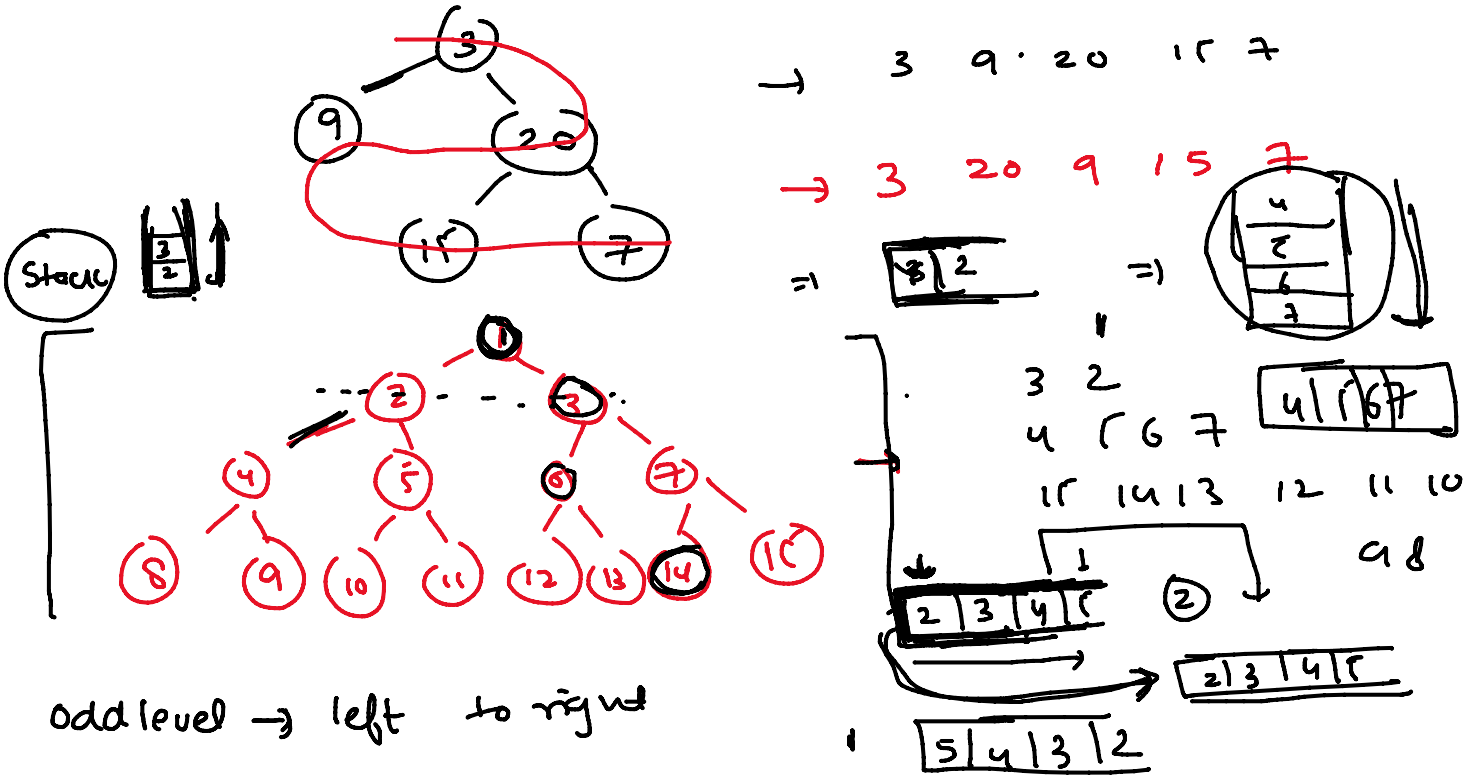
Class 63

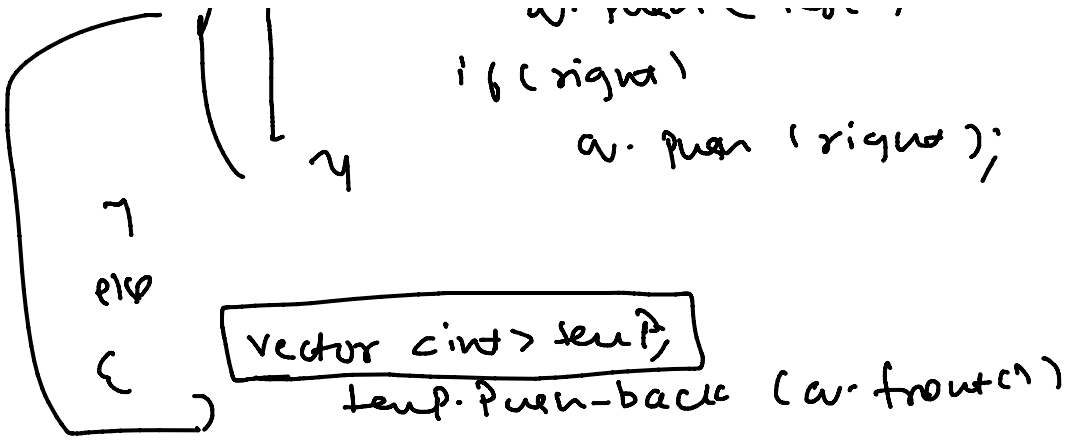
zig-zag level order print



Odd level → left to right

Even level → Right to left

```
queue<Node*>q;
q.push(root);        int level = 1;
while(!q.empty())
{ int n = q.size(); [No of Nodes in curr level)
  if( level %2 == 0)
  {  For( i=0; i<n; i++)
     {  Cout<< q.front();
        if(left)
           q.push(left)
        if(right)
           q.push(right);
     }
  }
```



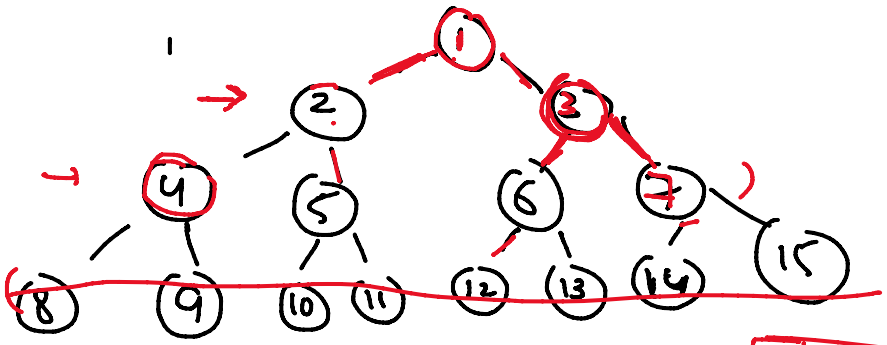
→ reverse (temp.begin(), temp.end());

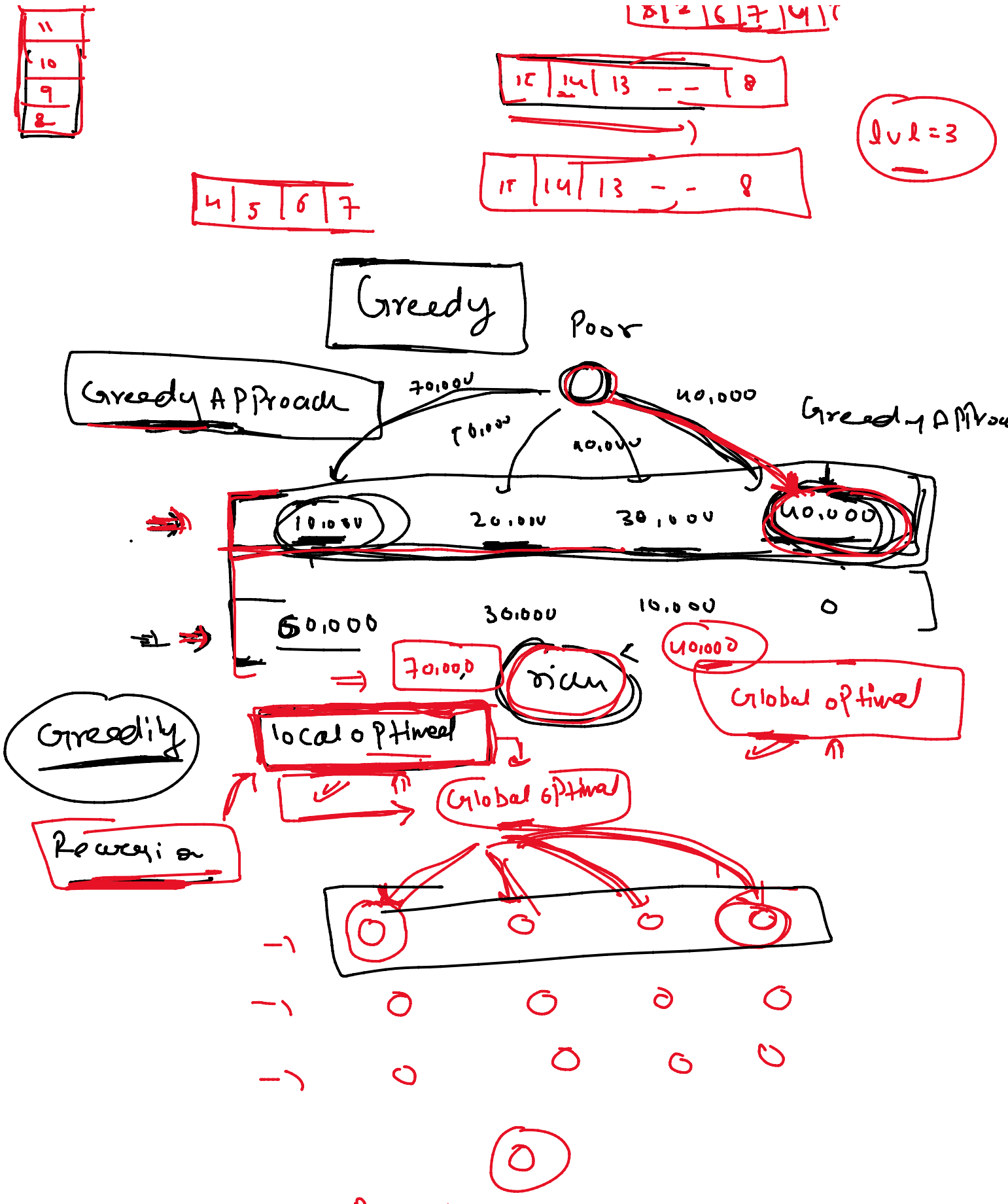
For

```
vector<vector<int>> ans;
queue<Node*> q;
int level = 1;

q.push(root);
while (!q.empty())
{
    vector<int> temp;
    int n = q.size();
    stack<Node*> s1;
    if (level % 2)
    {
        for (int i = 0; i < n; i++)
        {
            temp.push_back (q.front()->val);
            if (q.front() -> left)
                s1.push (q.front() -> left);
            if (q.front() -> right)
                s1.push (q.front() -> right)
        }
        q1.pop()
    }
```

```
        while ( ! s1.empty() )
        { qu.push( s1.top() );
          s1.pop();
        }

    else
    {   for ( i=0; i<n; i++ )
        {   temp.push_back( qu.front() -> val );

            if (qu.front() -> right)
              s1.push (qu.front() -> right)

            if ( qu.front() -> left)
              s1.push ( qu.front() -> left);

            qu.pop()
        }

        while ( ! s1.empty() )
        { qu.push (s1.top())
          s1.pop();
        }

        lvl++;  qu.push_back (temp);
    }
```

lvl = 2




Greedy
Poor
Greedy Approach
70,000
40,000
10,000
20,000
30,000
40,000
60,000
30,000
10,000
0
70,000
40,000
Global optimal
Greedily
local optimal
Global optimal
Recursion
Greedy T.C ↓
Recursion

Activity selection problem

12

- Cricket → 10hr
- Dance → 6
- Movie → 4
- Study → 3

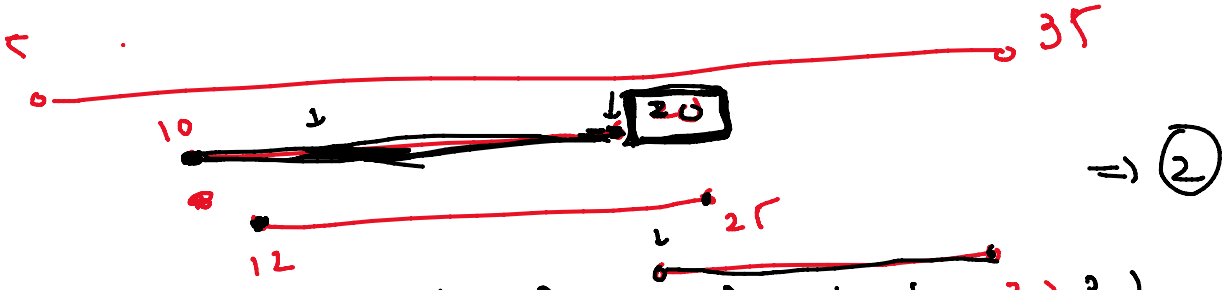
start = [x_1, x_2 - -]

end = [y_1, y_2 - -]

Greedy → Sorting

{ 10, 12, 20 } [(5,35)(10,20),(12,25)(20,30)]

{ 20, 25, 30 } (10,20) (12,25) (20,30) (5,35)



⇒ (2)

```
bool comp ( Pair<int,int> P1, Pair<int,int> P2)
{       if (P1.second == P2.second) return P1.first < P2.first;
        return P1.second < P2.second;
}
int maxActivities ( vector< Pair<int,int> > v)
{   int n = v.size();
    Sort ( v.begin(), v.end(), comp);
```

```
Sort ( v.begin (), v.end (), cmp) ;
int count = 1;
int last = 0;
For (i = 1; i < n; i++)
{   if ( v[i].first >= v[last].second )
    {   count++;
        last = i;
    }
}
return count
```

Greedy (Inc) < ⇒ N meetings in one Room — No of activit

Recursive

2 (exponential)

[1, 3, 0, 5, 8, 5]

[2, 4, 6, 7, 9, 9]

[(1,2) (3,4) (0,6) (5,7) (8,9) (5,9)]

10100 0 — 6 — 5

2, 3, 4, 5, 6

(7,2,6) [13,2]

Minimum cut to roped

[4, 3, 2, 6] ⇒

[10, 3, 2]

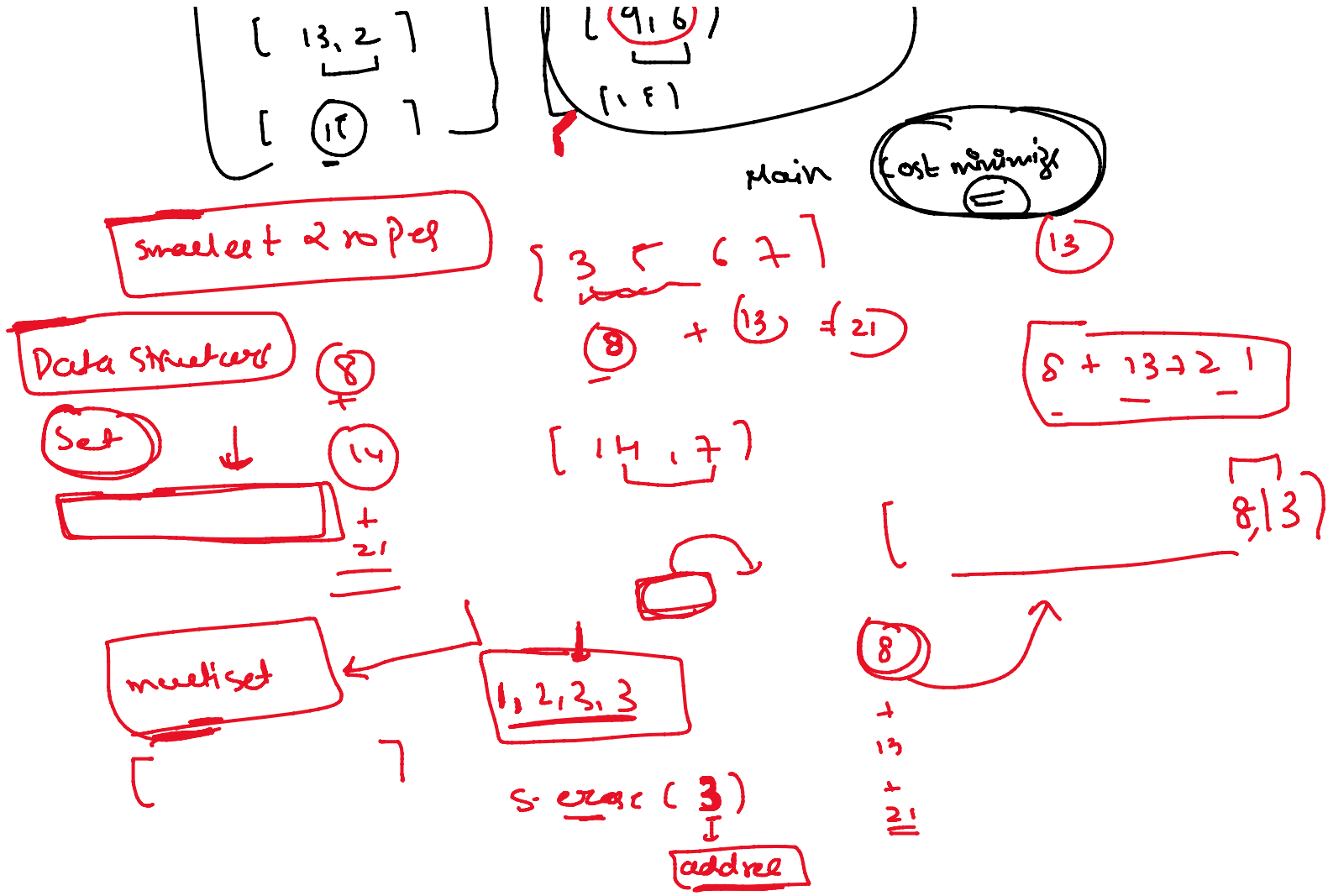
[13, 2]

[5, 4, 6]

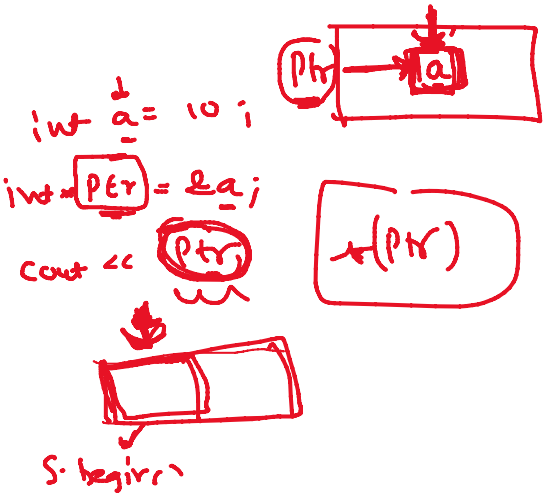
[9, 6]

10 + 13 + 15 = 38

5 + 9 + 15 = 29




```
multiset<int> s;
For(i=0; i<n; i++)
    s.insert(arr[i]);
int cost = 0;
while(s.size()>1)
{  int first = *(s.begin());
   s.erase(s.begin());
   int second = *(s.begin());
   s.erase(s.begin());
```



```
int second = ... ;
s.erase(s.begin());
cost += first + second;
s.insert(first + second);
}

return cost;
```