→ when find 1, cnt++ = 1 and max = max(cnt, max)
→ when find 0, cnt = 0.
Ans. max
TC → O(n) SC → O(1)

Greedy approaches.

Q43. N meetings in one room
→ one meeting room
→ n meetings in between a time interval
→ find maximum number of meetings that could be held
→ ending time of one meeting can't be starting time of another.

S[] = 1 0 5 8
F[] = 2 6 4. 9 7 9.
1 room

Date: Page:

Approach 1.
Sort the given meetings as per finishing time irrespective of starting time.

(1,2,1)
(3,4,3) → 1st meeting can take place limit = 2.
(0,6,2) → when go to next meet, make sure to
(5,7,5) check limit thus 3rd meeting limit = 4
(8,9,4) → if e[i] < limit, skip.
(8,9,6)
→ 5th meeting limit = 7
→ 4th meeting limit = 9.

∴ TC → O(n) + O(N log N) + O(N) ≈ O(N log N)
SC → O(N)
Most optimal Sol

Source code.

```
struct meeting {
  int start, end, pos; };
bool comparator (struct meeting m1, meeting m2)
  {
    if (m1.end < m2.end) return true
    else if (m1.end > m2.end) return false
    else if (m1.pos < m2.pos) return true.
    return false;
  }
Void maxMeet (int S[], int e[], int n) {
  struct meeting meet[n];
  for (int i = 0; i < n; i++) {
    meet[i].start = S[i];
    meet[i].end = e[i];
    meet[i].pos = i+1 ; }
  sort (meet, meet+n, comparator);
  vector<int> ans;
  int limit = meet[0].end;
```

ans.push_back(meet[0]);
for (int i = 1; i < n; i++)
if (meet[i].start > limit) {
int = meet[i].end;
answer.push_back(meet[i].pos);
print(answer)

Q44 Minimum number of platform required for a railway
→ given arrival & departure time for railway station
→ minimum no. of platforms required to avoid keeping trains waiting

→ all timings are of same day
→ a train departing platform can't be used for arrival of another train

start[] = 120 50 550 200 700 850
end[] = 600 550 700 500 900 1000

* ask interviewer whether starting time is sorted
→ sort all the starting time & departure time
→ take 2 ptrs as null as start & end e

Q45 job sequencing problem
→ n jobs with deadline & profit on each job
→ each jobs takes 1 unit of time and one job can be done at a time

id	deadline	profit
1	4	50
2	1	10
3	1	40
4	1	30

Aim: find max profit

→ sort all the jobs as per descending profit
→ perform all the jobs on last day (try)
→ later we will perform, better it'll be

TC → O(N log N) + O(N * M)
(sort)

SC → O(M)

Q46 Fractional knapsack ✓

Q47 Minimum number of coins
→ find minimum no. of coins needed to make change
→ unlimited coins
eg. 49 = 20 + 20 + 5 + 2 + 2 (5)
where coins [] = {1, 2, 5, 10, 20, 50, 100, 500, 1000}
87 = 50 + 20 + 10 + 5 + 2 (5)
→ start from back and iterate until you can't make change from denomination
→ if we can make we add ~~d[i]~~ change/d[i]
and change = change % d[i]

TC = O(V) SC = O(1)