CO322: Data Structures and Algorithms

Lab 05: HR problems set 04

WIJERATHNE E.S.G

E/18/397

---

01. Beautiful Pairs

```java
public static int beautifulPairs(List<Integer> A, List<Integer> B) {

        List<Integer> ATemp = new ArrayList<Integer>(A);


        int result = 0;

        for(Integer a : ATemp){
            if(A.contains(a) && B.contains(a)){
                A.remove(a);
                B.remove(a);
                result++;
            }
        }


        if(A.size() >= 1 && B.size() >= 1){
            result++;
        }else if(result == ATemp.size()){
            result--;
        }
        return result;
}
```

First, I made a copy of array A, which is saved as ATemp. Initialised integer To store the result. Iterate through the copy of array A and see whether both A and B arrays have the same element. If so, remove the element from both A and B arrays and increment the result value.

After that, I check the size of the remaining arrays, A and B. if the size of both arrays is larger than one, that means we can increment the result by one because we should exactly change an element in the array B. Then I check that the result is equal to the size of a copy of array A. if so that means both A and B arrays have identical elements. Since we are supposed to change exactly a single element in array B, we have to decrease the result value by one. Finally returns the result.

02. Sherlock and Array

```java
public static String balancedSums(List<Integer> arr) {
        long lsum = 0;
        long rsum = 0;
        String res = "NO";

        if(arr.size() ==1){
            return "YES";
        }

        for (int i =1; i<arr.size(); i++){
            rsum += arr.get(i);
        }

        for(int pos = 0; pos<arr.size()-1; pos++){
            if(lsum == rsum){
            res = "YES";
            break;
         }
            System.out.println(pos);
            lsum += arr.get(pos);
            rsum -= arr.get(pos+1);

        }

        return res;


    }
```

First, I initialize two long numbers to store the left sum value and the right sum value and initialize the string result value as “NO”. First, I check if the array size is equal to the one. If so, I can return “YES”. Then I iterate through the array from index one to the last element and get the initial right sum. Then I consider each position of the array until an element before the last one and check whether there is a match between the left sum and the right sum. If so, set the return value as “YES” and break the loop. Otherwise, increment the left sum by adding the current position value and decrement the right sum value by the current position value.

03. Short Palindrome

```java
public static int shortPalindrome(String s) {

        int length = s.length();
        int count = 0;
        char array[] = s.toCharArray();
        if (length < 4){
            System.out.println("0");
            return 0;
        }
        for (int i = 0; i < length-3; i++){
            for (int j = length-1; j > i+2; j--){
                if (array[i] == array[j]){
                    for (int k = i + 1; k < j ; k++ ){
                        for (int m = j - 1; m > k; m--){
                            if (array[k] == array[m]){
                                count++;
                            }
                        }
                    }
                }
            }

        }
        return count%(1000000000 + 7);

    }
```

First, look for if the array size is less than 4. If so, return 0. Then looping for the first and last letters of the palindrome. If find a match then search for 2nd and 3rd letters of the palindrome. If find a match, that means we find that palindrome occurrence. Then, the output palindrome count will be incremented by one.