codility

Ryan Wang

Candidate

E-mail: ryancode101@gmail.com
Phone: 613-890-6917

Session

ID: FMZ4M7-8K5
Time limit: 70 min.
Report recipients: alicia.blagrove@rbccm.com asad.munshi@rbccm.com
Accessed from: 173.34.163.38, 173.34.163.38
Invited by: alicia.blagrove@rbccm.com

Status: closed

Created on: 2018-05-18 17:29 UTC
Started on: 2018-05-21 18:52 UTC
Finished on: 2018-05-21 20:02 UTC

Notes:
N/A

Similarity Check

Status: not found
No similar solutions have been detected.

Tasks in test	Correctness	Performance	Task score
1 BracketStringSplit Submitted in: Java	83%	100%	90%
2 CountIdenticalPairs Submitted in: Java	80%	66%	75%
3 Fib Submitted in: Java	83%	0%	35%
4 BugfixingFrequentCharacter Submitted in: Java	50%	100%	70%

Test score

68%

270 out of 400 points

Next step: online coding interview

Start CodeLive Interview

TASKS DETAILS

MEDIUM

1. BracketStringSplit

Find a position in a given string such that the number of opening brackets to the left is equal to the number of closing brackets to the right.

Task Score	Correctness	Performance
90	83	100

Task description

You are given a string S consisting of N brackets, opening "(" and/or closing ")". The goal is to split S into two parts (left and right), such that the number of opening brackets in the left part is equal to the number of closing brackets in the right part.

More formally, we are looking for an integer K (the length of the first part of the split) such that:

- $0 \leq K \leq N$, and
- the number of opening brackets in the K leading characters of S is the same as the number of closing brackets in the N−K trailing characters of S.

Please keep in mind that input string and any of resulting strings do not need to be properly matched parentheses. The requirement is that the number of opening brackets in the left part of the split should be exactly the same as number of closing brackets in the right part.

Write a function:

```
class Solution { public int solution(String S); }
```

that, given string S, returns a value for K that satisfies the above conditions. It can be shown that such a number K always exists and is unique.

For example, given S = "(())", the function should return 2, because:

- the first two characters of S, "((", contain two opening brackets, and
- the remaining two characters of S, "))", contain two closing brackets.

In other example, given S = "(())))(", the function should return 4, because:

- the first four characters of S, "(())", contain two opening brackets, and
- the remaining three characters of S, "))(", contain two closing brackets.

In other example, given S = "))", the function should return 2, because:

- the first two characters of S, "))", contain zero opening brackets, and
- there are no remaining characters, so they contain also zero closing brackets.

Assume that:

- N is an integer within the range [0..100,000];
- string S consists only of the characters "(" and/or ")".

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(1) (not counting the storage required for input arguments).

Solution

SEE LIVE VERSION

Programming language used:	Java
Total time used:	38 minutes
Effective time used:	12 minutes
Notes:	*not defined yet*

Source code

Code: 19:30:34 UTC, java, final, score: **90**

```java
// you can also use imports, for example:
// import java.util.*;

// you can write to stdout for debugging purposes, e.g.
// System.out.println("this is a debug message");

class Solution {
    public int solution(String S) {
        // write your code in Java SE 8
        int len=S.length();
        int o[]=new int[len+1];
        int c[]=new int[len+1];
        int cur=-1;
        o[0]=0;
        c[len]=0;
        if(S.charAt(0)=='(')
            o[1]=1;
        if(S.charAt(len-1)==')')
            c[len-1]=1;
        for(int i=1;i<len;i++){
            if(S.charAt(i)=='(')
                o[i+1]=o[i]+1;
            else
                o[i+1]=o[i];
        }
        for(int i=len-2;i>=0;i--){
            if(S.charAt(i)==')')
                c[i]=c[i+1]+1;
            else
                c[i]=c[i+1];
        }
        if(o[len]==0)
            return len;
        if(c[0]==0)
            return 0;
        for(int i=0;i<=len;i++)
            if(o[i]==c[i])
                cur=i;
        return cur;
    }
}
```

Analysis summary

The following issues have been detected: runtime errors.

For example, for the input `''` the solution terminated unexpectedly.

Analysis

Detected time complexity: **O(N)**

Example tests	
example first example test	✔ OK
example2 second example test	✔ OK
example3 third example test	✔ OK
Correctness tests	
extreme empty string or one bracket	✘ RUNTIME ERROR tested program terminated with exit code 1
simple simple tests	✔ OK
single_double two brackets	✔ OK
small_random random string, N = 100	✔ OK
small_half '((...))' or '))...((', N = 100	✔ OK
small_bracket_expr random bracket expression, N = 100	✔ OK
Performance tests	
medium_random random string, N = 10,000	✔ OK
large_random random string, N = 100,000	✔ OK
large_bracket_expr random bracket expression, N = 100,000	✔ OK
all_the_same one type of brackets, N = 100,000	✔ OK
large_half '((...))' or '))...((', N = 100,000	✔ OK

HARD

2. CountIdenticalPairs

Calculate the number of identical pairs.

Task Score	Correctness	Performance
75	80	66

Task description

An array A consisting of N integers is given. We are looking for pairs of elements of the array that are equal but that occupy different positions in the array. More formally, a pair of indices (P, Q) is called *identical* if 0 ≤ P < Q < N and A[P] = A[Q]. The goal is to calculate the number of identical pairs of indices.

For example, consider array A such that:

```
A[0] = 3
A[1] = 5
A[2] = 6
A[3] = 3
A[4] = 3
A[5] = 5
```

There are four pairs of identical indices: (0, 3), (0, 4), (1, 5) and (3, 4). Note that pairs (2, 2) and (5, 1) are not counted since their first indices are not smaller than their second.

Write a function:

```
class Solution { public int solution(int[] A); }
```

that, given an array A of N integers, returns the number of identical pairs of indices.

If the number of identical pairs of indices is greater than 1,000,000,000, the function should return 1,000,000,000.

For example, given:

```
A[0] = 3
A[1] = 5
A[2] = 6
A[3] = 3
A[4] = 3
A[5] = 5
```

the function should return 4, as explained above.

Assume that:

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [−1,000,000,000..1,000,000,000].

Complexity:

- expected worst-case time complexity is O(N*log(N));
- expected worst-case space complexity is O(N) (not counting the storage required for input arguments).

Solution

SEE LIVE VERSION

Programming language used:	Java
Total time used:	62 minutes
Effective time used:	37 minutes
Notes:	*not defined yet*

Source code

Code: 19:54:02 UTC, java, final,
score: **75**

```java
// you can also use imports, for example:
import java.util.*;

// you can write to stdout for debugging purposes, e.g.
// System.out.println("this is a debug message");

class Solution {
    public int solution(int[] A) {
        // write your code in Java SE 8
        if(A==null||A.length==0)
            return 0;
        int len=A.length, res=0;
        Map<Integer, Integer> hm=new HashMap<>();
        for(int i=0;i<len;i++){
            hm.put(A[i], hm.getOrDefault(A[i], 0)+1);
        }
        for(Integer num: hm.values()){
            res+=countEach(num);
        }
        return res;
    }
    public int countEach(int num){
        return (num-1)*num/2;
    }
}
```

Analysis summary

The following issues have been detected: wrong answers.

Analysis

Example tests	
example example test	✔ OK
Correctness tests	
single empty/single element	✔ OK

double two elements	✔ OK
small_functional small functional tests	✔ OK
small_range range medium test, length = ~400	✔ OK
Correctness/performance tests	
medium_identical many identical pairs, length = ~50,000	✘ WRONG ANSWER got -897508641 expected 1000000000
large_functional large functional tests, length = ~100,000	✘ WRONG ANSWER got -245012293 expected 1000000000
Performance tests	
medium_range range medium test, length = ~40,000	✔ OK
large_random chaotic large sequences, length = ~100,000	✔ OK

MEDIUM

3. Fib

Find a few least significant digits of a large Fibonacci number.

Task Score	Correctness	Performance
35	83	0

Task description

The Fibonacci sequence is defined using the following recursive formula:

F(0) = 0
F(1) = 1
F(N) = F(N−1) + F(N−2) if N ≥ 2

Write a function:

```
class Solution { public int solution(int N); }
```

that, given a non-negative integer N, returns the six least significant decimal digits of number F(N).

For example, given N = 8, the function should return 21, because the six least significant decimal digits of F(8) are 000021 (the complete decimal representation of F(8) is 21). Similarly, given N = 36, the function should return 930352, because the six least significant decimal digits of F(36) are 930352 (the complete decimal representation of F(36) is 14930352).

Assume that:

- N is an integer within the range [0..2,147,483,647].

Complexity:

- expected worst-case time complexity is O(log(N));
- expected worst-case space complexity is O(log(N)).

Solution

SEE LIVE VERSION

Programming language used:	Java
Total time used:	69 minutes
Effective time used:	7 minutes
Notes:	*not defined yet*

Source code

Code: 20:01:06 UTC, java, final,
score: **35**

```
// you can also use imports, for example:
// import java.util.*;

// you can write to stdout for debugging purposes, e.g.
// System.out.println("this is a debug message");

class Solution {
    public int solution(int N) {
        // write your code in Java SE 8
        if(N==1)
            return 0;
        if(N==2)
            return 1;
        int n1=0, n2=1, n3=0;
        for(int i=2;i<=N;i++){
            n3=n1+n2;
            n1=n2;
            n2=n3;
        }
        return n3 % 1000000;
    }
}
```

Analysis summary

The following issues have been detected: wrong answers, timeout errors.

For example, for the input 1 the solution returned a wrong answer (got 0 expected 1).

Analysis

Example tests	
example1 example test, n=8	✔ OK
example2 example test, n=36	✔ OK
Correctness tests	

extreme0 zero-th element	✔ OK
extreme1 1-st element	✘ WRONG ANSWER got 0 expected 1
medium1 n=15	✔ OK
medium2 n=20	✔ OK
medium3 n=40	✔ OK
medium4 n=42	✔ OK
Performance tests	
medium5 n=50	✘ WRONG ANSWER got -632863 expected 269025
big1 n=100	✘ WRONG ANSWER got -107325 expected 915075
big2 n=1000	✘ WRONG ANSWER got 111435 expected 228875
big3 n=10K	✘ WRONG ANSWER got 44891 expected 366875
big4 n=100K	✘ WRONG ANSWER got 876091 expected 746875
big5 n=1M	✘ WRONG ANSWER got 755131 expected 546875
big6 n=100M+1	✘ TIMEOUT ERROR running time: 1.47 sec., time limit: 0.10 sec.
big7 n=1G+2	✘ TIMEOUT ERROR running time: >6.00 sec., time limit: 0.10 sec.

EASY

4. BugfixingFrequentCharacter

Find and correct bugs in a function that seeks the character that occurs most frequently in a given string and is the earliest alphabetically.

Task Score	**Correctness**	**Performance**
70	50	100

Task description

You are given an implementation of a function:

```
class Solution { public String solution(String S); }
```

that, given a non-empty string consisting of N lowercase English letters, returns the character which occurs most frequently in the string. If more than one character satisfies this requirement, the function should return the earliest alphabetically. For example, if both c and d are the most frequent letters, then the answer is c.

For example, given a string:

```
S = "hello"
```

the function should return `"l"`. It appears twice in S. No other characters appear as frequently.

The attached code is still **incorrect** on some inputs. Despite the error(s), the code may produce a correct answer for the example test cases. The goal of the exercise is to find and fix the bug(s) in the implementation. You can modify at most **four** lines.

Assume that:

- N is an integer within the range [1..100,000];
- string S consists only of lowercase letters (a−z).

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(1) (not counting the storage required for input arguments).

Solution

SEE LIVE VERSION

Programming language used:	Java
Total time used:	70 minutes
Effective time used:	15 minutes
Notes:	*not defined yet*

Source code

Code: 20:02:15 UTC, java, final, score: **70**

```
import java.util.*;
class Solution {
    String solution(String S) {
        int[] occurrences = new int[26];
        for (char ch : S.toCharArray()) {
            occurrences[ch - 'a']++;
        }

        char best_char = 'a';
        int  best_res  = 0;

        for (int i = 1; i < 26; i++) {
-           if (occurrences[i] >= best_res) {
+           if (occurrences[i] > best_res) { // indexOf compa
                best_char = (char)((int)'a' + i);
                best_res  = occurrences[i];
            }
        }

        return Character.toString(best_char);
    }
}
```

Analysis summary

The following issues have been detected: wrong answers.

For example, for the input 'aaabbb' the solution returned a wrong answer (got b expected a).

Analysis

Example tests	
example First example test.	✔ OK
Correctness tests	

one_character Tests with one character.	✓ OK
same_characters Tests with each character being the same.	✓ OK
two_characters Tests featuring two distinct characters.	✗ WRONG ANSWER got b expected a
short_random Short random tests.	✓ OK
random_ties Short random tests featuring many-way ties.	✗ WRONG ANSWER got b expected a
boundary_frequent Tests in which the removal of any (or both) of boundary characters changes the result.	✗ WRONG ANSWER got e expected a
Performance tests	
large_one_character Large tests with one character.	✓ OK
large_same_characters Large tests with two characters. The numbers of occurrences are the same.	✓ OK
large_two_characters Large test with two characters. The number of occurrences are different.	✓ OK
random_max_test Random max test.	✓ OK