Greedy

Intro To Greedy

String matching algorithms :-

1) Rabin Karp
2) Z Algo
3) KMP algo.

Intro To Greedy :-

Greedy algorithms are a class of algorithms that make locally optimal choices at each step with a hope of finding a global optimum solution.

Car Sale

Problem :

There is a limited time sale going on for toys.
A[i] → sale end time for i^{th} toy.
B[i] → Beauty of i^{th} toy.
Time start at $t=0$ & it takes 1 unit of time to buy one toy. & toy can be brought if $T < A[i]$
Buy toys such that sum of beauty of toys is maximized.

A[] → {3 1 3 2 3} (0 1 2 3 4)
B[] → {6 5 3 1 9}

$t = 0 + 2 = 3$
$9 + 6 + 3 = 18$

$t = 0 \; 1 \; 2 \; 3$
toys → 5 + 9 + 6 = 20 ↑

A[] → {1, 2}
B[] → {3, 1500}

$t = 0 \; 1$
$3 + 1500 = 1503$ ↑

Idea :- Try to buy everything on the order of sale end time.

A[] : {1 3 3 3 5 5 5 8} → based on time.
B[] : {5 2 7 1 4 3 8 1}

t = 0 → 5 2 7 8 4 3 1 → cannot be added as it is less as per this particular sale time.

→ getMin()
extractMin()
insert()

We can use min heap as if max value we are not able to insert we will remove min

Pseudo Code

1) Sort the data in ascending order of time.

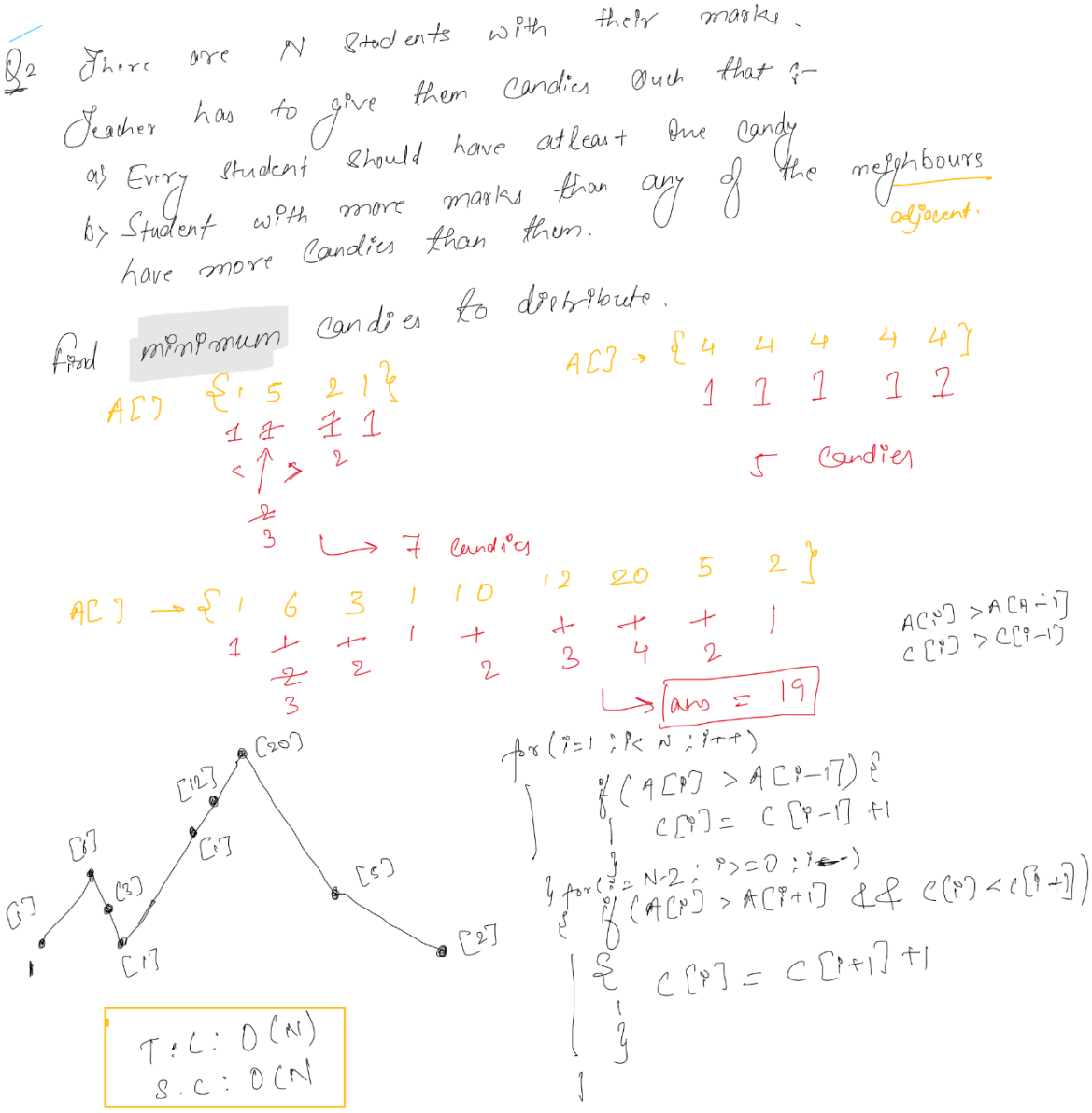
2) Minheap mh
t = 0;
for(int i=0; i<N; i++) {
  if (t < A[i]){
    mh.insert(B[i]);
    t++;
  else if (B[i] > getMin()) {
    mh.extractMin()
    mh.insert(B[i])
  }
}

3) Ans = sum of all elements in heap

S.C : O(N)
Take an array of pairs & sort based on time using comparators.

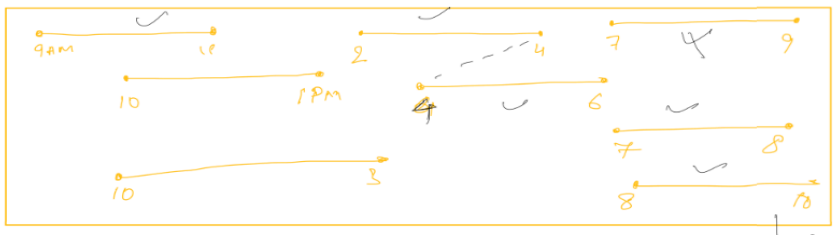
T.C : O(N log N)

Maximum Candies

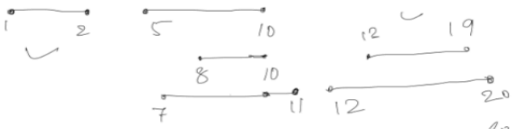


Maximum Job

Q3 Given N jobs with their start & end time, Find the max jobs that can be completed if only job can be done at a time. S[i] >= e [i-1]



Ex: S[] → {1 5 8 7 12 13}
E[] → {2 10 10 11 20 19}



ans = 3

Ideas :
1) Sort all the jobs based on start time. ✗ will not work
2) Pick based on duration of jobs ✗ will not work
3) Start early + Minm duration = End early.
→ Sort the jobs based on end time.
→ use a pair

S[] → {1 5 8 7 13 12}
E[] → {2 < 10 < 10 < 11 < 19 < 20}

S[i] >= lastEndTime

ans = ✗ 3 , lastEndTime = ~~2~~ 10 19

✗ fails for 12

ans = 3

Pseudo code

1) Sort based on end time
2) ans = 1, lastEndTime = E[0]
3) for(i=1; i<N; i++) {
   if(S[i] >= lastEndTime) {
      lastEndTime = E[i];
      ans++;
   }
}
4) return ans;

for 0 based

T.C : O(N log N)
S.C : O(N) because of array of past.

Greedy → Maximize / Minimize