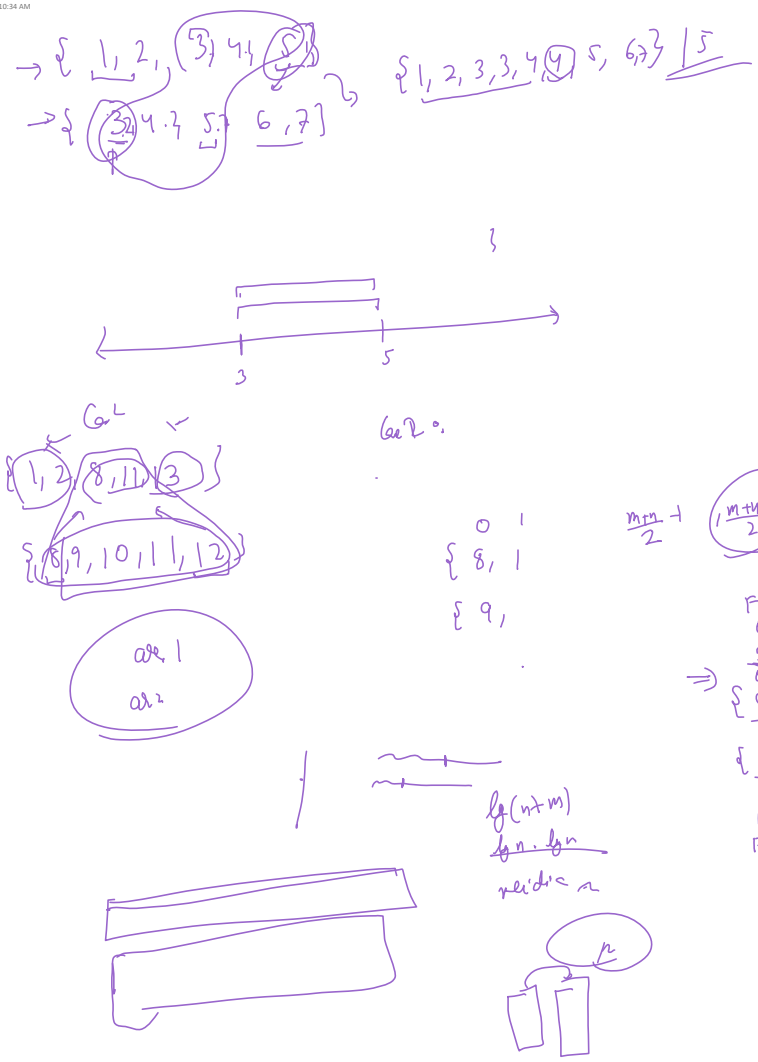
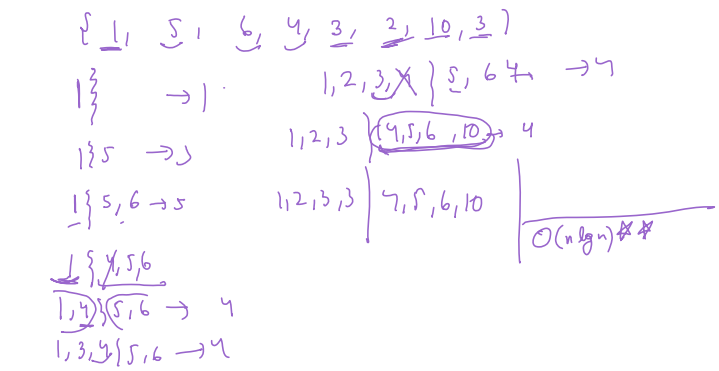
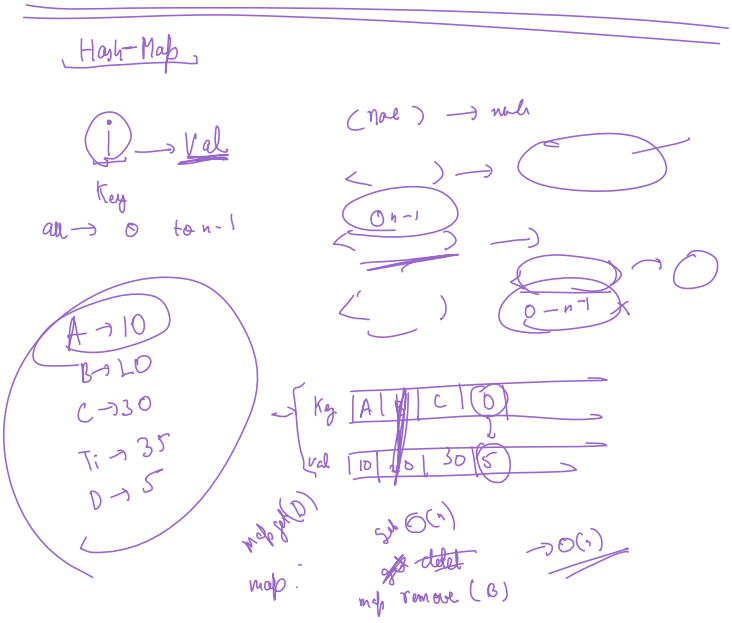


Q [[10,20,30], [15,35], [5,7,12,25], [17, 22, 40]]



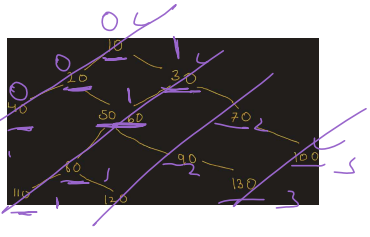


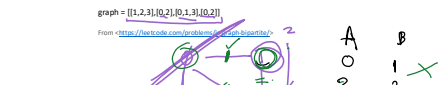

```
int[] arr1 = { 30, 20, 40, 50, 70, 30, 20, 20, 50, 50 };
int[] arr2 = { 50, 80, 30, 20, 20, 20, 90, 50, 20 };
```

M-1) n^3

M-2) $n \log n$

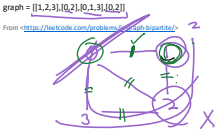
M-3) $n+m$

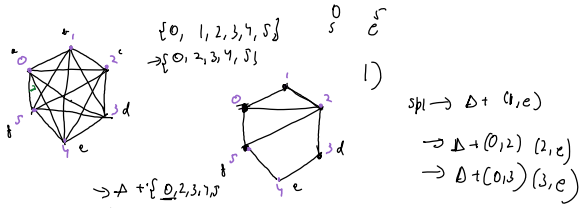


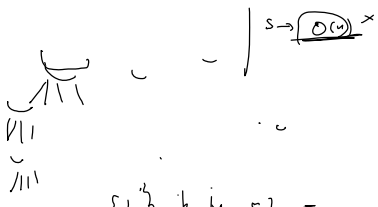


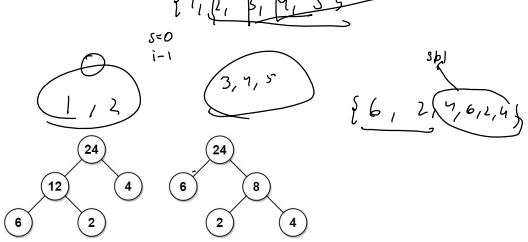
graph = [[1,2,3],[0,2],[0,1,3],[0,2]]

From <https://leetcode.com/problems/is-graph-bipartite/>



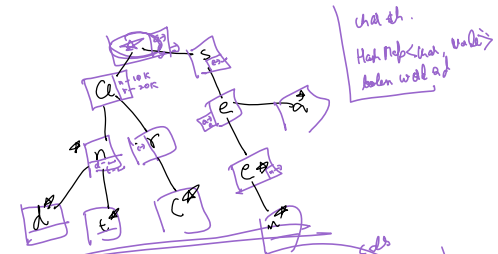


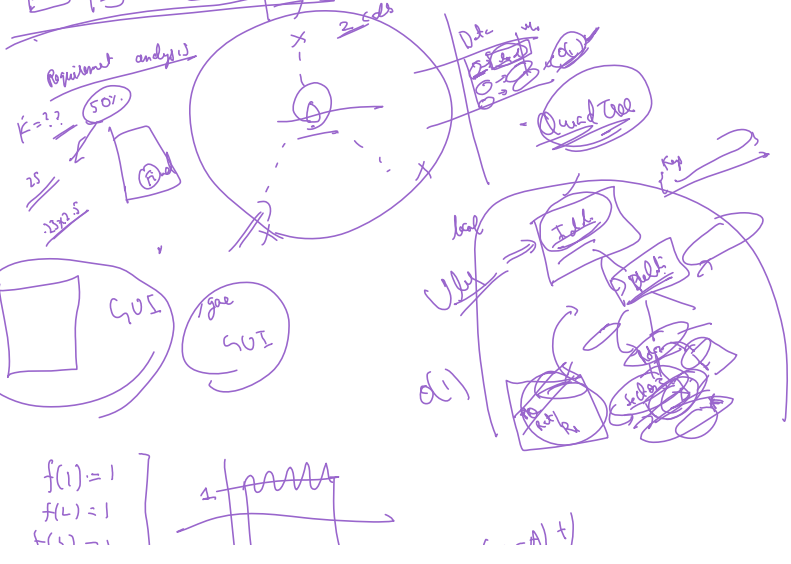


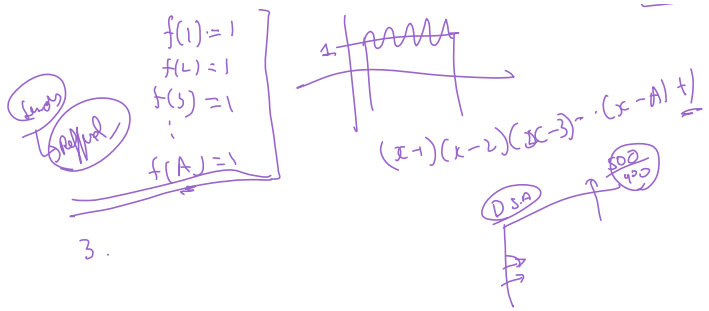


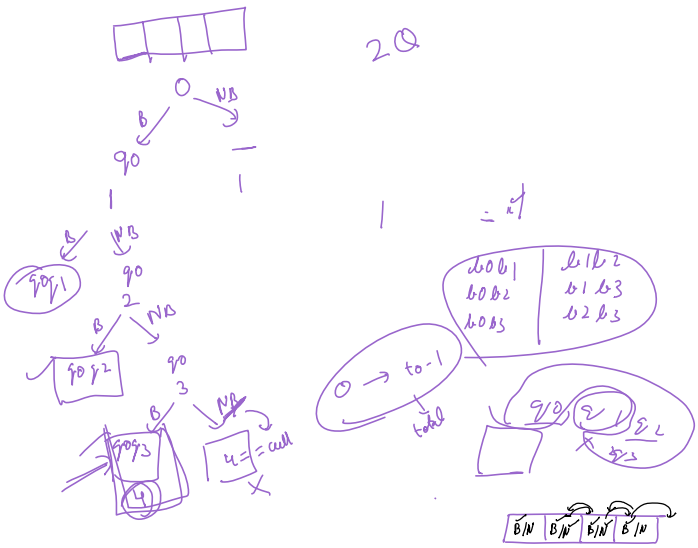
```
public int mctFromLeafValues(int[] arr) {
    return solve(arr,0,arr.length-1);
  }
  public int solve(int[] arr, int s, int e){
    if(s==e){
      return 0;
    }
    int ans = Integer.MAX_VALUE;
    for(int i=s+1;i<=e;i++){
//      i=1;s=0;e=2
      int sp = max(arr,s,i-1)*max(arr,i,e)+solve(arr,s,i-1)+solve(arr,i,e);
      ans = Math.min(ans,sp);
    }
    return ans;
  }
  public int max(int[]arr, int s, int e){
    int ans = arr[s];
    for(int i=s;i<=e;i++){
      ans = Math.max(ans,arr[i]);
    }
    return ans;
  }
}
```

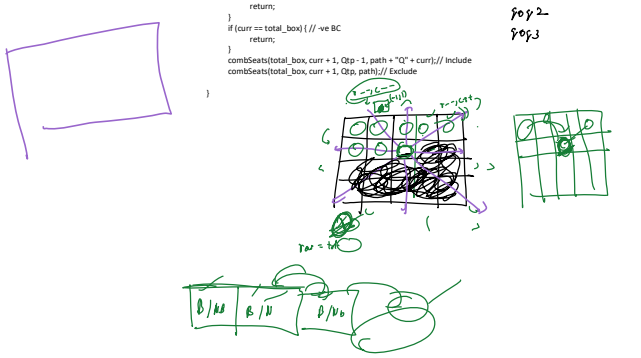
```
trie.addWord("ant");
trie.addWord("arc");
trie.addWord("and");
trie.addWord("see");
trie.addWord("sea");
trie.addWord("seen");
trie.addWord("an");
```





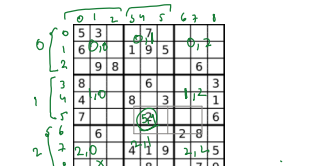


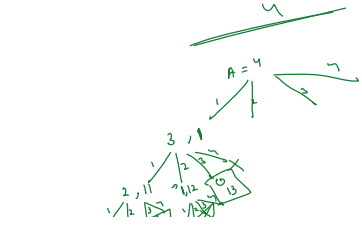


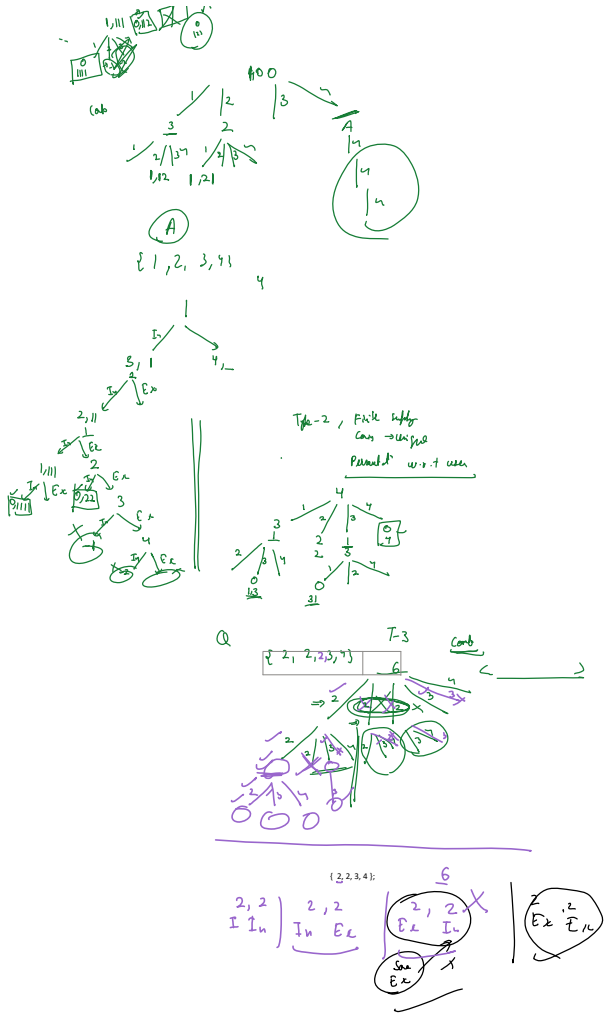


```
public static void combSeats(int total_box, int curr, int Qtp, String path) {
        if (Qtp == 0) { // +ve BC
            System.out.println(path);
            return;
        }
        if (curr == total_box) { // -ve BC
            return;
        }
        combSeats(total_box, curr + 1, Qtp - 1, path + "Q" + curr);// Include
        combSeats(total_box, curr + 1, Qtp, path);// Exclude

    }
```






Comb
A
{1, 2, 3, 4}
Type-2, Fixed suffix
Comb → unique
Permute w.r.t comb
Q
T-3
Comb
{2, 2, 3, 4};
6