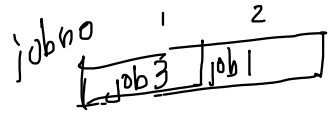
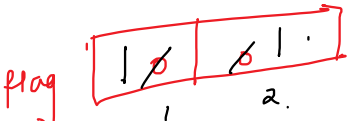
Consider → Practical Algo

job	Profit	Deadline
1	100	2
2	15	1
3	27	2
4	10	1

Sol





flag = 0 ⇒ Empty
1 ⇒ Full

Consider jobs in Descending order of profit

Step 1 → Consider Job 1

d(job1) = 2

check slot(2) = Empty

S(2).jobno = job1

S2 flag = 1

profit = 0 + 100 = 100

Step 2 →

Consider job 3

d(job3) = 2

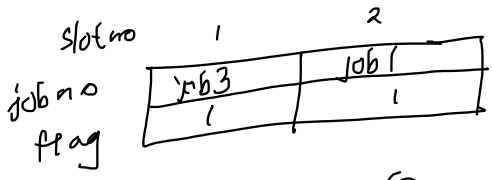
check slot 2 ⇒ not Empty.

Check slot 1 ⇒ Empty.

S(1).jobno = job3

S(1).flag = 1

Profit = 100 + 27 = 127



(3) Consider job 2

d(job2) = 1

check slot 1 ⇒ Not Empty.

job2 cannot be scheduled

(4) Consider job4

d(job4) = 1

check slot 1 ⇒ not Empty

job4 cannot be scheduled

Profit = 127

jobs = { job3, job1 }

(6) $(P_1 \dots P_7) = (3, 5, 20, 18, 1, 6, 30)$ } ✓

⑥ $(P_1 \ldots P_7) = (5, 5, 20, 18, 1, 6, 30)$ ✓

$(d_1 \ldots d_7) = (1, 3, 4, 3, 2, 1, 2)$ ✓

j	p	d
1	3	1
2	5	3
✓3	20	4
4	18	3
5	1	2
6	6	1
✓7	30	2

last deadline = 4

no of slots = 4

slot no	1	2	3	4
job no	job6	job7	job4	job3
flag	1	1	1	1

① Consider job no 7

d(job 7) = 2

chk slot 2 ⇒ Empty

s(2).(job no) = job7

s(2).flag = 1

② Consider job no 3

d(job3) = 4

check slot 4 ⇒ Empty

s(4).job no = job3

s(4).flag = 1

③ Consider job no 4

d(job4) = 3

check slot 3 ⇒ Empty

s(3).job no = job 4

s(3).flag = 1

④ Consider job no 6

d(job6) = 1

check slot 1 = Empty.

s(1).job no = job6

s(1).flag = 1

⑤ Now all the slots are occupied.

So no other job can be scheduled.

profit = 30 + 20 + 18 + 6

= 50 + 24 = 74

Schedule = job6 – job3 – job4 → job7

Algo for Job Sequencing with Deadline

function GetSlot (S, dl) : Integer

// slot bhejega jaha job rakh sakte hai

S(1:k) Here the schedule contains jobno & flag

dl = deadline

```
1. start
2. for(i = dl to 1)
3.     if (S(i).flag == 0) then
4.         return i;
5.     return -1;
```

Return the i^{th} pos^n/slot where this job can be placed.

If no valid pos/slot is Empty then return -1.

Algo for function JobSequencing (X, n, S, k)

where

X(1:n) ⇒ Input array containing x.j (jobno), x.p (profit) and x.d (deadline)

n = no of slots in Schedule

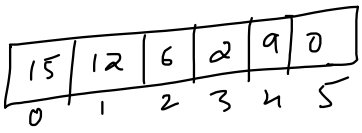
S(1:k) ⇒ Schedule containing s.jobno and s.flag.

```
1. start
2. profit = 0
3. Sort all the jobs in descending order of profit.
4. for(i = 1 to n)  ⇒ sorted job (in order of profit)
   {
```

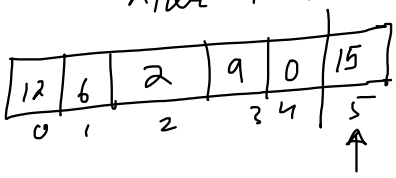
i=1 means job with highest profit

```
5.  avail = GetSlot(S, x(i).d)
6   if (avail == -1) then
7.  print "job" x(i).j "cannot be Scheduled"
8   else
    {
9       S(avail).jobno = x(i).j
10      S(avail).flag = 1
11      profit = profit + x(i).p
    }
}
12  print S (Schedule)
13. Print Profit.
14  Return
```

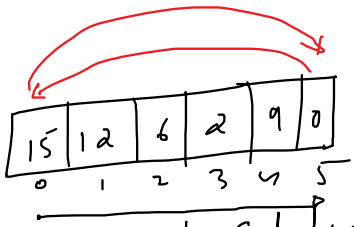
Selection Sort ->



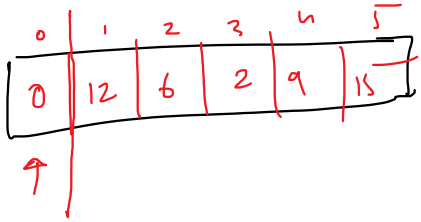
↓ Bubble Sort
After 1st Iteration.



Largest Element at bottom

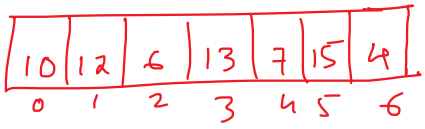


Selection Sort
↓ After 1st Iteration.



Smallest Element is at posn first 0th

Working ->

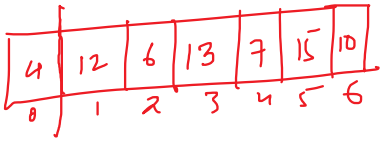


n=7

from 0 ⟶ 6 (n-1)

Smallest Element is 4 at posn = 6

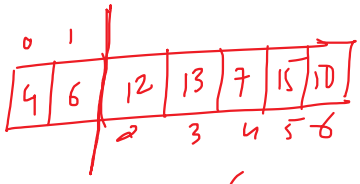
Swap a[0] & a[6]



from 1 ⟶ 6

Smallest Element = 6 at pos = 2

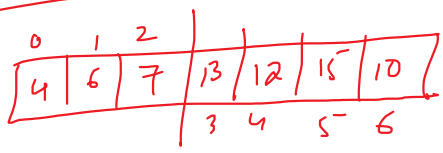
Swap a[1] & a[2]



from 2 ⟶ 6

Smallest Element = 7 and posn = 4

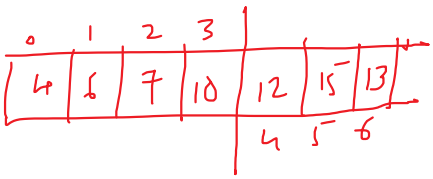
Swap a[2] & a[4]



from 3 ⟶ 6

Smallest Element = 10 Position = 6

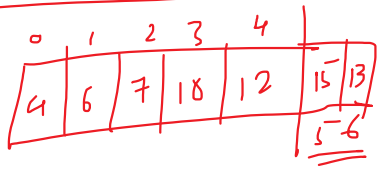
Swap a[3] & a[6]



from 4 ⟶ 6

Smallest Element = 12 at posn = 4

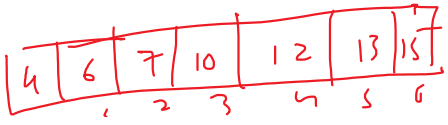
Swap a[4] & a[4]

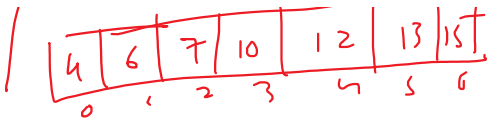


from 5 ⟶ 6

Smallest Element = 13 at pos = 6

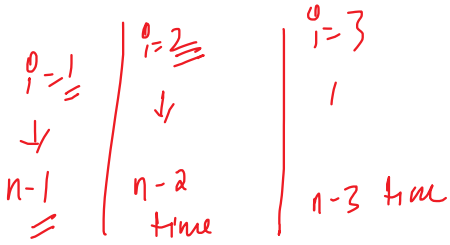
So Swap a[5] and a[6]





Sorted

Analysis =>



n-1 th

n - (n-1) = 1 time

Total no of time execution : $(n-1)+(n-2)+(n-3)+\cdots 1$ $\quad \frac{n(n+1)}{2}$

$= \frac{(n-1)(n+1+1)}{2} = \frac{n(n-1)}{2} = O(n^2)$

Worst case
Best case
Average case

n^2 — $O(n^2)$, $\Omega(n^2)$, $\Theta(n^2)$

⇒

Job Sequencing with deadline

Job j[4],

4

~~Job j1, j2, j3, j4~~

typedef struct {
int id;
int deadline,
int profit
} job