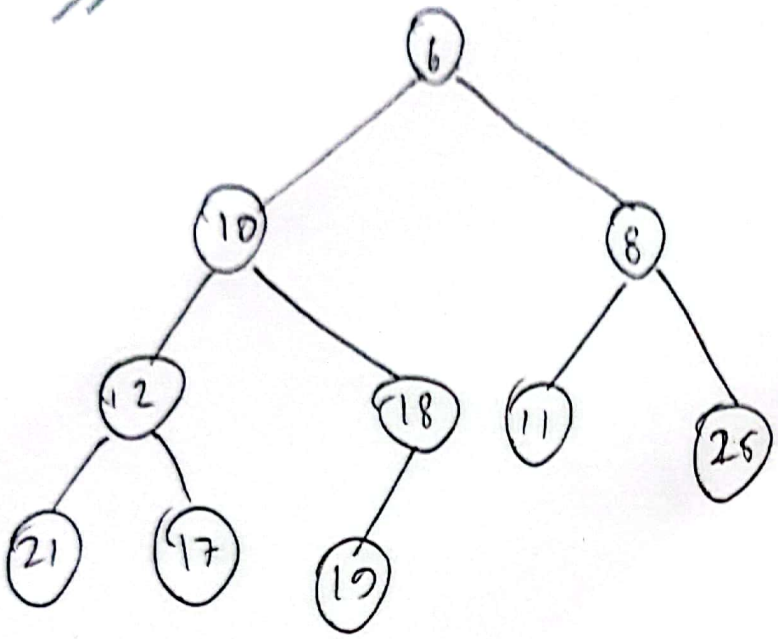
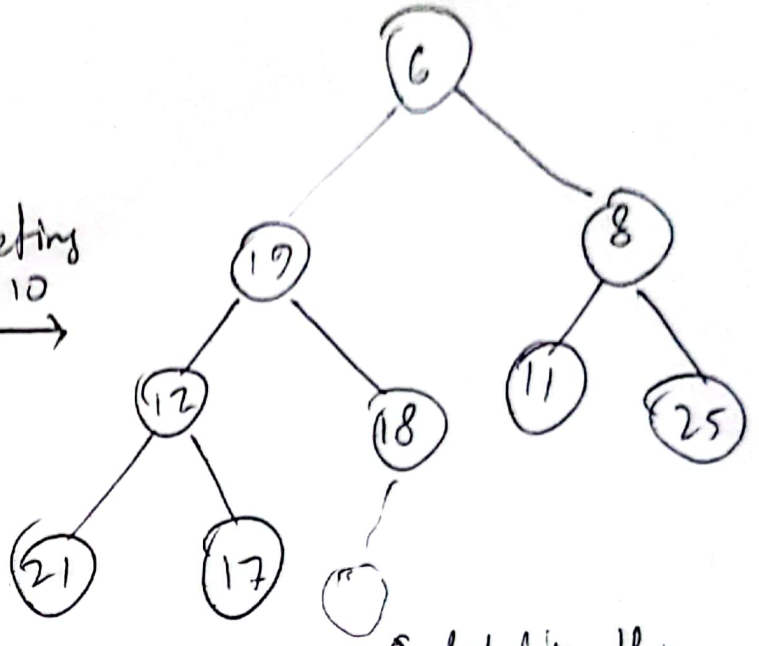
2a)

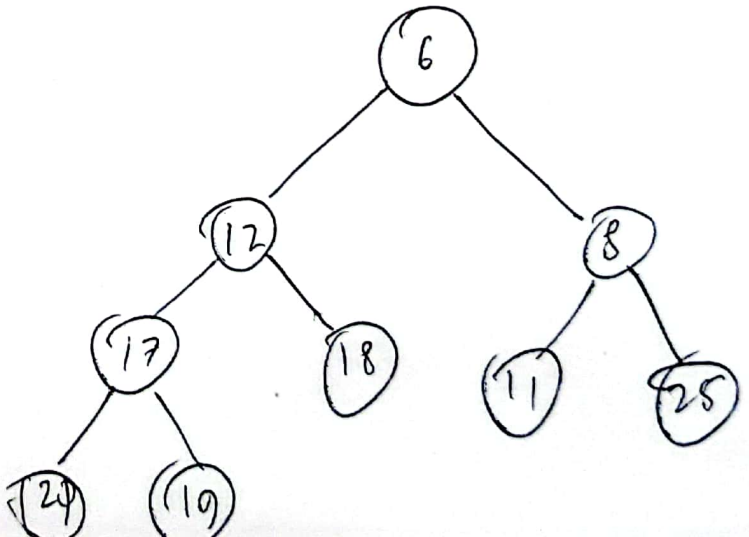



deleting 10 →




↑ deleting this node as well

minHeapify ↓



3(b)

```
MakeIntervalSet(S){
  Let S be the set which contains the numbers.
  R be the set of intervals.
  while S is not empty:
      Get the first element of S, let it be xi
      Push [xi, xi+1] into R
      Keep popping the element of S until all the elements
      in the interval has been popped out.
  return R
}
```

This is not the formal proof.

The main idea of correctness is that we are choosing the left most interval greedily at any time which is not in an interval. Thus, our algorithm is ensured to covered all the points and no space would be wasted even if there is some large gap between interval.

The runtime is O(n)

4(b)

| | | A | L | G | O | R | I | T | H | M |
|---|---|---|---|---|---|---|---|---|---|---|
| " " | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| L | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| I | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| T | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 |
| H | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 3 | 4 | 4 |
| I | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 3 | 4 | 4 |
| U | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 3 | 4 | 4 |
| M | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 3 | 4 | 5 |

LITHM

initial call LCS ("Algorithm, "Lithm")

Declare an arry [s1.size()+1] [s2.size()+1]

LCS ( s1 , s2 ) {

if (s1.length()==0 || s2.length==0)

return 0;

if (arry [s1.size()][s2.size()] != -1) return arry

return arry [s1.size()] [s2.size()];

compare the last characters.

if they are equal

return arry [s1.size()]

dist = LCS ( s1 - last character , s2 - last character) +1

else

dist = max ( LCS (s1 - last character , s2), LCS (s1 , s2 - last cha

return arry [s1.size()] [s2.size()] = max (dist1 , dist2) .