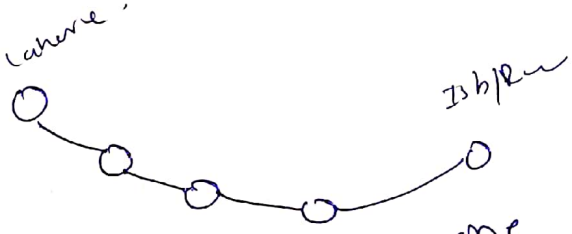
Q no 3

As we know that it is undirected graph which means we can go from one to second and can go back to city one from second.
So here if we use DFS then we can use backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

It can be implemented by using stacks.

```
Preprocess(g, v)
  visited = true
  visited(v).distance = 0
   stack s
   s.push(v)
   ~~while(s.push~~
   while(s.size()>0)
    v' = s.top()
    flag = 0
    for each edge(v', v) in E
    if not visited(v):
       visited(v) = true
       v.distance = v'.distance + edge(v', v)
        s.push(v)
        flag = 1
        break
     if flag == 0
       stack.pop()

return |v.distance - v.distance|.
```



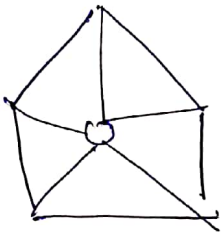
we can use only one loop to find the shortest path.

Here the preprocess has run-time of DFS, $O(V+E)$. thus the best run time must be linear As Distance have runtime of $O(1)$ ~~since~~ preprocess will have distance from source.

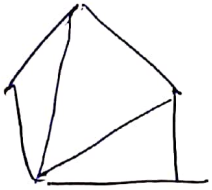
QNO4

if we have 5 cities

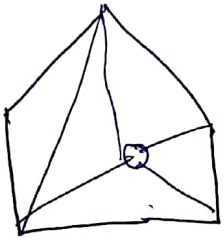
if we introduce bridge b/w them



→ but this not the requirement



This also cannot fulfil our requirement

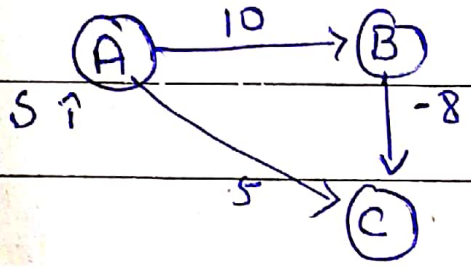


This also cannot fulfil our requirement

So hence we cannot satisfy NHA requirement without intersection or building bridges we cannot connect each city with each other city.

Q NO2

Lets explain with an example suppose we have one graph with negative weight



we know the dijkstra relax the edge only once

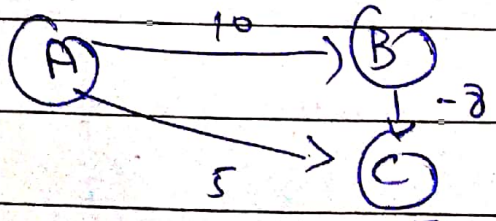
A	B	C
0	∞	∞

initially it will be like this.

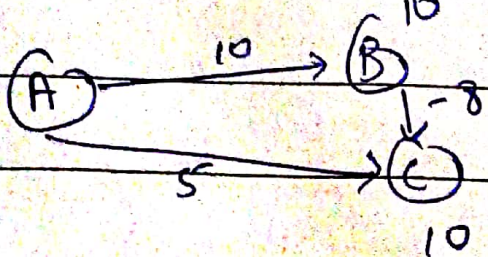
Now when we relax the edges

A	B	C
0	∞	∞
A	10	5
AC	10	5

so here the minimum is 5

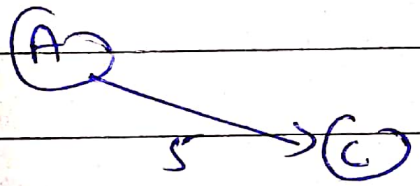


so here the next minimum is 10



So at this stage dijstraw will stop because it relax the edge only once. so if we look at this we can clearly see that it is giving wrong shortest path so it fails here.

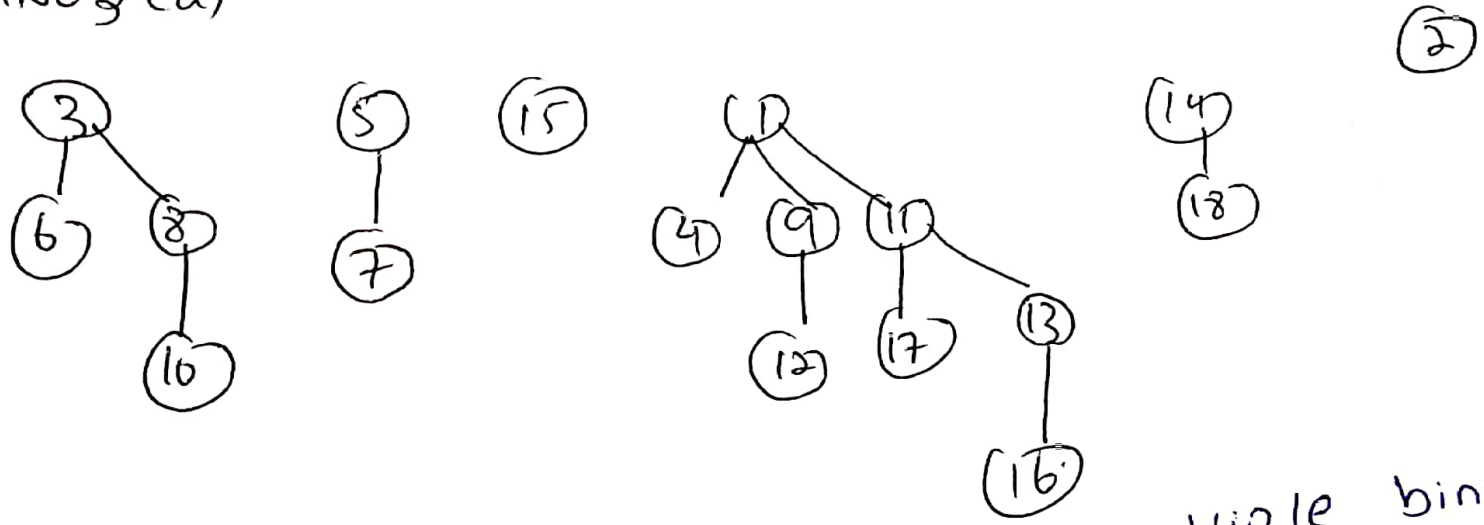
xo. the shortestest here was 2 but it will give 5



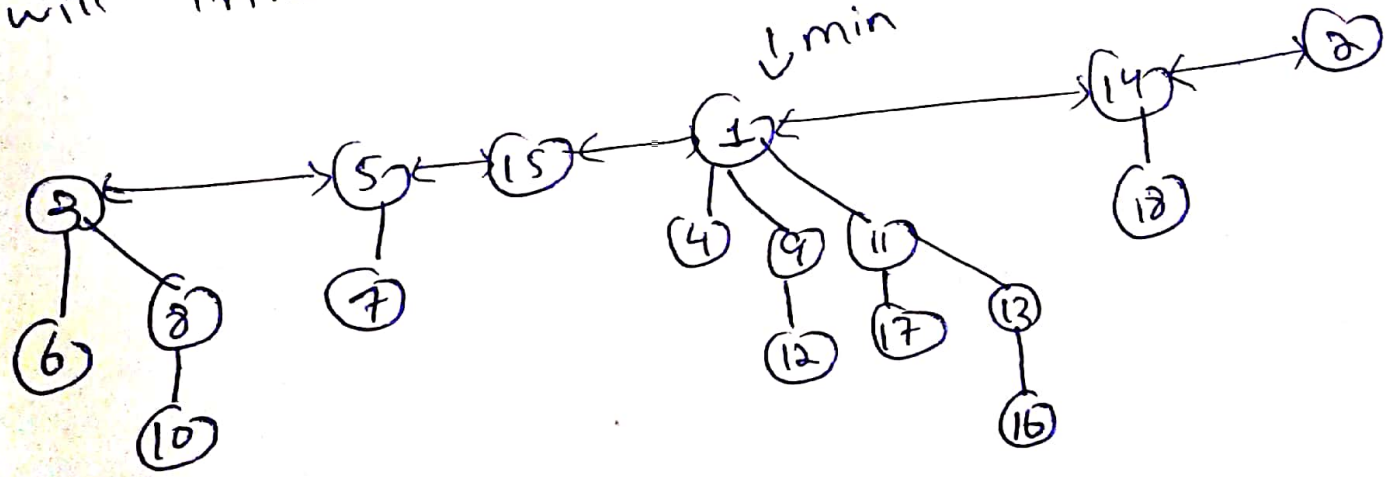
For this we use bellanford to check negative weight edges. because it relax the edge (v-1) times.

Hadoop Distributed

Qno 3 (a)

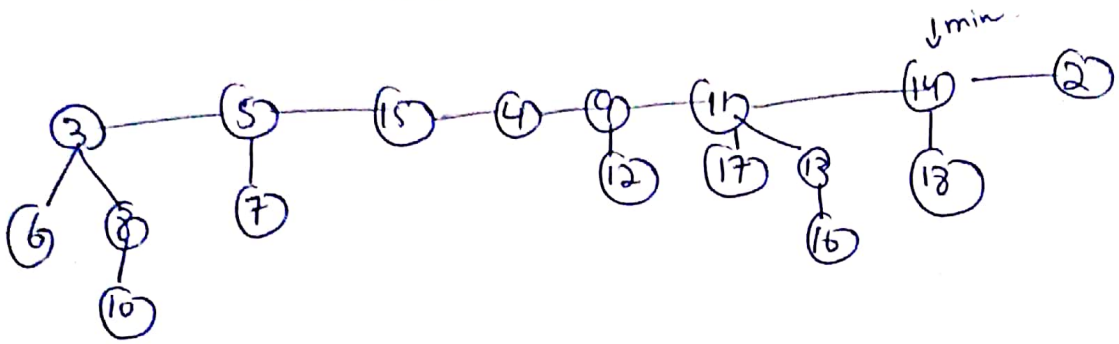


As we know that these are multiple binomial trees that sutisfies the Min-heap property so we will link them to make Fibonacci heap

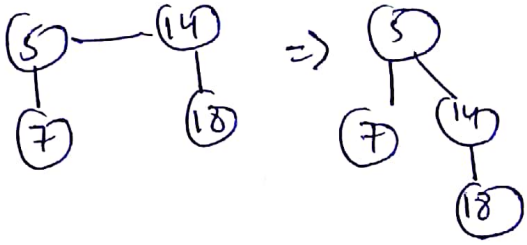


(b) Delete Min

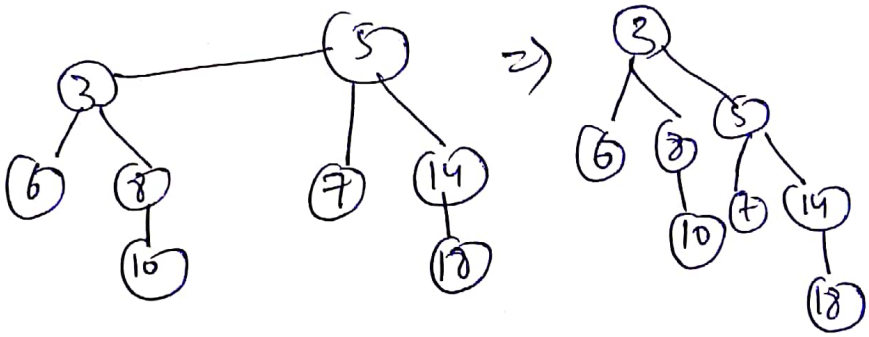
So here we will remove (1) because it is minimum here.



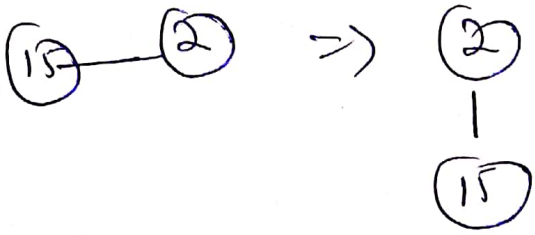
step 1



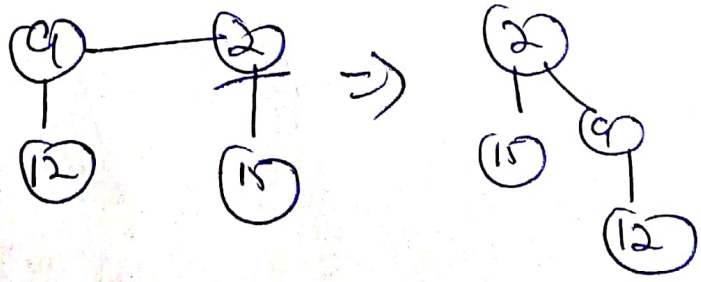
step 2



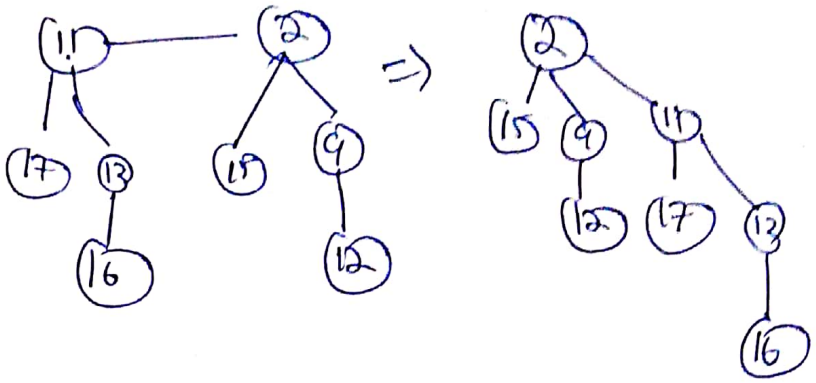
step 3



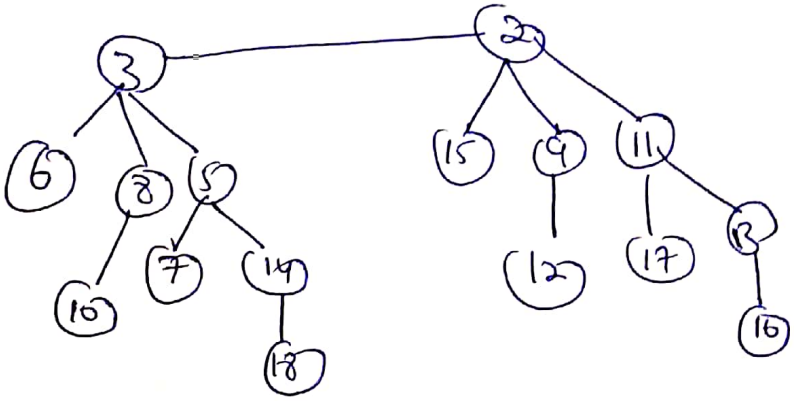
step 4

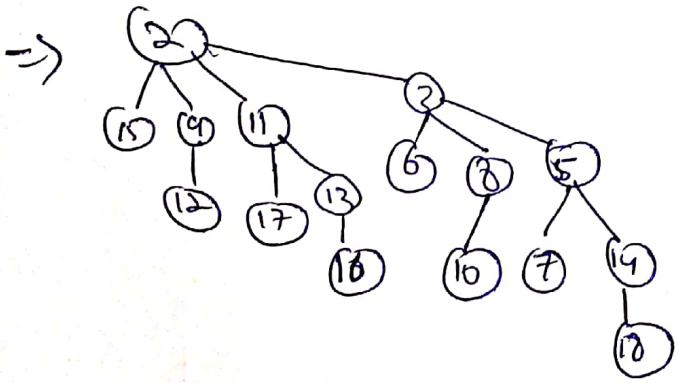


Step 5

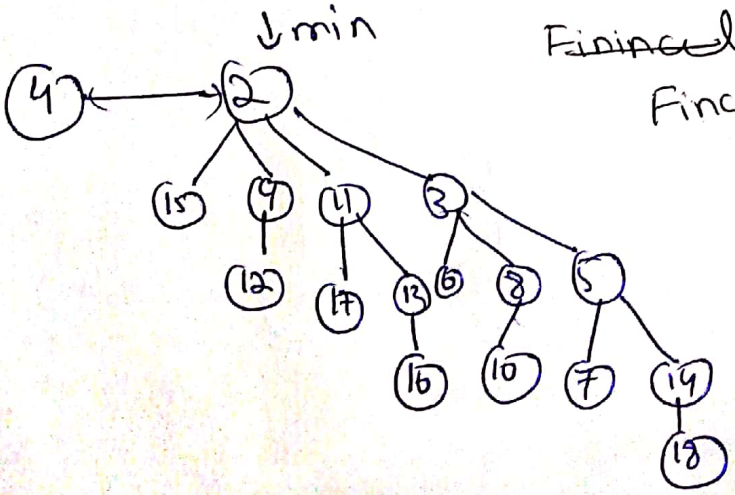


Step 6





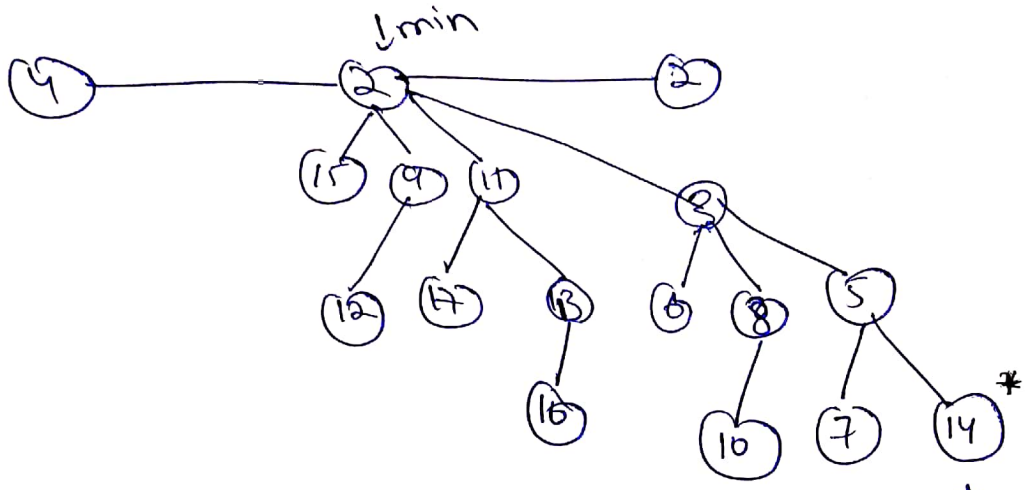
step 7



Finaced

Final answer

(c) Decrease-key 18 to 2



Here the pointer will be at 2 in the same position of previous. It doesn't matter in case of 2 min (having same values). The pointer will be at previous min.