2021.05.25 鄧祺文 410821316

ALGO CLASS ASSIGNMENT QUESTION ONE & TWO

QEUSTION ONE

Statement:

For question one, we assume the condition that each parking lot (total ‘N’ parking lots) can store only 1 car. Therefor total ‘N’ cars stored in ‘N’ parking lots are ‘N’ every 1 unit movement (either to the left or to the right) costs of 1 unit time.

The goal for it is to minimize the sum of time which moves all cars to the parking lots, once at a time. So, we need to find the min[summation($k_i – p_j$)], which denotes the difference between the positions of cars and the position of parking lots.

Design:

1. An array ‘C’ to store the position of the cars on the one-dimensional axis with C_i denoting of the position of the ith car.
2. An array ‘P’ to store the position of the parking lots on the one-dimensional axis with P_i denoting the position of the ith slot.
3. An array ‘Parked’ to store the result whether a car is parked in the slot or not. If Parked[C_i] = 1, the parking slot is full. On the contrary, the parking slot will be empty.

4. Create an integer ‘minSum’ = 0 to store the final answer (minimum time spend).
5. For every P_i find the car C_j that hasn’t been parked in the slot and find its minimum distance from P_i. Then increment the value to variable ‘minSum’ (minSum = minSum + distance difference between the parking slot and the non-parked car).
6. Output the final answer ‘minSum’

<u>Time Complexity:</u>

$O(N^2)$

<u>Pseudo Algo:</u>

```
// input the car position & parking lot position
int C[10] = {car position};
int P[10] = {parking lot position};
// assume the maximum parking lots to be 10
// initial the array for no car parked
Int Parked[10] = {0};
// variable for the final answer
minSum = 0;
// for every parking lot in P
for p in P;
CurrentMinimumCarPosition = 1000;
for c in C;
// only check if the parking slot is empty
if (Parked[c] != 1);
// check the minimum difference distance
```

```
if (CurrentMinimumCarPosition > c) {
    CurrentMinimumCarPosition = c;
}
// update the parked array with the minimum car pos as
parked
Parked[CurrentMinimumCarPosition] = 1;
minSum = minSum + abs(p - CurrentMinimumCarPosition);
cout << minSum;
```

QUESTION TWO

Statement:

For question two, we assume the condition that every key word given will be unique and with no key word to be a part of another key word, the goal for this question is to find the minimum combination sentence for all the key words, by only combining the word from the left side or the right side.

Design:

1. An array ‘W’ for storing all the key words given.
2. An array ‘L’ for storing the maximum combination length from the left.
3. An array ‘R’ for storing the maximum combination length from the right.
4. A string ‘S’ for storing the result of the combination sentence.
5. For every two key words, combine them from the left and store the maximum length.
6. For every two key words, combine them from the right and store the maximum length.

7. Compare the maximum combination length for left and right, apply the bigger value to combine two words, which will be stored in the string ‘S’.
8. Going through the array ‘W’, keep doing the above given steps and combining the key words one by one till all of the key words have been combined together.
9. Output the final result ‘S’.

Time Complexity:

$O(N^2)$