Day - 7 07-09-2021

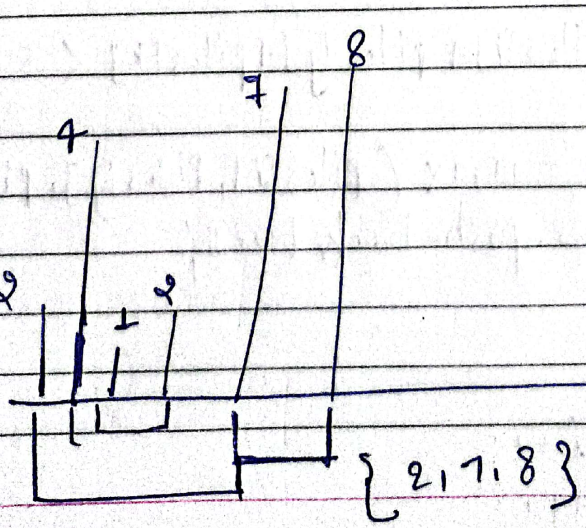
level - medium (challenging question)

Q1 Maximum number of coins you can get.

→ There are 3n piles of coins of varying size, you and your friends will take piles of coin as follows.

- In each step, you will choose any 3 piles of coins (not consecutive).
- Of your choice, Alice will pick the piles with the maximum no. of coins.
- Bob will pick the last pile.
- Repeat until there are no more piles of coins.

Input: piles = [2,4,1,2,7,8]
Output: 9
Explanation: Choose the triplet (2,7,8), Alice pick the pile with 8 coins, you the pile with 7 coins and bob the last one. - - - - - //



logic of take two max and one min

// Sort the given piles
// take two piles from right side
// one from left side
// after getting all triplet then from the triplet take second max and do same from all triplet and add the all second max and get the answer.

Brute force :

```
// int sum=0; int m=INT_MIN; int res;
for(int i=0; i<n-2; i++)
{
  for(int j=0; j<n-2; j++)
  {
    for(int k=0; k<n-2; k++)
    {
      if(piles[i]+piles[j]+piles[k]==sum)
      {
        sum = sum+piles[m];
      }
      else if(piles[i]+piles[j]+piles[k]<sum)
      {
        m = max(piles[i], piles[j], piles[k])
        res.push_back[m];
      }
      else
      {
        i++;
        j++;
        k++;
      }
```

Optimal soln

```
int maxCoins(vector<int>& piles)
{
    sort(piles.begin(), piles.end());
    int sum = 0;
    int n = piles.size()/3;
    int idx = piles.size()-2;
    for(int i=0; i<n; i++, idx=idx-2)
        sum = sum + piles[idx];
    return sum;
}
```

Que-2 : Plus One

Level - Easy

Input: digits = [1,2,3]
Output: [1,2,4]

```
vector<int> plusOne(vector<int>& digits)
{
    int n = digits.size()-1;
    while(n>=0)
    {
        if(digits[n]==9)
            digits[n]=0;
        else
        {
            digits[n]+=1;
            return digits;
        }
        n--;
    }
```

```
        digits.insert(digits.begin(), 1);
        return digits;
    }
};
```

Level- medium

Que-2 Insert Interval (overlap interval)

Input: [[1,3],[6,9]]
[2,5]
output: [[1,5],[6,9]]

```
// two insert the interval.
vector<vector<int>> insert(vector<vector<int>>& intervals,
~~int~~ vector<int>& newinterval)
{
    int flag=1; // if conditions satisfied then flag
                   will be zero.

for(int i=0; i<intervals.size(); i++)
    {
        if(newinterval[0] < intervals[i][0])
    {
        flag=0;
        intervals.insert(intervals.begin()+i, newinterval);
        break;
    }
}
}
```

```
if (flag)
  {
    intervals.insert(intervals.end(), newinterval);
  }

// merge intervals.

Vector<vector<int>>ans;
int start = intervals[0][0];
int end=intervals[0][1];

for(int i=1; i<intervals.size(); i++)
    {
      if (intervals[i][0] <end)
  {
      end = max(end, intervals[i][1]);
  }
  else
      {
        ans.push_back({start,end});
      start = intervals[i][0];
      end = intervals[i][1];
    }
}
    ans.push_back({start,end});
  return ans;
}
```