Page :
Date : / /

Day - 223

Greedy Algorithm

* **Greedy Algorithm:**

⇒ It is a problem solving approach that makes a sequence of choice, each of which looks best at that moment.

⇒ This methods aim to find global optimal solution by making series of locally optimal choice.

⇒ Here, we do smaller choices that leads to the solution of our bigger problem.

⇒ If ~~the~~ we have to find min or max in the question then it's a hint that here we can use greedy.

* **Min no. of coins:**

⇒ Here, we are given some coins i.e. 2000, 500, 200, 100, 50, 20, 10, 5, 2, 1

⇒ N = 143, we have to make 143.
100 + 20 + 20 + 3 + 1

Page:
Date: / /

⇒ Here, we will use greedy & find the max that is near to 143 the subtract that coin value from 143.

⇒ And again apply same process to the remaining amount.

→ 143 → 43 → 23 → 3 → 1

→ 100 → 20 → 20 → 2 → 1

⇒ Greedy don't work every where.

⇒ We will divide the no. to check that coin can be used or not.

* Shop in candy store:

5	3	2	1	4

N = 5 , k = 2

⇒ We have to buy all the candies and the price are given in the array.

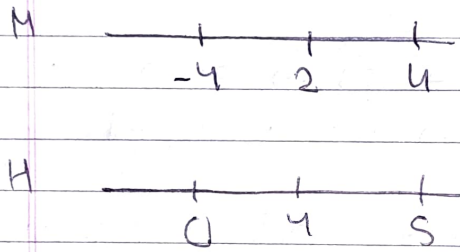
⇒ We also have an extra condition that if we buy a candy, we can buy extra k candy for free.

⇒ So, we will become greedy here, we will buy the min cost candy & for free, we will ~~but~~ take max cost candy.

Page.
Date / /

* Assign mice to holes:

M =	4	-4	2
H =	4	0	5



⇒ So, we ~~~~ have to put mouse in the hole & return the ~~max~~ time taken.

⇒ So, time will be calculated as M[i] - H[i].

⇒ we have to return the min time. So, we will sort the arrays & ~~de~~ calculate the time.

* Min Rotation to unlock a circular Lock:

R = 2 2 2
D = 3 3 3

⇒ we can rotate clockwise or anticlockwise.
⇒ So, we will do this —

min(3-2, 10-2) = min(1, 8)
= 1

Page:
Date: / /

* N - meeting in one Room:

Start → | 1 | 3 | 0 | 5 | 8 | 5 |

End → | 2 | 4 | 6 | 7 | 9 | 9 |

⇒ We can organize meetings in such a way that end time of previous meeting should be smaller than the start time of the next meeting.

⇒ We can't take start time to find this.

=| But we can end time to solve this problem.

=| So, we will using the approach which meeting will end first.

=| So, we will sort on the basis of end array.

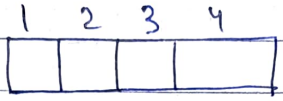
* Job Sequencing Problem:

Job id	Deadline	Profit
1	4	20
2	1	10
3	1	40
4	1	30

1 task = 1 unit of time

⇒ So, we have to maximize the profit.

⇒ We will make a deadline array to check if we have space on that to do that task.



⇒ We will select the max. profit job first.

T.C. → $O(n^2)$

⇒ Now, how to do this $O(n \log n)$ T.C.

⇒ For that we will use ~~an p~~ a parent array that store the value of available space.

⇒ Also, we use recursion to update the values.

* Single Threaded CPU:

Enqueue time →	1	2	→ Processing time
	2	4	
	3	2	
	4	1	

⇒ If more than one task are available at a given time then CPU will

Page:
Date: / /

select the task that have least processing time.

⇒ we have to print the order of tasks.

=⇒ we will start from timer = 0.

⇒ When there is no task available at a given time then CPU will do nothing.

⇒ So, when we will ~~match~~ get any task, we will execute it & increase the timer value.

⇒ We will take a min heap & before that we will sort the array based on enqueue time.

→ After that start the timer value with first task.

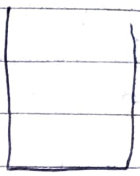
=⇒ Then push the tasks of that timer value in the min heap.

=⇒ And select the min processing time task. & increase the timer value.

able =⇒ If the queue is empty then init the value of timer to the next task.

* Fractional knapsack:

	0	1	2
value:	60	100	120
Weight:	10	20	30



w = 50 (max)

=> We have to put the ~~value~~ items in knapsacks in this manner that the value will be max.

=> Also, only 'w' amount of ~~item~~ weights equivalent item can come.

=> Also, we can put items in fractional value.

=> First, we get the value by weight value and become greedy for this.

=> Select the max. value by weight item.

=> Sort on the basis of value by weight.

* Non-overlapping Intervals:

Page:
Date: / /

⇒ [[1,2], [2,3], [3,4], [1,3]]

⇒ end <= start → non-overlapping interval
[1,3] [4,6]

⇒ Here, we use the end time.
⇒ we will select the interval that have least end time and do this process and remove the overlapping interval & increase the count value.

* Insert Interval:

[[1,3], [6,9], [10,20]] | [2,5]

→ overlapping then make them one

⇒ [[1,3], [2,5]]

=> [[1,5], [6,9], [10,20]]

⇒ If we get any overlapping interval then mix them.

* Task Scheduler:

Page :
Date : / /

① A A A B B C , n = 2

⇒ If we schedule a task suppose A then we can't again schedule the same task A until 2 seconds ~~get over.~~ completed.

⇒ But we can execute other task in between them.

⇒ We have to return the min time required to execute all the tasks.

⇒ We will first execute the max frequency task first.

⇒ So, → A B C A B _ A

⇒ A B C A B _ A → 7 cycles

⇒ (count - 1) * (n + 1) + 1

max frequency

⇒ But, A A A B B B

A B _ A B _ A B → 8

But → (3-1) * (2+1) + 1 = 7

Page:
Date: / /

⇒ We will add the no. of task that have same frequency i.e. max. frequency task.

⇒ A A A A B B B B C C C D D D E , n = 2
4 4 3 3 1 → 15

⇒ A B C A B C A B C A B

⇒ But now D & E are remaining.

⇒ A B C D E A B C D A B C D A B → 15

⇒ (count - 1) * (n + 1)

⇒ 3 X 3 = 9 + 2 = 11 < 15

⇒ So our answer will be max(11, 15) → 15.

* Huffman Coding:

⇒ message: aa bc dae fabc ....

a = 5
b = 9
c = 12
d = 13
e = 16
f = 45

Page
Date / /

⇒ It is a very important compression algorithm.

⇒ Message will be send in this format –
01100001 ≠ 01100001 ...
one char

⇒ So, ~~that~~ = total size will be –
800 bits

⇒ Usually, we follow ASCII values.
⇒ But here, we will create our own.

⇒ So, we will given six diff. chars.
⇒ To represent these six diff. chars, we will require 3 bits.

01100001 ⇐

a →	000
b →	001
c →	010
d →	011
e →	100
f →	101

→ table

⇒ So, now total size will be 300 bits.

Page:
Date: / /

ion

at —

ues,
own.

ars.
ars.

0 bits.

⇒ we will also send a table to the receivers side so that receiver will identify which type of conversion we have used.

⇒ So ~~what wil~~ table size will be —
$6 \times 8 + 6 \times 3 = 48 + 18 = 66$

⇒ Total size = $300 + 66 = 366$

⇒ So can we further reduce the size?

⇒ We know that 'f' came 45 times so if we denote it by least bit then we can decrease the size more.

⇒ $\$ \leftarrow a \rightarrow 101 = 15$
$9 \leftarrow b \rightarrow 100 = 27$
$12 \leftarrow c \rightarrow 11 = 24$
$13 \leftarrow d \rightarrow 10 = 26$
$16 \leftarrow e \rightarrow 1 = 16$
$45 \leftarrow f \rightarrow 0 = +45$
153
$+60$
213

Table
$= 6 \times 8 + 12$
$= 60$

Page
Date / /

⇒ There is a problem in this —

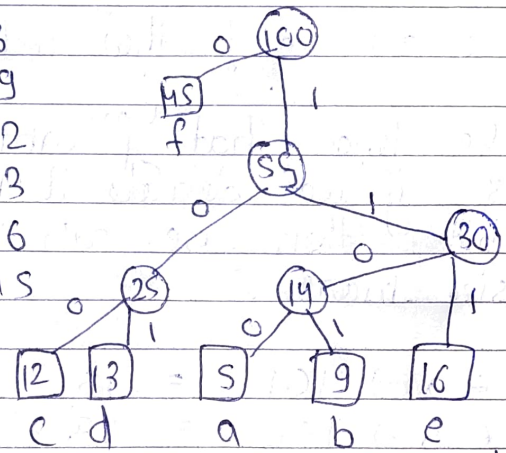
⇒ 11 or 11

e e c

⇒ It can be ee or c.

⇒ So, how Huff man solve this problem?

a = 5
b = 9
c = 12
d = 13
e = 16
f = 45

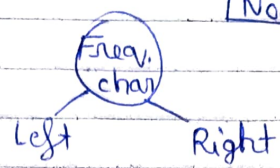


⇒	5 ← a =	1100	→	20
⇒	9 ← b =	1101	→	36
⇒	12 ← c =	100	→	36
⇒	13 ← d =	101	→	39
⇒	16 ← e =	111	→	48
⇒	45 ← f =	0	→	45
				224
				+ 66
				290

Table
= 6x8 + 18
= 66

Page:
Date: / /

⇒ Node



⇒ If there is no char then use '$'.

⇒ This is a greedy algo ~~questi~~ problem because we are combining two min.

⇒ Also we will use min-heap.

Table
x8 +18
66