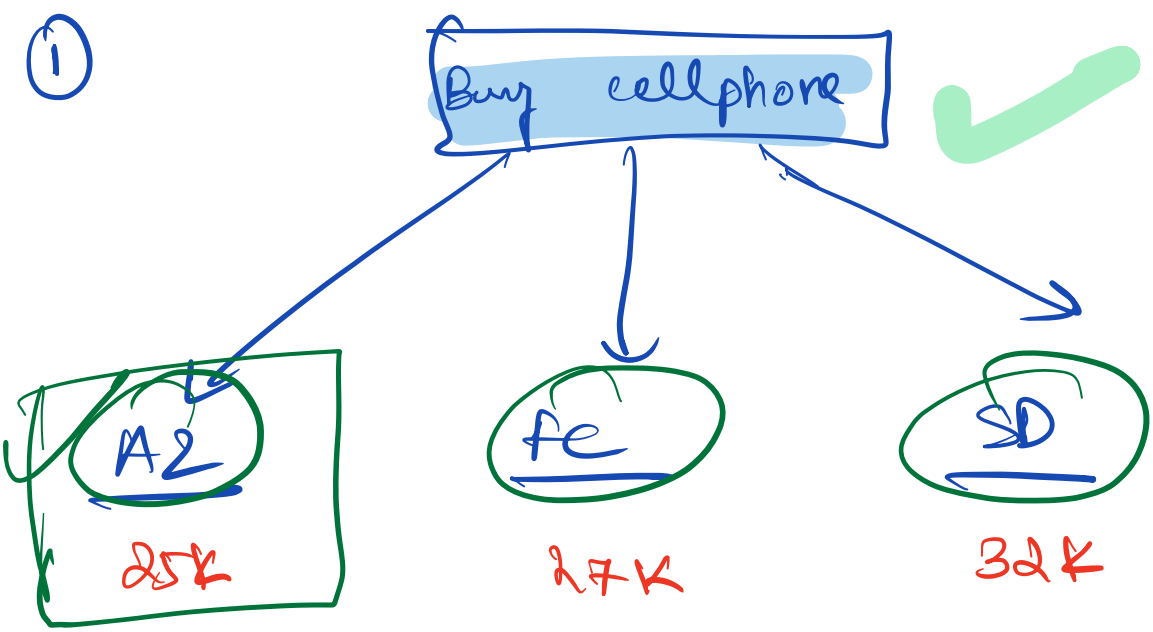
"Hello Everyone!"

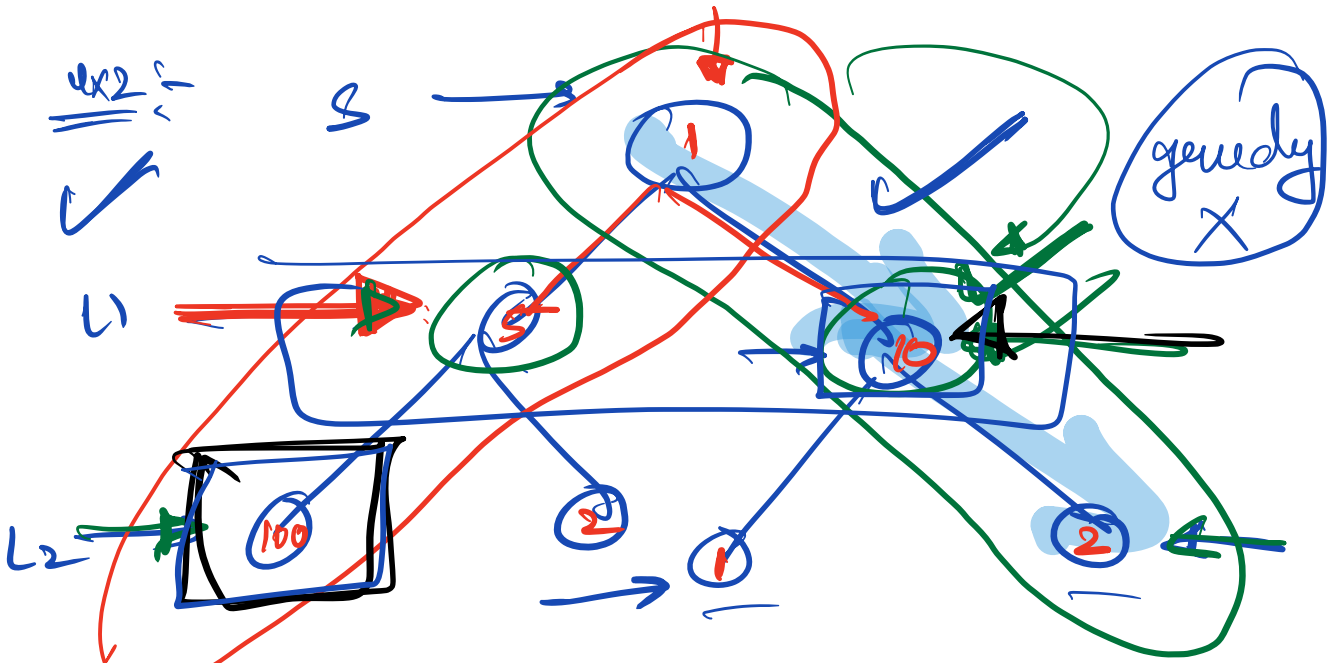
GREEDY

Disclaimer → ① Example ✓

② Easiest → Toughest



Goal:- Minimise the cost!



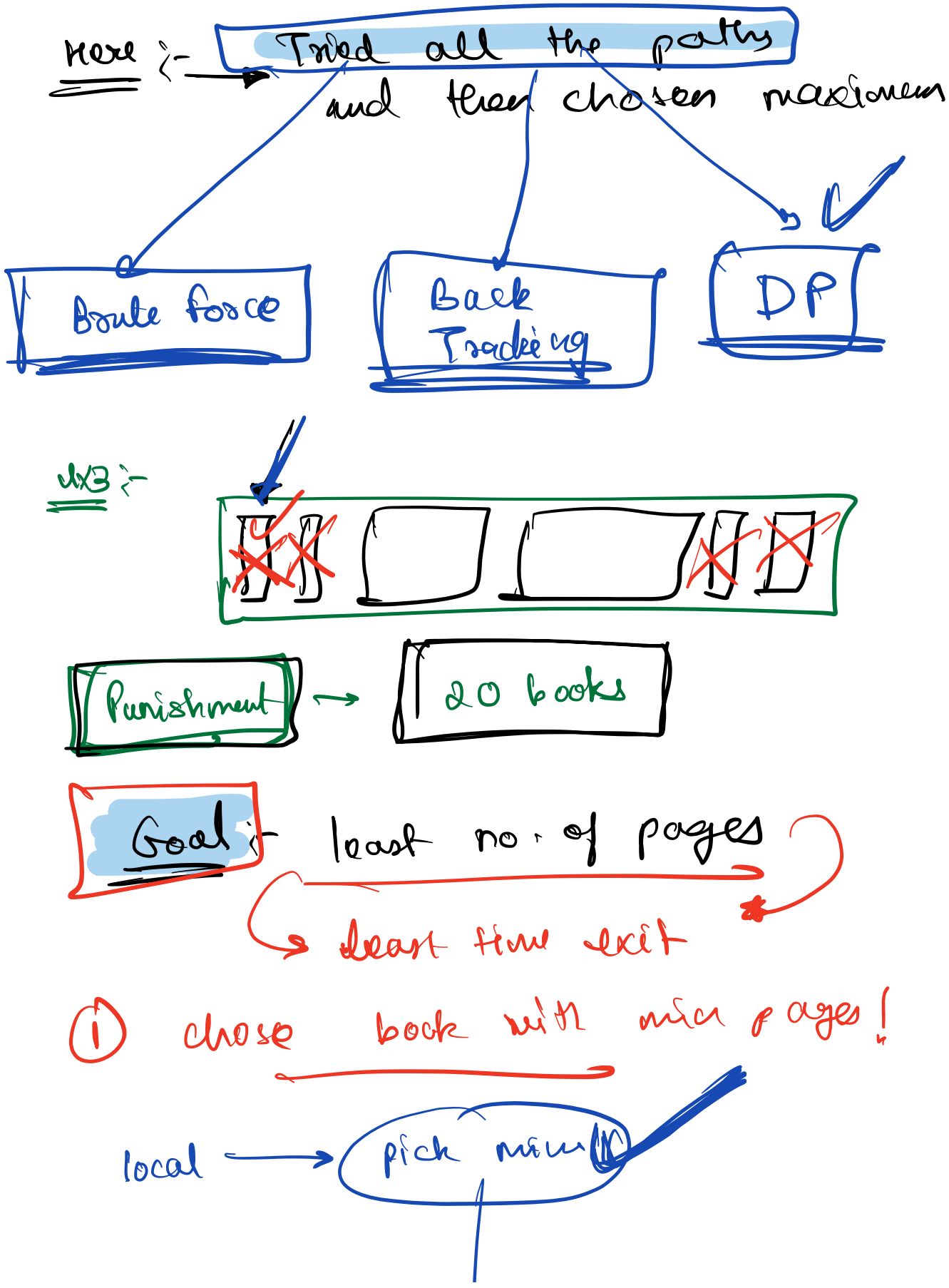
Total gold → 1 + 10 + 2 → 13

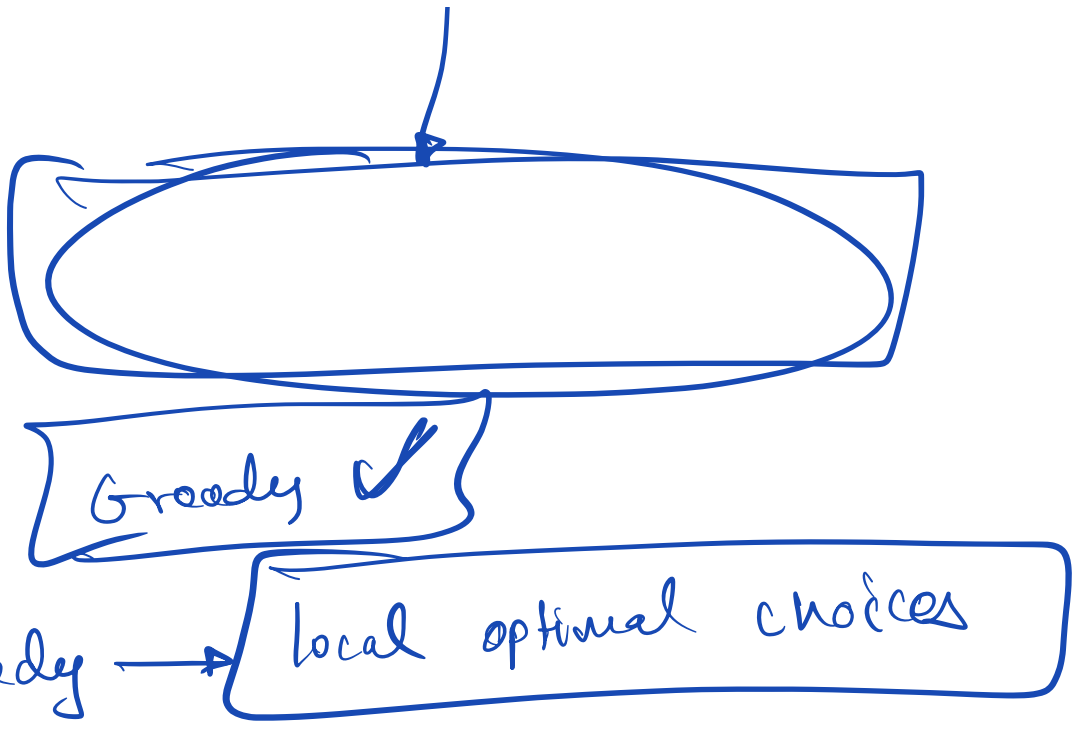
Goal :- Collect Max Gold coins.

1 + 5 + 100 → 106 ✓

Greedy → At each step, You pick the most valid optn available

→ local maximum


Here :-
Tried all the paths and then chosen maximum
Brute force
Back Tracking
DP
Ex3 :-
Punishment
20 books
Goal :- least no. of pages
least time exit
① chose book with min pages!
local
pick min

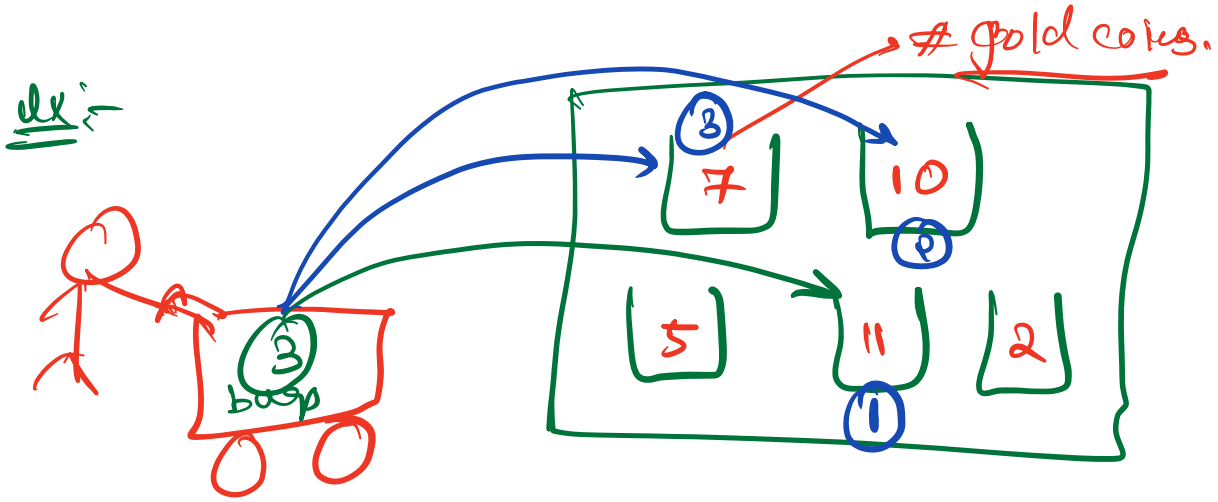


Imp obs :- Greedy strategies might NOT always lead to most optimal solution.

↳ Gold coin ex

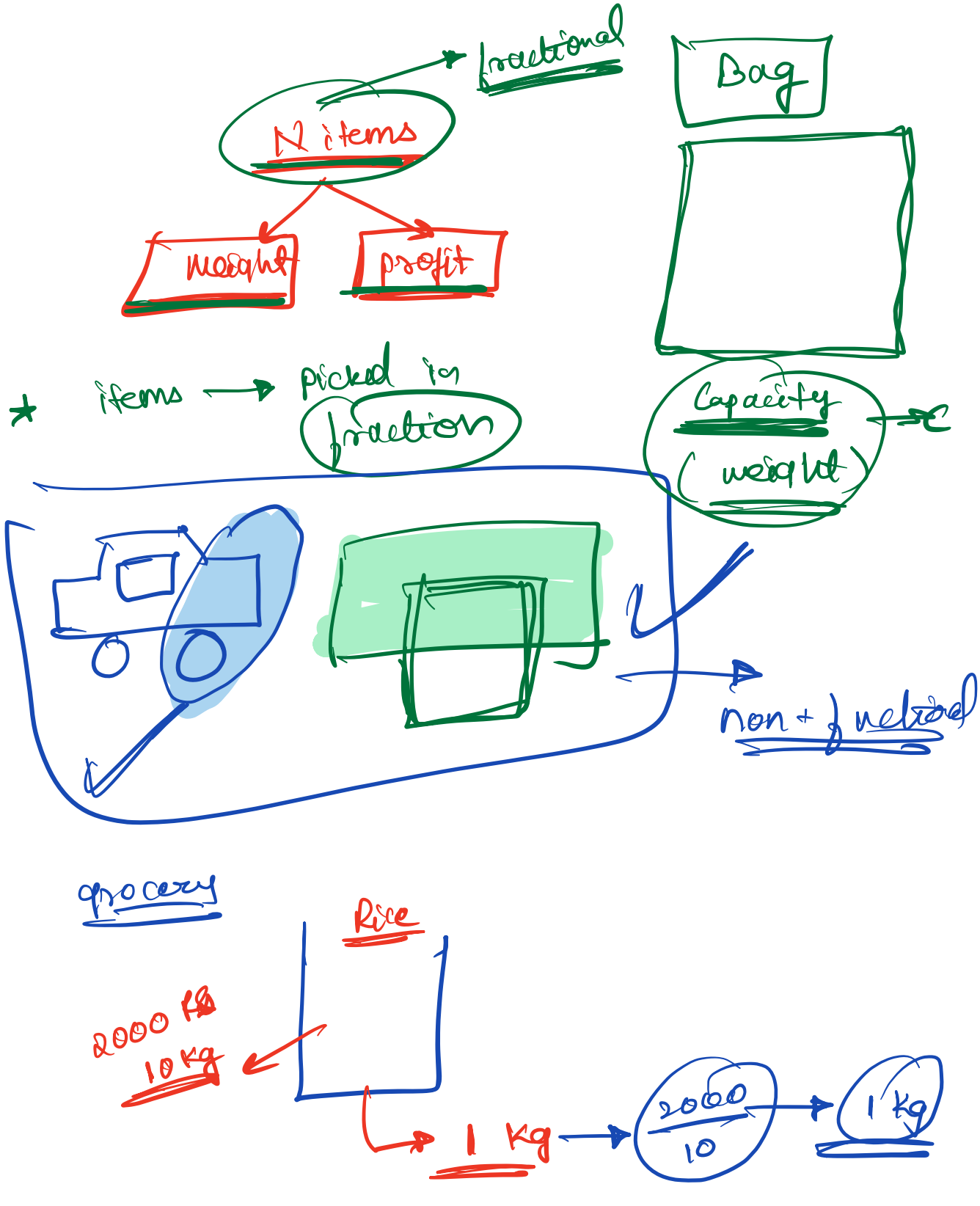
Obs :-

Makes a choice at every step without considering future outcomes.



greedy → Pick max among avail

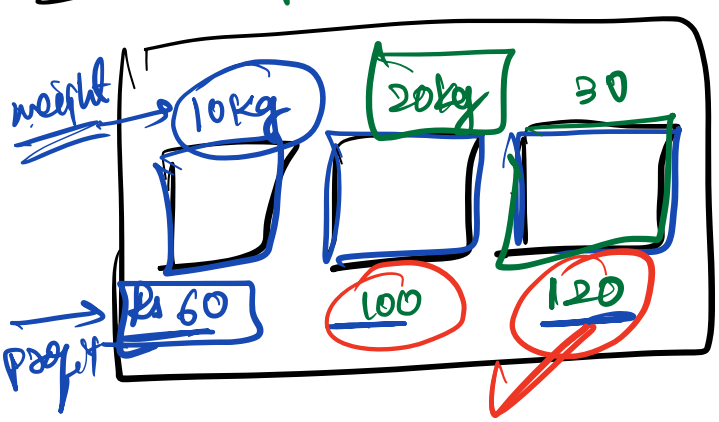
① Fractional Knapsack



Goal: Maximize Profit

→ Pick some items such that profit is maximized.

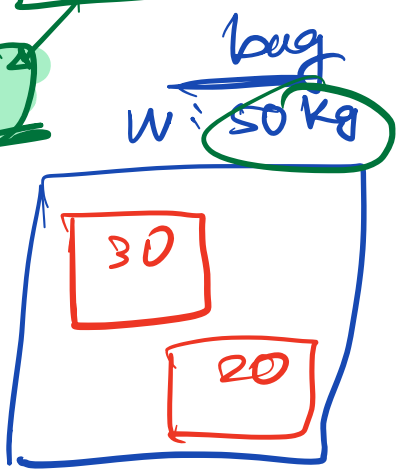
Ex 1:-



App 1:- I1 → 6 ✓
I2 → 50 ✓
I3 → 4

W: 10 + 20 + 20
P: 60 + 100 + 20*4

240



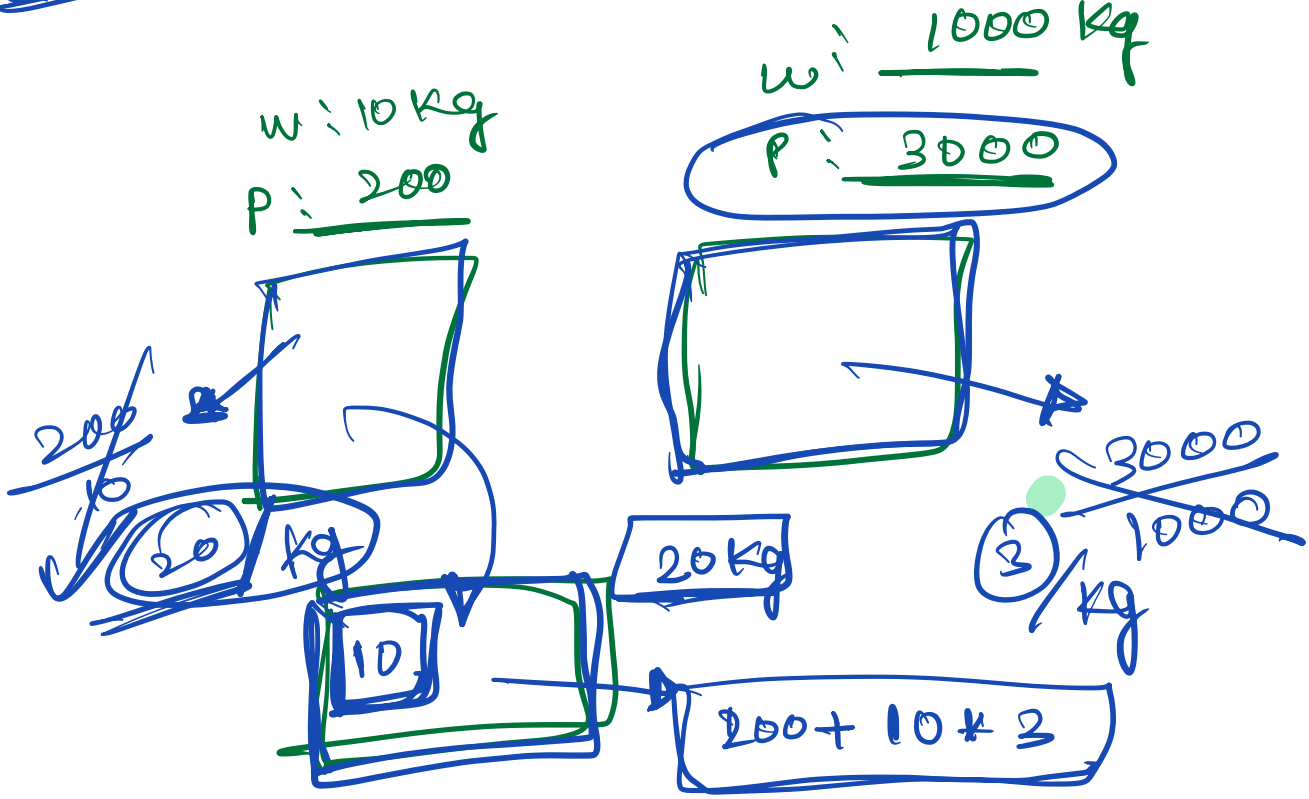
Max:-

220
240

Profit :- 100 + 120
220

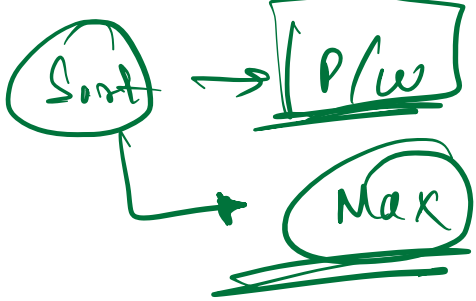
App 1 :- Pick item with Max Profit

Ap2 :- Check the item with max P/w first



* Calc per unit Profit

Impl :-

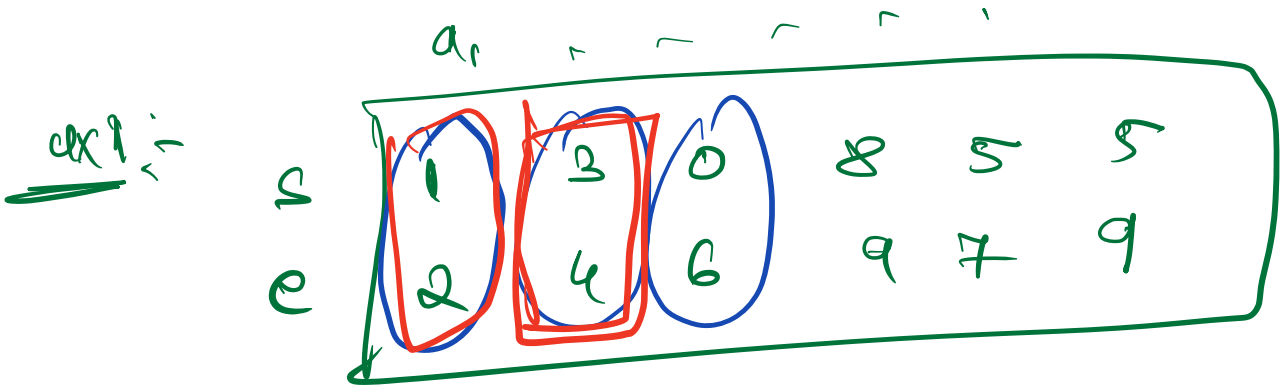


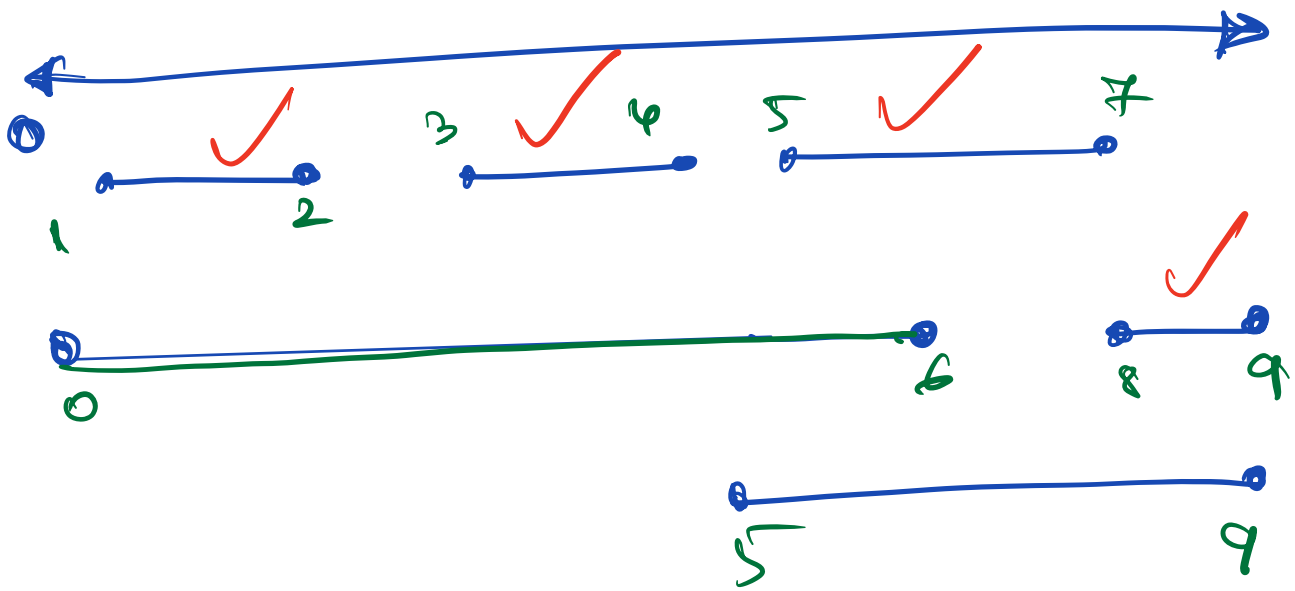
* Comparator

(9.2) Activity Selection Problem

N activities
↳ [s, e]

At a time, you can perform 1 activity
find the max number of activities
that you can perform?

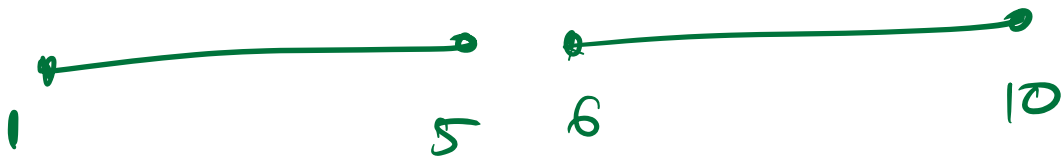




Ans : 4

App 1 : Pick the activity with least duration

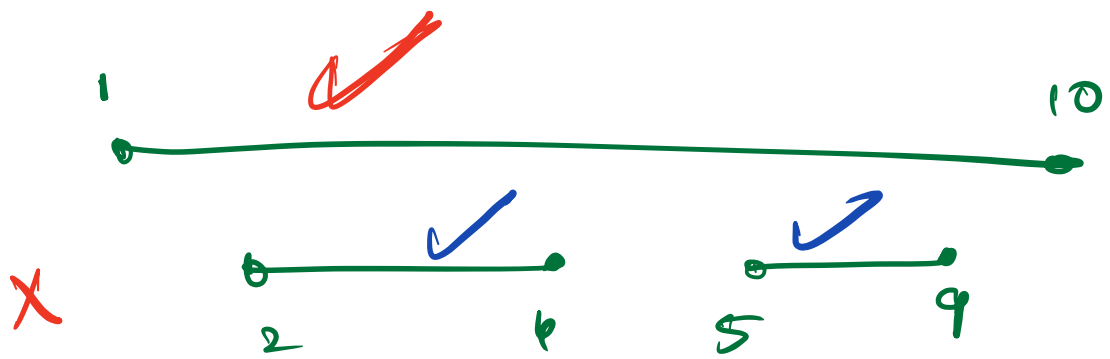
(1)



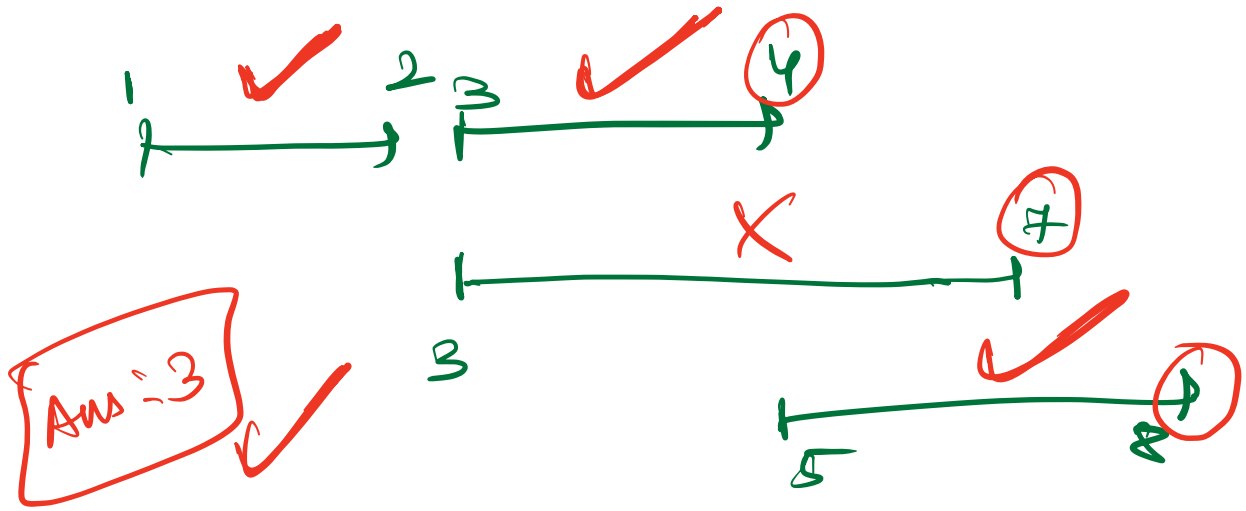
Proof contradiction

App 2 : Pick the activity starting earlier (smaller ST)

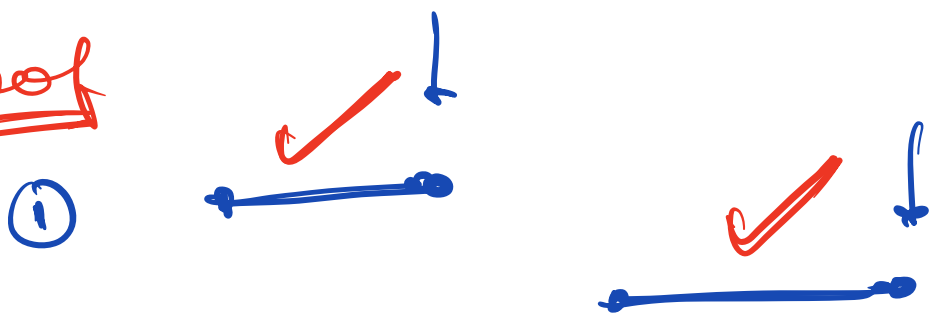
X

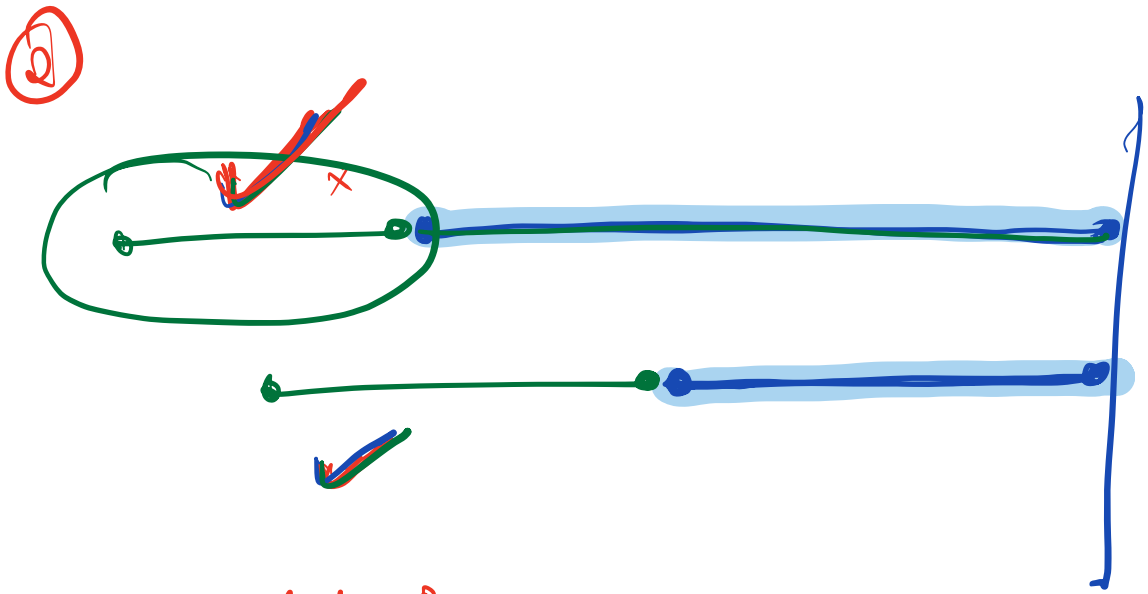


Ans :- Pick the one with Smallest ET.



Proof



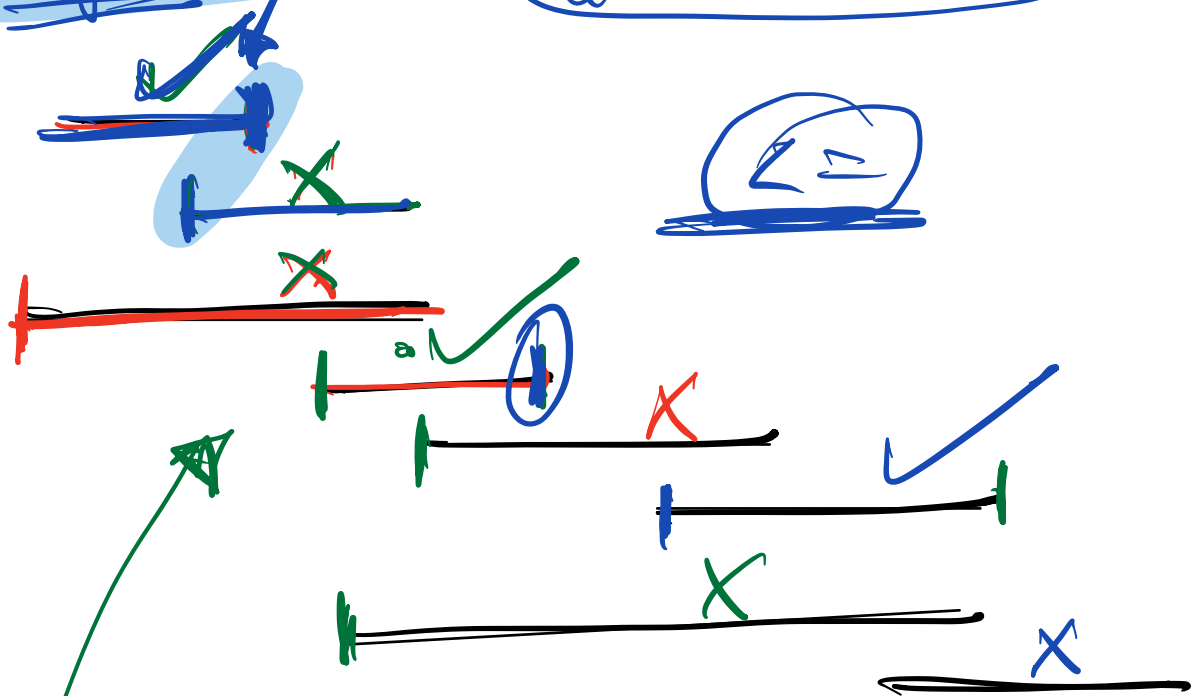


Implementation

→ keep a track of the ET of the last chosen activity.

Dry Run :-

last_end_time



Ans : 3

Sorting on end time

and maintain the end time of last selected activity.

check curr_start >= last end time

no : of activity

TC : O(n log n)

SC : O(1)

10:37 → 10:43

③ Job Scheduling (Engineering !)

(Q) N Jobs → Reward
N Jobs → Deadline

① At a time, a single job can be performed.

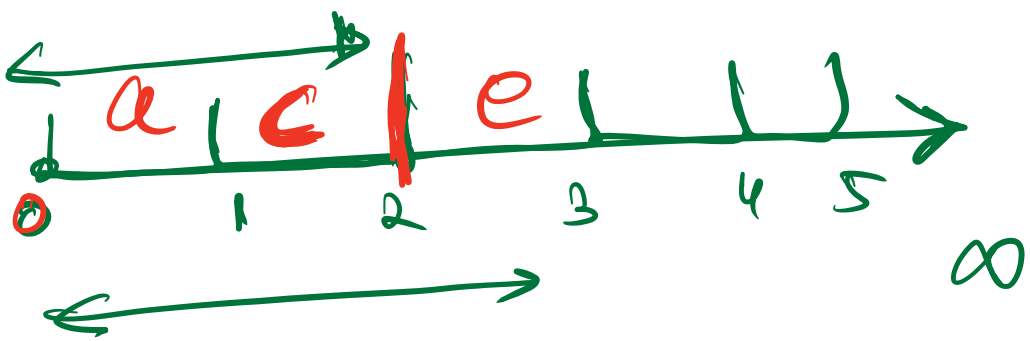
*② A Job takes 1 unit of time to complete.

**③ Reward → only when done within deadline.

→ Max Reward you can get !

Ex1 :

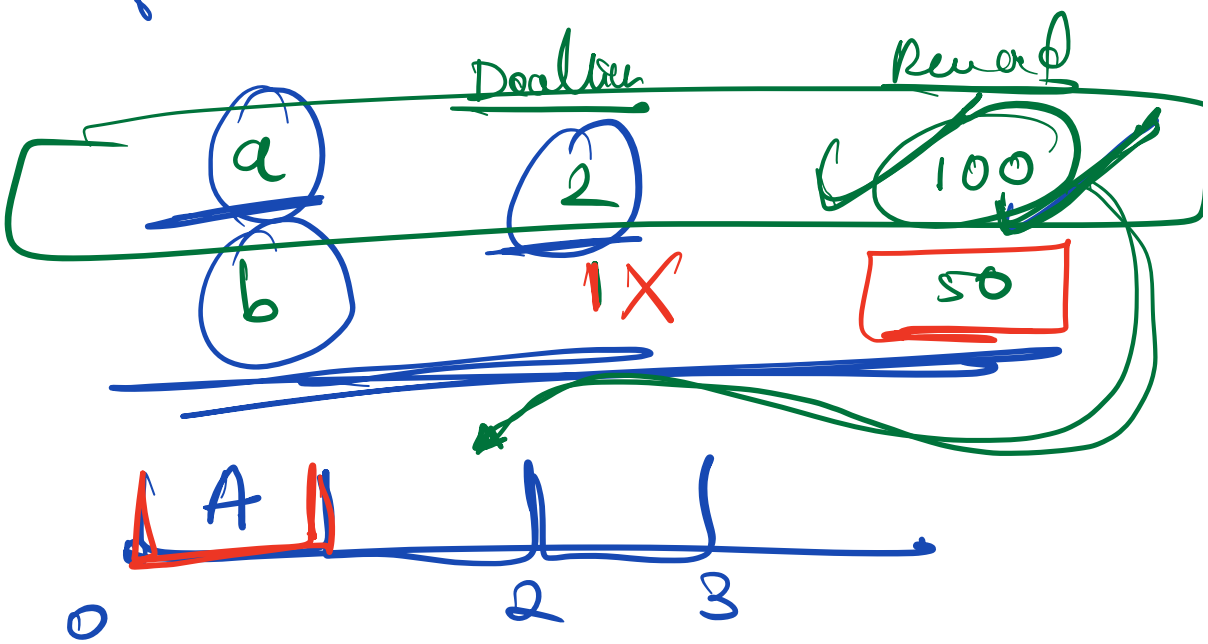
Job	Deadline	Reward
A	3	100
B	1	15
C	2	27
D	3	25
E		30

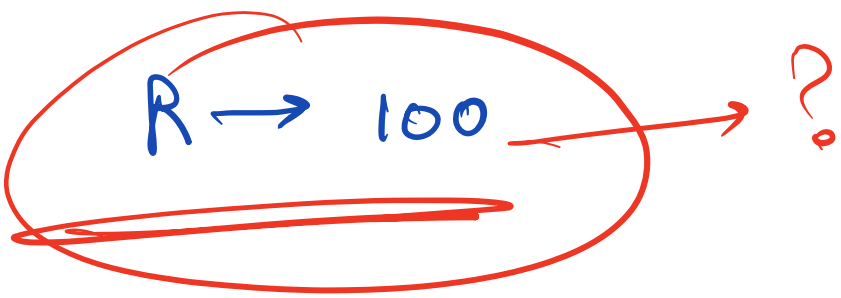


100 + 27 + 30

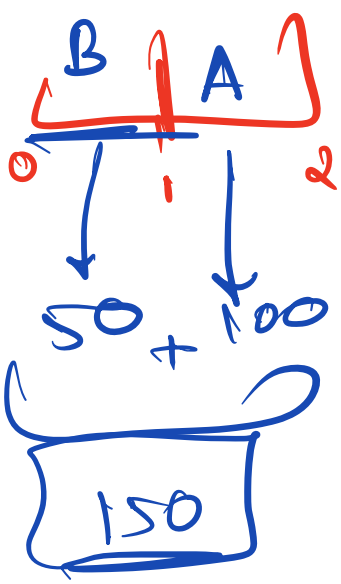
157

① Pick Job with highest reward first.





MAX = 150



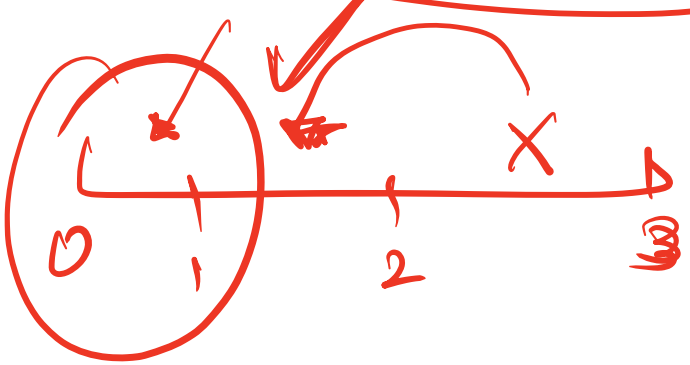
★ Engineering Mindset!

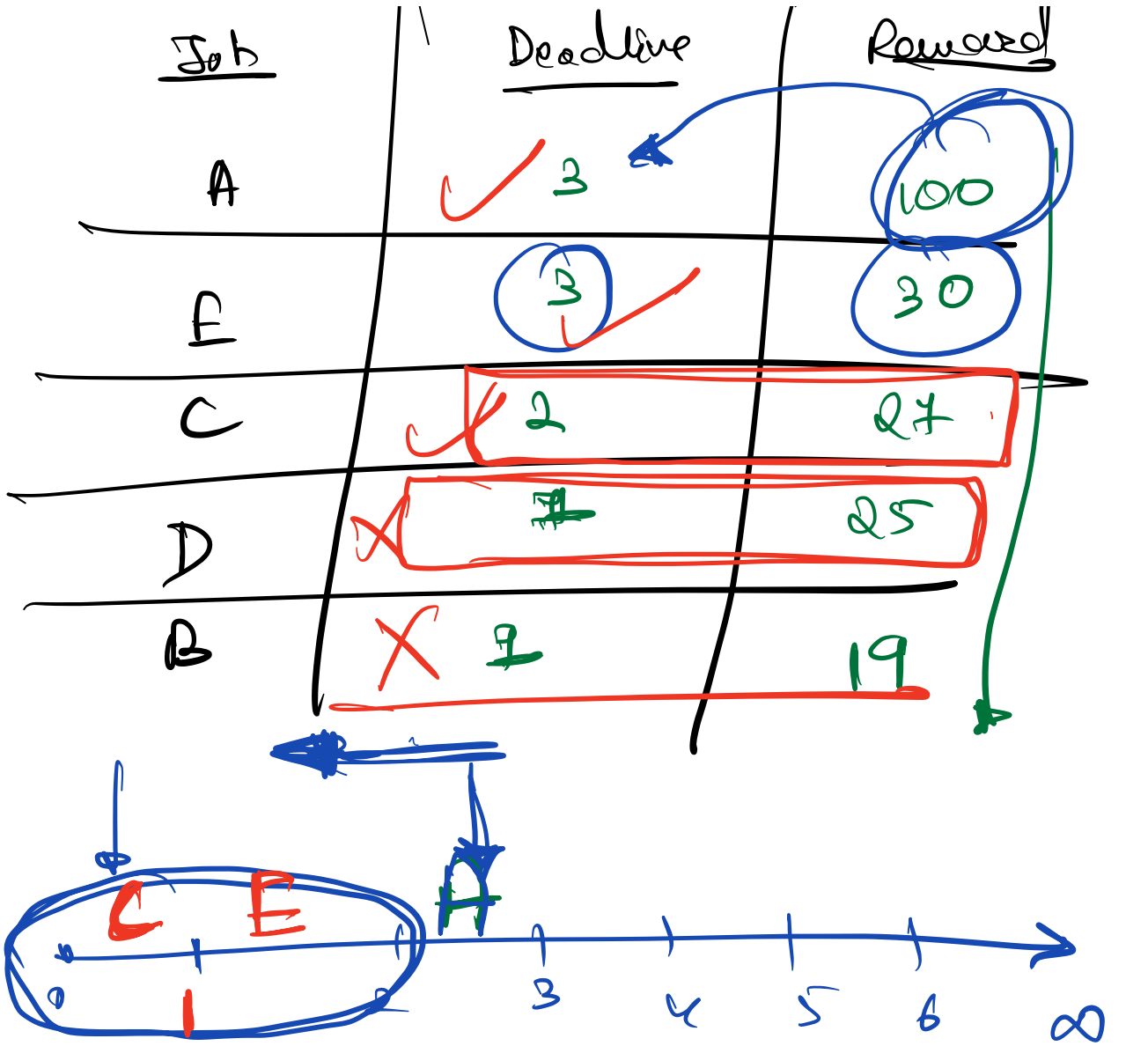
→ once decided to schedule a job on the basis of reward

Schedule it to the farthest possible time?

OR

Closest to its deadline.



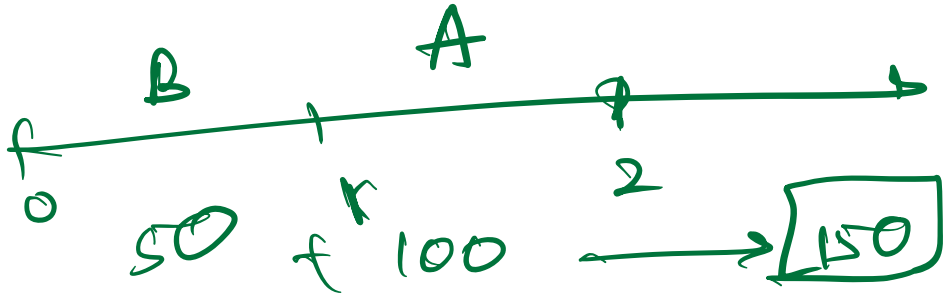


Reward = 100 + 30 + 27

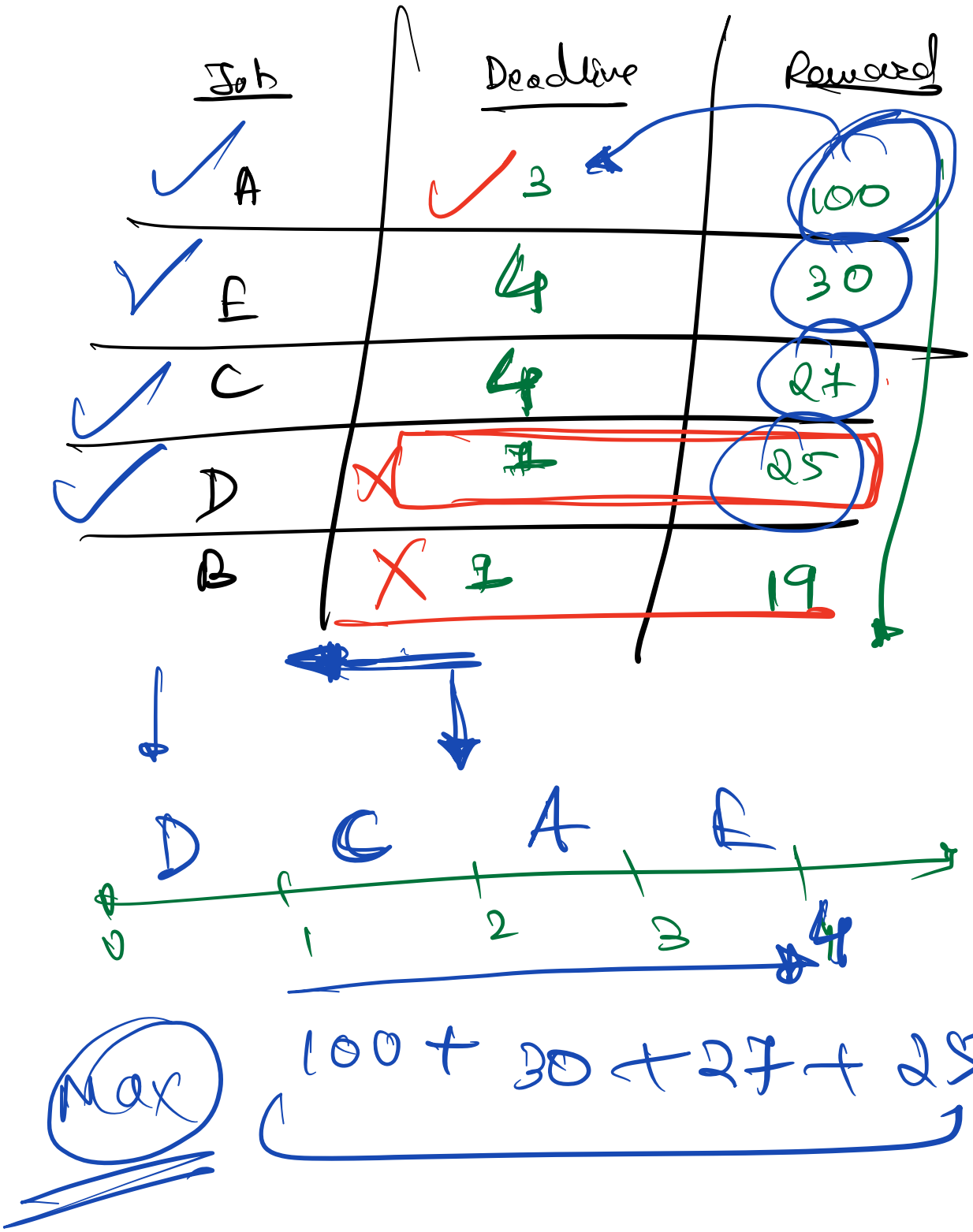
157

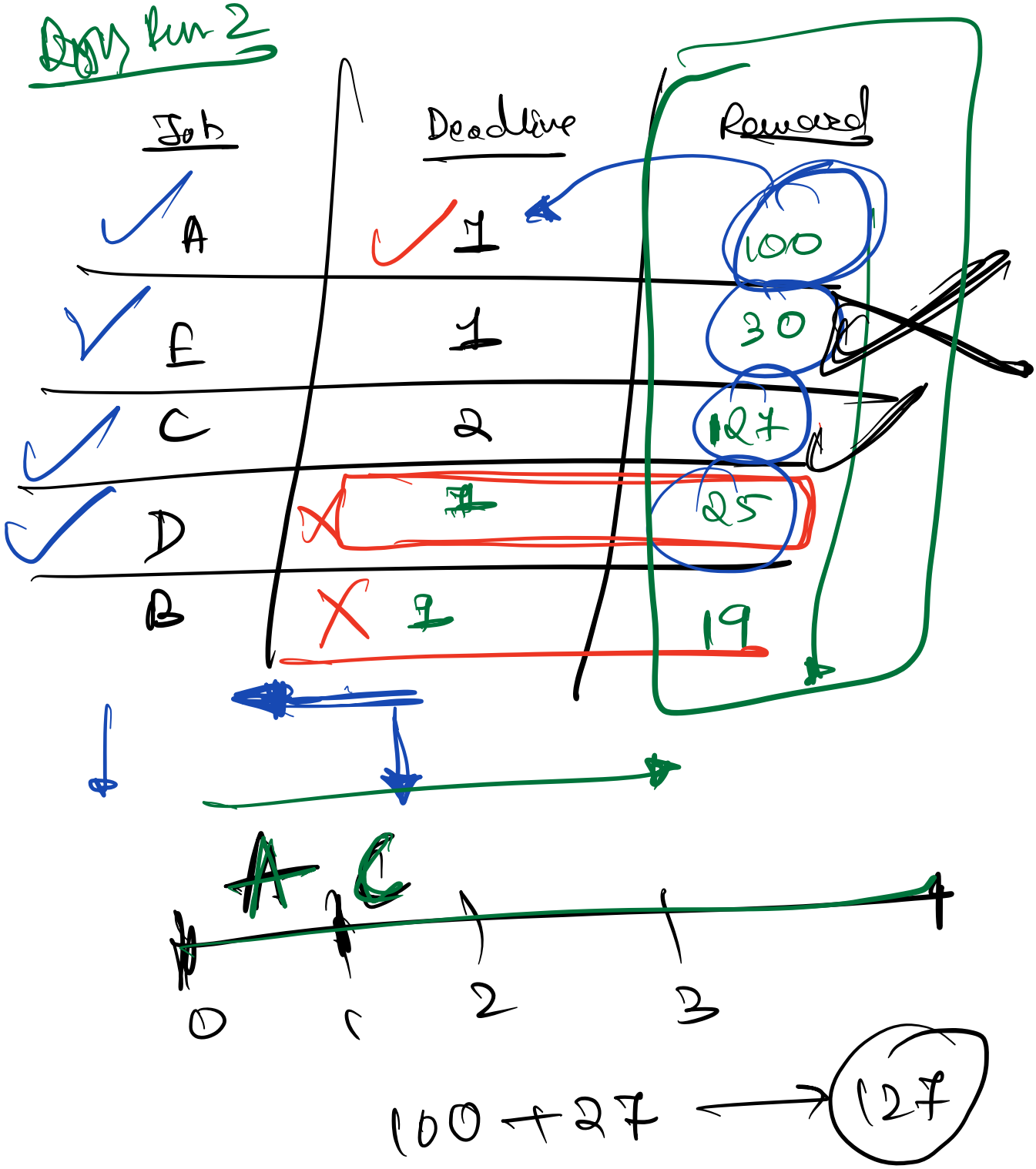
Dry Run 2 :-

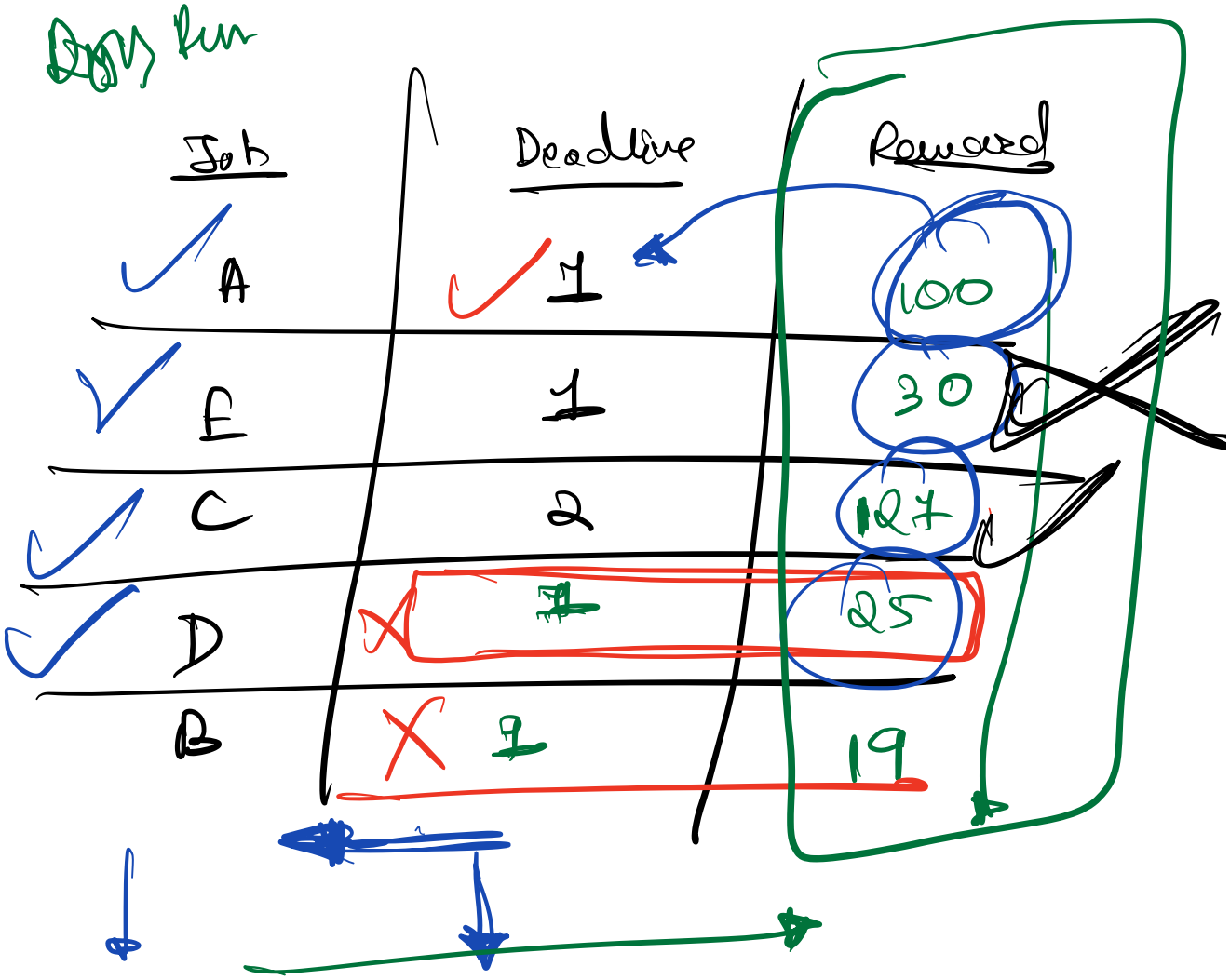
a	2	100 ✓
b	1	50



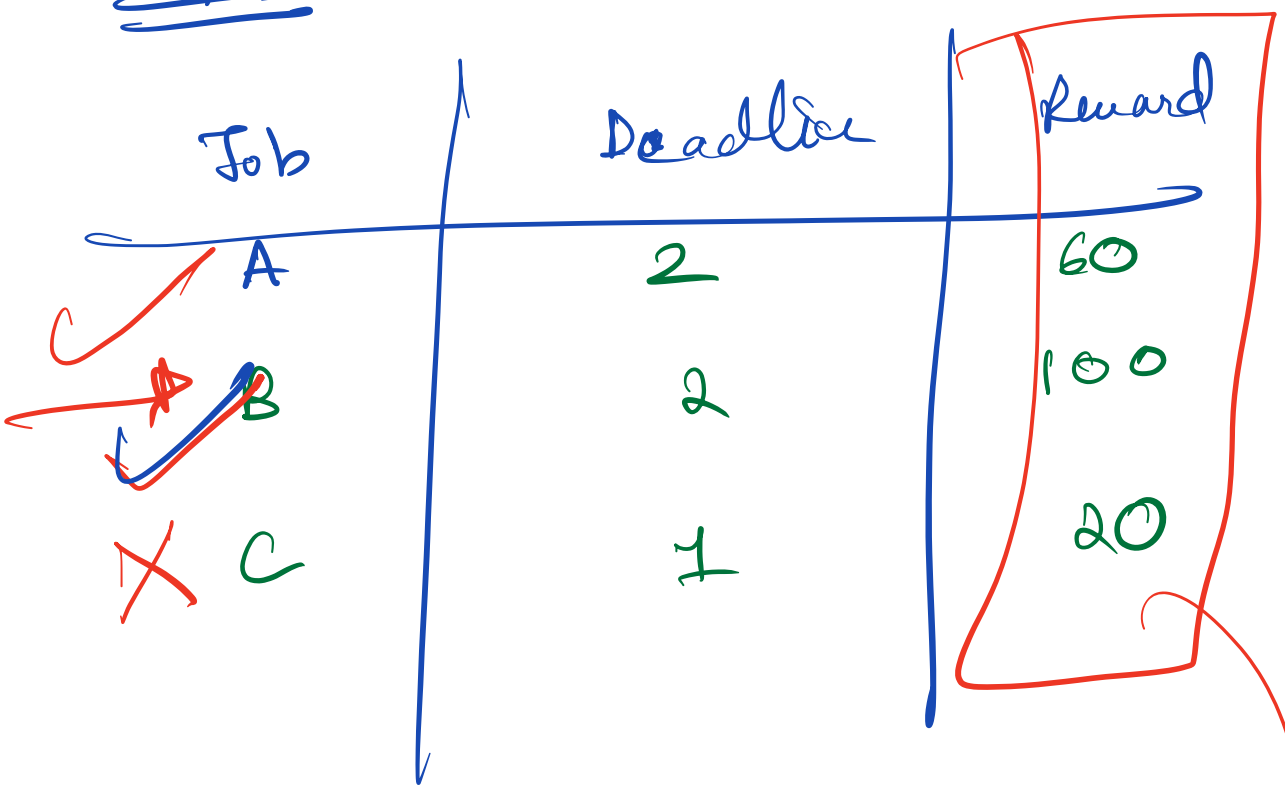
50 f 100 → 150

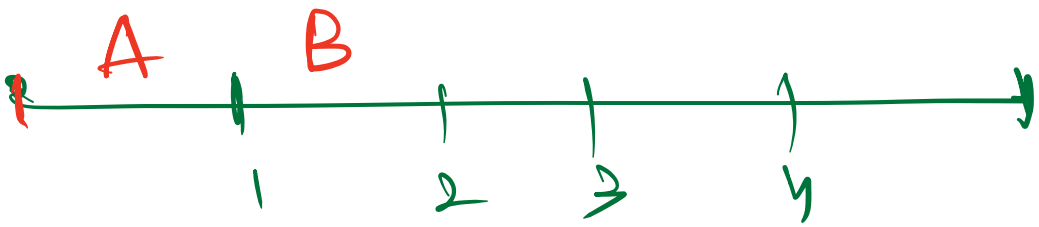

Job
Deadline
Reward
A
3
100
F
4
30
C
4
27
D
1
25
B
1
19
D
C
A
F
0
1
2
3
4
Max
100 + 30 + 27 + 25


Dry run 2
Job
Deadline
Reward
A
1
100
B
2
19
C
2
27
D
1
25
E
1
30
A
C
0
1
2
3
100 + 27
127


Dry Run
Job
Deadline
Reward
A
1
100
B
2
19
C
2
27
D
1
25
E
1
30

Dry Run 3





R = 100 + 60 → 160

B A C

Implementation

① Sort by Reward (Dec order)

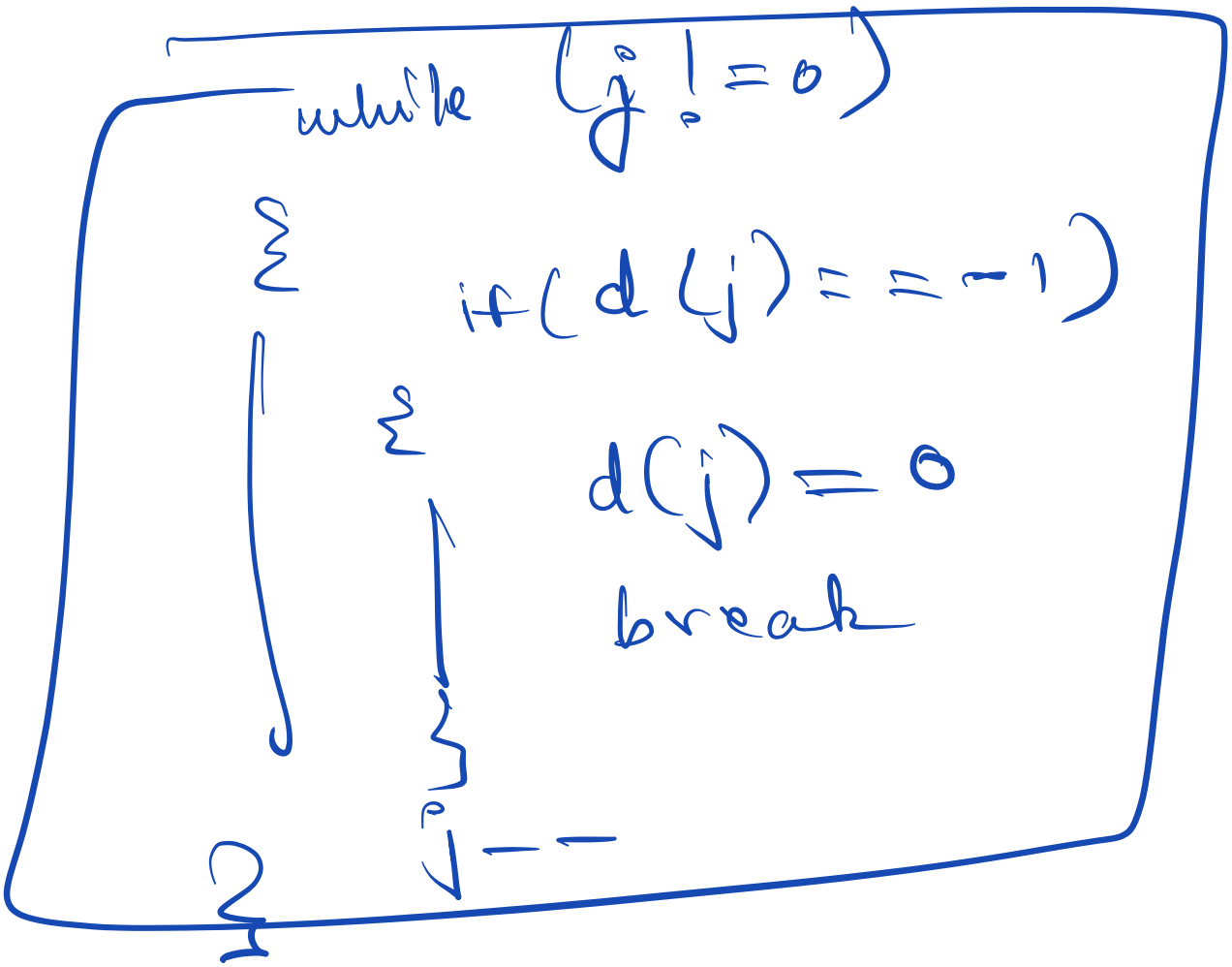
② | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
0 1 2 3 4 5 6

max deadline

A → 100 → 3

if (d(3) == -1) → avail

Job	Deadline	Reward
A	3	100
E	3	30
C	2	27
D	~~1~~	25
B	~~1~~	19

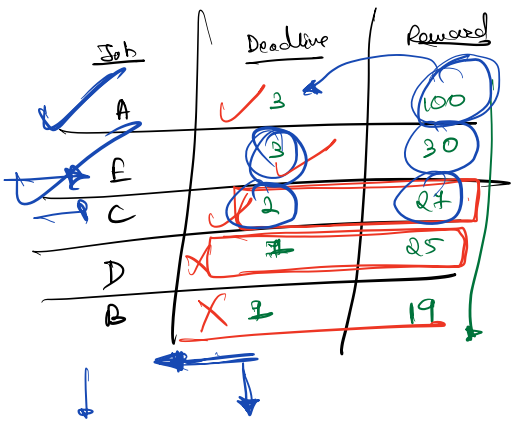


worst case → sort → $(n \log n)$

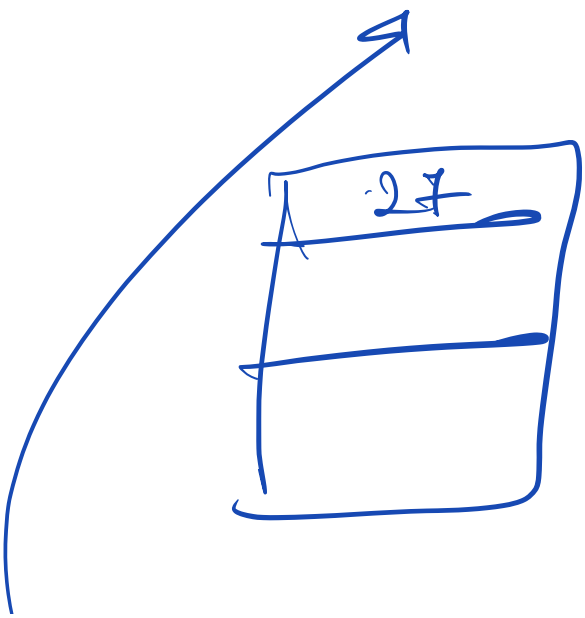
$O(n^2)$

Extra Apr | Apr 3

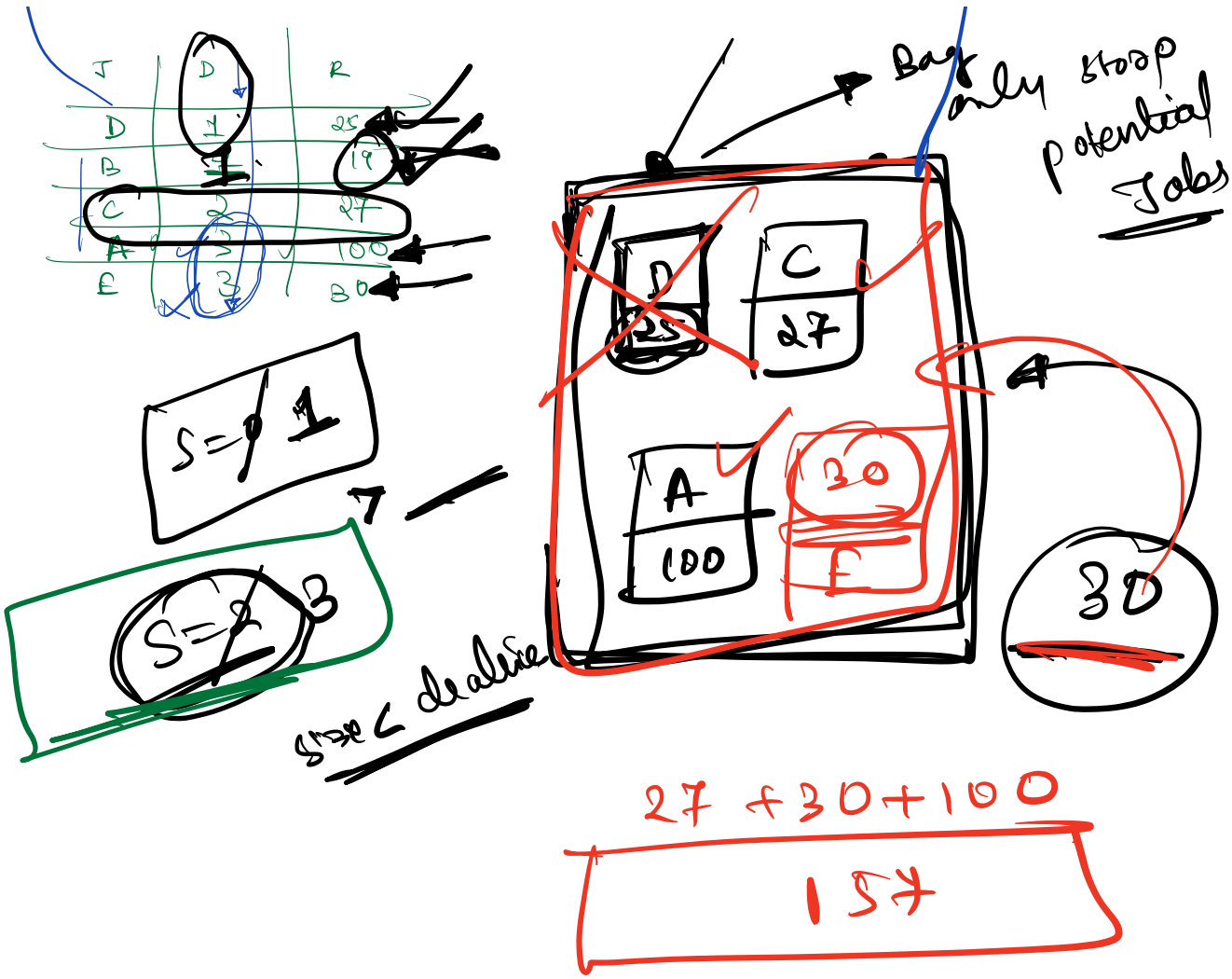
→ Sort Acc to deadlines in Inc ordr



Apr 3 → Similar to K largest + Max Min-Heap

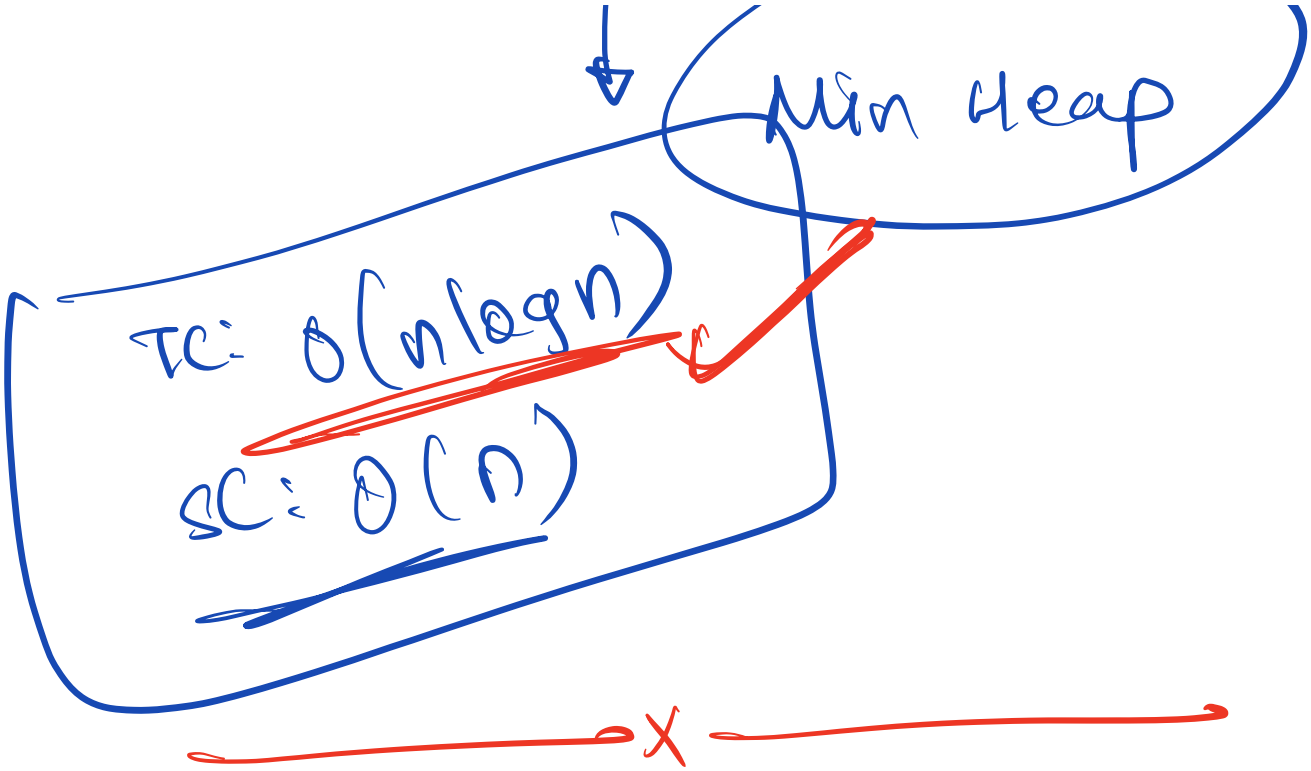


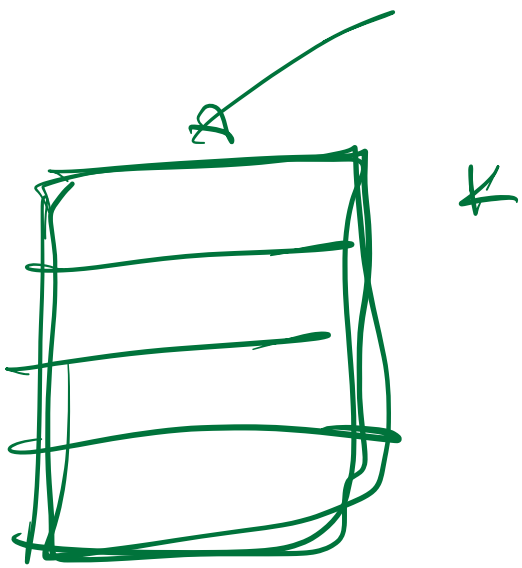
Min-heap

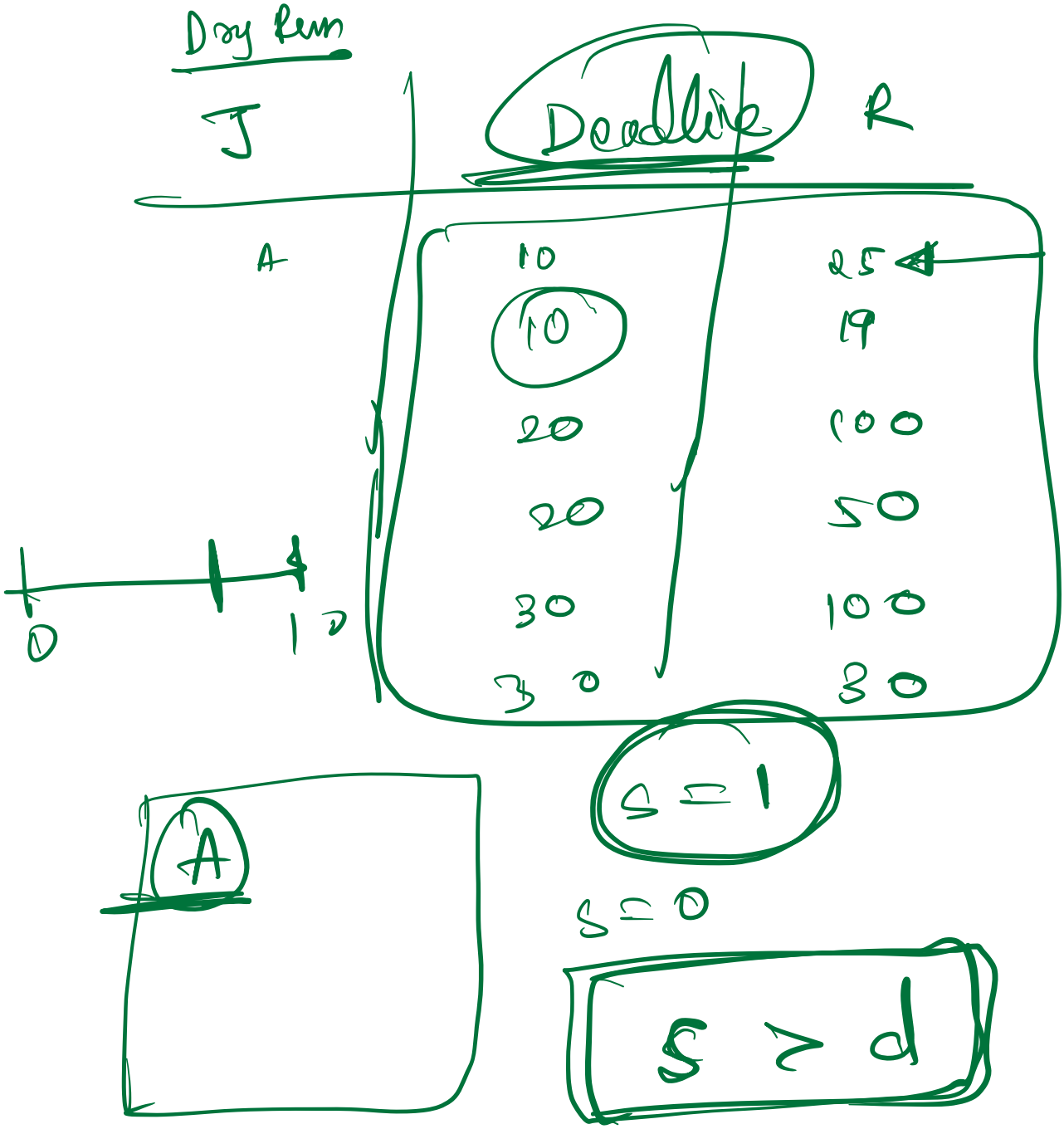


Requirements

1. Insert
2. get min
3. delete min


Min Heap
TC: O(nlogn)
SC: O(n)
X


K


Dry Run
J
Deadlik
R
A
10
25
10
19
20
100
20
50
30
100
30
30
0
10
A
S = 1
S = 0
S > d