MID SEM ASSIGNMENT CS-204

Tarusi Mittal
1901Cs65

QUE 1:

Given: An array of sized n (named arr)
Key[n] – Ai = key value of ith element of the array
Weight[n] – Bi = weight value of ith element of the array
To find: Weighted position corresponding to f(f<=1)

Solution:

Traverse the array in key values and store the value of largest key as **maxkey**

Now using **radix sort** to sort the elements according to the key values (i.e. **using count sort at each digit)**

```
Initialize i=1 ;

(Running for loop with i such that (maxkey / i)>0 and incrementing it each time with i=i*10){

initialise an output array final[n][2]

initialise an array count [10]={0};

initialise an array index[n];


running a for loop(j=0 to j=n){
  index[ j ]=(arr[ j ][ 0 ]/t)%10
}


running a for loop(j=0 to j=n){
  count[index[ j ]]++
}


running a for loop(k=1 to k=10){
  count[ k ] = count[ k ] + count [k-1]
}
```

```
running a for loop(j=0 to j=n){
  final[count[index[ j ] ] ][ 0 ]  = arr[ j ][ 0 ]
  final[count[index[ j ] ] ][ 1 ]  = arr[ j ][ 1 ]
  count[index[ j ] ]--
}
running a for loop(j=0 to j=n){
  arr[ j ][ 0 ] = final[ j ][ 0 ]
  arr[ j ][ 1 ] = final[ j ][ 0 ]
}
}
The  array arr received now is sorted according to the key values ;
Initialising a variable sum=0
Given f the frequency
Initialising a variable ans=0;
Running a for loop(i=1 to i=10){
  sum+=arr[ i ][ 1 ]
  if ( sum >= f){
     break;
}
else{
     ans = arr[ i ][ 1 ]
}
Output the ans;
```

Radux sort in O(n) algorithm. Since we are using radix sort, the running time would be O(d∗n).

D is a small number, Hence
T(n) = O(n)

QUE 2:

Given: An input array of size n
Integer m, the number of times the negation operation is required to be done

Steps:

1. Take the input of the array of size n and the integer m.
2. Sort the array in ascending order
3. Initialise an integer temp to count the no of negative integers in the given array
4. Compare temp with the given m
    - If the value of temp is less than or equal to the given integer m:
      we will perform negation operation on the first m integers in the sorted array
    - If the temp value is more than the integer m
      First do the negation operation on all the negative elements, Then update m value to m-temp; Then sort the array again in ascending order and negate the first m-temp values
5. Take the sum of all the elements and print the output

Proving that greedy approach is optimal:

The greedy choice property is that here we select the local optimal solution and combine them to find the Global optimal solution

In the above greedy approach:
We are using the negation of the negative number which has the biggest absolute value and if we will not do this let us consider that the negative number which has the highest absolute value is y and say there is an another negative number whose absolute value is less than y and let us name is x.

Case 1: y in included in the final solution
In this case so the final answer will not depend as we have also negation of y as well so the value and the sum will be same

Case 2: y is not included in the final solution
The solution which we have provided earlier using the greedy approach will have y in its negative value therefore it would be larger than the solution performed by this step as the absolute value of y is greater than the absolute value of x so the sum provided by the greedy solution will automatically be larger than the sum provided by the non-greedy solution.

We can say that the greedy approach provides more optimal solution to this problem so we should first negate the value of the integers which have the highest absolute value in the negative range.

We can see that we are using the local optimal solutions to find out the Global optimum solutions so this is indeed a greedy approach and as specified above it is **the most optimal approach** for this question

________________________________****************END*****************__________________________