EXPERIMENT - 6

Student Name: Sandeep Kumar **UID: 20BCS4885**

Branch: CSE **Section/Group: 603/A**

Semester: 6th semester **Subject: Competitive Coding**

Aim: Graphs

Objective:

a) Find the difference

b) Predict the winner

Problem 1: find the difference

Solution code:

```
class Solution {
public:
   char findTheDifference(string s, string t) {
      int n = max(s.size(), t.size()) ;
      sort(s.begin(), s.end()) ;
      sort(t.begin(), t.end()) ;
      for (int i = 0 ; i < n ; i ++)
      {
         if (s[i] == t[i])
         {
            continue ;
         }
         else if (s[i] != t[i])
         {
            return t[i] ;
         }
      }
      return t[n] ;
   }
};
```

Approach:

Approach [using Sorting]

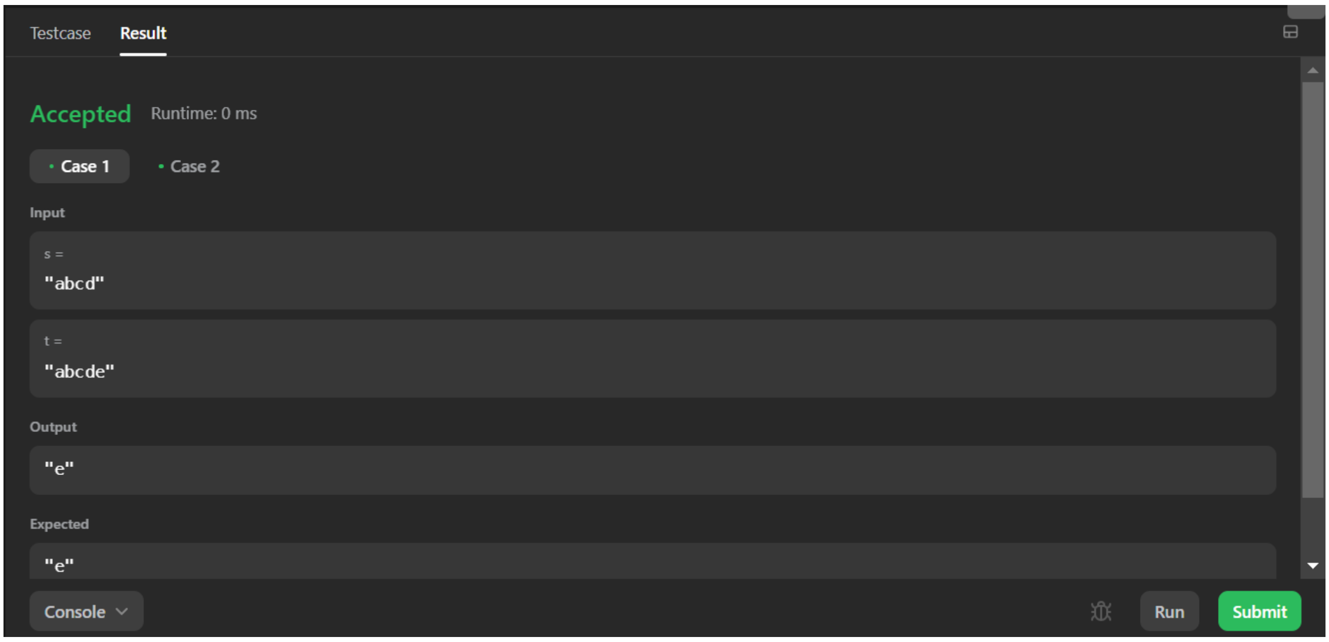
- Sort both the strings
- Iterate through the strings, now the first character that is not the same in both strings is the difference, so return that
- Else, if all characters match and we have reached the end of the strings so return the last character of the 't' string

Complexity:

Time Complexity: O(n∗logn)

Space Complexity: O(n)

Output:



```
Testcase   Result

Accepted   Runtime: 0 ms

Case 1   Case 2

Input
s =
"abcd"
t =
"abcde"

Output
"e"

Expected
"e"
```

Problem 2: Predict the winner

Input Code:

```
class Solution {
public:
   int solve(vector<int>& nums, bool turn ,int i, int j ){
      if(i>j)
         return 0;
      if(turn){
       return max(nums[i]+solve(nums,false,i+1,j),nums[j]+solve(nums,false,i,j-1));
      }
      else
        return min(solve(nums,true,i+1,j),solve(nums,true,i,j-1));
   }
      bool PredictTheWinner(vector<int>& nums) {
      int totalSum=0;
      for(int i=0;i<nums.size();i++){
         totalSum+=nums[i];
      }
      int sum1= solve(nums,true,0,nums.size()-1);
      int sum2=totalSum-sum1;
      return sum1>=sum2;
   }
};
```

Approach:

here we will just calculate the maximum possible sum for player 1 , and after that we will subtract this from total sum of nums , then we will get the sum for player 2.

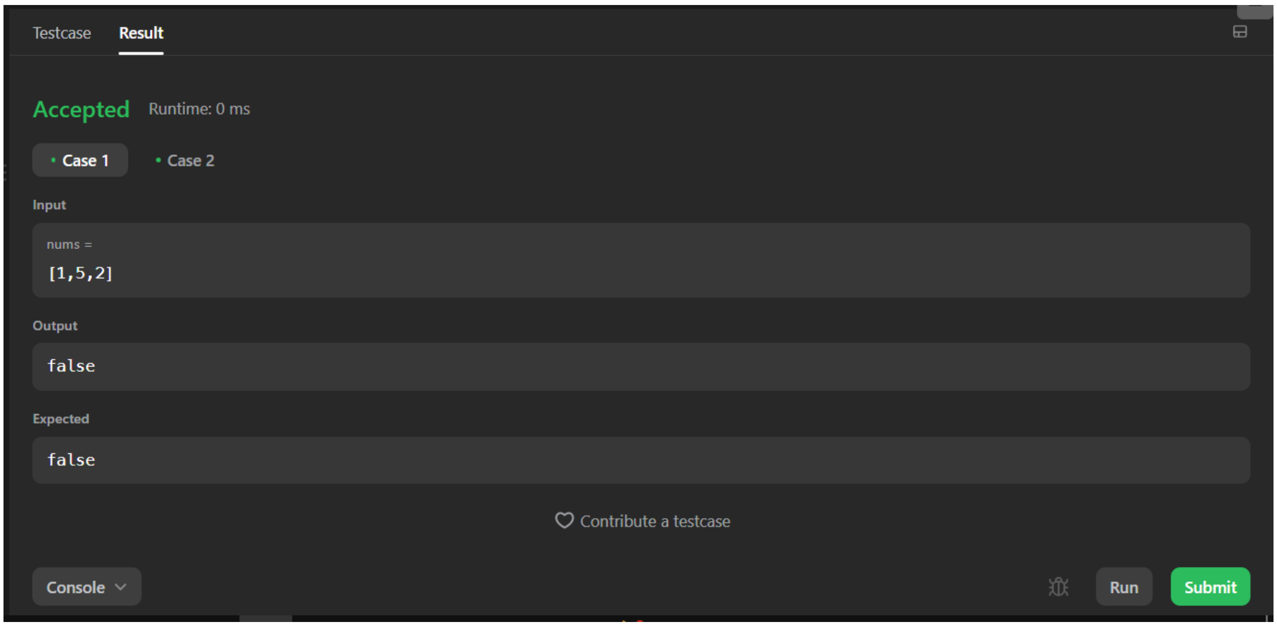
here is just one catch , while writing recursive calls for player 2 we will not add nums[i]/ nums[j], and just return the minimum because we are calculating the sum for player1.

Complexity:

Time Complexity: O(n)

Space Complexity: O(1)

Output:



Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
nums =
[1,5,2]
```

Output

```
false
```

Expected

```
false
```