What is greedy?

phone → Amazon (32k)
phone → paytm (28k)
phone → flipkart (30k)

} greedy

Company

22 lpa 30 lpa 31 lpa

other factors come into play

→ not greedy

{done}

Job Scheduling

N jobs/tasks → reward
→ deadline

Any job takes one day
Cant do 2 jobs at the same day

Eg -

Task	deadline	reward
A	3	100
B	1	19
C	2	27
D	1	25
E	3	30

1	2	
3	1500	⇒ 1503

Greedy → First sort by deadline

deadline	1	1	2	3	3
reward	19	25	27	30	100

day = 4

day	1	2	3
task	100	27	30

100
27
30

dead	1	2	3	3	3
rew	5	1	6	3	9

day = ~~1~~ ~~2~~ ~~3~~ 4

5
9
6

final = 5+6+9 = 20

min heap

1	1	1	2	2	2	3	3	3
5	2	7	6	4	9	6	3	9

day 1 → 7

Code

1) sort acc to deadline

```
day = 1
minheap h
for(i=0; i<n ; i++) {
    if ( deadline[i] >= day ) {
        heap.insert( reward[i])
        day ++
    }
    else {
        if( rew [i] > h.root() ) {
            h.removeMin()
            h.insert( rew [i])
        }
    }
}
return sum of heap
```

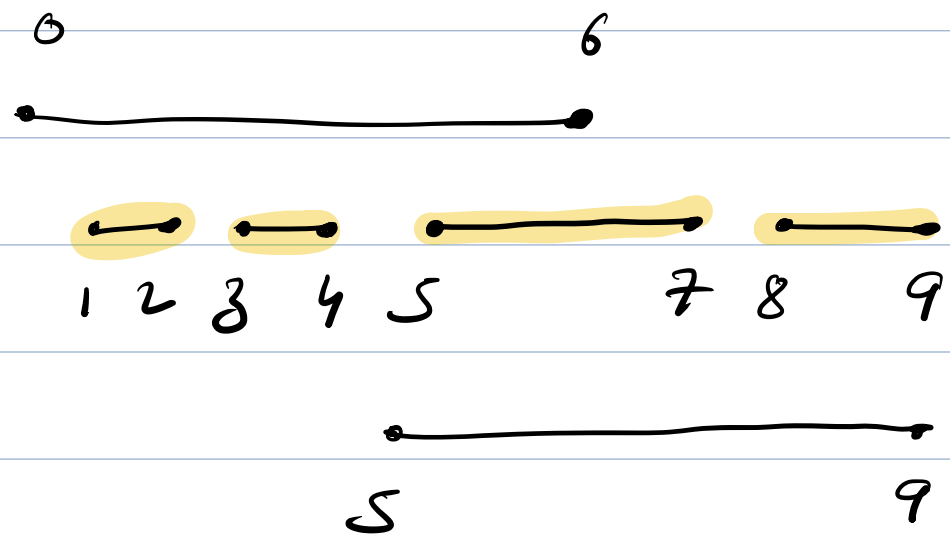
TC: N log N

SC: N

Activity selection problem.

N activities start / end

	a_1	a_2	a_3	a_4	a_5	a_6
start	1	3	0	8	5	5
end	2	4	6	9	7	9



Sort acc to end time

1	3	0	5	5	8
2	4	6	7	9	9

ans = 4

time = ~~2~~ ~~4~~ ~~7~~ 9

Code

1) Sort acc to end time

2) ans = 1 end = ending [0]

```
for (i=1 ; i<n; i++) {
    if( start [i] > end ) {
        ans++
        end = ending [i]
    }
    else
        continue
}
```

		1	5	8	7	13	12
return	ans	2	10	10	11	19	20

TC : $O(n \log n)$

SC : $O(1)$

Q2 Distribute Candy

N children → ratings
Give candies to all children

1) ≥ 1 to everyone
2) child with higher rating than neighbour should have more candies than neighbour

Minimize total no of candies

A: [8 10 6 2]

L: 1 2 1 1

R: 1 3 2 1

fin 1 3 2 1 tot = 7

	1	6	3	1	10	12	20	5	2
L[]	1	2	1	1	2	3	4	1	1
R[]	1	3	2	1	1	1	3	2	1
fin	1	3	2	1	2	3	4	2	1

ans = 19

Code

```
int candies ( int A[], int N) {
    int L[n]
    int R[n]
    L[0] = 1
    for (i = 1 ; i < n ; i++) {
        if (a(i) > a(i-1))
            L[i] = L[i-1] + 1
        else
            L(i) = 1
    }
```

```
R[n-1] = 1
for ( i = n-2 ; i >= 0 ; i--) {
    if ( a(i) > a(i+1) )
        R[i] = R[i+1] + 1
    else
        R[i] = 1
}

Candies = 0
for (i=0 ; i<n ; i++) {
    candies += max(L[i], R[i])
}
return candies
}
```

TC : } O(n)
SC : }