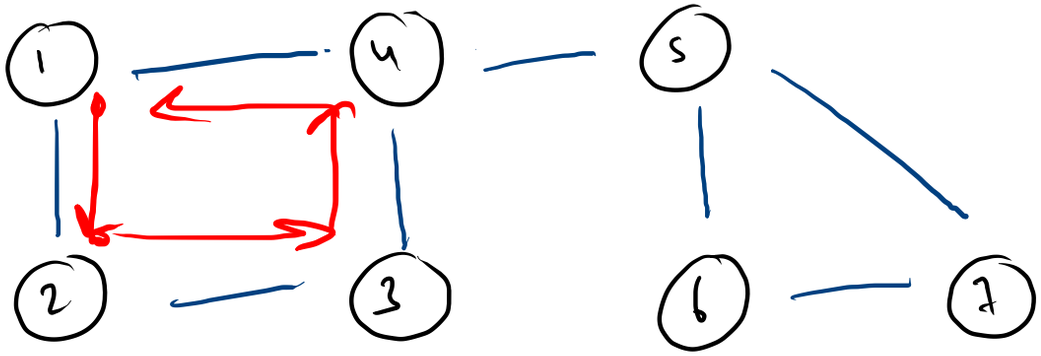
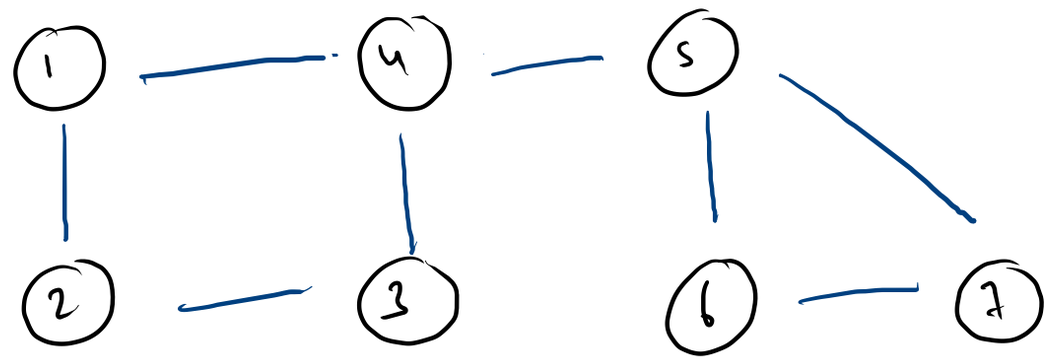
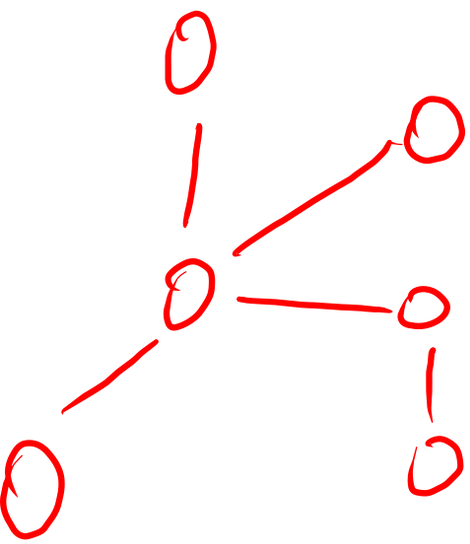
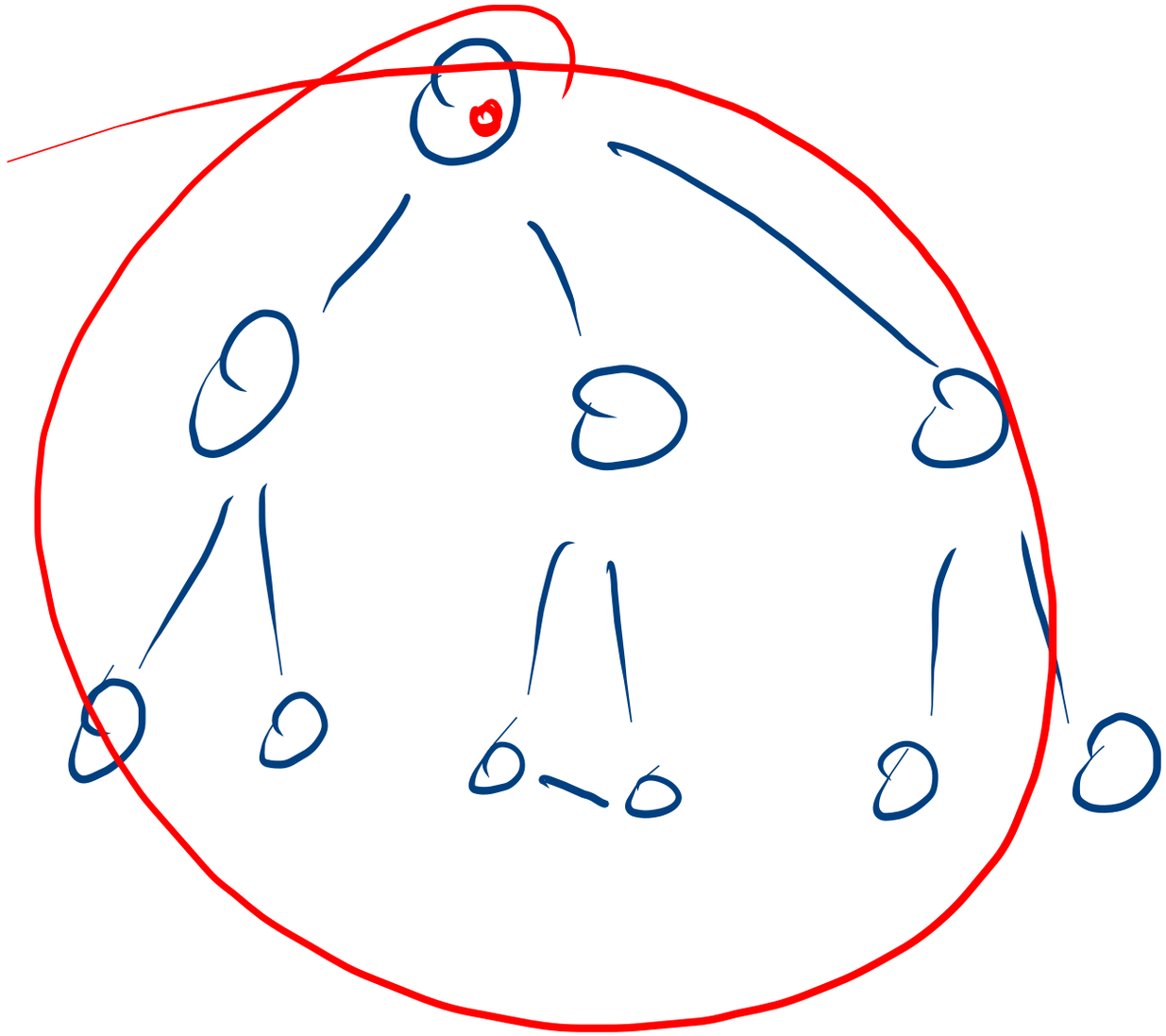
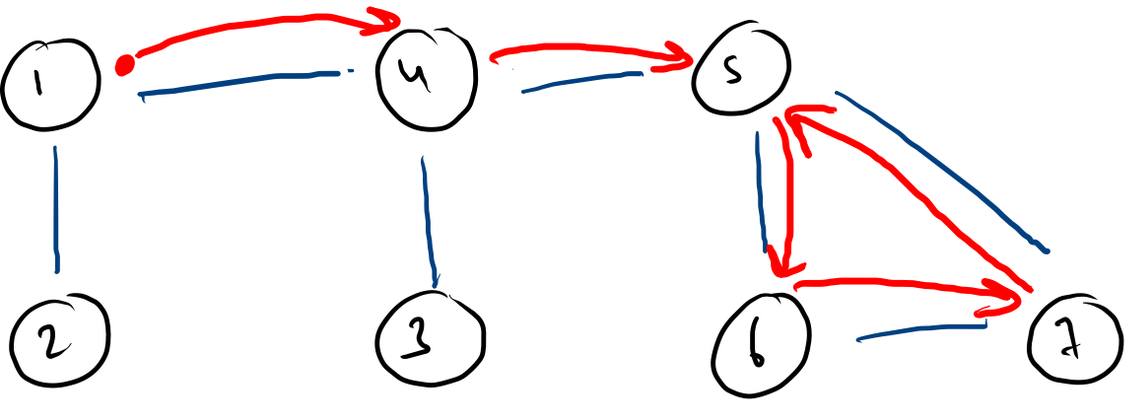
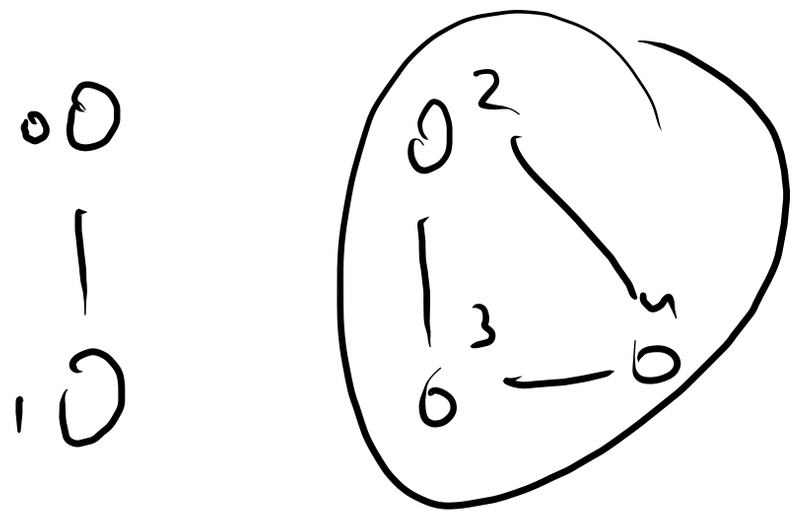

1
4
5
2
3
6
7

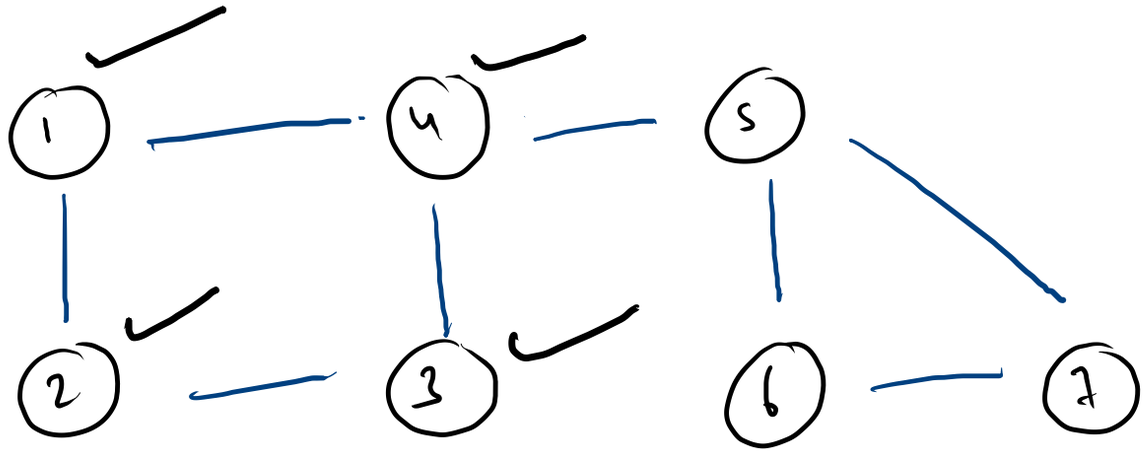

1
4
5
2
3
6
7



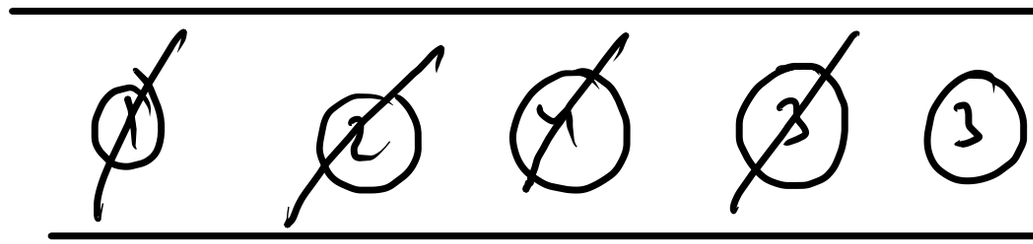


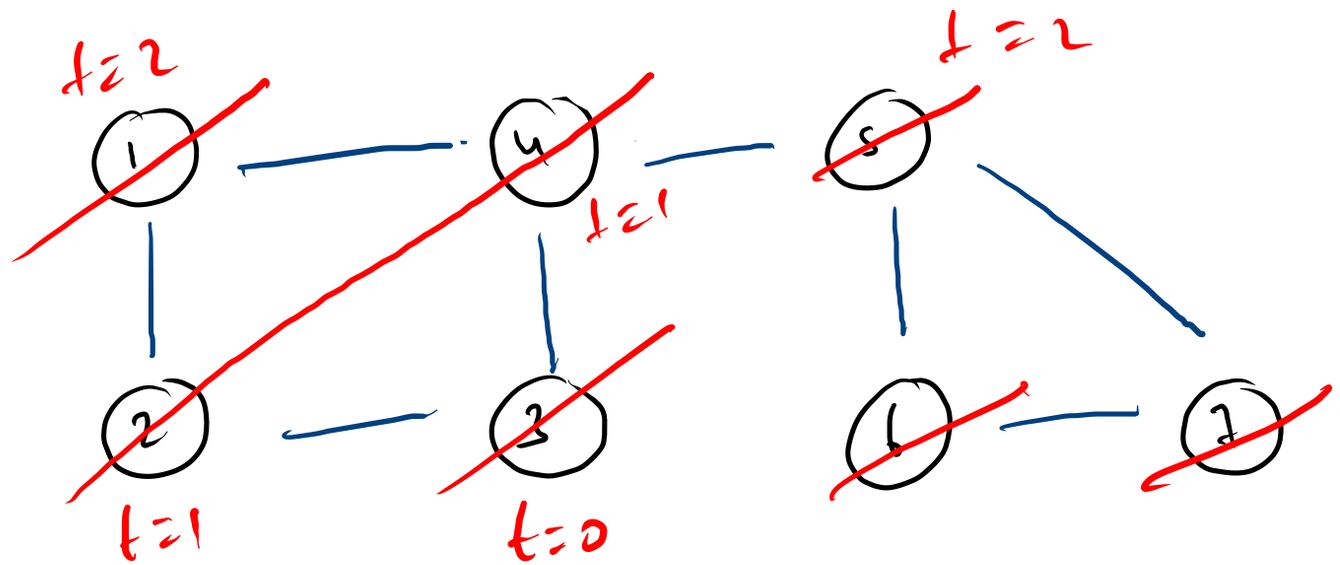

1
4
5
2
3
6
7


0
1
0
0
2
1
3
6
5
4



g mark child



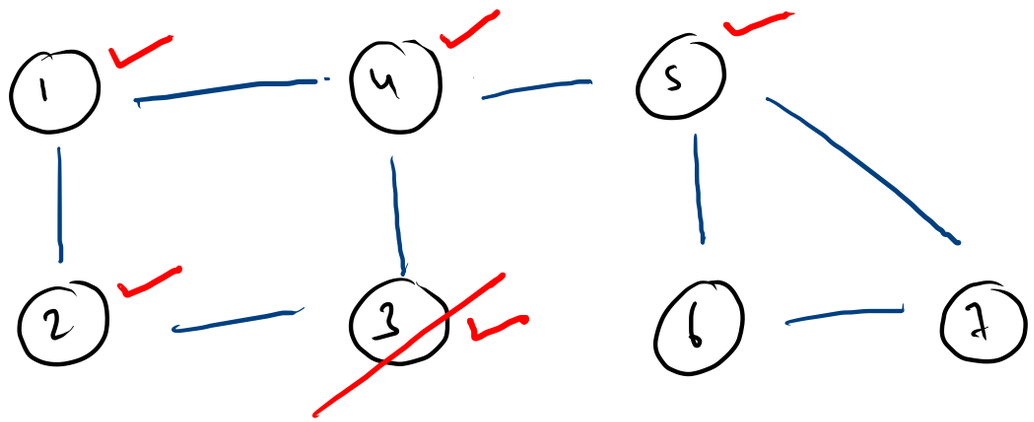



```
for(→    )
```

t=20

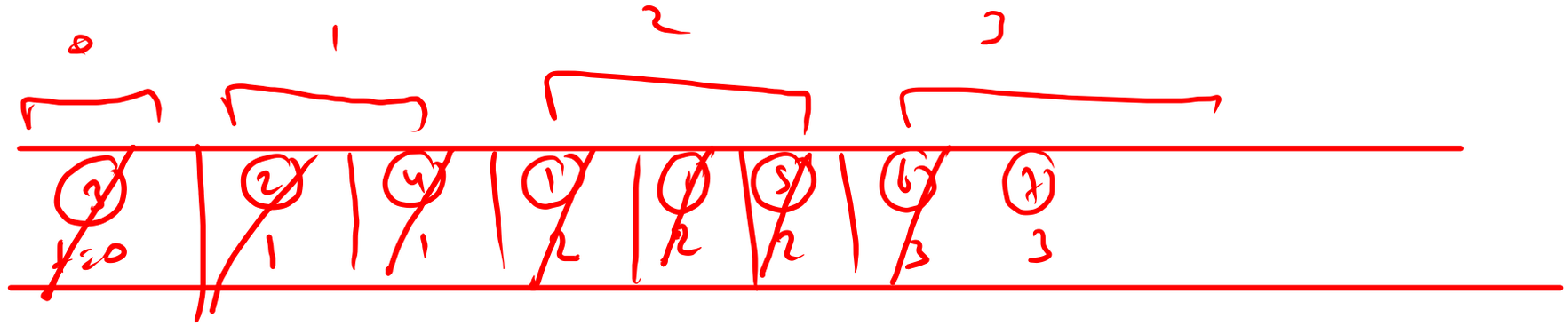
	count
t=2	5
t=3	7

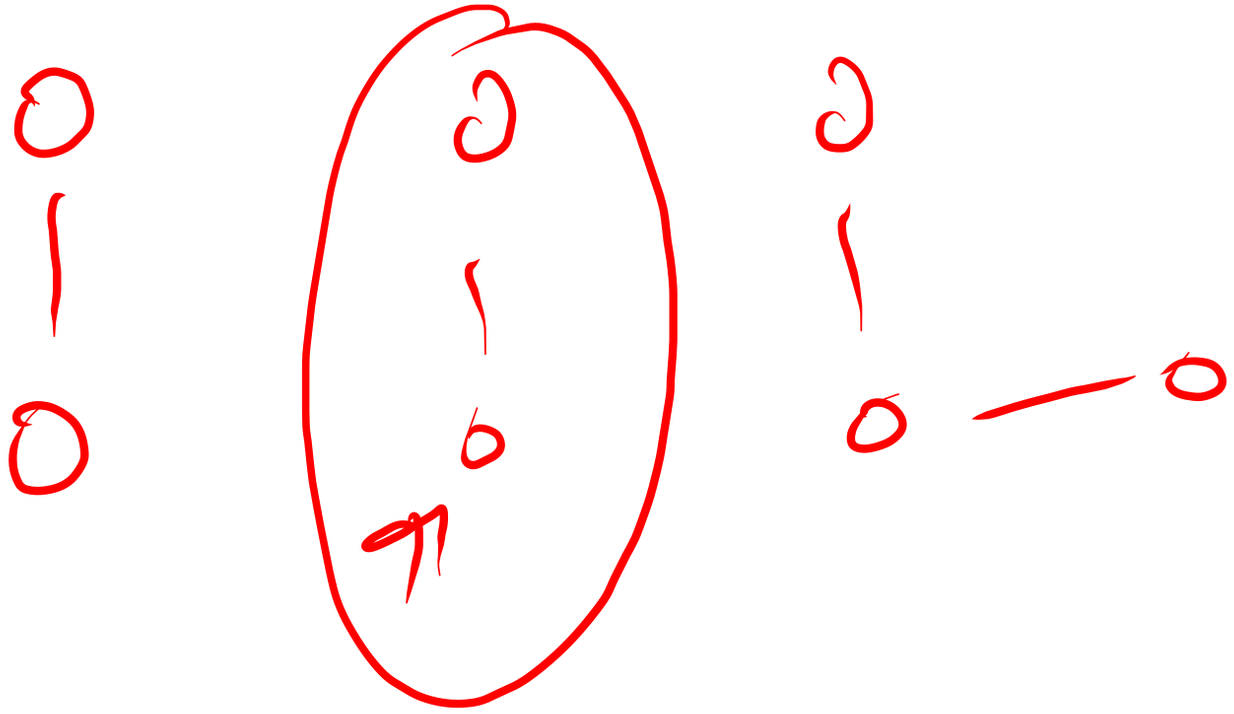
γ mark work child



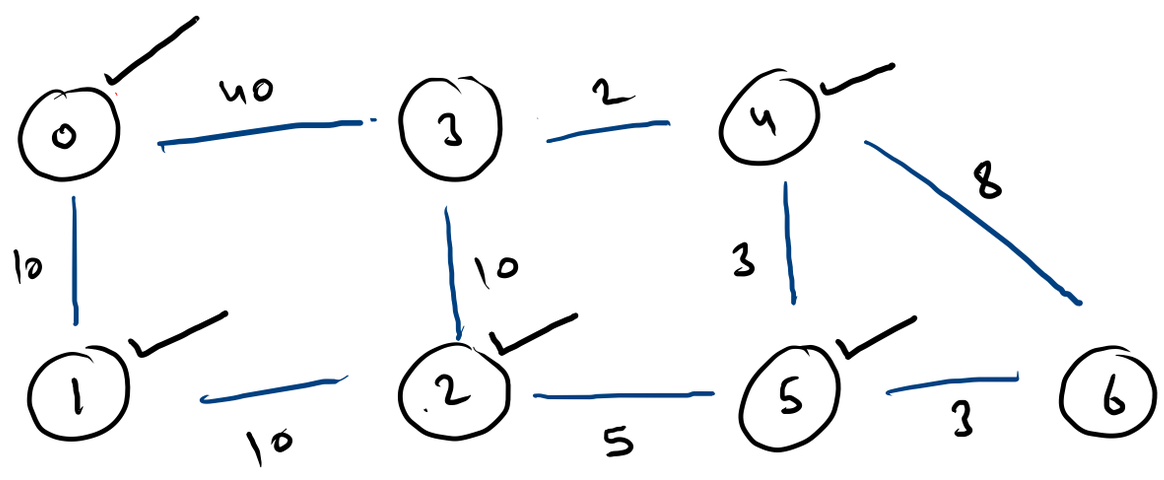
Conn = 1 2 3 4 5

t=2





src=0



y mane wore child

0 @ 0 0

1 @ 10 01

2 @ 20 012

3 @ 30 0123

5 via 0125 @ 25

4 via 01254 @ 28

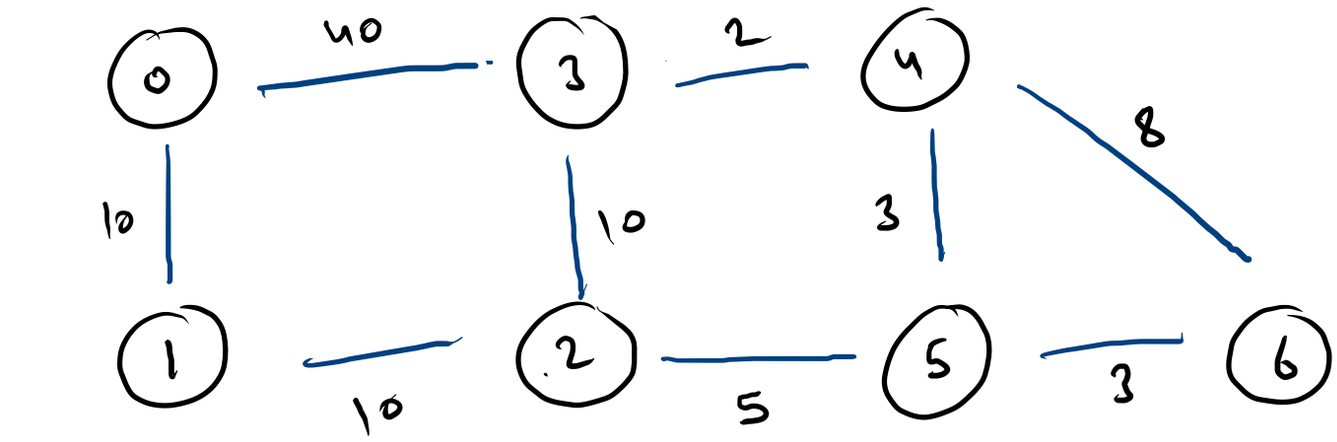
~~4/28/"01254"~~ ~~5/25/"0125"~~

3/40/"03" 6/31/"012546"

6/28/"01256"

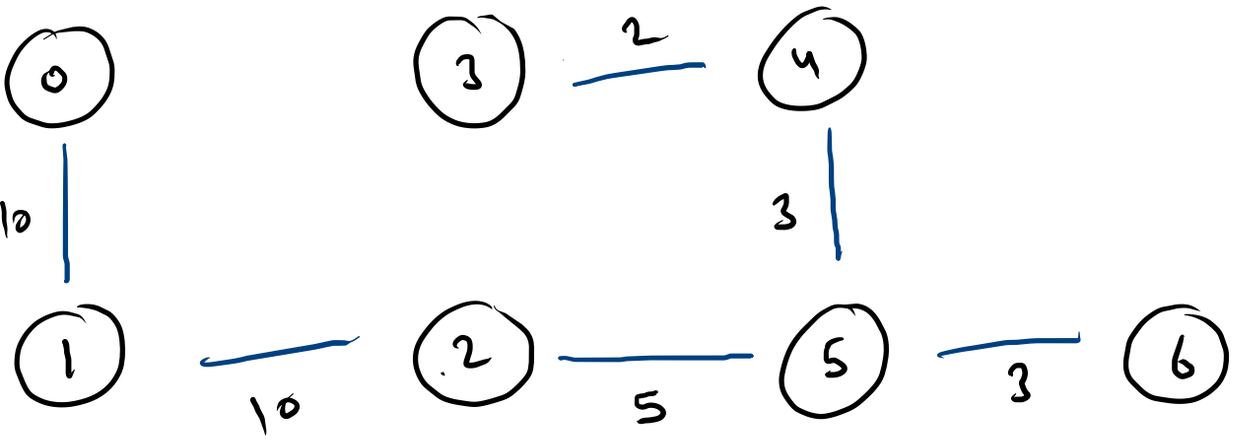
3/30/"01543"

3/30/"0123"

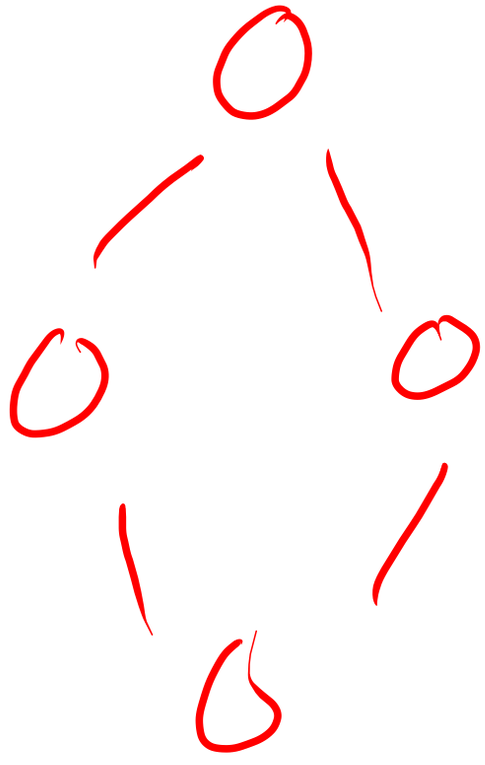

0
40
3
2
4
8
10
10
3
1
10
2
5
5
3
6

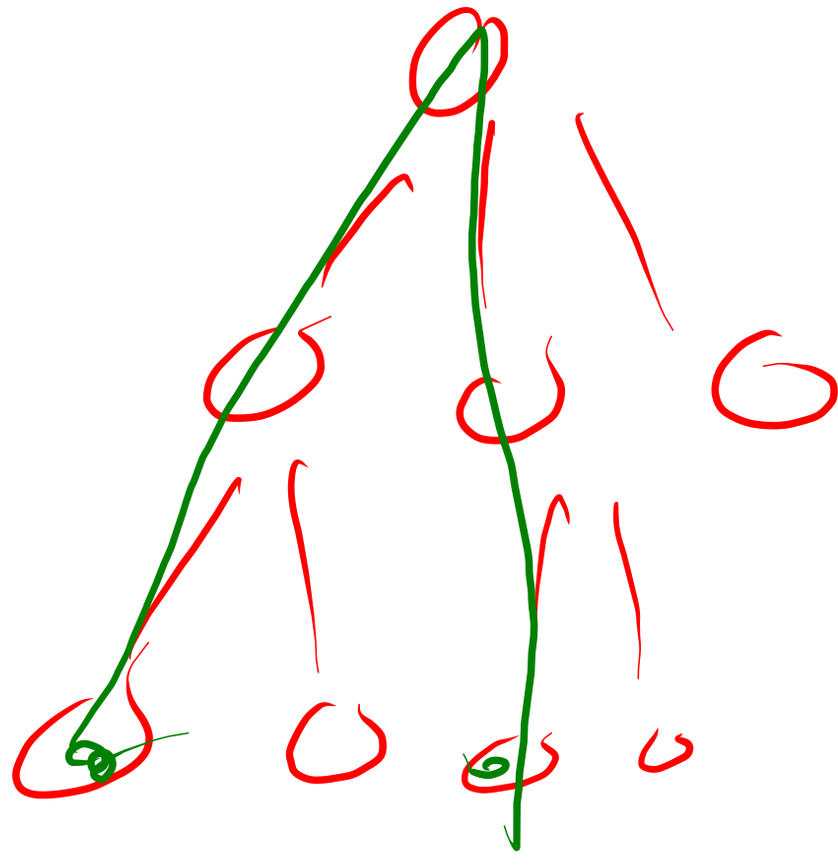

MST

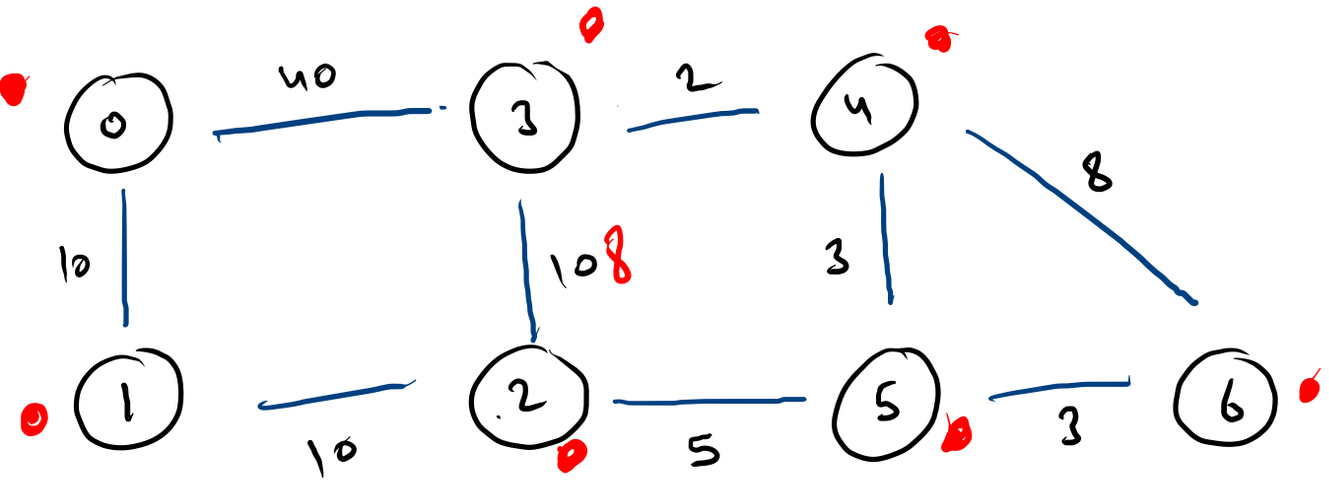

ST
No Cycle

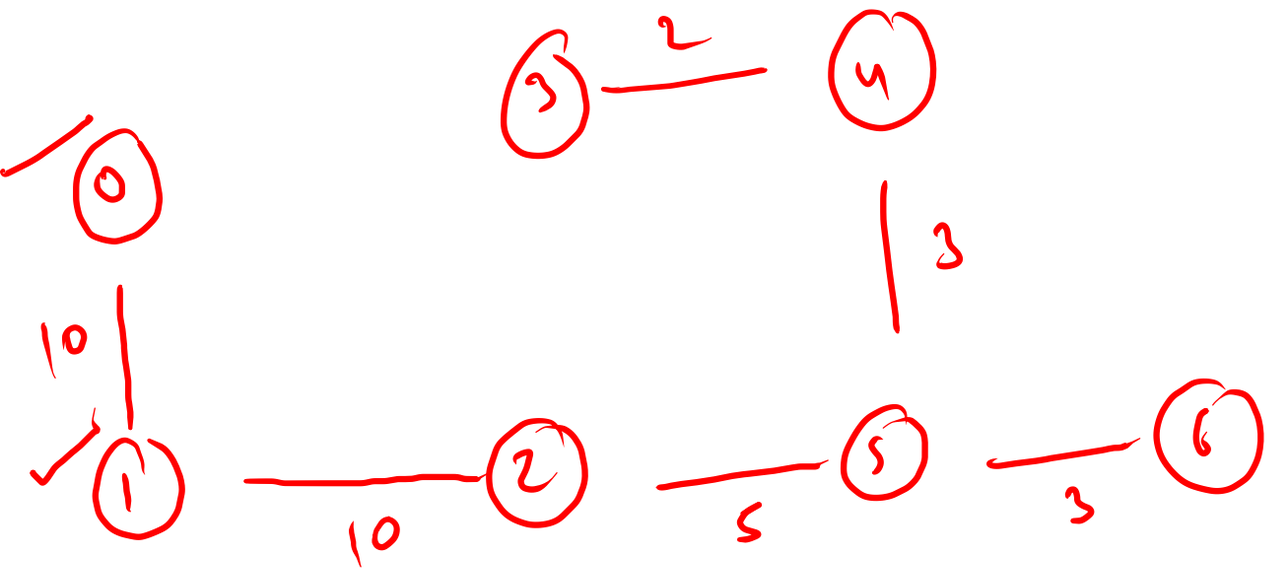

0
3
2
4
10
3
1
10
2
5
5
3
6

Sp









0-1/10

5-4/3

0-3/40

5-6/3

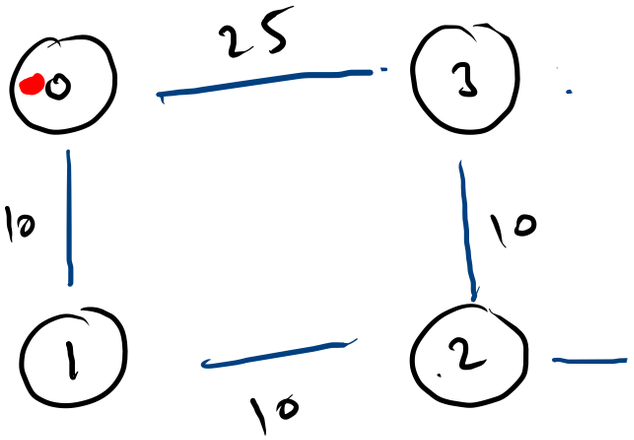
1-2/10

4-3/2

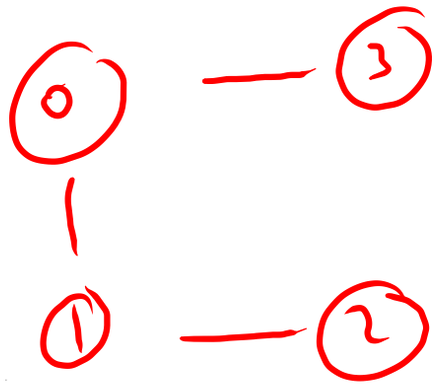
2-3/10

4-6/8

2-5/5

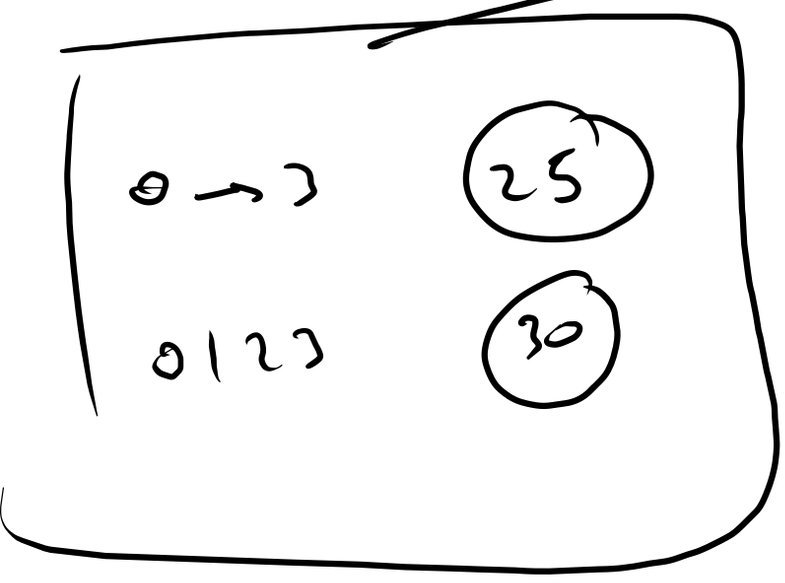


Dikes

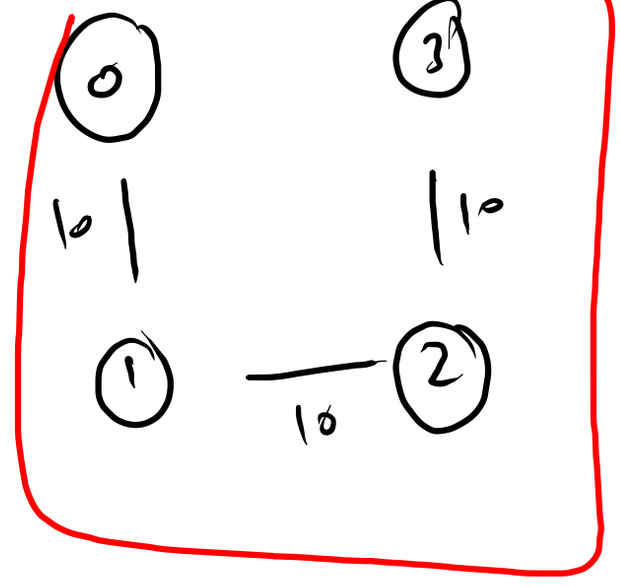


path
Delhi min
can to
all destin

path Dikes



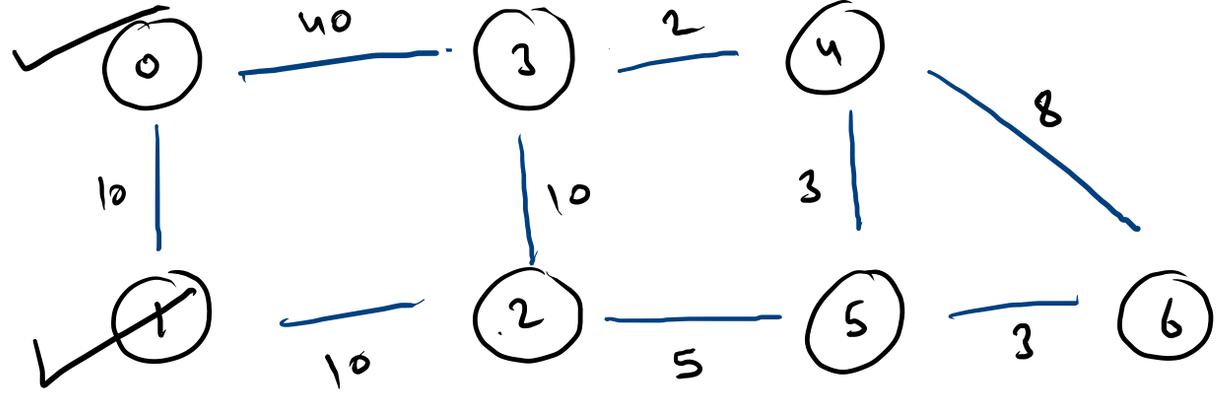
Prim



connectivity

Ex railways all over India

```java
PriorityQueue<Pair> pq = new PriorityQueue<>();
pq.add(new Pair(0, 0, 0));

boolean visited[] = new boolean[vtces];
while(pq.size() > 0){

    Pair p = pq.remove();

    if(visited[p.nbr]){
        continue;
    }

    visited[p.nbr] = true;

    if(p.src != p.nbr)
    System.out.println("["+p.nbr+"-"+p.src+"@"+p.wt+"]");

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

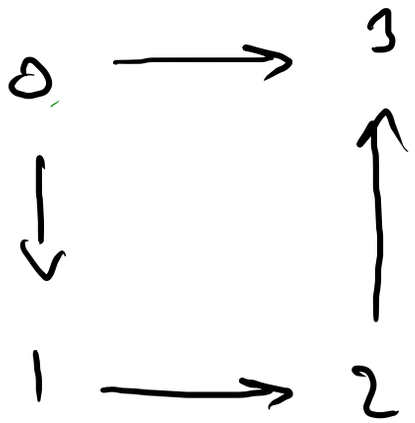


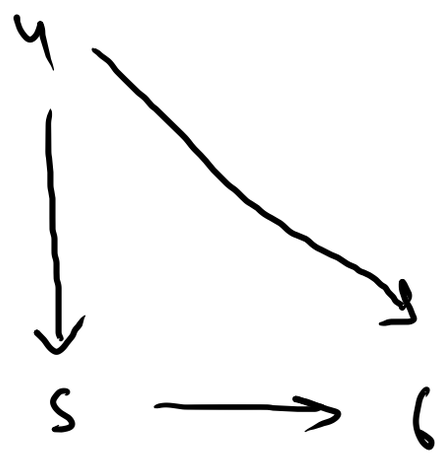
1 - 0 @ 10

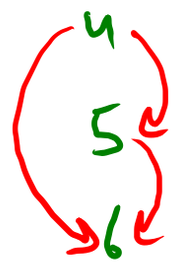
src nbr w

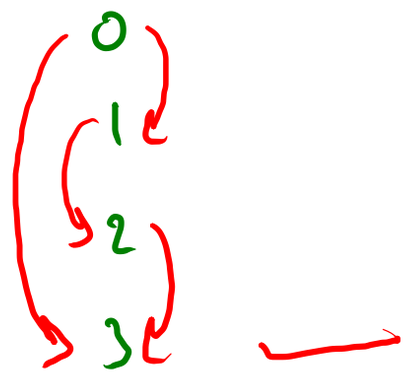
0 | 0 | 0

```java
static class Pair implements Comparable<Pair> {
    int src;
    int nbr;
    int wt;

    Pair(int v, int u, int w){
        this.src = v;
        nbr = u;
        wt = w;
    }

    public int compareTo(Pair b){
        return this.wt - b.wt;
    }

}
```







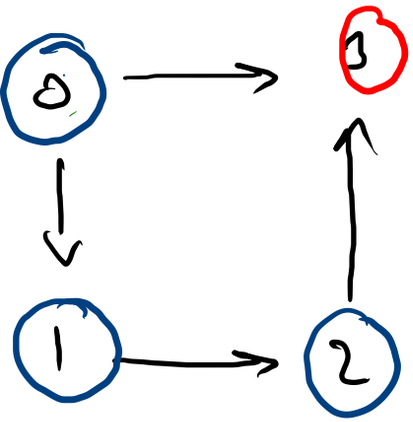


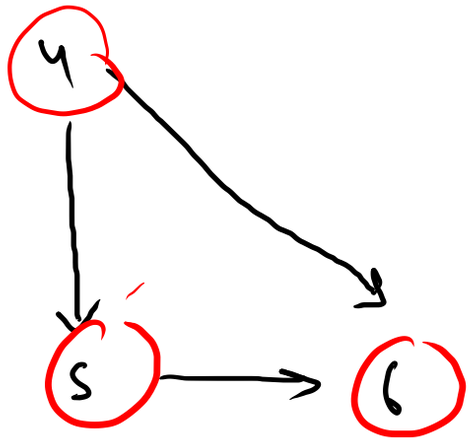
4
5
6
0
1
2
3

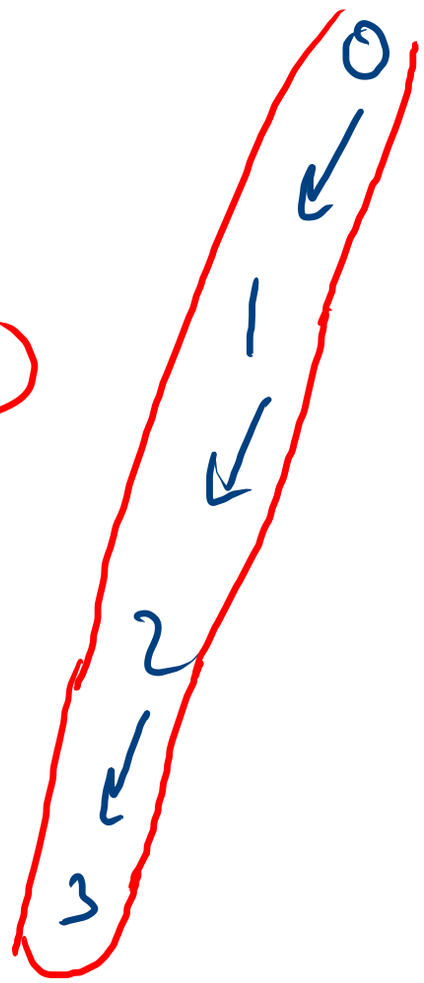
7

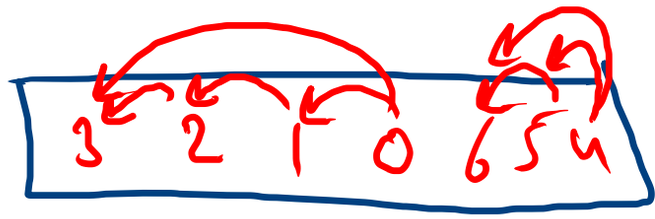
0 1 1 2 2 3 0 3 4

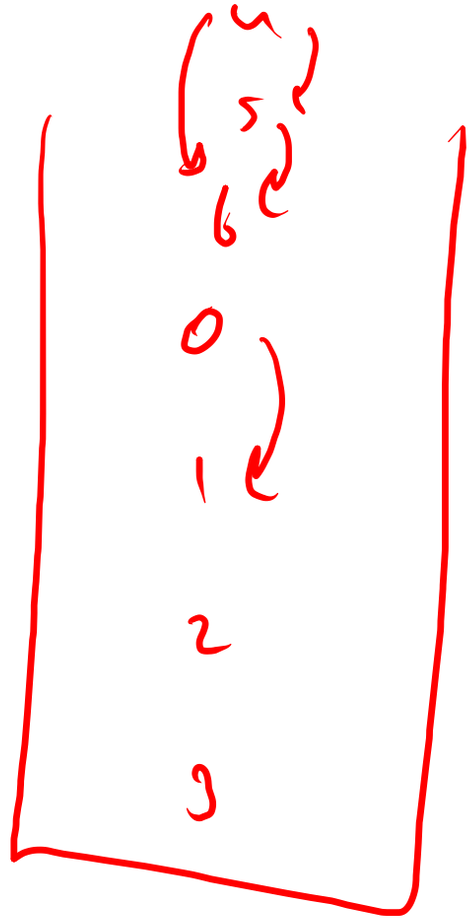
5 5 6 4 6


0
3
1
2

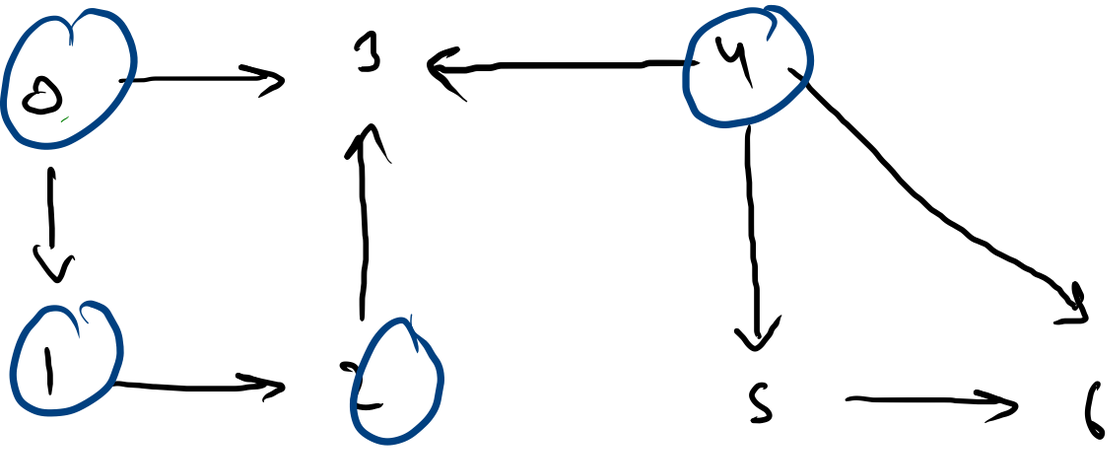

4
5
6

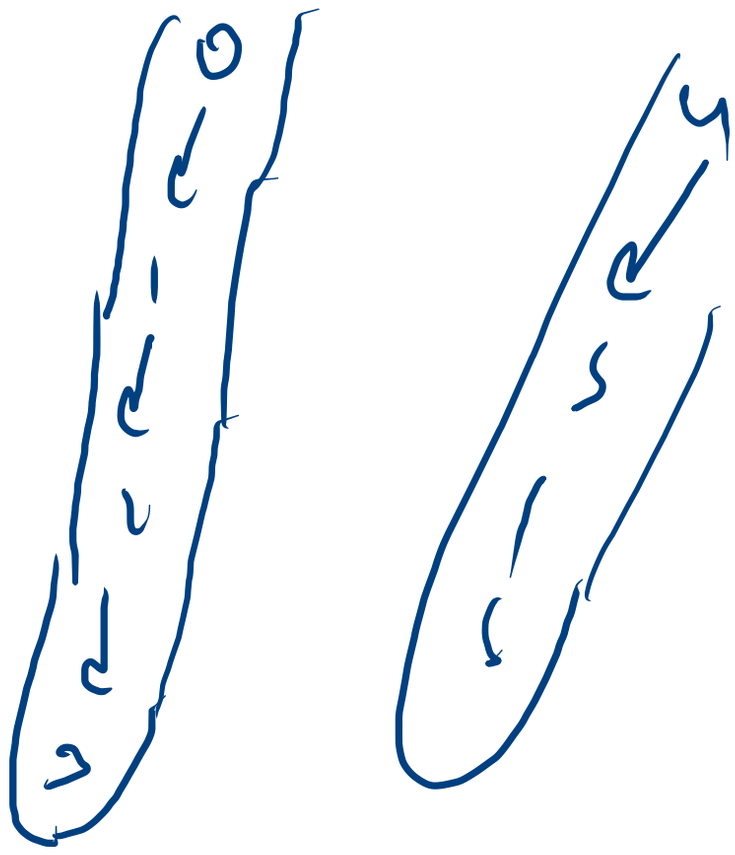

0
1
2
3

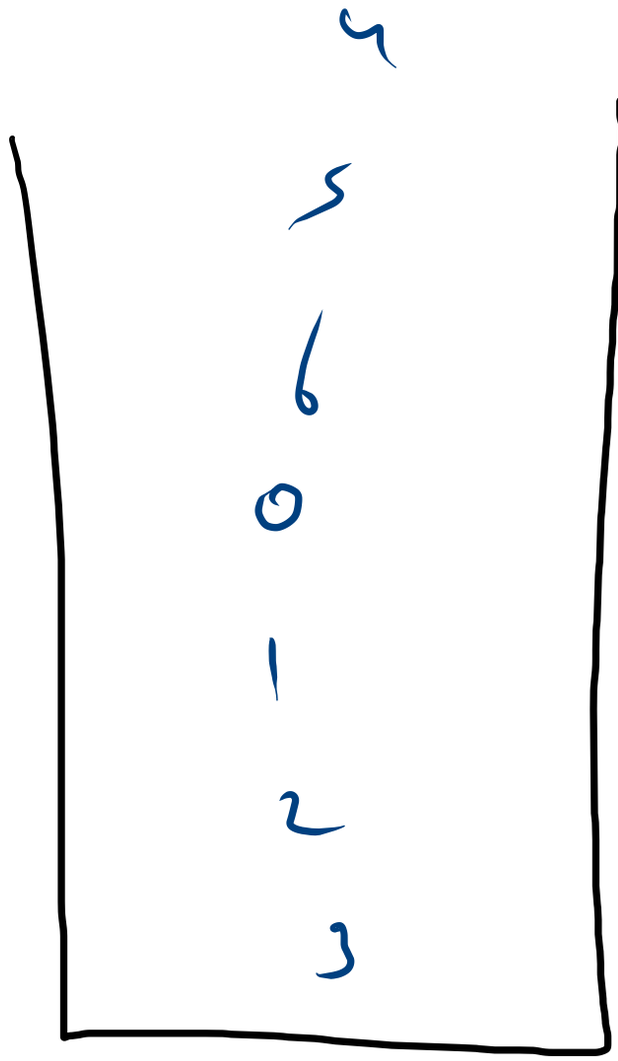

3
2
1
0
6
5
4

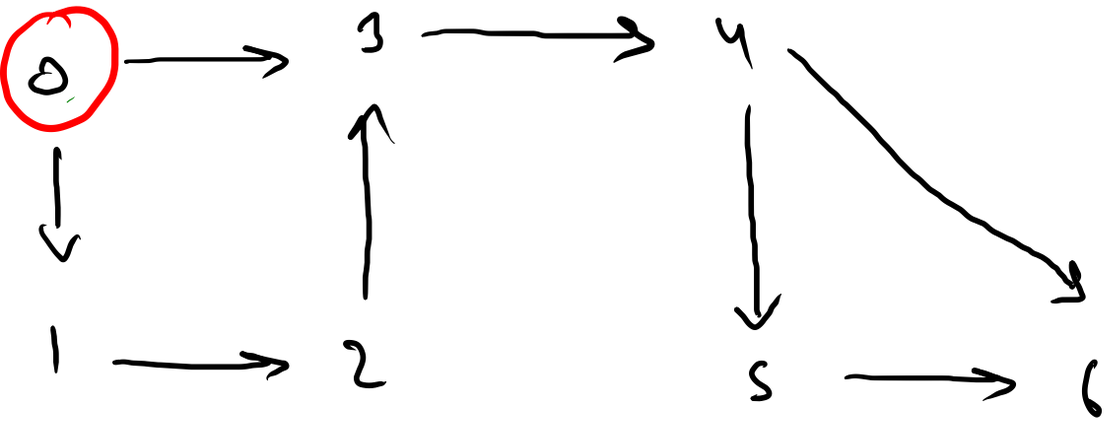

4
5
6
0
1
2
3

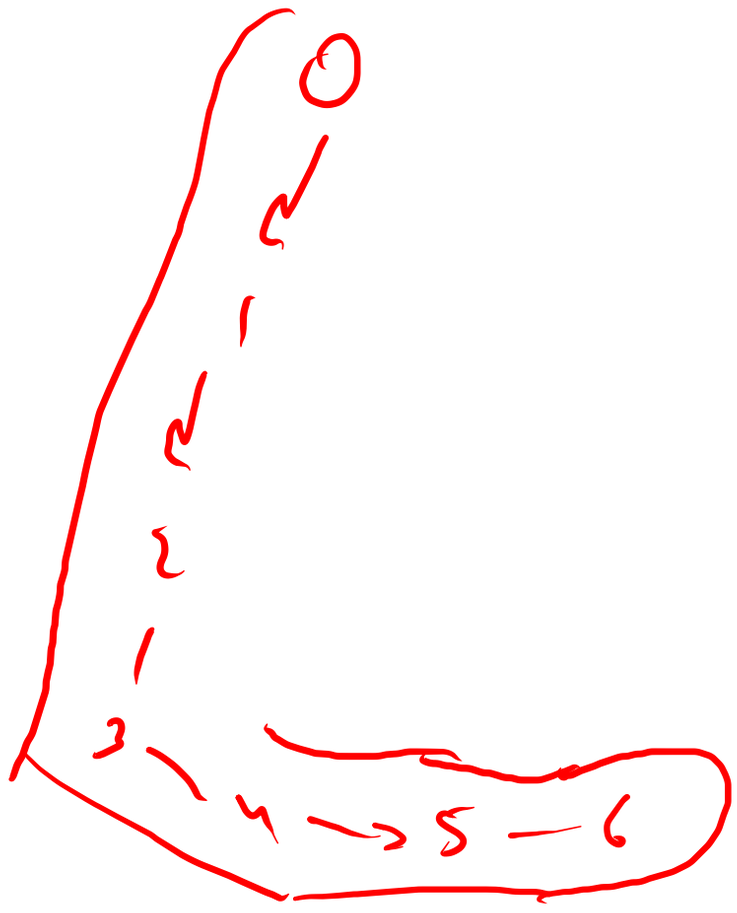

5
6

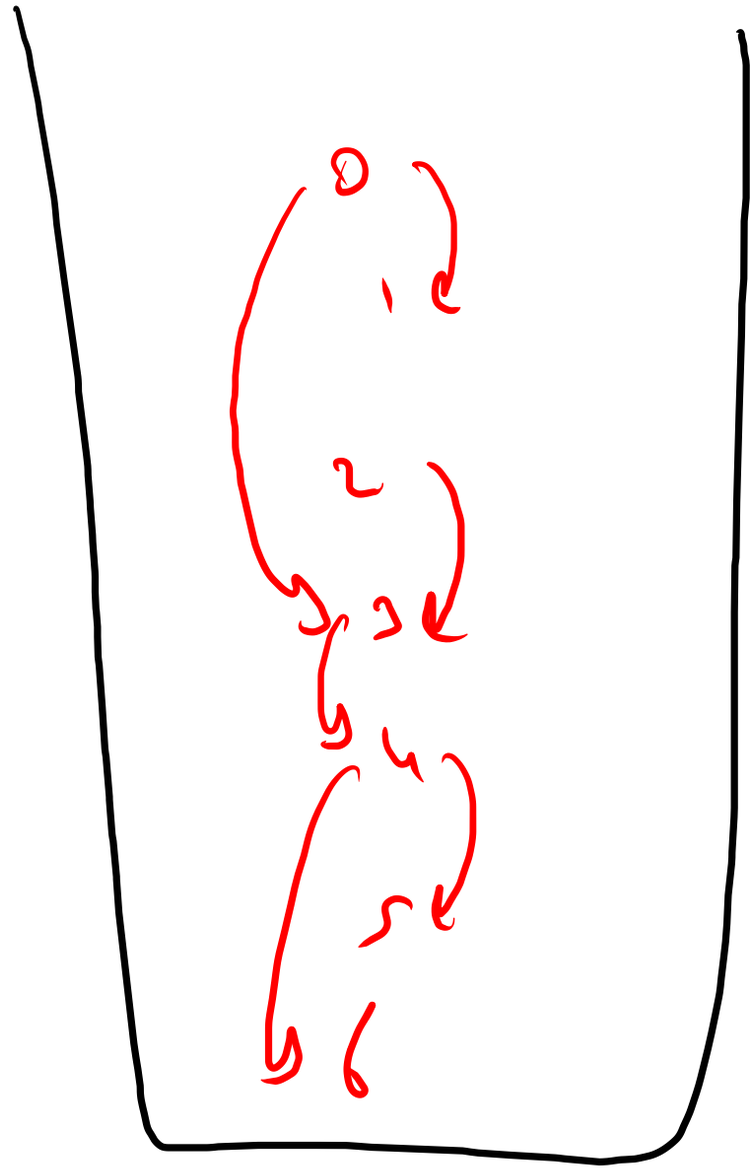

0
1
2
3
4
5
6

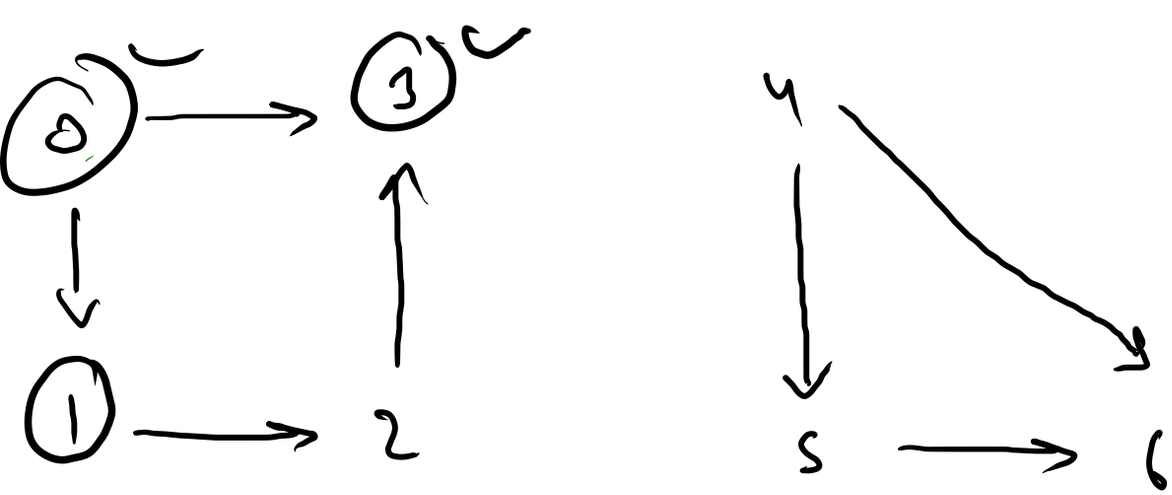


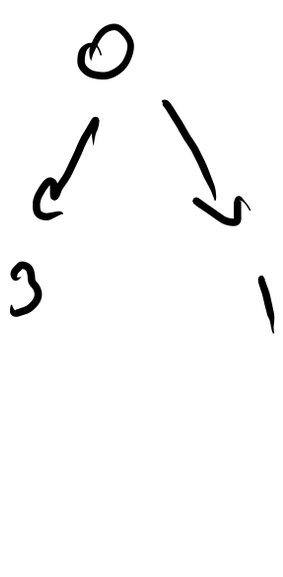


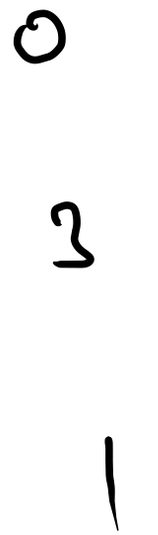

0
3
4
1
2
5
6

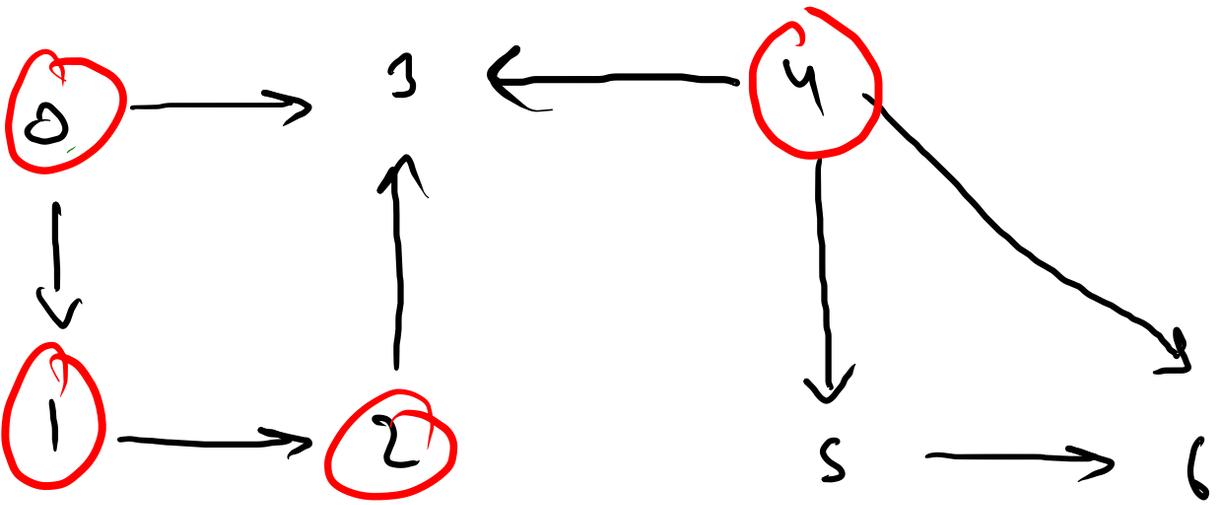

0
1
2
3
4
5
6

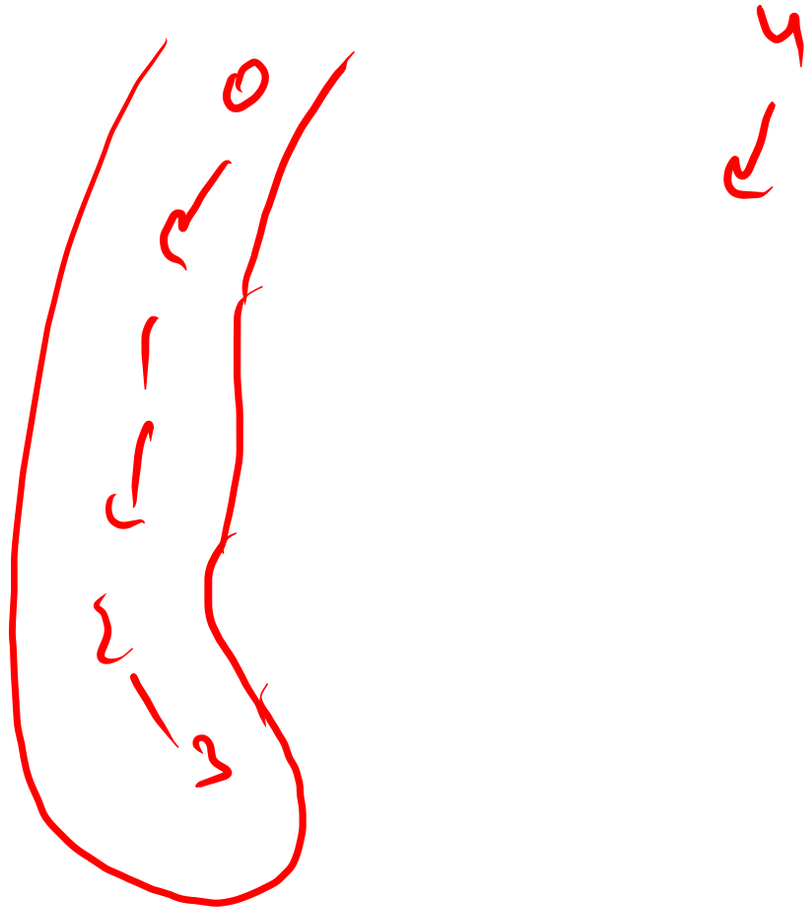

0
1
2
3
4
5
6

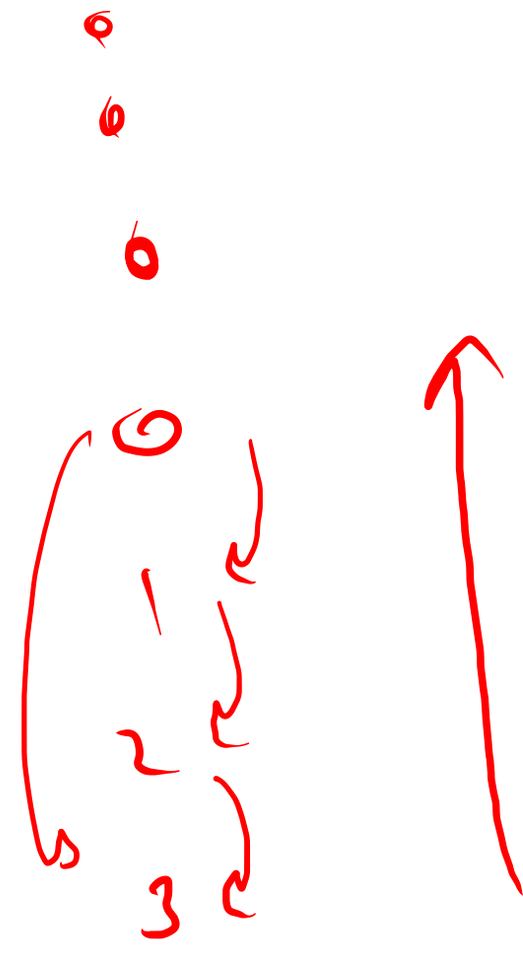

0
3
1
2
4
5
6

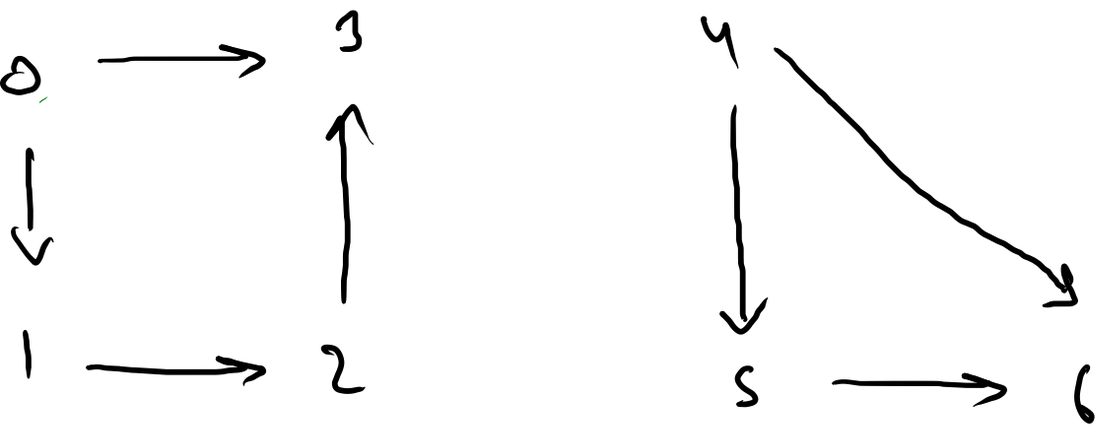

0
3
1
2

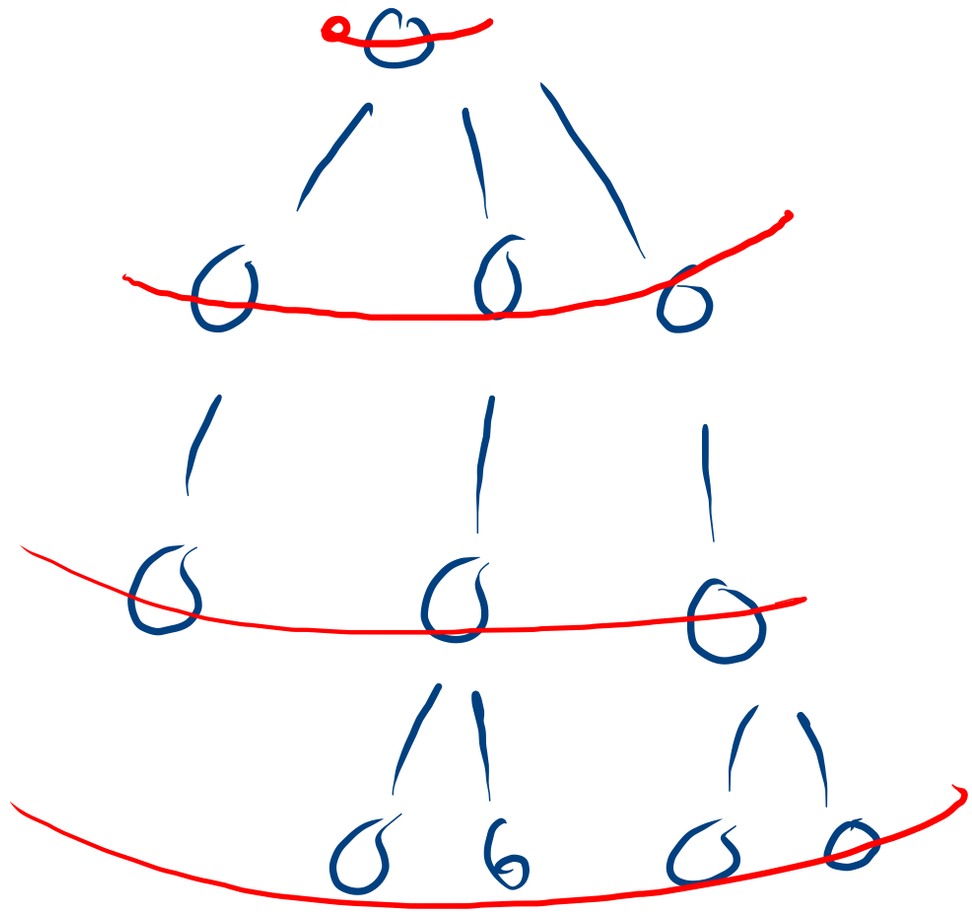

0
3
1
2


0
1
4
1
2
5
6

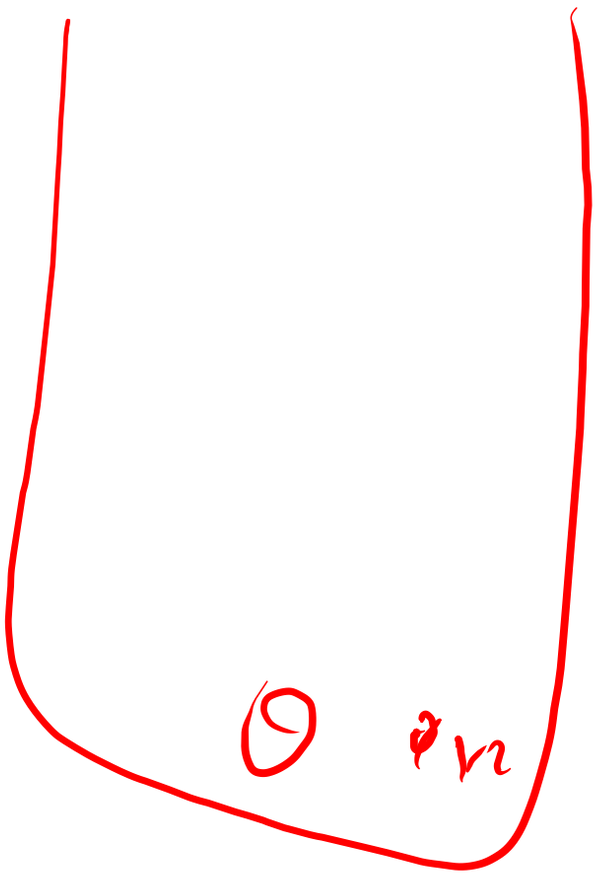

0
4
1
2
3

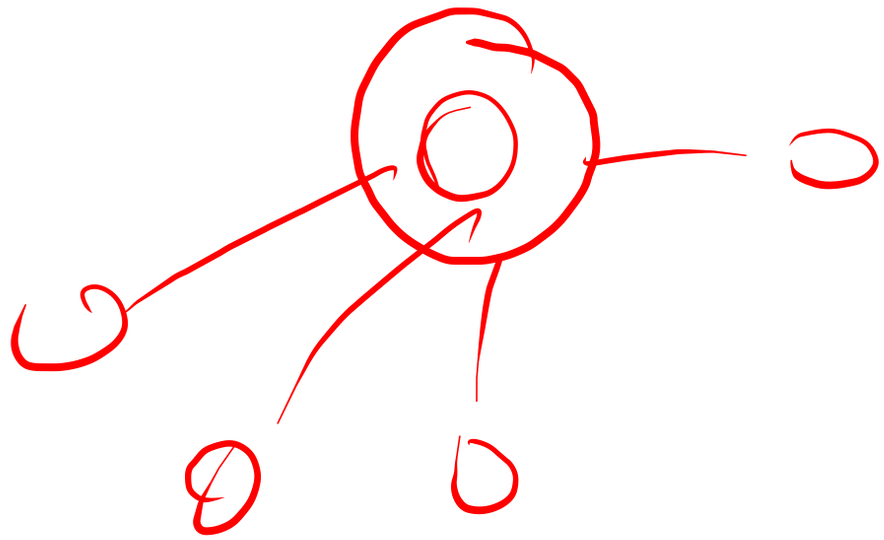

0
1
2
3

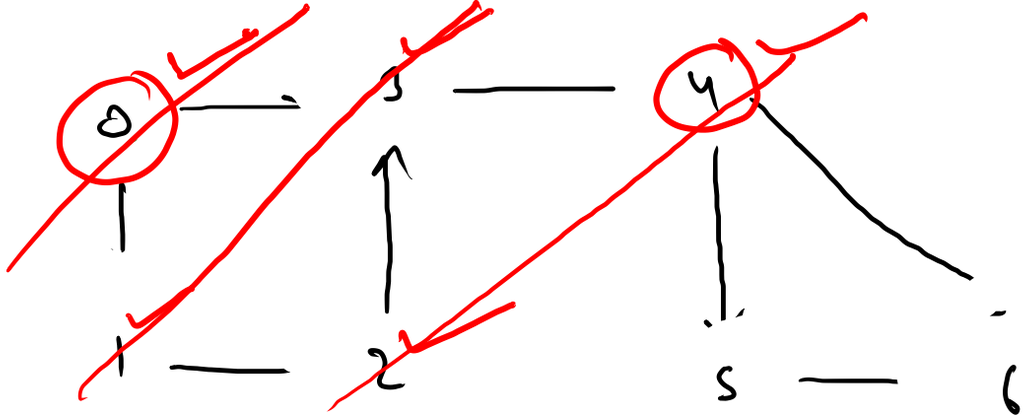

0
3
1
2
4
5
6



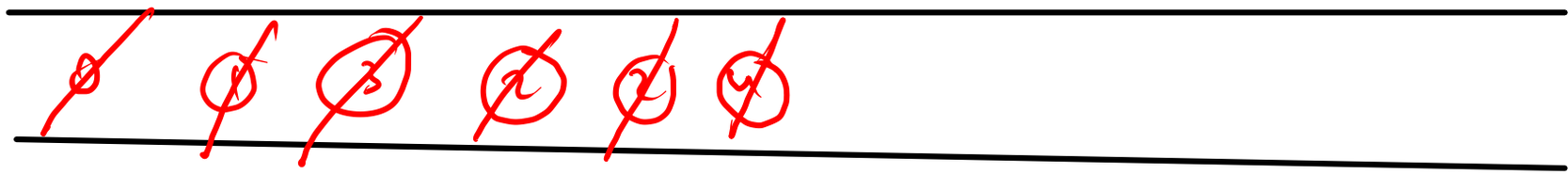
recursive stack







y mark wale chil



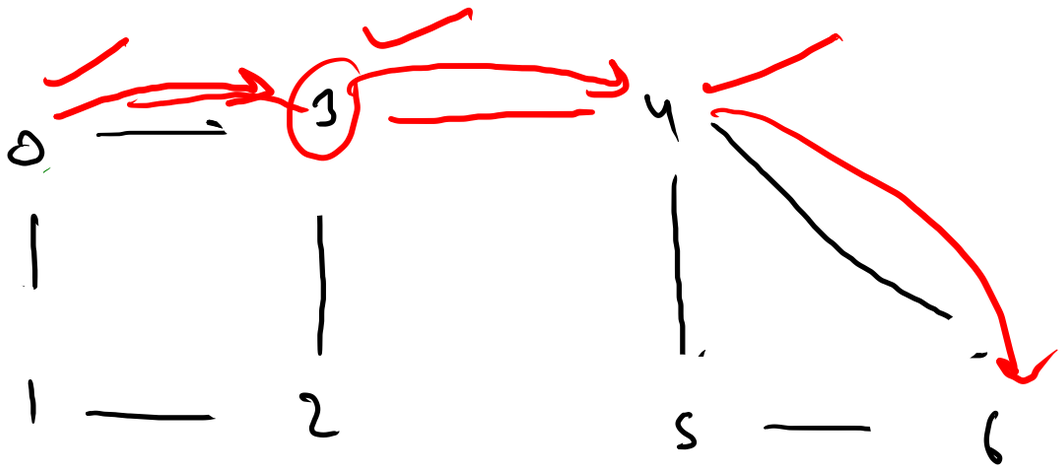
0

1

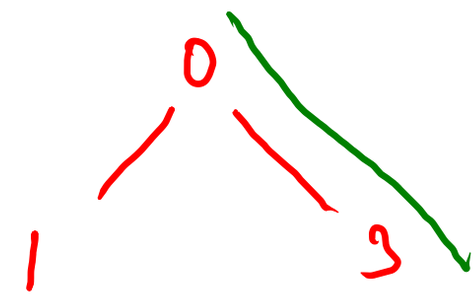
3

2

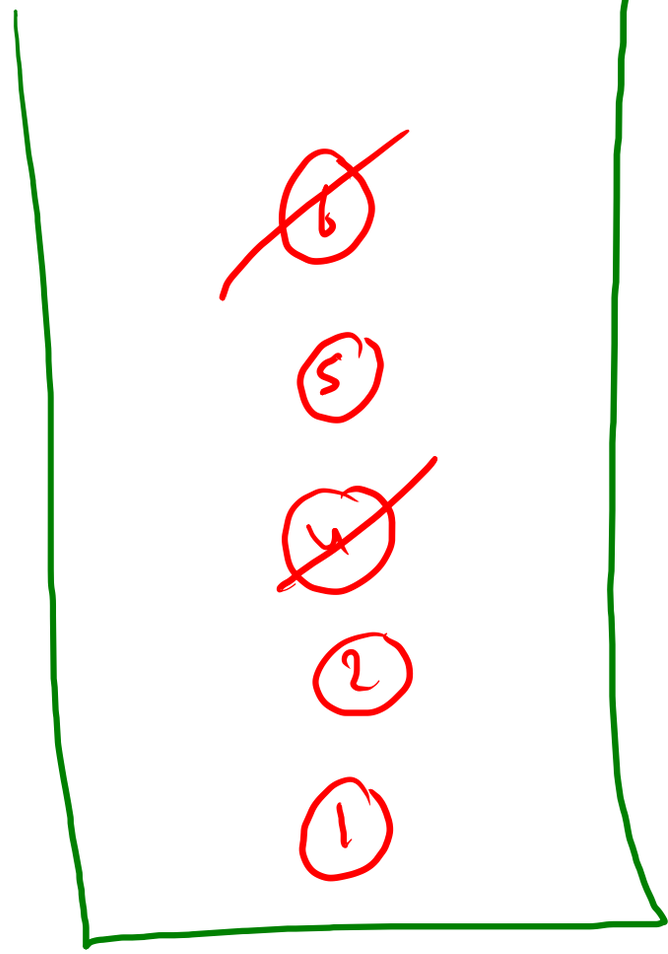
4

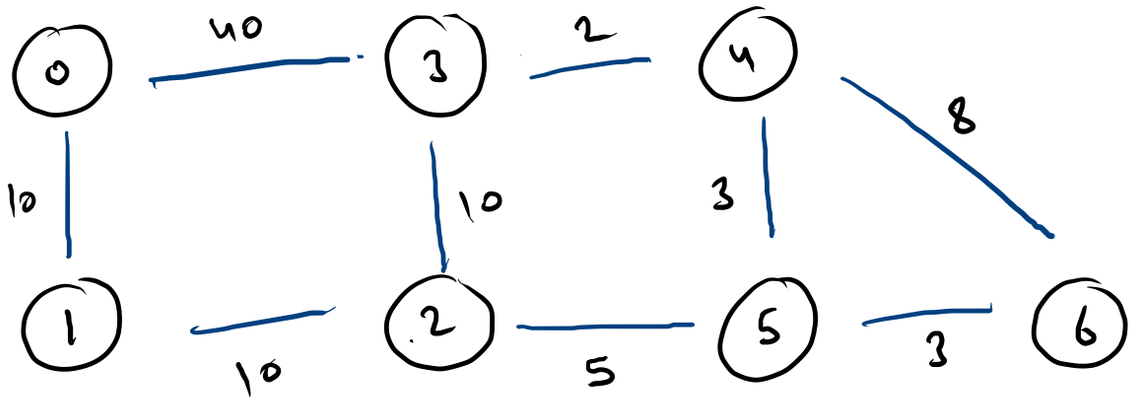


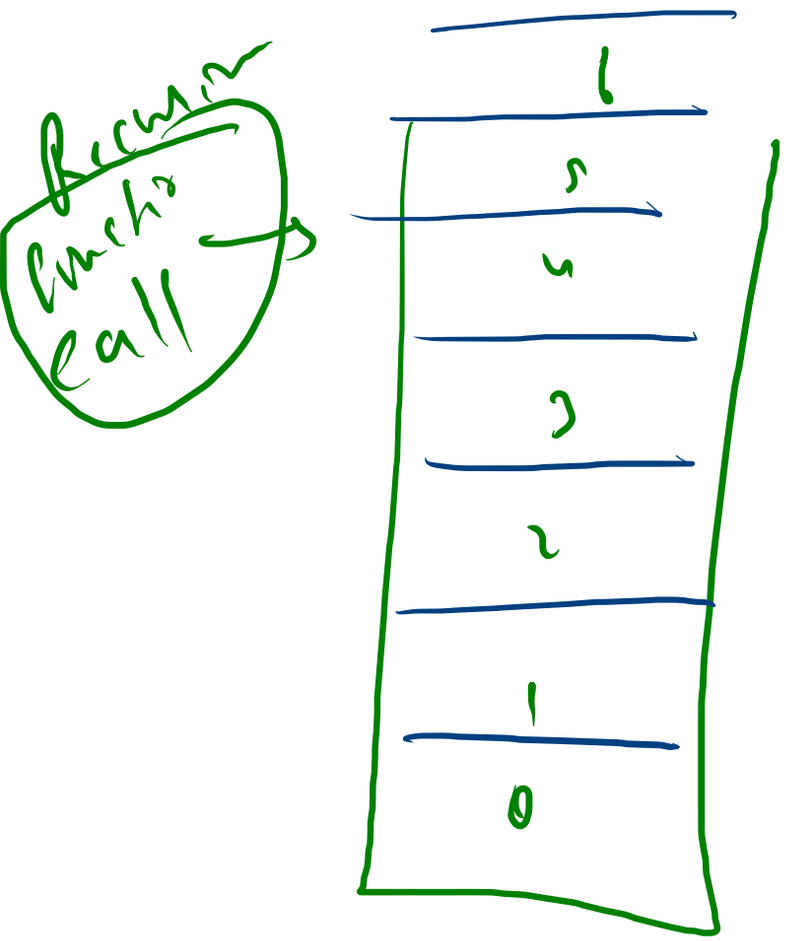
γ make wait child

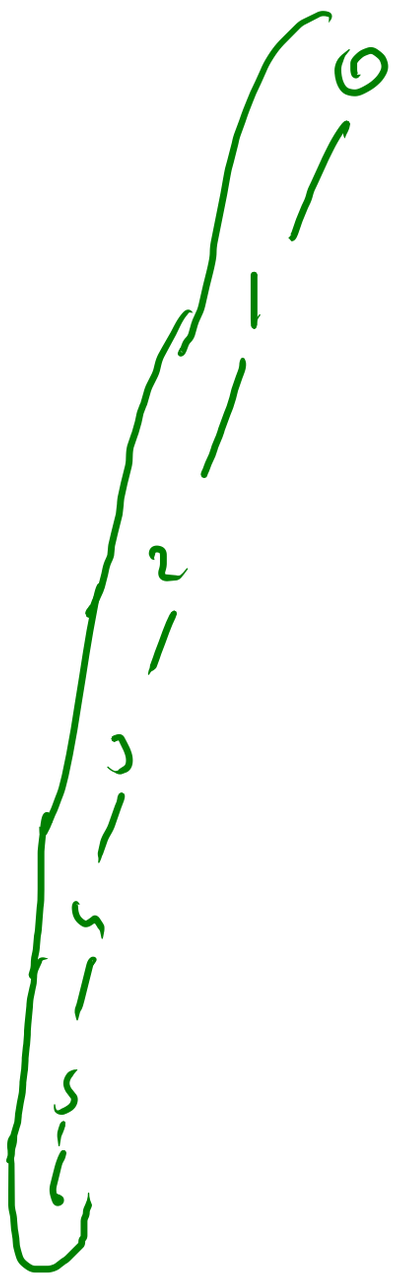


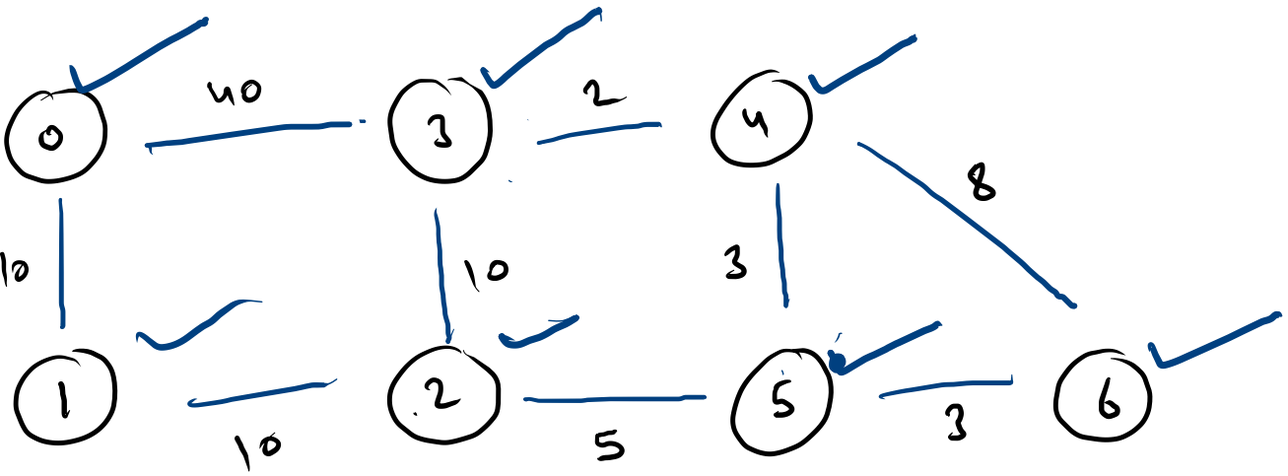
0
3
4

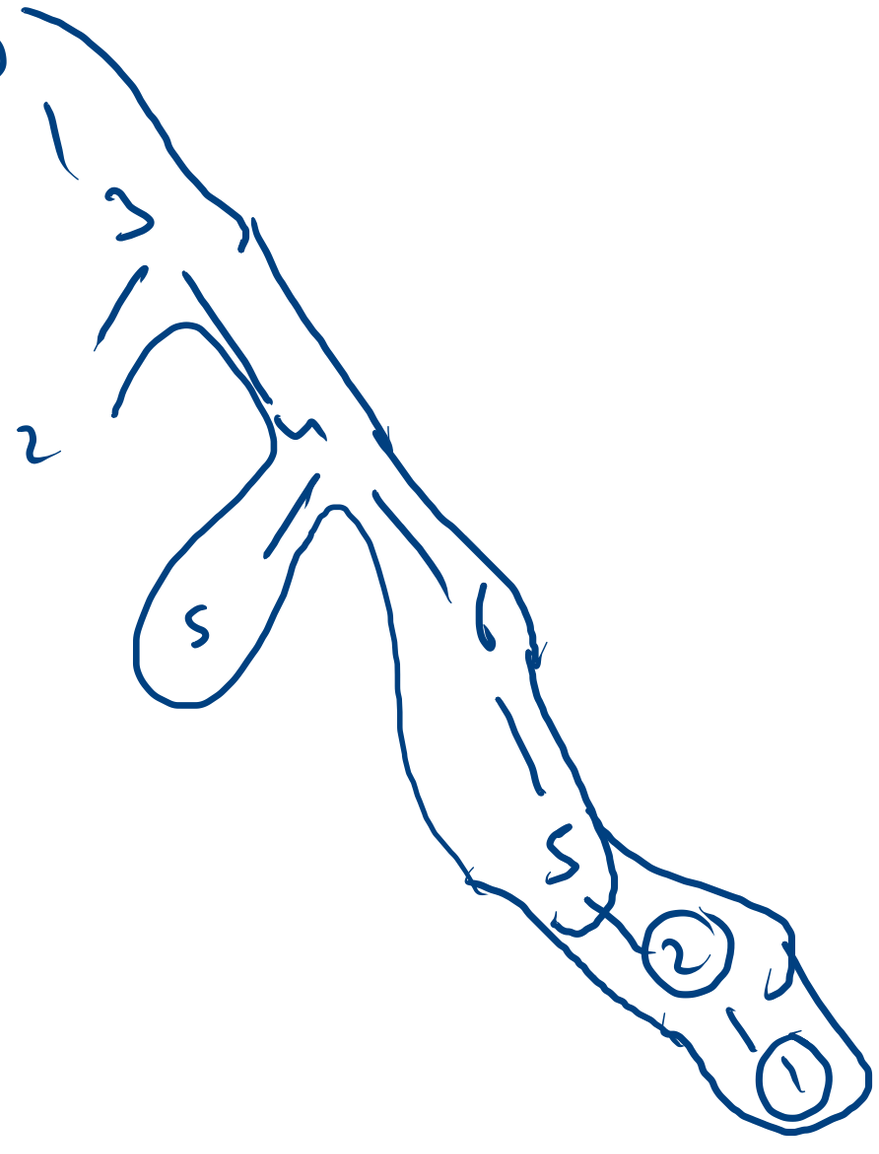


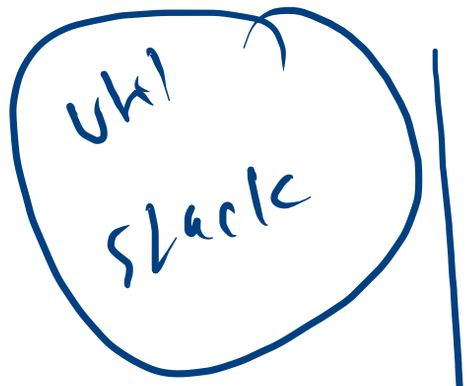

0
40
2
4
8
10
10
3
1
10
2
5
3
6

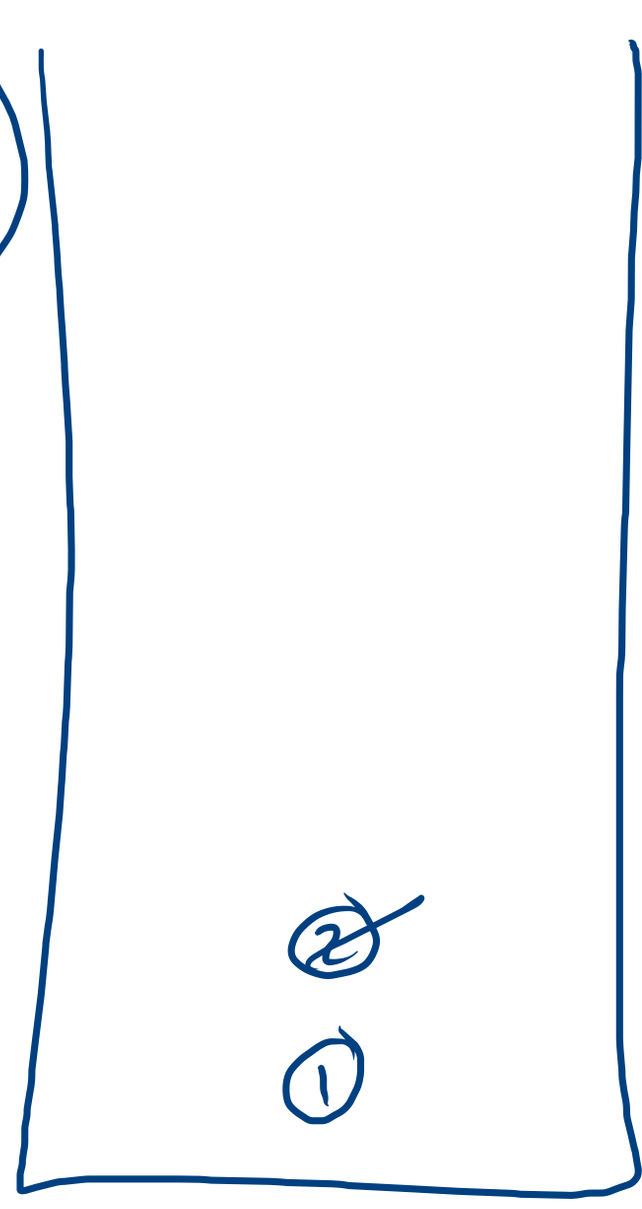

Recursive
Function
call
1
5
4
3
2
1
0

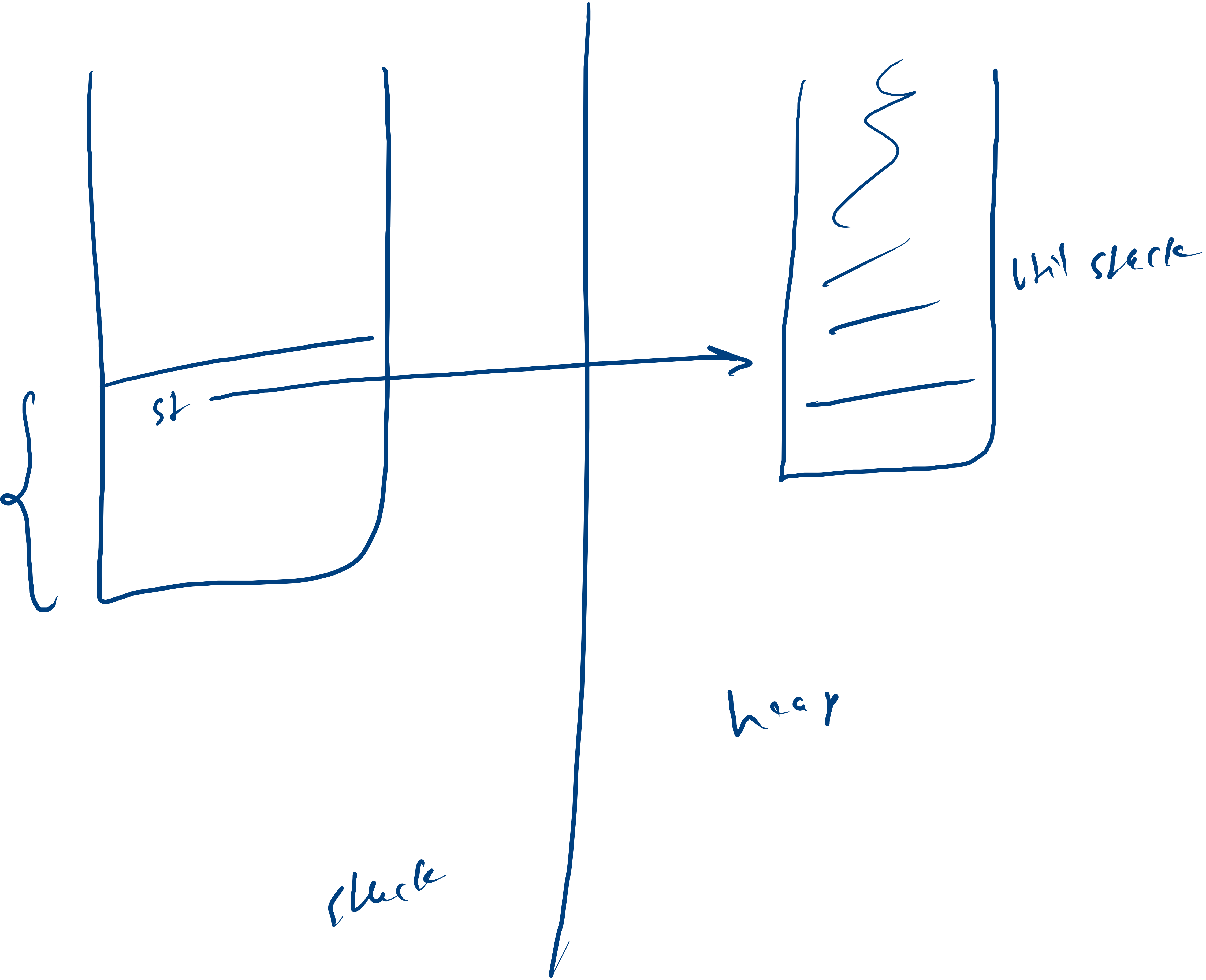

0
1
2
3
4
5


0
40
3
2
4
10
10
3
8
1
2
5
6
10
5
3

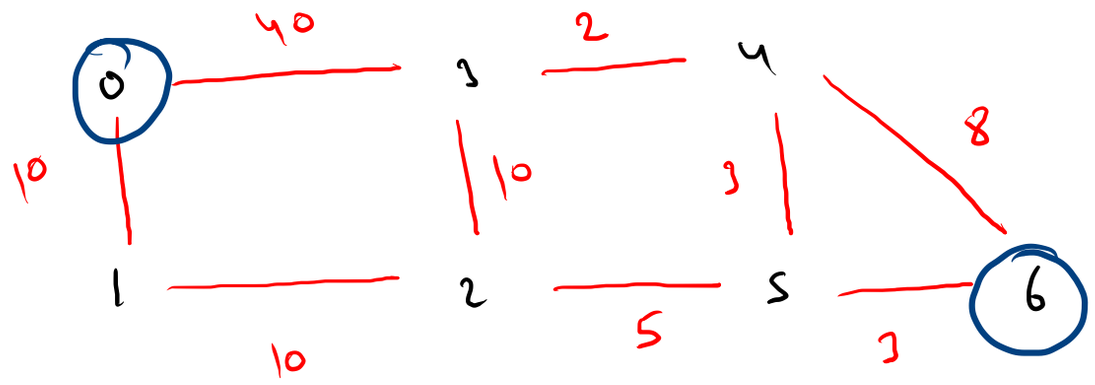

0
1
3
2
5
5
2
1


ukl
slack


2
1


st
heap
stack

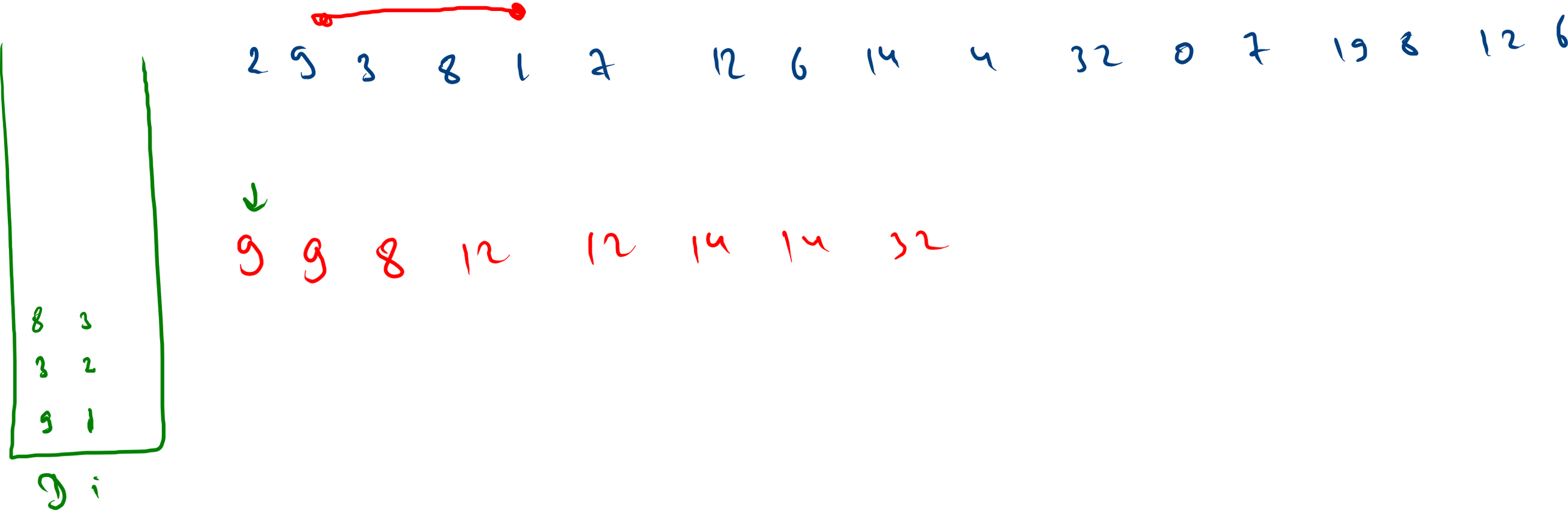
7
9
0 1 10
1 2 10
2 3 10
0 3 40
3 4 2
4 5 3
5 6 3
4 6 8
2 5 5
0
6
30
4



Smallest Path = 01256@28
Largest Path = 032546@66
Just Larger Path than 30 = 012546@36
Just Smaller Path than 30 = 01256@28
4th largest path = 03456@48

Sr → 0

Ds → 6



for the array [2 9 3 8 1 7 12 6 14 4 32 0 7 19 8 12 6] and k = 4, the answer is [9 9 8 12 12 14 14 32 32 32 32 19 19 19]

k=4

gi

i

2 3 4 5

2	5	3	8	1	7	12	6	14	4	32	0	7	19	8	12	6
0	1	2	3	4	5	6	7	8	9	10						

nger

1	6	3	-1	5	6	6

gi

9 9 8 12

if (gi<i)

gi=i

while (ngr[gi] < i+k)

gi = ngr[gi]