... Algorithm

- Approach jisme hum locally best decision lete hain.
- bina future a jyda sochte hain
- assuming ki final answer is the optimal one

Eg → Coins → {1, 2, 5, 10}

minimum coins to make target (18)

→ to largest lenge. (10) → coin (I)

18 - 10 = 8

→ if 10 > 8 ×
  5 < 8 ✓ so (5) → coin (II)

→ 8 - 5 = 3
  if 5 > 3 ×
  2 < 3 ✓ so (2) → coin (III)

→ 3 - 2 = 1
  if 2 > 1 ×
  1 = 1 ✓ so (1) → coin (IV)

minimum no. of coins = (4)

ans (4)

★ har jgha useful in hai quki, aur algorithm useful hoskte hai sirf unme lagega jinme local optimal hi global optimal hai.

2 properties → Greedy choice :
  (1) Ex optimal solution →
  (2) phla decision hi greedy chaia.

Structure
- phele optimal solution jo banega
- whoi uski subproblems ka solution bhi hoga

★ Same as DP but approach alg hai

How to identify greedy ?

(1) Keywords → maximum & minimum hoga
(2) least time
(3) minimum number of operations
(4) Schedule
(5) selecting non-overlapping
(6) optimized

Mostly → sorting
  single pass
  local decision

★ Patterns
- Interval Based
- Selection based
- cost optimization
- heap + greedy

Greedy	/ DP
→ fast	→ slower
→ Simple	→ complex
→ local decision	→ all possible
→ May fail	→ correct 100%

Mistakes :
- har jgha applicable in hai;
- Sorting glat basis pe
- proof nahi samjhna
- greedy ko DP samajhna

① N meetings in one Room

→ I/P → n meetings with (s/e) time find maximum ~~time~~ meetings to be done in single room.

Start = [1, 3, 0, 5, 8, 5]
End = [2, 4, 6, 7, 9, 9]

intuition : Sort according to end time ☆

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

0 1 2 3 4 5

pick i = 0

ans = [st = 1, end = 2] cnt++ = 1

next only when st time of pair just next to greater than (st) of recent

(3 > 2) cnt++ = 2

end = 3

→ 0 > 3 ✗

→ 5 > 3 ✓ cnt++ ⇒ 3

→ 8 > 5 ✓ cnt++ = 4

end = 8

Code vector< pair<int, int>> v; # sorted

→ for(int i=0; i<n; i++)

{ pair<int, int> p = make_pair(start[i], end[i]);

v.push_back(p);

}

sort(v.begin(), v.end(), cmp);

static bool cmp(pair<int, int> a, pair<int, int> b) } cmp

{ return a.second < b.second;

}

int cnt = 1;

int anend = v[0].second;

for(int i=1; i<n; i++)

{ if(v[i].first > anend) } if prev's end chota hai next kai start se,

{ cnt++;

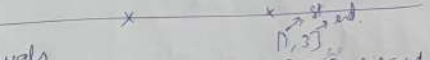
anend = v[i].second;

}

}

return cnt;

}



Merge intervals

interval[i] = st[i] & e[i];

merge all overlapping intervals

i.e. [[1,3] [2,6] [8,10] [15,18]]

→ [1 → 6] time merges all [1-3] [2-6]

[8-10] } ans

[15-18]

☆ greedy lagao aur sort karke, fir (pair mai values) compute karo.

Non-Overlapping intervals

[1, 2]
[2, 3]
[3, 4]
[1, 3]

→ Sort →

st	end
1	2
1	3
2	3
3	4

```
st = 1, end = 2
(i = 1 ⟶ v.size())
    v[i].first == ansst;
        ansend = v[i].second;
    v[i].second >= ansend
        end = v[i].second;
    else {
        push_back(st,end);
        st = v[i].first;
        end = v[i].second;
    }
```

intervals ⇒ [1,2] [1,3] [2,3] [3,4]

```
prevend = v[0].second;
agr next kai first < chota hai
    cnt++;
else {
    prevend = v[i].second;
}
return cnt;
```

* Selection / Maximum / minimum Removal
  → End time se sort (jo phele khtm hoga wo hi future k liye zyda gagah dega)
* Merging / Overlapping / timeline
  ⇒ Start time se sort
  → Agar question Choice line ka hai → END;
    question timeline ka hai → Start;
* Selection / Min / Max → END
  Merging / combine / timeline → Start.

LC→452 : [1,6] prevend = v[0].second = 6
[2,8] if(v[i].second > prevend) continue; ①
[7,12] else { cnt++;
[10,16]

* ballon ki range = (Start, end);
  [Start <= x <= end]

Shop in Candy store

n types of candy

N = 7 = {1, 2, 3, 4, 5, 6, 7}
types

if I buy, I can take (K types) of candy, find minimum money,

C[7] = [3, 2, 1, 4]

sort → [1, 2, 3, 4]

take ①, K = ②

→ ab sabse mehnge lelo

→ ab ≠ 2 dekr ② wali bhi leli.

prices ⇒ price[i];
visited[K-1];
visited[K-2]
cnt = ①, [candy = 1, 3, 4]

10	12	13	14
F	F	F	F
0	1	2	3

visited[i] == false;

4-2 ⇒ ②

price = (10)

[1, 2, 3, 4] [1, 2, 3, 4, 5] 5-2 → 3 ✓

for(int i=0; i<n; i++)

if(prices[i] == false) exit

price = priceof[i] + prices;

visited[n-K+i] = true;

visited[n-K] ≠ true;

(n-K+i) / (n-K+i+1)

④ 1 → n-K+1

5-2+2 = ④ / ②

(n-K+1) → (n-K+1)

5-2+1 → 3+1 = ④

5-2+0 → ②

---

i	j				
1	2	3	4	5	
0	1	2	3	4	

K = ①

j = ④

while(i<=j) (min)

cost += price[i];

i++;

j = j - K;

return cost

int i=0; j = prices.size()-1; cost=0;

while(i<=j) (max)

cost = cost + priceof[j];

j--;

i = i+K;

cout;

S → ~~each day mai kitne ke sakta hai~~ Survival days

N → each day buy krskta hai;

M → each day requirements;

S → (10) N = (16), M = ②

⇒ No of sundays → 10/7 → ① (S/7 → no. of sundays)

14/7 = ②

totalfood = S * M → 10*2 → (20) % N == 0

multiple → return (S, N)

if not → 20/16 → ① return (S, N/N + 1)

```
int miniday(int S, int N, int M)
{
    int sundays = S/7;
    int buysigdays = S - sundays;
    int totalfood = S * M;
    if (totalfood % N == 0)
        return totalfood/N;
    else {
        totalfood/N + 1;
    }
}
if (ans <= buysigdays)
    return ans;
else {
    return -1;
}
```

S = 10
N = 1
M = 1

sunday → 1
buysigdays = 9

(10 % 1) == 0;
total/N

else {
10/1 + 1/1 → 11/1 → 11

if (ans <= buysigdays)
10 <= X
else { return -1;

```
string reverseword(string s)
{
    string ans = "";
    string ch = "";
    for(int i = s.size()-1; i >= 0; i--)
    {
        if (s[i] != '.')
            ch = ch + s[i];
        else {
            reverse(ch.begin(), ch.end());
            ans = ans + ch;
            ch.clear();
            ans.push_back('.');
        }
    }
// last word reh gya hoga
reverse(ch.begin(), ch.end());
ans += ch;
return ans;
}
```