Lecture - 78

(Greedy - 1)

✪ Greedy Algorithm : they build sol^n piece by piece.

optimal Structure ~ getting ans. of a Problem using optimal ans. of its Sub Problems.

✪ Applications
↳ game, combinatorics, graph, Network, ML,

✪ Advantage:
↳ easy to implement, T.C

✪ Disadvantage: Not applicable to every problem

Q1 ↳ Fractional Knapsack
Given the weights and Profits of N items, in the form of {profit, weight} put these items in a knapsack of capacity W to get the maximum total profit in the knapsack. In fractional knapsack, we can break items for maximizing the total value of the knapsack.

input: arr[] = {{60, 10}, {100, 20}, {120, 30}}, W = 50
Output:- 240

Explanation:- By taking items of weight 10 and 20 kg and 2/3 fraction of 30 kg. Hence total price will be $60 + 100 + (2/3)(120) = 240$

⇝ how to choose which item to pick??
↳ $\boxed{\frac{\text{Value}}{\text{Weight}}}$ → $\boxed{\text{Comparison factor}}$

CLASSTIME Pg. No.
Date / /

Sol:-

```
# include < iostream>
# include < vector>
# include <algorithm>
 using namespace std;
 struct Item {
         int Value;
         int Weight;
 };
 bool cmp (Item i1, Item i2) {
      //Custom comparator for Sorting
      double V-W-i1 = static_cast<double>(i1.value)/i1.weight;
      double V-W-i2 = static_cast<double>(i2.Value)/i2.weight;
      return V-W-i1 > V-W-i2;
 }
 double fractional (int W, Vector <item> & items) {
      double ans = 0;
      sort(items.begin(), items.end(), cmp);
      for (Const auto& item : items) {
           if (item.Weight ≤ W) {
                ans += item.Value;
                W -= item.Weight;
           } else {
                // we cant pick the whole item
                   as space in knapsack is less
                double fraction = static_cast <double>(W)/item.weight;
                ans += fraction * item.value;
                W = 0;
           }
      }
      return ans;
 }
```

T.C → O(n log n)
S.C → O(sorting)

```
int main (){
        int n, W;
        Cin >> n >> W;
        Vector <Item> items;
        for(int i=0 ;i<n;i++) {
            int v, w;
            Cin >> v >> w
            Item it;
            it.value = v;
            it.weight = w;
            items.push_back (it);
        }
        Cout << fractional (W, items) <<"\n";
        return 0;
}
```

Ques2→ Maximum Meetings in one room

There is one meating room in a firm. There are N meetings in the form of (S[i], F[i]) where S[i] is the start time of meeting i and F[i] is the finish time of meeting i. The task is to find the maximum no. of meetings that can be accommodated in the meeting room. Print all meeting no.s.

Input: S[] = {1,3,0,5,8,5}
       f[] = {2,4,6,7,9,9}

Output: 1 2 4 5

CLASSTIME Pg. No.
Date / /

```
struct meeting {
        int start;
        int end;
        int idx;
};
bool cmp( meeting m1, meeting m2) {
        return m1.end < m2.end;
}
void print_max_meeting (Vector<meeting> &arr) {
        Sort (arr.begin (), arr.end (), cmp);
        Cout << arr[0].idx << " ";
        meeting last = arr[0];
        for (int i=1; i <arr.size(); i++) {
                if (arr[i].start > last.end) {
                        Cout << arr[i].idx << " ";
                        last = arr[i];
                }
        }
}
int main() {
        int n;
        Cin>> n;
        Vector <meeting> arr; int i=0;
        while (n--) {
                int s, e;
                Cin>> s >> e;
                ~~int~~ i++;
                meeting m;
                m. start = s
                m. end = e
                m. idx = i
                arr. push_back (m)
        }
```

CLASSTIME Pg. No.
Date / /

```
    print_max_meeting(arr);
    return 0;
}
```

Ques 3: Activity selection Problem
Given N activities with their start and finish day given in array start[] and end[]. select the max. no. of activities that can be performed by a single person, assuming that a person can only work on a single activity at a given day.
Note: duration of the activity includes both starting and ending day.
input: N = 4
start[] = {1, 3, 2, 5}
end[] = {2, 4, 3, 6}

$\{(1,2), (2,3), (3,4), (5,6)\}$
① ② ④

Output: 3
Explanation: A person can perform activities 1, 2, and 4.

Sol: Same as prev. Ques

Ques 14 Check if it is possible to survive on Island

You are a person in Island there is a shop in this Island, this shop is open on all days of the week except for sunday. Consider fallowing constraints:

S - No. of days you are req. to survive
N - Maximum unit of food you can buy each day.
M - Unit of food required each day to survive.

Currently it's Monday, and you need to survive for the next S days.

N - M

Find the minimum no. of days on which you need to buy food from the shop so that you can survive the next S days, or determine that it is not possible to survive.

Condition →
$$\underbrace{\underbrace{\left(\underbrace{S}_{\text{Total Survive days}} - \underbrace{\frac{S}{7}}_{\text{No. of Sunday}}\right) \times \underbrace{N}_{\text{Food can buy}}}_{\text{Total Food which can buy}} \geq \underbrace{SM}_{\text{Food req. to survive}}}_{\text{Survive}}$$

$$\left(\frac{Sm}{N}\right) \to 2.3 \downarrow \underbrace{3}_{\text{No. of days on which Food buy}}$$

✱ Jitna bhi khana chahiye hoga starting me hi kharid lenge So that baad me Sunday aajaye to koi dikkat na ho.

Input:1 → S=10, N=16, M=2
9 2 Din ~~K~~kharedenge
Day 1(16) → 8 Din Chalega
Day 2 (16) → 9 & 10th Din chalega

Minimum No. of Days to buy Food = 2

Ques5→ Given N (very Large), the task is to print the largest palindromic no. obtain by permuting the digits of N. If it is not possible to make a palindromic no., then print an appropriate message.
Input: 313551
Output: 531135
To be Palindrome.

Soln Ek se Jyada No. ni honge Jinki freq. odd hogi.
i.e. odd freq. Vala bas ek hi digit hoga. baaki sare digits ki even freq. hogi
→ Jo bhi largest no. hoga Usse MSB Pe or LSB Pe Rakhenge
↳ Most Significant Bit.

```
#include <iostream>
# include <unordered_map>
# include <vector>

    using namespace std;

    bool isPossible(unordered_map <int, int> &mp) {
        int count = 0;
        for( int i=0 ; i<9 ; i++) {
            if (mp.count(i)) {
                if(mp[i]%2 != 0) count++;
                if ((count > 1) return false;
            }
        }
        return true;
    }
```

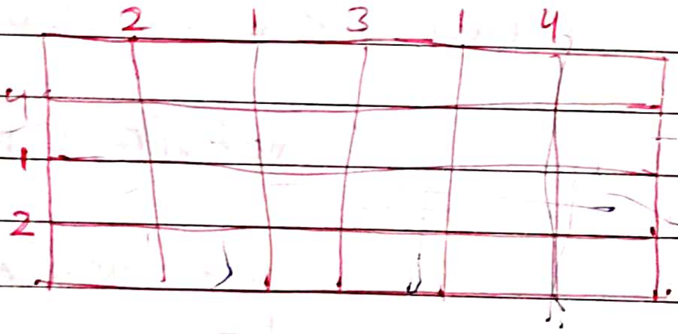
```
string Max_Palindrome(string num){
    int l = num.size(); unordered_map<int, int> mp;
    for (int i=0; i<l; i++){
        mp[num[i] - '0']++;
    }
    if(!isPossible(mp)) {
        return "NP";
    }
    vector<char> V(l);
    int front=0;
    for(i=9; i≥0; i--){
        if(mp[i]%2!=0){
            V[l/2] = char(i+48);
            mp[i]--;
        }
        while(mp[i]>0){
            V[front] = chars(i+48);
            V[l-front-1] = char(i+48);
            mp[i] -= 2;
            front++;
        }
    }
    string res = "";
    for(int i=0; i<V.size(); i++) res += V[i];
    return res;
}
int main(){
    cout << Max_Palindrome("5315122");
    return 0.
}
```

Greedy-2 | Lecture-78

• **Problem 1:** Minimum cost to cut a board into squares.

Q → A board of length M and width N is given. The task is to break this board into M*N squares such that cost of breaking is minimum. The cutting cost for each edge will be given for the board in two arrays X[] and Y[]. In short you need to choose a sequence of cutting (such) such that cost is minimized. Return the minimized cost.

Inputs
M=6, N=4
X[] = [2,1,3,1,4]
Y[] = {4,1,2}
Output: 4



① If we have divided the grids in rectangles & a cut later will be done on ~~later~~ multiple rectangles then cost of the cut should be considered on each rect.

② Vertical cuts increase horizontal blocks and ||ly horizontal cuts increase vertical blocks

③ Phle Jyada cost Valo ko cut karenge So, that km pieces ki cost nikalni pde

Sol:→
```
#include <vector>
#define ll long long int
 using namespace std;
 bool cmp(int x, int y) {        // Used in sorting
       return x>y;                   for dec. order
 }
```

CLASSTIME Pg. No.
Date / /

```
ll minCostToBreakGrid(int n, int m, Vector <ll> &vertical,
                          Vector <ll> &horizontal) {

    sort (Vertical ~~x~~.begin(), Vertical ~~x~~.end(), comp) // Sort each array
    sort (~~y~~.begin(), ~~y~~.end(), comp);                  in dec order
          horizontal     horizontal

    int hz = 1                 // Pieces at starting
    int vr = 1
    int h = 0 , v = 0 ;               // indices of array
    ll ans = 0;

    while (h < horizontal.size() && v < vertical.size()) {

        if (vertical[v] > horizontal[h]) {

            ans += vertical[v] * vr;
            hz++;

            v++;
        } else {
            ans += horizontal[h] * hz;
            vr++;

            h++;
        }
    }
    while (h < horizontal.size()) {
        ans += horizontal[h] * hz;
        vr++;

        h++;
    }
```

CLASSTIME Pg. No.
Date / /

```
while (v < ~~&~~vertical.size()) {
        ans += vertical[v] * vr;
        hz++;

        v++;
}
return ans;
}
int main() {
        int n, m;
        cin>>n>>m;
        Vector<ll> horizontal, vertical;
        for( int i=0; i<m-1; i++) {
                int x;
                cin>>x;
                Vertical.push_back(x);
        }
        for (int i=0; i< n-1; i++) {
                int x;
                cin>>x;
                horizontal.push_back(x);
        }
        Cout<<minCostToBreakGrid(~~&~~ n, m, horizontal, vertical);

        return 0;
}
```

Q21) Leet code $\boxed{435}$ Non-overlapping intervals.

Input $\Rightarrow$ [[1,2] [2,3] [3,4] [1,3]]
↳ will be removed

output $\Rightarrow$ 1
[[1,2], [2,3], [3,4]]

concept
if
(interval[last picked][1] >
interval[i][0])

Count++;

CLASSTIME Pg. No.
Date / /

Q) Problems → Smallest no.

Q → The task is to find the smallest no. with given sum of digits as S and number of digits as D.

input: S = 9, D = 2
output: 18

$1 + 8 \Rightarrow S$

$\uparrow + \uparrow \Rightarrow D$
1 1

Concept

D ⇒ _ _ _ _ _
msb [Chote Se Chota]
lsb [Bde Se bda]

Sol:-
```
# include <iostream>
# include <vector>
  using namespace std;

~~Vector~~
Void smallestNumber(int d, int s) {

        Vector<int> v(d, '0');                // Vector in which no. stored

        if (9*d < S) {                         // if Total Sum is greater than the digits * 9
            Cout<<-1;                          9+9+9 → 27
            return;                            S = 30 not possible
        }
        S--                                    // Ek to pehle hi minus karlenge so that starting pe 1 rakhein
        int i;
        for (i = d - 1; i >= 0; i--) {
            if (S < 9) break;
            v[i] = 9;                          // lsb → 9
            S -= 9;
        }

        v[0] = 1;                              // msb → 1
        v[i] = Char (S + '0')                  // lsb after 9
```

CLASSTIME Pg. No.
Date / /

```
~~for~~ for(int i=0; i<v.size(); i++){
        cout<<v[i];
    }
}
int main(){
    int d,s;
    cin>>d>>s;
    smallestNumber(d,s);
    return 0;
}
```

Problem: leetcode $\boxed{1235}$. Maximum profit in Job scheduling.