Code Jam 2015 - Qualification Round

Analysis: Standing Ovation

What kind of shyness level friends should we invite?

Since we can invite any friend with any shyness level, the problem seems really complicated. However, if you think about it you will realize that inviting friends with shyness level 0 is optimal and makes this problem simpler. Friends with shyness level 0 are always better than those with other shyness levels because they always stand up and clap, therefore helping all other audience members who have shyness level greater than 0. This kind of choice is greedy but makes sense as for any scenario: if you can invite friends with shyness level greater than 0 and solve the problem, then you can always replace them with friends whose shyness level is 0 and still solve the problem.

How many friends should we invite?

For audience members with shyness level ***k***, to meet their needs, we must have at least ***k*** people who already stood up before them. These include both the audience members whose shyness level is less than ***k*** and the friends we invited (with shyness level 0). Assuming that there are ***t*** audience members who stood up already, the number of friends we need to invite is **max(*k* - *t*, 0)**. This provides us with an algorithm. Letâ€™s say we are given an array of audience members for which the k^{th} entry contains the total audience members of shyness level ***k***. Now, for each shyness level, we compute the number of friends we need to invite to get the audience members of that level to stand up and clap, and we record the maximum value out of all such values. This max value is the minimum number of friends we need to invite to let every audience member stand up and clap for the prima donna.

Below is a sample implementation in Python:

```
for tc in range(input()):
  smax, string = raw_input().split()
  t = 0
  min_invite = 0
  for k in range(int(smax) + 1):
    min_invite = max(min_invite, k - t)
    t += int(string[k])
  print "Case #%d: %d" % (tc + 1, min_invite)
```

Because the input of audience members is already given in increasing order, we only need to walk through this array once and compute the result. Since min_invite is initially 0, max(k - t, 0) is redundant (i.e., k - t is sufficient). The overall time complexity is $O(\mathbf{S}_{max})$.