Greedy Algorithm

Indian Currency :

1, 2, 5, 10, 20, 50, 100, 200, 500, 2000

Cash : 5548 Rs

min # of notes/coins to make up this cash

		left
2000	2	1548
500	3	48
20	2	8
5	1	3
2	1	1
1	1	0
	1	

≥ 2 times

Currency : 1, 10, 10

Money : 20 Rs

20

18 1 2 10, 10 $\Rightarrow$ 2

1 1

(1

3

Q2 ⇒

Q ⇒ Super Market : If needed we can eat a single Kg from each item

veg	Protien gained
Tamato : 20Kg	200 p
Apple : 15 Kg	180 p
Onion : 50 Kg	250 p
Chicken : 10 Kg	150 p
Patato : 25 Kg	200 p
Mango : 12 Kg	132 p
Seafood : 5 Kg	100 p

we can rax ⇒ 70 Kg

10p/Kg - 20Kg - 200

12p/Kg - 15Kg - 180

5p/Kg - 50Kg - 250

15p/Kg - 10Kg - 150

8p/Kg - 25Kg - 200 (64)

11p/Kg - 12Kg - 132

20p/Kg - 5Kg - 100

Max protien : 826

70 → 65Kg → 55Kg → 40Kg

↓

8Kg ← 28Kg

8 * 8 = 64

Greedy Properties

① For optimization related problem

② On what parameter apply greedy

③ Counter Example

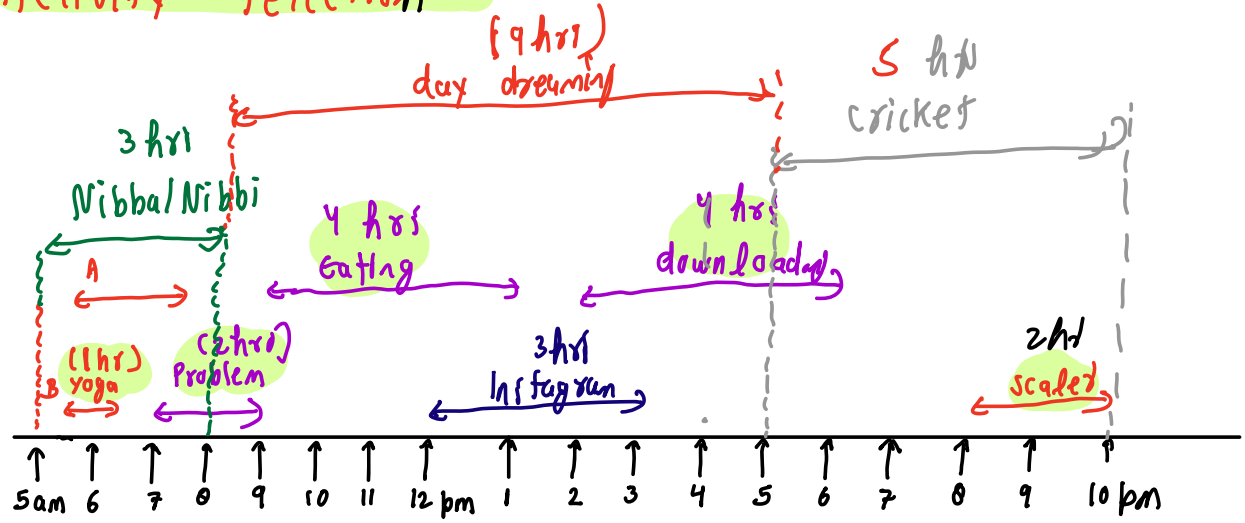
→ Prim Algo / Kruksal Algo

→ Dijkstra.

→ Huffman encoding

Break : 8 ! 53

Activity selection



Tasks:

yoga
Problem
Eating
downloading
Scaler

① Min duration

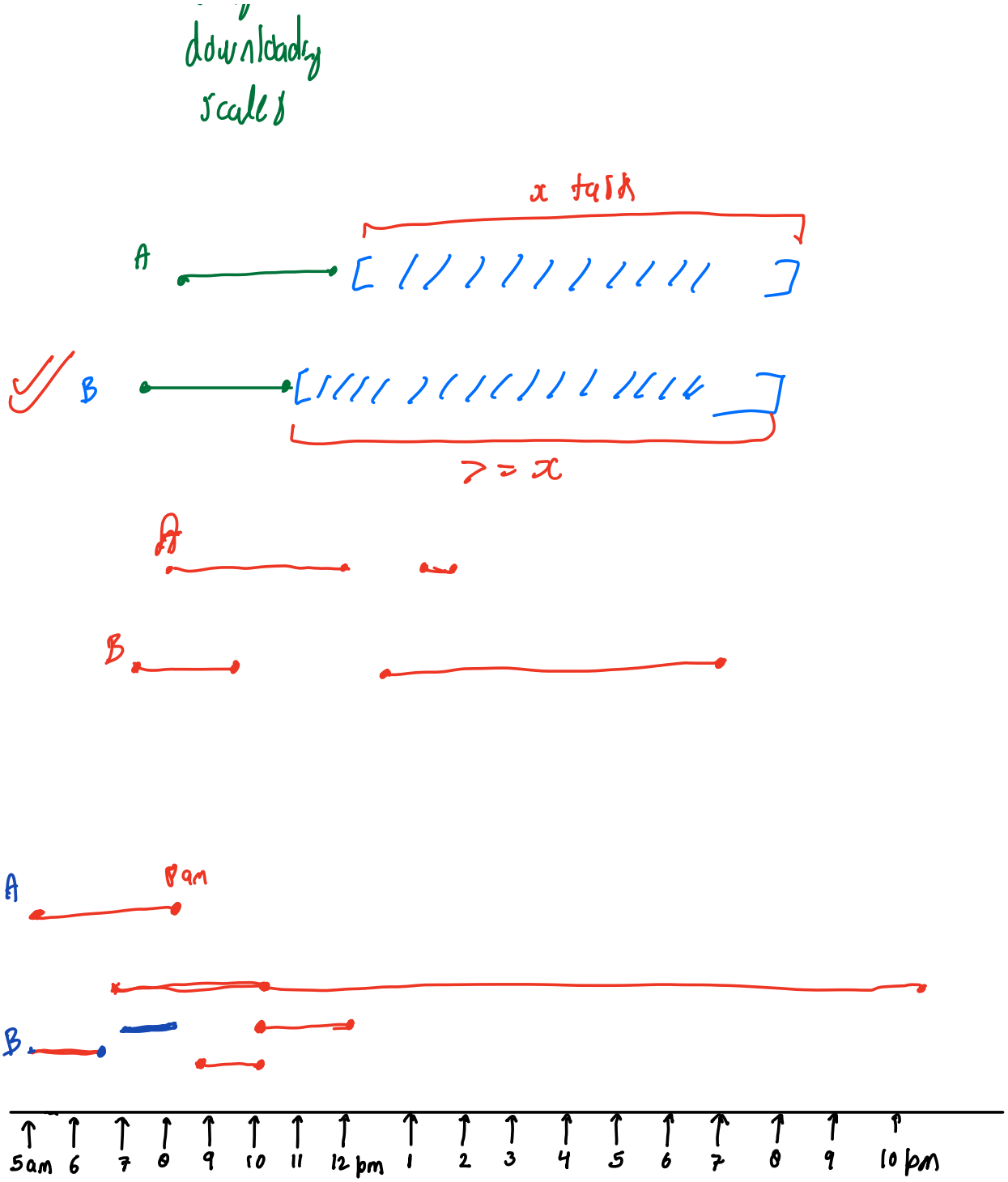
yoga
Problem
Scaler
Instagram

② Pick with min start time

Nibba / Nibbi
Activity
day dreaming
cricket

③ Pick task which end first

yoga
Problem
Eating


downloading
scales
x task
A
B
>= x
A
8am
B
5am
6
7
8
9
10
11
12 pm
1
2
3
4
5
6
7
8
9
10 pm

Job sheduling

Given N task to complete

→ Deadline assigned on each task, day or before that

→ Payment assigned for each task

→ 1 day / 1 task

→ Find max payment.

Job	deadline day	Payment
a	3	100
b	1	19
c	2	27
d	1	25
e	3	30

d, c, a ⇒ 152

c, e, a ⇒ 157

a, e, d

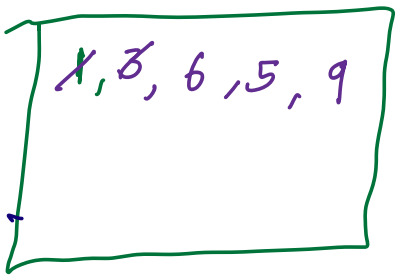
1 1 2 3 3

19 25 27 100 30

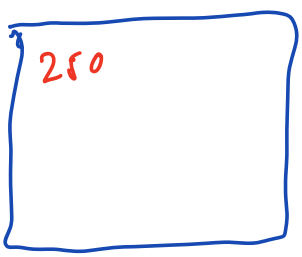
19 25, 27, 100, 30

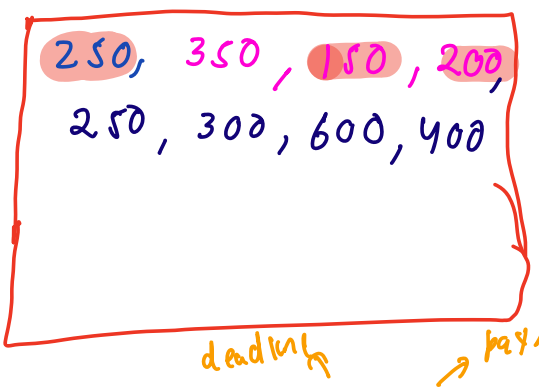
1 2 3 3 3

1 3 6 5 9

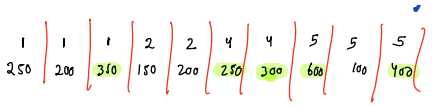


1	1	1	2	2	4	4	5	5	5
250	200	350	150	200	250	300	600	100	400



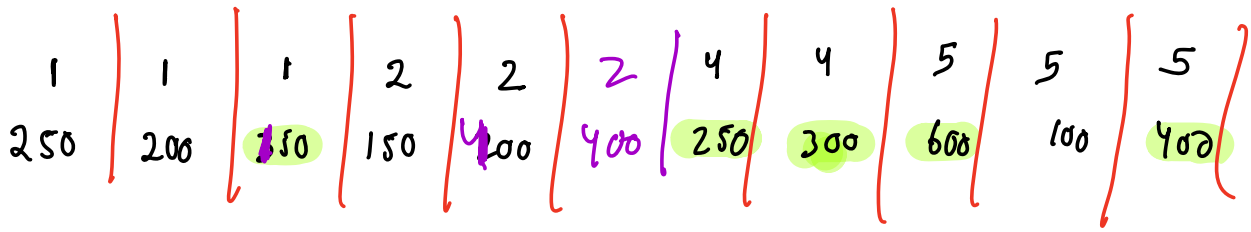



```
int maxcost( list <pair <int, int>> data )
{
    int n = data.size();
    Minheap <int> mh;
  // data.sort ( based on deadline )
    for( int i = 0 ; i<N; i++)
    {
        pair<int, int> x = data[i]
        int day = x.first;
        int pay = x.second;
      if ( day > mh.size() )
```




```
                mh.insert(pay)
        else if (pay > mh.getMin())
            {
                mh.deleteMin()
                mh.insert(pay);
            }
    }
    int ans=0;
    while (mh.size >0)
      {
        ans = ans + mh.getMin();
        mh.deleteMin();
      }
return ans;
  }
```

250, 350,
150, 200



~~250~~, 150, 400,
400