PS5

Chieh Lee

You Zhang

Part I

1)

We can sort the given list of job into ascending order. Then the algorithm will return the optimal result.

t0 + (t0 + t0+1) + … + (t0+ t0+1 + …. tn)/n

where tn-1 < tn

We can use known sorting method that is efficient such as quicksort which is divide and conquer algorithm. And the running time is bounded to O (n log n)

2)

We can use greedy algorithm, compare the unhappy score for each person,

For I to N person P

Compare UiA and UiB

If UiA > UiB, send Pi to team A else send to team B

The whole process will take O(n),

3)

“messiness” is the cube of the number of spaces left at the end of a line between the final word of the line and W.

The goal here is to minimize the messiness

Each word w have length l + 1(including the separating space), except the last word of the line.

W is the total length of a line, thus (l1 + 1) + (l2 + 1) + … (lp-1 + 1) + lp < W, where lp is the last word of the first line and so on

First we sort w = {w1, w2, … , wn} into a descending order. And insert w1 to first line.

For i = w

Int counter k = 0;

k += (li + 1)

If k < W

Remove wi from w

i++

if k > W,

skip to wi and proceed to wi+1 until wn

endfor

if w !isEmtpy()

run for loop again with w

messiness of each line will be optimal, function is taking O(n2)

Part II

1.

7

6

5

4

3

2

1

2.

4

2 6

1 3 5 7

3.

1 2 4

| / |

3 6 5

|

7

4.

1 2 3 4 5 6 7

Part III

1. **NO!** There has no prove that NP problem have no polynomial algorithm to solve, especially breaking the cryptography requires factoring and factoring isn’t even a NP-complete
2. **YES!** solving 3-SAT hence we can also solve 2-SAT only prove that 2-SAT problem is at least as hard as 3-SAT. In other word, 2-SAT problem can be easier. 2-SAT cannot be proved as NP-Complete
3. **YES!** The solution can be check in polynomial time so this is an NP problem

this problem is reducible from NP problem such as Hamilton path problem in a polynomial time. Therefore, this problem is surely NP-complete problem.

Part IV

Optimally solution (Brute force) has running bounded to O(2N). Therefore slightly larger input will take enormous amount of time to compute.