Q Prakash went to get cake at bakery

N types of cakes present (Array)

Each cake — weight
\ happiness

W Limit for bag = 7 Kgs of cake

Max total happiness

	0	1	2	3
H ⇒	1	4	5	8
W ⇒	3	3	4	5
⇒	0.3	1.33	1.25	1.6

= 8

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

greedy

2 Kgs

⇒ You are allowed to break the cake into per unit weight

Knapsack

⇒ Given N objects
⇒ 2 properties ⇒ weight & value
⇒ Total weight capacity (W)

Goal ⇒ Either maximise or minimise the select some objects

sum of values s.t.

sum of weights $\leq W$

⇒ Fractional Knapsack

⇒ To find the optimal solution of the overall problem, we were optimizing local subproblems

Greedy Algorithm

⇒ Taking a greedy decision without caring about future.

⇒ Take a local optimal solution, keep on repeating the same process till the end. you will get the global optimal solution.

	0	1	2	3
H ⇒	1,	4,	5,	8
W ⇒	3,	3,	4,	5
	0.3	1.33	1.25	1.6

⇒ W = 7

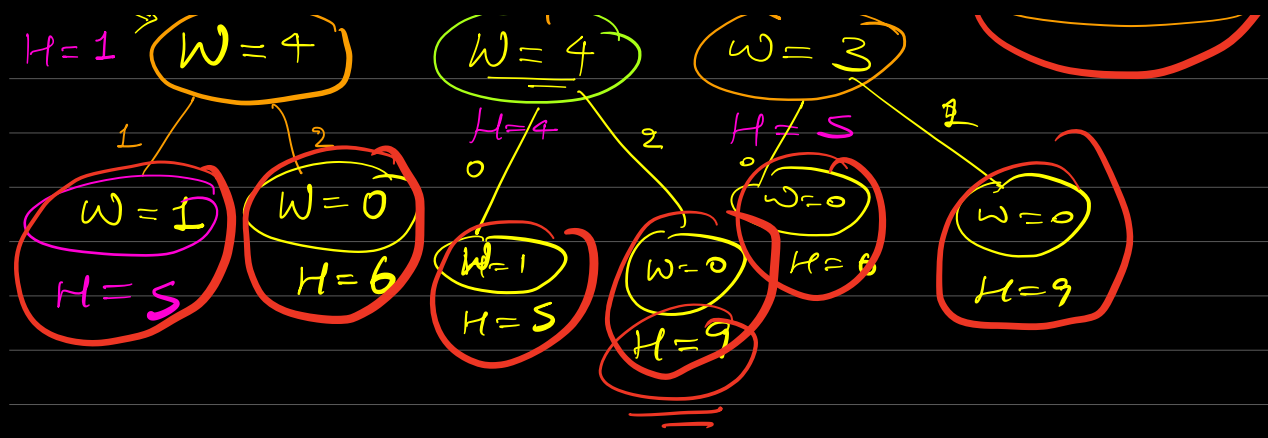
H = 0

max ×

0 1 2 3

H = 8

W = 2



1) At any given point, you might have multiple choices (Element of Choice)

2) If you can select the best choice available & that gives you an overall optimal solution

⇒ Greedy

But if greedy is not the correct solution you need to check for all Elements of choice

⇒ Recursion

Boomer

⇒ Applying greedy is not difficult

But choosing if greedy can be applied or not is difficult.

Cake

(fraction not allowed)

1) H ⇒ [1, 4, 5, 50]
W ⇒ [3, 3, 4, 5]
0.3 1.3 1.25 10

W = 7

H ⇒ [1, 4, 5, 8]
W ⇒ [3, 3, 4, 5]
0.3 1.3 1.2 1.6

(negative case)

W = 7

Lakshmi (CTO) [famous]

BMW showroom called Lakshmi

Special sale

N cars.

A[i] ⇒ Time till which the ith car is free of cost (sale is active)

B[i] ⇒ Beauty of the car.

V O

Maximize the beauty

⇒ It takes 1 unit of time to purchase a car

$T \le 2$

eg

	1	2	3	4	5
A ⇒	3,	1,	3,	2,	3
B ⇒	6,	5,	3,	1,	9

i^{th} car can be purchased till $T = A[i]-1$

15 ✗	T = 0	5	
9 ✗	⇒ T = 1	6	= 20
18 ✗	⇒ T = 2	9	
14 ✗	T = 3	✗	
29 ✗	⋮		

A ⇒ [1, 5, 5, 5, 3, 3, 3]

B ⇒ [5, 4, 3, 8, 20, 7, 10]

27

⇒ Sort on the basis of A

1 2 3 4 5 6 7

$A \Rightarrow [1, 3, 3, 3, 5, 5, 5]$

$B \Rightarrow [5, 10, 7, 20, 4, 3, 8]$

$T = 0$ ~~1 (5)~~ 3 (7)

$T = 1$ 4 (20)

$T = 2$ 2 (10)

$\Rightarrow T = 2$

	1	2	3	4	5	6	7
A ⇒	1,	3,	3,	3,	5,	5,	5
B ⇒	5,	10,	7,	20,	4,	3,	8

Min Heap

$\log N$

, 10, 7,
20, 4, 3
8.

✓	✓	~~T = 0~~	= 5
✓	✓	~~T = 1~~	= 10
✓	✓	~~T = 2~~	= 7
✓	⇒	~~T = 3~~	= 4
✓		~~T = 4~~	= ~~3~~ 8
		T = 5	

20

size of arr = $O(N)$

$$T.C. = O(N \log N + N \log N)$$

$$= O(N \log N)$$

S.C = $O(N)$

Break Try to code

Code

A , B , N ⇒ inputs

T = 0 , Heap h ⇒ Global

sum = 0;

Sort (A, B) ⇒ H.W.

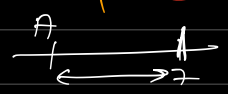
```
for (i=0; i<N; i++) {
        if (A[i] > T) {
                h.insert(B[i]); sum+=B[i];
                T++;
        }
        else {
                minBeauty = h.root();
                if (B[i] > minBeauty) {
                        h.deletemin(); sum-=min_Be
                        h.insert(B[i]); sum+=B[i];
                }
        }
}
```

Q Given N jobs ⇒ Each job has a start time & an end time.

Paypal assessment

	A	B	C	D
Start ⇒	1	5	7	1
End ⇒	7	8	8	8

= 2

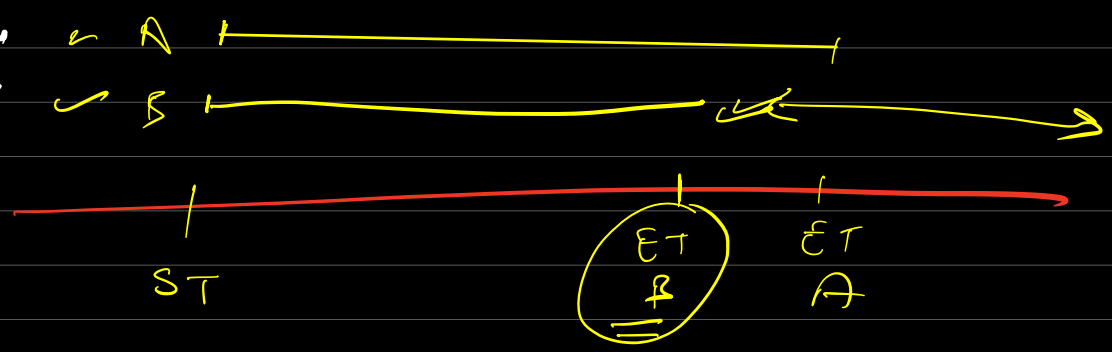


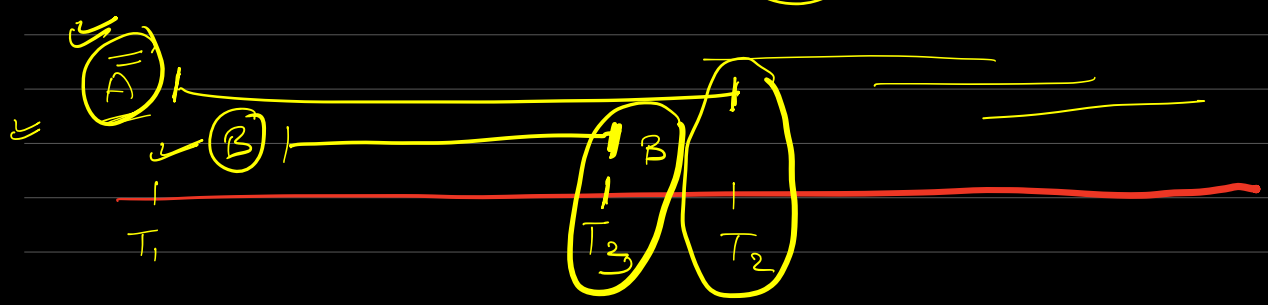
Goal ⇒ Perform max jobs possible

Constraint ⇒ Only perform 1 job at a time.

⇒ If a job ends at a time T, you can start another job at the same time.

Sol^n





Goal ⇒ Select a job with lesser end time

	A	B	C
S	3	1	6
E	8	5	8

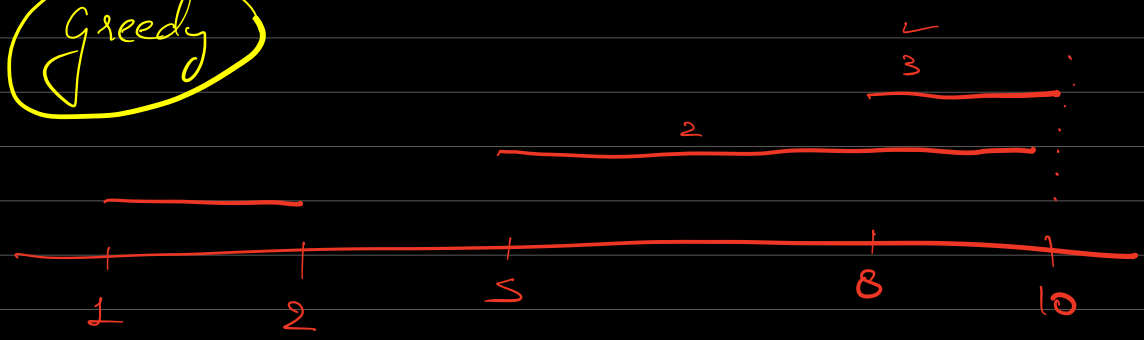
⇒ Start time is not imp.

⇒ We select the job with the min end time

1) Sort the jobs on the basis of end time

	1	2	3	4	5	6
S ⇒	1	5	8	7	15	12
E ⇒	2	10	10	11	19	20

Greedy



2 × 2

Code ↑ ↓

$T.C. = O(N \log N + N)$

$= O(N \log N)$

$S.C. = ???$

Q N children

Every child has a rating

$A \Rightarrow [\overset{1}{1}, \overset{2}{5}, \overset{3}{2}, \overset{4}{1}]$ $N = 4$

$1 + 3 + 2 + 1 = 7$

1) Every child should have atleast one candy

2) If the ith person has more ratings than the neighbours, then he/she should also have more chocolates

if $(A[i] > A[i+1]) \Rightarrow C[i] > C[i+1]$

if $(A[i] > A[i-1]) \Rightarrow C[i] > C[i-1]$

Find the min no. of candies that would be required

Sol^n

1) Sorting solution

	1	2	3	4	5	6	7	8	9
R ⇒	1,	6,	3,	1,	10,	12,	20,	5,	2
Ans ⇒	1	3	2	1	2	3	4	2,	1

if (A[i] > A[i+1]) {

C[i] > C[i+1] {

C[i] = C[i+1] + 1

1) Sort the children on the basis of ratings (index of the element in the original array);

R' =	1,	1,	2,	3,	5,	6,	10,	12,	20
	1	4	9	3	8	2	5	6	7
⇒	1	1	1	2					

Map <index, chocolate>

⇓

Array

② Observations

	1	2	3	4	5	6	7	8	9
A ⇒	1,	6,	3,	1,	10,	12,	20,	5,	2
G1 (L To R) →	1	2	1	1	2	3	4	1	1
G2 (R to L) ←	1	3	2	1	1	1	3	2	1
max	1	3							

Greedy

T.C. = $O(N+N) = O(2N)$

S.C. = $O(N)$