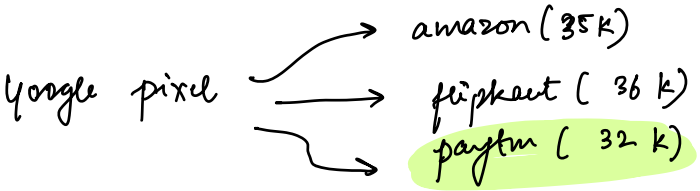


20 books ⟶ 100 books

thinnest books

SD

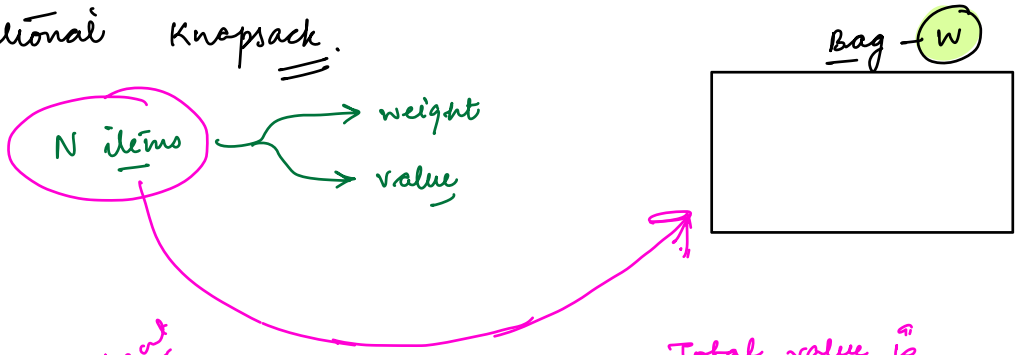
100 books

20 books

ropes — minimum cost

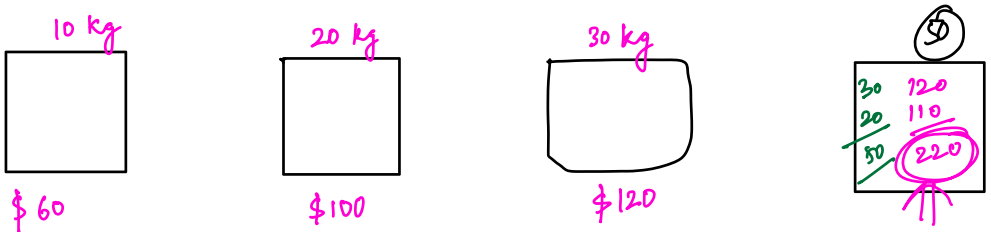
array → a[i]+a[j] ⇒ two max

1) Fractional Knapsack.



you don't have to choose an item completely 1kg at

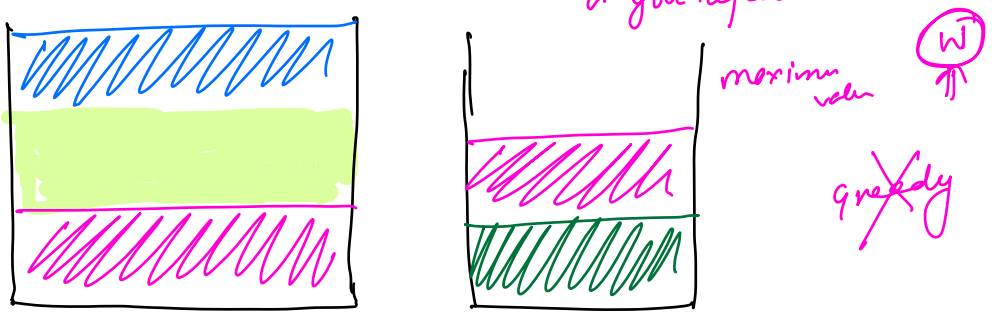
Total value is maximised.



$\frac{60}{10}$ = $6/kg $5/kg $4/kg

1	10	$60
2	20	$100
3	20	$80
		$240

0/1 knapsack — can't take fraction of item, either you choose complete it or you reject.



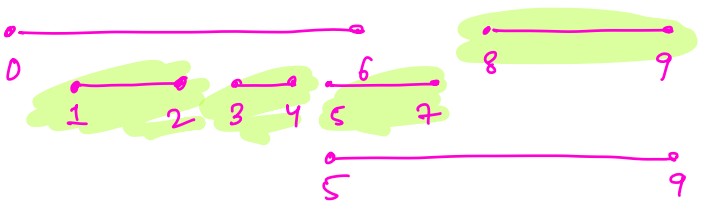
Activity selection problem

N activities → for every activity ≡ start → end

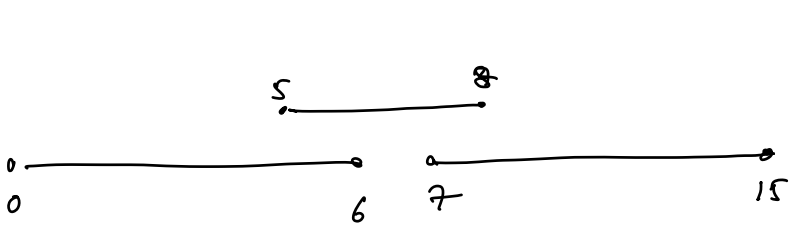
1 activity at a time.

Max activities you can do.

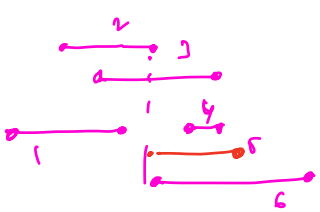
	a_1	a_2	a_3	a_4	a_5	a_6
start	1	3	0	8	5	5
end	2	4	6	9	7	9



Pick by the duration sum smallest duration.



Sort according to start times.



we need activities in order of ending time

|

sort acc to end time

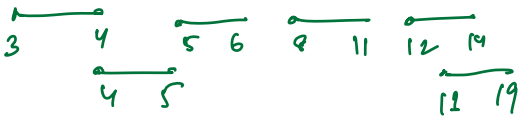
1) sort
2) single iteration

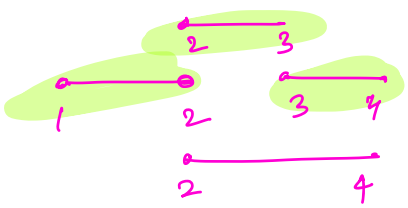
8	12	1	3	4	5	13
11	14	5	4	5	6	19

3	1	4	5	8	12	13
4	5	5	6	11	14	19

start 4

n log n





job scheduling problem.

N jobs → reward
↳ deadline

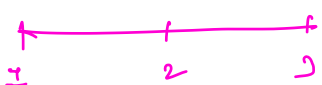
maximize reward

job will only require 1 day to complete

1) on a day, a single job can be performed
2) job can be performed any day before deadline

Job	deadline		
a	3	(3)	100
b	1		19
c	2	(1)	27
d	1		25
e	3	(2)	30

100
27
30



reward ≡ when two tasks are fighting to claim a day

deadline

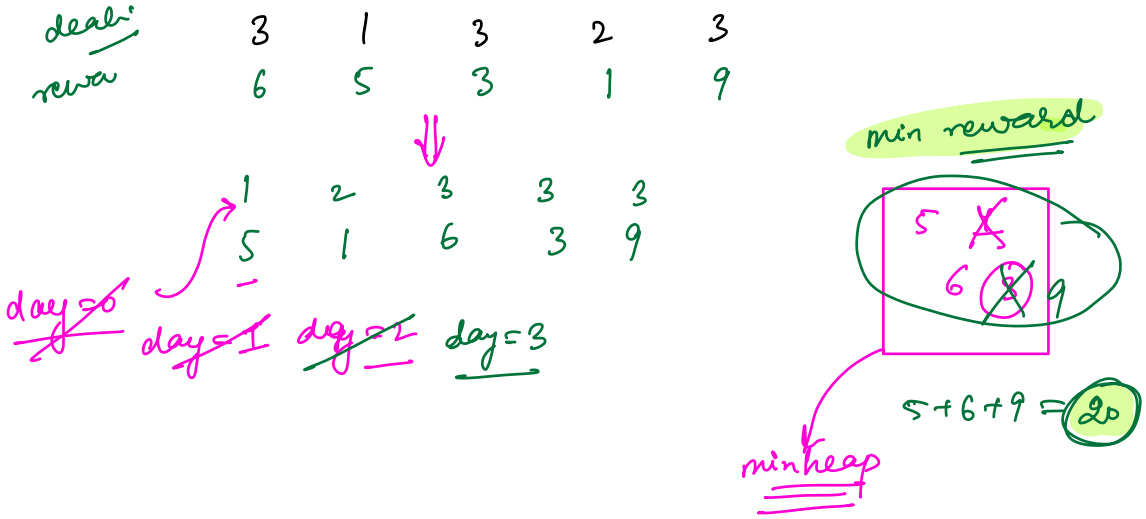
try to complete those task first which has lesse deadline

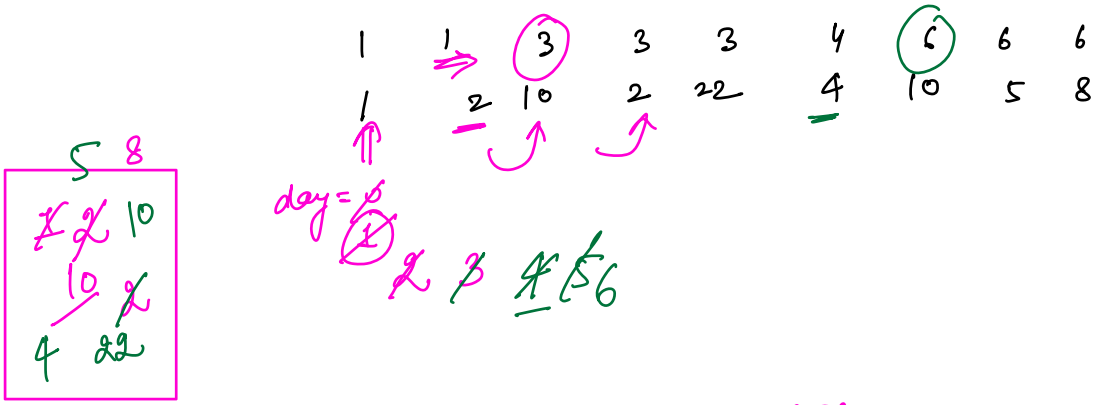
sort acc to deadline

1	1	2	3	3
19	25	27	100	30

20
~~19~~ ~~25~~
27 100

day = 3






```
// sort based on deadline
int day=0                    (i=0 → n)
{
    if( day < deadline[i]) day++;
    {   heap.insert(reward[i]);
    }
    else {  if( root of heap < reward[i]
            {   extractMin() & insert
            }
    }
}
```

T.C: nlogn + n(logn) (max no of days)

sort — loop — heap

- Distribute candies

minimum candies

N Students ⟶ marks

1) Every student must get at least one candy

2) a student should more candies than its neighbours if it has more marks

Marks 1 5 2 1 6

1 3 2 1 2

marks :- 2 6 3 1 10 12 20 5 2

1 2 2 1 2 3 4 2 1

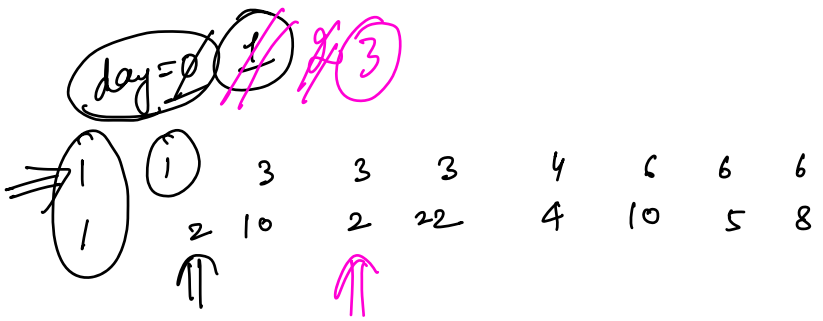
a thr i

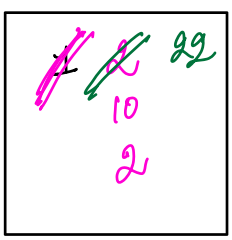
marks[i]

marks[i] > marks[i-1] ⇒ candies[i] > cand[i-1]

& marks[i] > mars[i+1] ⇒ candie[i] > cand[i+1]

marks :-	2	6	3	1	10	12	20	5	2
left	1	2	1	1	2	3	4	1	1
right	1	3	2	1	1	1	3	2	1
	1	3	2	1	2	3	4	2	1


day=0
1
3
1
1
3 3 3 4 6 6 6
2 10 2 22 4 10 5 8


22
10
2