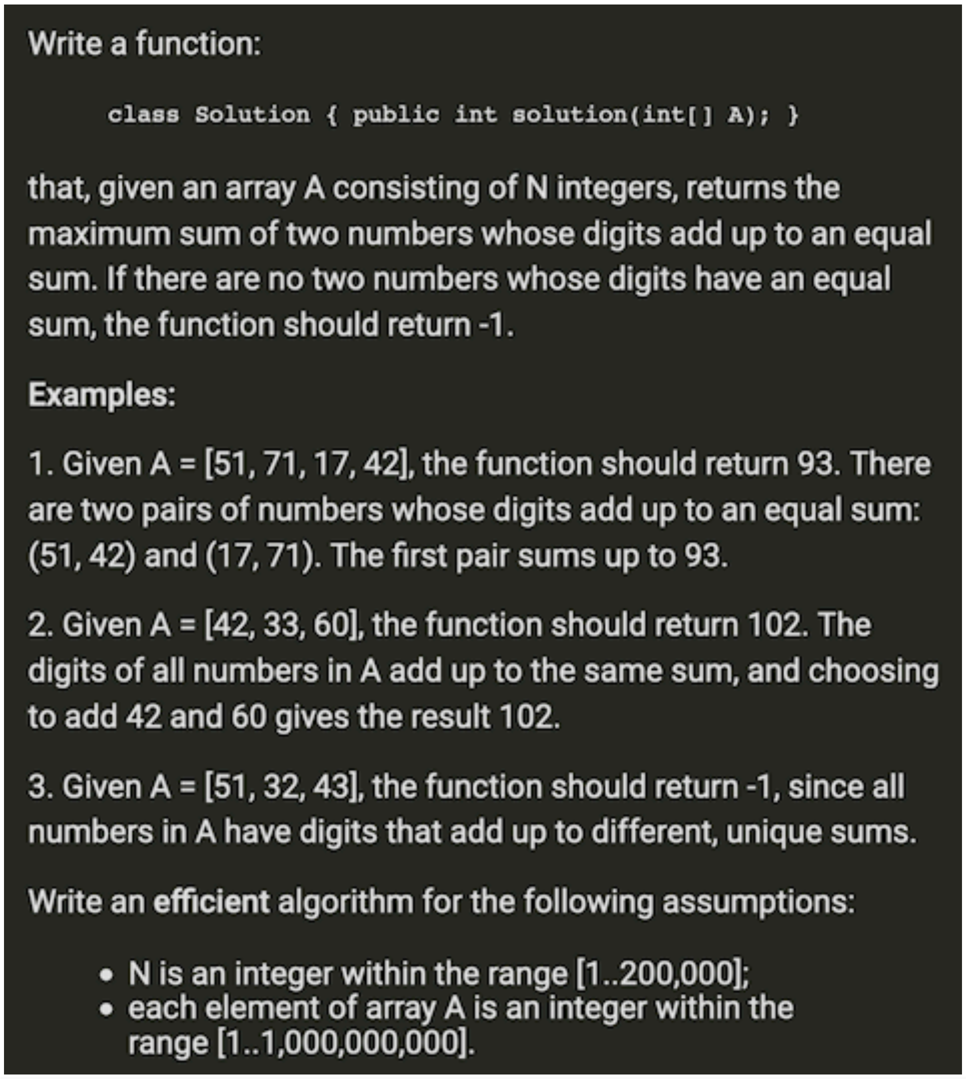
Microsoft | OA 2019 | Numbers With Equal Digit Sum

10

captaincode826

Last Edit: October 5, 2019 7:39 PM

5.2K VIEWS



Write a function:

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

that, given an array A consisting of N integers, returns the maximum sum of two numbers whose digits add up to an equal sum. If there are no two numbers whose digits have an equal sum, the function should return -1.

Examples:

1. Given A = [51, 71, 17, 42], the function should return 93. There are two pairs of numbers whose digits add up to an equal sum: (51, 42) and (17, 71). The first pair sums up to 93.

2. Given A = [42, 33, 60], the function should return 102. The digits of all numbers in A add up to the same sum, and choosing to add 42 and 60 gives the result 102.

3. Given A = [51, 32, 43], the function should return -1, since all numbers in A have digits that add up to different, unique sums.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..200,000];
- each element of array A is an integer within the range [1..1,000,000,000].

Comments: 23

BestMost VotesNewest to OldestOldest to Newest

Login to Comment

SiyC120

Last Edit: August 28, 2019 1:04 AM

Read More

Java O(NlogK) time complexity & O(N) space complexity solution with playground.
There contains logK since computing the digit sum of A[i], which complexity is log(A[i]) with base 10.

```
    private int computeDigitSum(int a){

        // supposed to be valid for negative numbers and the output must be non-
negative integer.

        a = Math.abs(a);

        int res = 0;

        while(a > 0){

            res += a % 10;

            a /= 10;

        }

        return res;

    }

    public int maxSum(int[] A){

        int N = A.length;

        if(N <= 1) return -1;

        Map<Integer, Integer> map = new HashMap<>();

        int res = -1;

        for(int i = 0; i < N; ++i){

            int digitsum = computeDigitSum(A[i]);

            if(!map.containsKey(digitsum)){

                map.put(digitsum, A[i]);

            }

            else{

                res = Math.max(res, map.get(digitsum) + A[i]);

                map.put(digitsum, Math.max(A[i], map.get(digitsum)));

            }
        }

        return res;

    }
```

https://leetcode.com/discuss/interview-question/365872/

2.

https://leetcode.com/discuss/interview-question/398026/

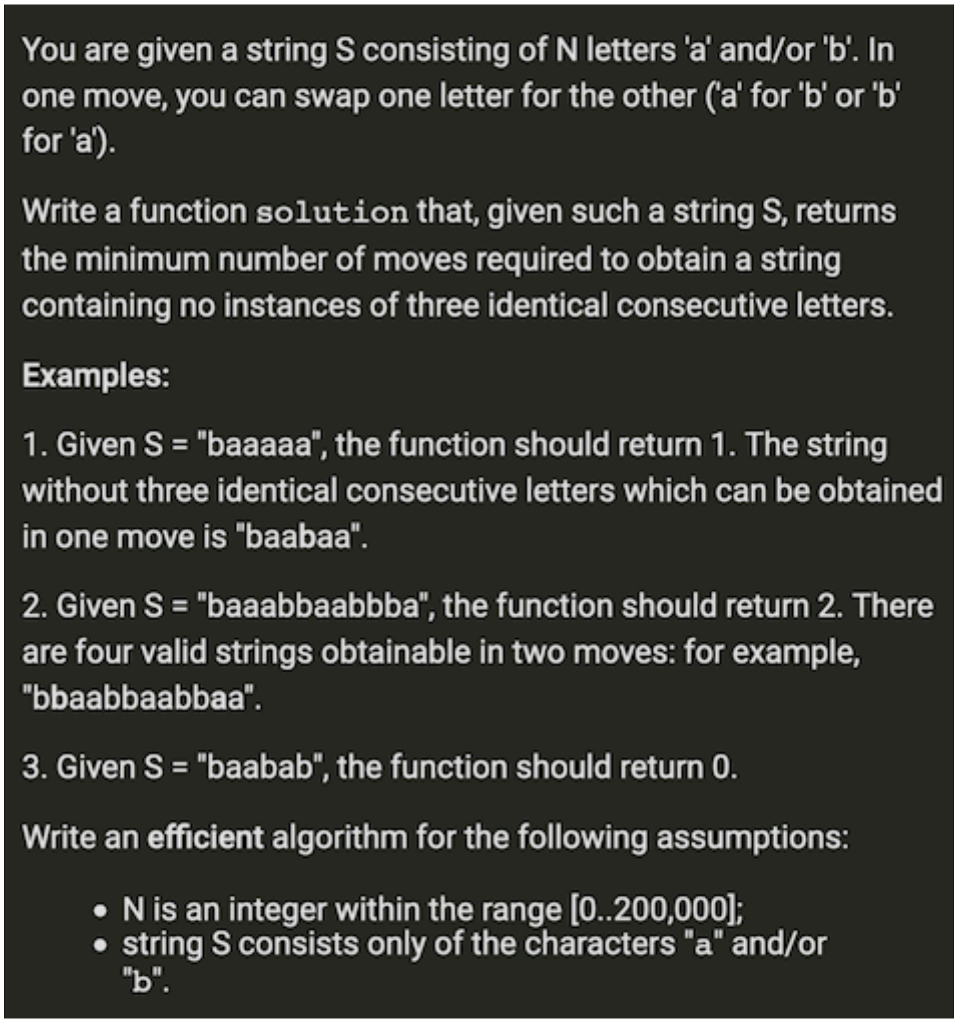
Microsoft | OA 2019 | Min Moves to Obtain String Without 3 Identical Consecutive Letters

2

SithisModerator5580

Last Edit: October 6, 2019 7:36 PM

2.3K VIEWS


You are given a string S consisting of N letters 'a' and/or 'b'. In one move, you can swap one letter for the other ('a' for 'b' or 'b' for 'a').

Write a function `solution` that, given such a string S, returns the minimum number of moves required to obtain a string containing no instances of three identical consecutive letters.

Examples:

1. Given S = "baaaaa", the function should return 1. The string without three identical consecutive letters which can be obtained in one move is "baabaa".

2. Given S = "baaabbaabbba", the function should return 2. There are four valid strings obtainable in two moves: for example, "bbaabbaabbaa".

3. Given S = "baabab", the function should return 0.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [0..200,000];
- string S consists only of the characters "a" and/or "b".

Java solution

Time complexity: O(n).

Space complexity: O(1).

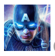
```
public int solution(String s) {

    int moves = 0;

    for (int i = 0 ; i < s.length(); i++) {

        int runLength = 1;

        for (; i + 1 < s.length() && s.charAt(i) == s.charAt(i + 1); i++) {

            runLength++;

        }

        moves += runLength / 3;

    }

    return moves;

}
```

3.

https://leetcode.com/discuss/interview-question/364760/

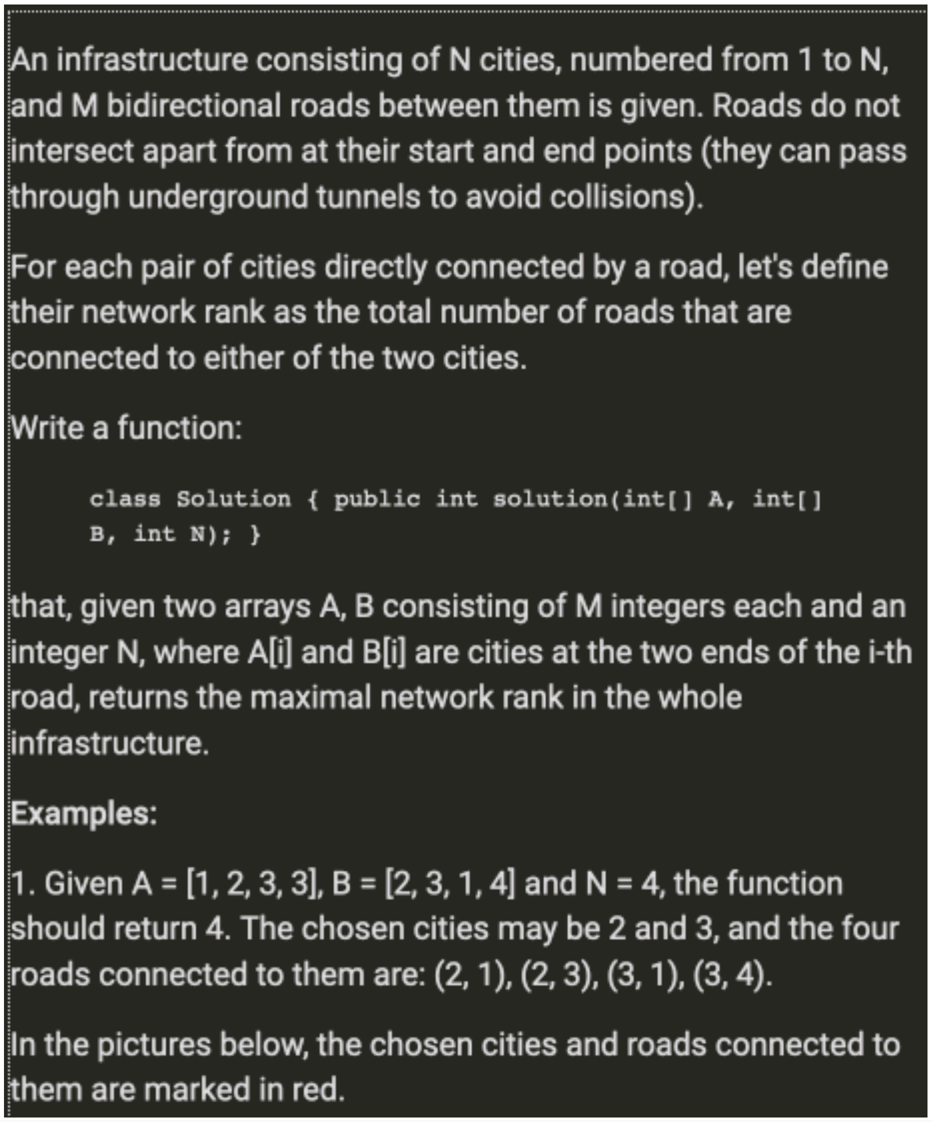
Microsoft | OA 2019 | Max Network Rank

3

captaincode826

Last Edit: October 5, 2019 7:52 PM

2.3K VIEWS



An infrastructure consisting of N cities, numbered from 1 to N, and M bidirectional roads between them is given. Roads do not intersect apart from at their start and end points (they can pass through underground tunnels to avoid collisions).

For each pair of cities directly connected by a road, let's define their network rank as the total number of roads that are connected to either of the two cities.

Write a function:

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

that, given two arrays A, B consisting of M integers each and an integer N, where A[i] and B[i] are cities at the two ends of the i-th road, returns the maximal network rank in the whole infrastructure.

Examples:

1. Given A = [1, 2, 3, 3], B = [2, 3, 1, 4] and N = 4, the function should return 4. The chosen cities may be 2 and 3, and the four roads connected to them are: (2, 1), (2, 3), (3, 1), (3, 4).

In the pictures below, the chosen cities and roads connected to them are marked in red.

```
// "static void main" must be defined in a public class.

public class Main {

    public static void main(String[] args) {
        System.out.println(countEdges(new int[]{1,2,3,3}, new int[]{2,3,1,4}, 4));

        System.out.println(countEdges(new int[]{1,2,4,5}, new int[]{2,3,5,6}, 6));

    }



    public static int countEdges(int[] A, int[] B, int N){

        if (A == null || B == null || A.length == 0 || B.length == 0 || A.length != 
B.length){

            return 0;

        }



        //Form an adjacency list

        Map<Integer, List<Integer>> map = new HashMap<>();

        for (int i=0; i<A.length; i++){

            if(!map.containsKey(A[i])){

                map.put(A[i], new ArrayList<Integer>());

            }

            if(!map.containsKey(B[i])){

                map.put(B[i], new ArrayList<Integer>());

            }

            map.get(A[i]).add(B[i]);

            map.get(B[i]).add(A[i]);

        }



        // Iterate through all nodes and perform DFS on the node not present in seen 
set

        Set<Integer> seen = new HashSet<>();

        int res = 0;

        for(int j=1; j<=N; j++){
            if(!seen.contains(j)){

                int edges = dfs(seen, map, j);

                res = Math.max(res, edges);

            }

        }

        // Since each edge is counted twice in the dfs method we return res/2

        return res/2;

    }


    public static int dfs(Set<Integer> seen, Map<Integer, List<Integer>> map, int
cur){

        if (seen.contains(cur) || !map.containsKey(cur)){

            return 0;

        }


        seen.add(cur);

        List<Integer> nodes = new ArrayList<>();

        nodes = map.get(cur);

        int res = nodes.size();


        for(Integer node : nodes){

            if (!seen.contains(node)){

                res += dfs(seen, map, node);

            }

        }


        return res;

    }
```

```
}
```

4. https://leetcode.com/discuss/interview-question/351783/

Microsoft | OA 2019 | Min Swaps to Make Palindrome

8

rithiksingh09828

Last Edit: October 5, 2019 7:46 PM

3.6K VIEWS

Given a string, what is the minimum number of **adjacent swaps** required to convert a string into a palindrome. If not possible, return `-1`.

Practice online: https://www.codechef.com/problems/ENCD12

Example 1:

```
Input: "mamad"

Output: 3
```

Example 2:

```
Input: "asflkj"

Output: -1
```

Example 3:

```
Input: "aabb"

Output: 2
```

Example 4:

```
Input: "ntiin"

Output: 1

Explanation: swap 't' with 'i' => "nitin"
```

Have someone managed to solve this in less than O(n^2)??

My Java solution, seems to be O(n^2)

```
public static int minSwapPalindrome(String s) {

  if (s == null) throw new IllegalArgumentException();

  if (!canFormPalindrome(s)) return -1;


  int n = s.length(), swaps = 0;

  StringBuilder sb = new StringBuilder(s);


  for (int i = 0; i < n / 2; i++) {

    boolean found = false;

    for (int j = n - i - 1; j > i; j--) {

      //System.out.println(i + ":" + j);

      if (sb.charAt(j) == sb.charAt(i)) {

        found = true;

        for (int k = j; k < n - i - 1; k++) {

          swap(sb, k, k + 1);

          swaps++;

        }

        break;

      }

    }


    if (!found && n % 2 != 0) {

      for (int k = i; k < n/2; k++) {

        swap(sb, k, k + 1);

        swaps++;
      }
    }
  }

  return swaps;
}

private static void swap(StringBuilder sb, int i, int j) {
  char c = sb.charAt(i);
  sb.setCharAt(i, sb.charAt(j));
  sb.setCharAt(j, c);
}


// only for lowercase letter
private static boolean canFormPalindrome(String s) {
  int[] counts = new int[26];
  for (char c : s.toCharArray())
    counts[c - 'a']++;

  boolean hasOdd = false;
  for (int count : counts) {
    if (count % 2 == 0) continue;
    else {
      if (hasOdd)
        return false;
      hasOdd = true;
    }

  }



  return true;

}
```

5. https://leetcode.com/discuss/interview-question/398031/

Given a string `s` containing only `a` and `b`, find longest substring of `s` such that `s` does not contain more than two contiguous occurrences of `a` and `b`.

Example 1:

```
Input: "aabbaaaaabb"

Output: "aabbaa"
```

Example 2:

```
Input: "aabbaabbaabbaa"

Output: "aabbaabbaabbaa"
```

Time complexity O(N)

Space complexity O(1)

```
public static String validLongestSubstring(String s) {

        if (s.length() < 3)

                return s;

        int cur = 0, end = 1;

        char c = s.charAt(0);

        int count = 1;

        int maxLen = 1;

        int start = 0;

        while (end < s.length()) {

                if (s.charAt(end) == c) {

                        count ++;

                        if (count == 2) {

                                if (end - cur + 1 > maxLen) {

                                        maxLen = end - cur + 1;

                                        start = cur;

                                }
```

```
            }
            else {
                cur = end - 1;
            }
        }
        else {
            c = s.charAt(end);
            count = 1;
            if (end - cur + 1 > maxLen) {
                maxLen = end - cur + 1;
                start = cur;
            }
        }
        end ++;
    }
    return s.substring(start, start + maxLen);
}
```

6. https://leetcode.com/discuss/interview-question/366869/

Lexicographically smallest string formed by removing at most one character.

Example 1:

```
Input: "abczd"

Output: "abcd"
```

```
public static void main(String[] args) {

        String s1 = "abczd";

        System.out.println(getSmallString(s1));

        String s2 = "abcde";

        System.out.println(getSmallString(s2));

}


private static String getSmallString(String s) {

        StringBuilder sb = new StringBuilder();

        Stack<Character> stack = new Stack<>();

        int cnt = 0;

        for(int i=0;i<s.length();i++) {

                char c  = s.charAt(i);

                if(!stack.isEmpty() && stack.peek() > c && cnt < 1) {
                        stack.pop();

                        cnt++;

                }

                stack.push(c);

        }

        if(cnt == 0)

                stack.pop();

        while(!stack.isEmpty()) {

                sb.insert(0,  stack.pop());

        }

        return sb.toString();

}
```

---

abcd
abcd
1
Reply
Share

7. https://leetcode.com/discuss/interview-question/398035/

Given a string `s` consisting of `n` lowercase letters, you have to delete the minimum number of characters from `s` so that every letter in `s` appears a unique number of times. We only care about the occurrences of letters that appear at least once in result.

Example 1:

```
Input: "eeeeffff"

Output: 1

Explanation:

We can delete one occurence of e or one occurence of 'f'. Then one letter will occur
four times and the other three times.
```

Example 2:

```
Input: "aabbffddeaee"

Output: 6

Explanation:

For example, we can delete all occurences of 'e' and 'f' and one occurence of 'd' to
obtain the word "aabbda".

Note that both 'e' and 'f' will occur zero times in the new word, but that's fine,
since we only care about the letter that appear at least once.
```

Example 3:

```
Input: "llll"

Output: 0

Explanation:

There is no need to delete any character.
```

Example 4:

```
Input: "example"

Output: 4
```

Java solution: Thanks for the idea from @manidh

```
public static void main(String[] args) {
        String s1 = "aaaabbbb";

        System.out.println(s1 + ":" + minDeletion(s1));

        String s2 = "aabbbbccccdddd";

        System.out.println(s2 + ":" + minDeletion(s2));

        String s3 = "aaaaaabbbbbccccddddeeeeee";

        System.out.println(s3 + ":" + minDeletion(s3));

        String s4 = "abcdefghijkl";

        System.out.println(s4 + ":" + minDeletion(s4));

        String s5 = "aaaaaa";

        System.out.println(s5 + ":" + minDeletion(s5));

        String s6 = "aabbffddeaee";

        System.out.println(s6 + ":" + minDeletion(s6));

        String s7 = "llll";

        System.out.println(s7 + ":" + minDeletion(s7));

        String s8 = "example";

        System.out.println(s8 + ":" + minDeletion(s8));

}


private static int minDeletion(String s) {

        Map<Character, Integer> map = new HashMap<>();

        for(char c : s.toCharArray())

                map.put(c, map.getOrDefault(c, 0) + 1);

        Map<Integer, Integer> cnt = new HashMap<>();

        for(char c : map.keySet()) {

                cnt.put(map.get(c), cnt.getOrDefault(map.get(c), 0) + 1);

        }

        Queue<Map.Entry<Integer, Integer>> maxHeap = new PriorityQueue<>((a,
b)->b.getKey() - a.getKey());
```

```
        maxHeap.addAll(cnt.entrySet());

        int res = 0;

        while(maxHeap.size() > 1) {

                Map.Entry<Integer, Integer> e1 = maxHeap.poll();

                Map.Entry<Integer, Integer> e2 = maxHeap.poll();

                res += e1.getValue() - 1;

                e2.setValue(e2.getValue() + e1.getValue() - 1);

                maxHeap.offer(e2);

        }

        Map.Entry<Integer, Integer> lastEntry = maxHeap.poll();

        if(lastEntry.getValue() > lastEntry.getKey()){

                res += lastEntry.getKey() * (lastEntry.getValue() -
lastEntry.getKey());

                lastEntry.setValue(lastEntry.getKey());

        }

        res += getSum(lastEntry.getValue() - 1);

        return res;

}


static int getSum(int n) {

        if(n <= 1)

                return n;

        return n + getSum(n-1);

}
```

--'

aaaabbbb:1

aabbbbcccdddd:3

aaaaaabbbbbccccddddeeeeee:5

abcdefghijkl:11

aaaaaa:0

aabbffddeaee:6

llll:0

example:4

8. https://leetcode.com/discuss/interview-question/398039/

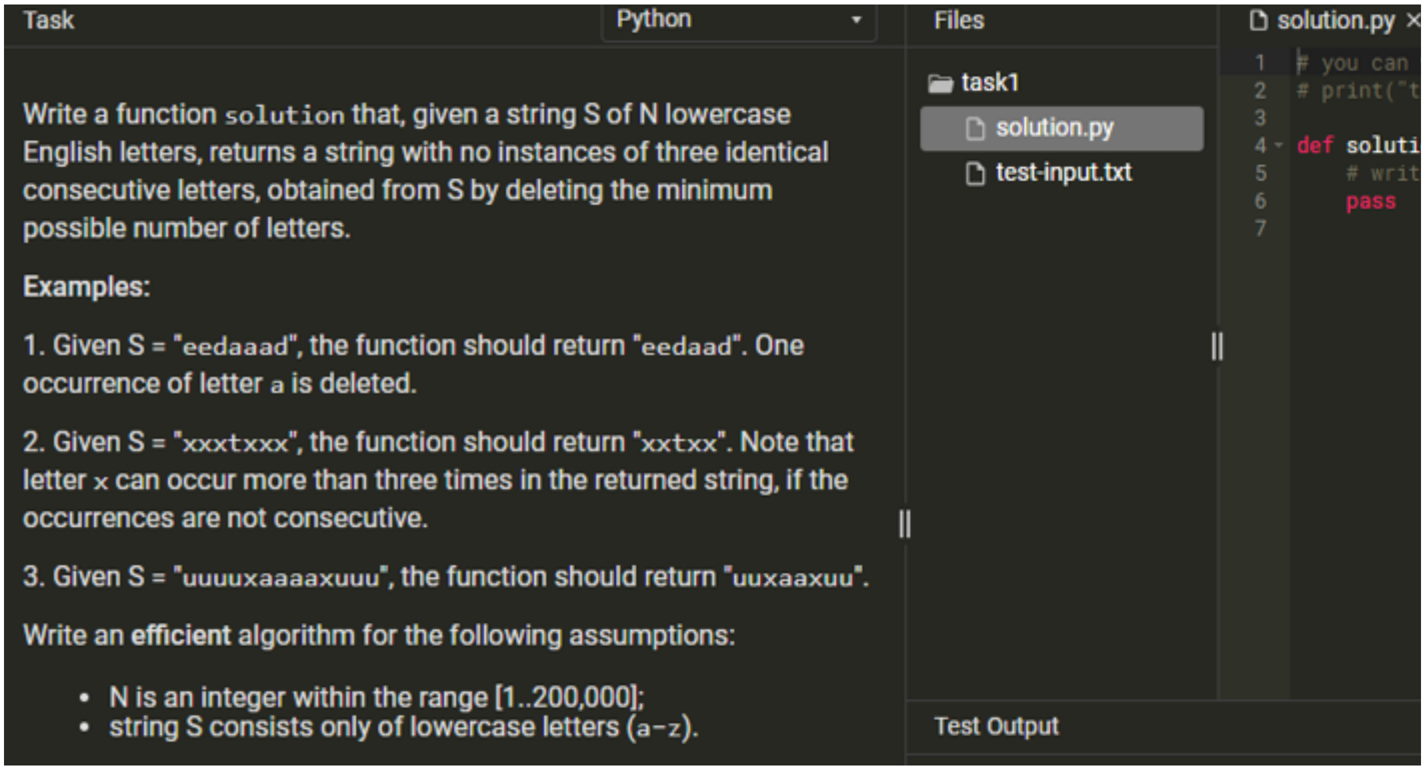
Microsoft | OA 2019 | String Without 3 Identical Consecutive Letters

2

SithisModerator5581

October 5, 2019 8:12 PM

1.2K VIEWS



Write a function `solution` that, given a string S of N lowercase English letters, returns a string with no instances of three identical consecutive letters, obtained from S by deleting the minimum possible number of letters.

Examples:

1. Given S = "`eedaaad`", the function should return "`eedaad`". One occurrence of letter `a` is deleted.

2. Given S = "`xxxtxxx`", the function should return "`xxtxx`". Note that letter `x` can occur more than three times in the returned string, if the occurrences are not consecutive.

3. Given S = "`uuuuxaaaaxuuu`", the function should return "`uuxaaxuu`".

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..200,000];
- string S consists only of lowercase letters (`a-z`).

```
# you can
# print("t
def soluti
    # writ
    pass
```

Java Solution:

```
public static void main(String[] args) {

        String s1 = "eedaaad";

        String s2 = "xxxtxxx";

        String s3 = "uuuuxaaaaxuuu";

        System.out.println(getLongestSubstring(s1));

        System.out.println(getLongestSubstring(s2));

        System.out.println(getLongestSubstring(s3));
```

```java
}


private static String getLongestSubstring(String s) {
        StringBuilder sb = new StringBuilder();
        sb.append(s.charAt(0));
        int cnt = 1;
        for(int r=1;r<s.length();r++) {
                char c = s.charAt(r);
                if(c == s.charAt(r-1))
                        cnt++;
                else {
                        cnt = 1;
                }
                if(cnt < 3)
                        sb.append(c);
        }
        return sb.toString();
}
```

---

eedaad

xxtxx

uuxaaxuu

9. https://leetcode.com/discuss/interview-question/398037/

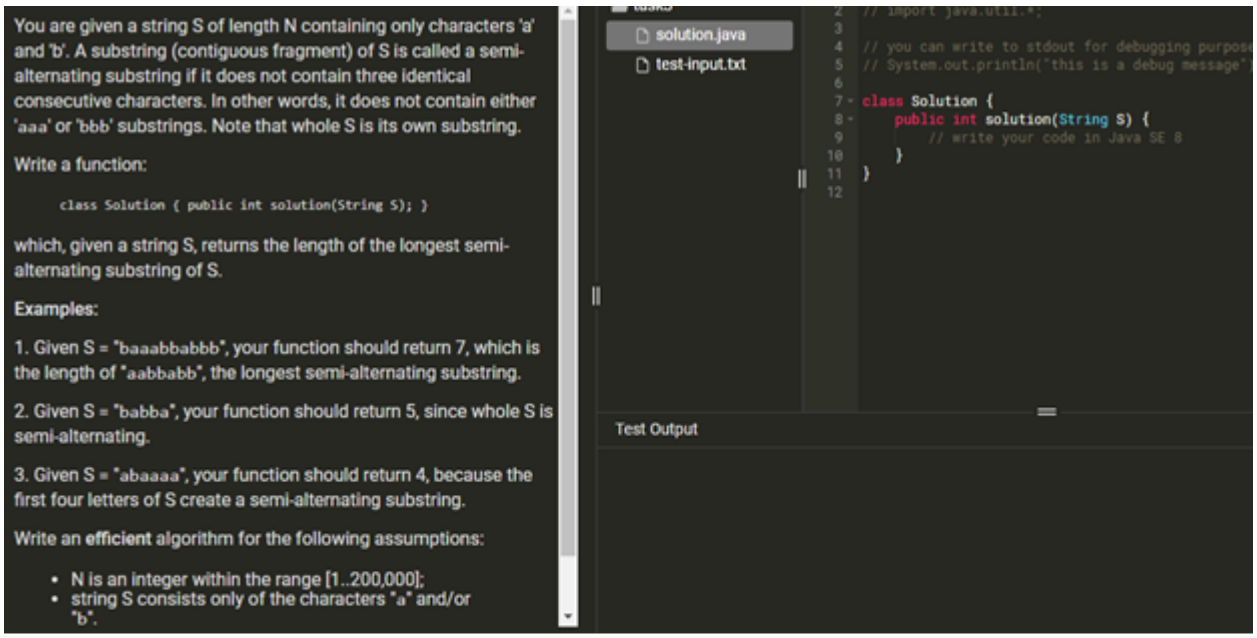
Microsoft | OA 2019 | Longest Semi-Alternating Substring

2

SithisModerator5581

October 5, 2019 8:10 PM

1.3K VIEWS



You are given a string S of length N containing only characters 'a' and 'b'. A substring (contiguous fragment) of S is called a semi-alternating substring if it does not contain three identical consecutive characters. In other words, it does not contain either 'aaa' or 'bbb' substrings. Note that whole S is its own substring.

Write a function:

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

which, given a string S, returns the length of the longest semi-alternating substring of S.

Examples:

1. Given S = "baaabbabbb", your function should return 7, which is the length of "aabbabb", the longest semi-alternating substring.

2. Given S = "babba", your function should return 5, since whole S is semi-alternating.

3. Given S = "abaaaa", your function should return 4, because the first four letters of S create a semi-alternating substring.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..200,000];
- string S consists only of the characters "a" and/or "b".

```
// import java.util.*;

// you can write to stdout for debugging purpose
// System.out.println("this is a debug message")

class Solution {
    public int solution(String S) {
        // write your code in Java SE 8
    }
}
```

Comments: 7

Java solution:

```
public static void main(String[] args) {

        String s1 = "baaabbabbb";

        String s2 = "babba";

        String s3 = "abaaaa";

        System.out.println(getLongest(s1));

        System.out.println(getLongest(s2));

        System.out.println(getLongest(s3));
}


private static int getLongest(String s) {
        int cnt = 1, l = 0, lastSeen = 0;
        int res = 0;
        for(int r = 1;r < s.length();r++) {
                char c = s.charAt(r);
                if(s.charAt(r-1) == c) {
                        cnt++;
                }else {
                        cnt = 1;
                        lastSeen = r;
                }
                if(cnt > 2 && l < lastSeen)
                        l = lastSeen;
                while(cnt > 2) {
                        l++;
                        cnt--;
                }
                res = Math.max(res, r - l + 1);
        }
        return res;
}
```

7

5

4

10. https://leetcode.com/discuss/interview-question/364618/

Microsoft | OA 2019 | Min Steps to Make Piles Equal Height

6

Anonymous User

Last Edit: August 22, 2019 3:43 PM

1.7K VIEWS

Alexa is given `n` piles of equal or unequal heights. In one step, Alexa can remove any number of boxes from the pile which has the maximum height and try to make it equal to the one which is just lower than the maximum height of the stack. Determine the minimum number of steps required to make all of the piles equal in height.

Example 1:

```
Input: piles = [5, 2, 1]

Output: 3

Explanation:

Step 1: reducing 5 -> 2 [2, 2, 1]

Step 2: reducing 2 -> 1 [2, 1, 1]

Step 3: reducing 2 -> 1 [1, 1, 1]

So final number of steps required is 3.
```

Java O(NlogN) solution with playground.

For `piles = [5, 2, 1]`, 5 needs 2 steps to come to 1(5 –> 2 –> 1) and 2 needs 1 step to 1(2 –> 1) and 1 for 0 step. We just need to sort the array and record how many different numbers appeared before and sum them up.

```
    public int minSteps(int[] piles){

        int len = piles.length;

        if(len <= 1) return 0;

        Arrays.sort(piles);

        int res = 0, distinctNums = 0;

        for(int i = 1; i < len; ++i){
            if(piles[i] == piles[i - 1]){

                res += distinctNums;

            }

            else{

                ++distinctNums;

                res += distinctNums;

            }

        }

        return res;

    }
```

11. https://leetcode.com/discuss/interview-question/398047/

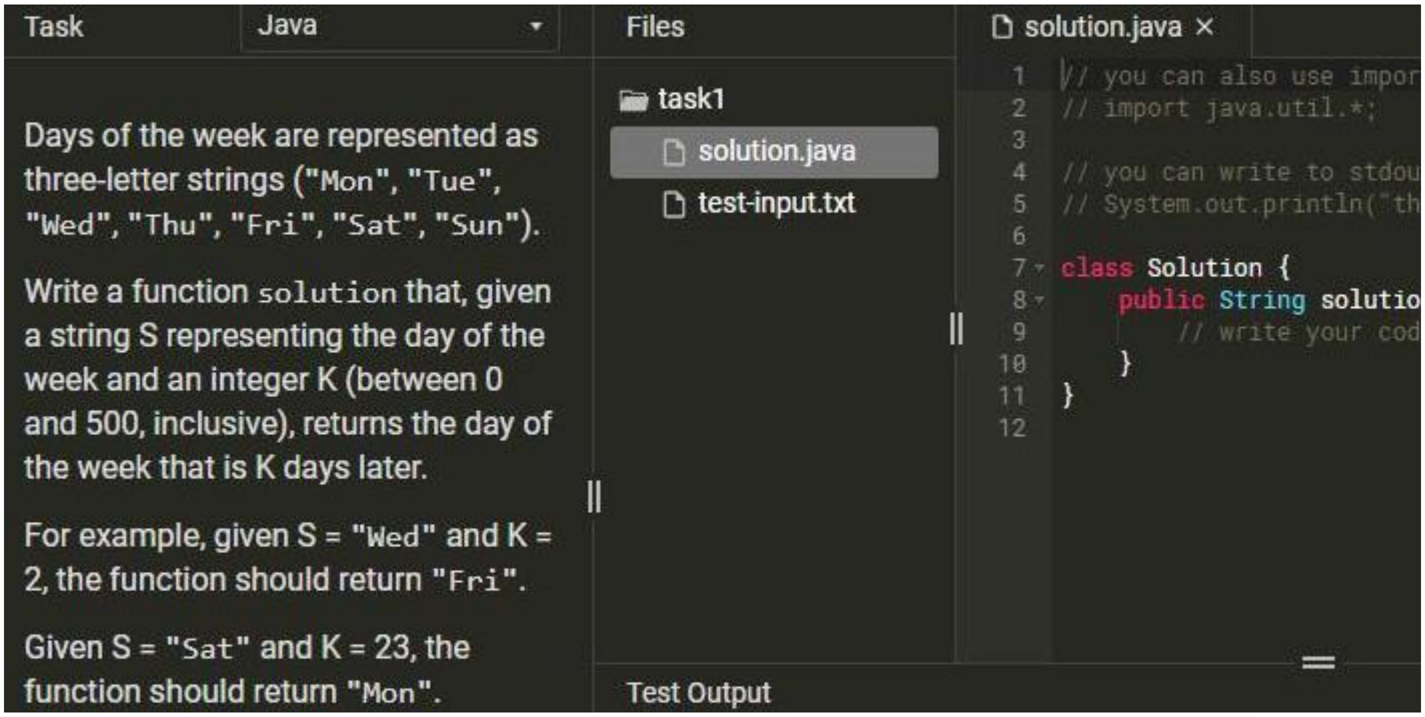
Microsoft | OA 2019 | Day of Week

4

SithisModerator5581

October 5, 2019 8:24 PM

1.4K VIEWS



Days of the week are represented as three-letter strings ("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun").

Write a function solution that, given a string S representing the day of the week and an integer K (between 0 and 500, inclusive), returns the day of the week that is K days later.

For example, given S = "Wed" and K = 2, the function should return "Fri".

Given S = "Sat" and K = 23, the function should return "Mon".

```java
// you can also use impor
// import java.util.*;

// you can write to stdou
// System.out.println("th

class Solution {
    public String solutio
        // write your cod
    }
}
```

Comments: 5

Java O(1) solution

```java
  public static String dayAfter(String day, int k){


        String[] days = new String[]{"Mon", "Tue", "Wed", "Thu", "Fri", "Sat",
"Sun"};

        Map<String, Integer> dayMap = new HashMap<>();


        for( int i=0; i<days.length; i++)

            dayMap.put(days[i], i);
```

```
    int numOfDay = (k+dayMap.get(day)) % 7;

    return days[numOfDay];

}
```

12. https://leetcode.com/discuss/interview-question/398050/

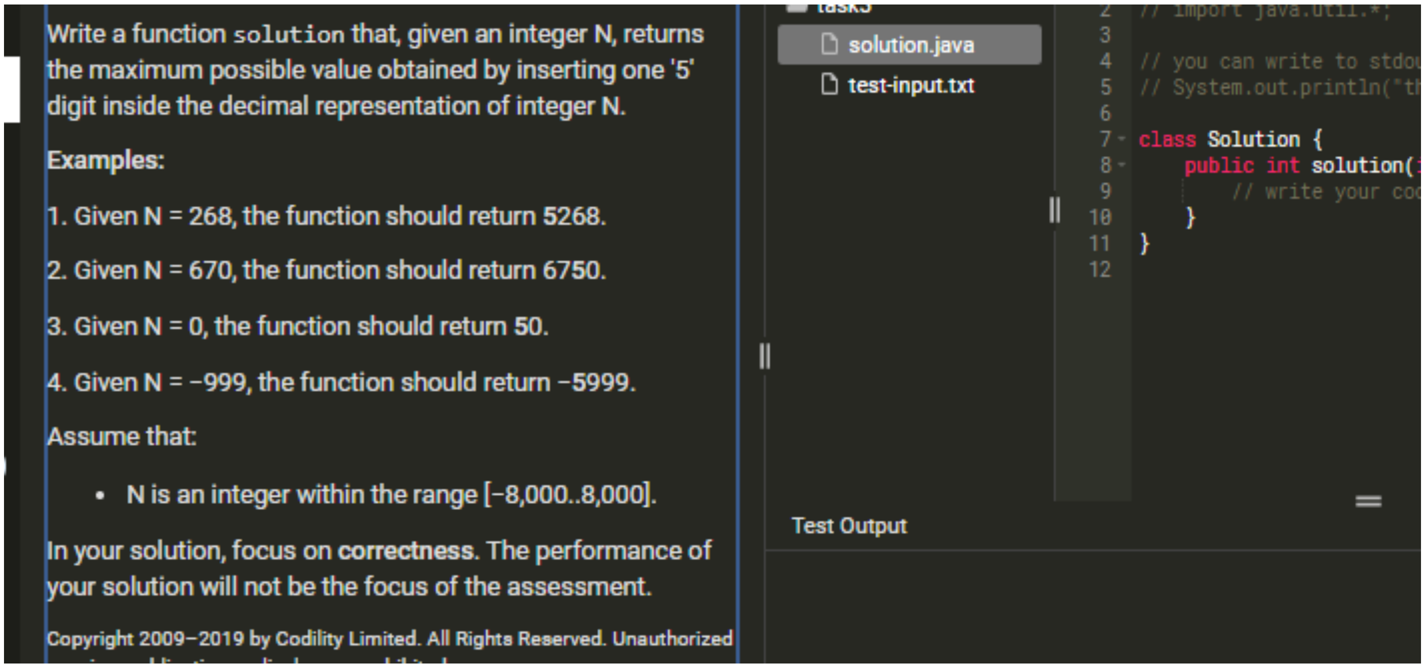
Microsoft | OA 2019 | Max Possible Value

1

SithisModerator5581

October 5, 2019 8:26 PM

1.3K VIEWS



Write a function solution that, given an integer N, returns the maximum possible value obtained by inserting one '5' digit inside the decimal representation of integer N.

Examples:

1. Given N = 268, the function should return 5268.
2. Given N = 670, the function should return 6750.
3. Given N = 0, the function should return 50.
4. Given N = −999, the function should return −5999.

Assume that:

- N is an integer within the range [−8,000..8,000].

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Copyright 2009–2019 by Codility Limited. All Rights Reserved. Unauthorized

```
// import java.util.*;

// you can write to stdou
// System.out.println("th

class Solution {
    public int solution(
        // write your coo
    }
}
```

Slightly messy but here was my go at it in Java

```
public static void main(String[] args)

{

    System.out.println(maxDidget(268));

    System.out.println(maxDidget(670));

    System.out.println(maxDidget(0));

    System.out.println(maxDidget(-999));

    System.out.println(maxDidget(-462));

    System.out.println(maxDidget(000));
```

```
}


public static int maxDidget(int N)

{

    String strN = N + "";

    final int i = helper(strN, isNeg(N));


    return i;

}


public static Integer helper(String n, boolean neg)

{

    String s = "";

    boolean changed = false;


    if(neg)

    {

        // go as far right as possible till next number is greater than 5

        for(int i = 0; i < n.length(); i++)

        {

            if(n.charAt(i) < '5')

            {

                s+= n.charAt(i);

            }

            else

            {

                s += '5' + n.substring(i);
}
            changed = true;
            break;
        }
    }

    if(!changed)
        s += '5';
}
else
{
    // add to front of first number less than 5
    for(int i = 0; i < n.length(); i++)
    {
        if(n.charAt(i) > '5')
        {
            s+= n.charAt(i);
        }
        else
        {
            s += '5' + n.substring(i);
            changed = true;
            break;
        }
    }
    if(!changed)
        s += '5';
}
```

```
    return Integer.parseInt(s);

}



public static boolean isNeg(int N)

{

    if(N < 0)

        return true;



    return false;

}
```

13. https://leetcode.com/discuss/interview-question/398056/

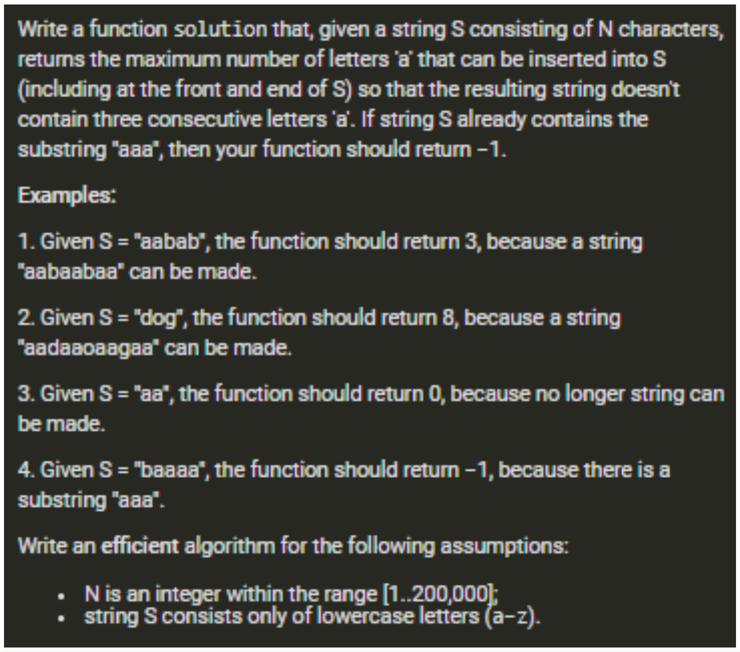
Microsoft | OA 2019 | Max Inserts to Obtain String Without 3 Consecutive 'a'

0

SithisModerator5581

October 5, 2019 8:29 PM

1.1K VIEWS


Write a function `solution` that, given a string S consisting of N characters, returns the maximum number of letters 'a' that can be inserted into S (including at the front and end of S) so that the resulting string doesn't contain three consecutive letters 'a'. If string S already contains the substring "aaa", then your function should return −1.

Examples:

1. Given S = "aabab", the function should return 3, because a string "aabaabaa" can be made.

2. Given S = "dog", the function should return 8, because a string "aadaaoaagaa" can be made.

3. Given S = "aa", the function should return 0, because no longer string can be made.

4. Given S = "baaaa", the function should return −1, because there is a substring "aaa".

Write an efficient algorithm for the following assumptions:

- N is an integer within the range [1..200,000];
- string S consists only of lowercase letters (a−z).

```
public int solution(String input){

    if(input.length() == 0)

        return 2;

    int result = 0;

    char c = input.charAt(0);

    int countOfAs = c == 'a'?1:0;

    result += c != 'a'?2:0;

    for (int i = 1; i <= input.length(); i++) {

        if(i == input.length()){
            result += 2-countOfAs;

            break;

        }

        c = input.charAt(i);

        if(c != 'a'){

            result += 2-countOfAs;

            countOfAs = 0;


        }else{

            countOfAs++;

        }

        if(countOfAs>=3)

            return -1;


    }

    return result;

}
```

14. https://leetcode.com/discuss/interview-question/401826/

6

Anonymous User

Last Edit: October 24, 2019 2:52 PM

2.2K VIEWS

Given an Array A consisting of N Strings, calculate the length of the longest string S such that:

- S is a concatenation of some of the Strings from A.
- every letter in S is different.

Example –

A = ["co","dil","ity"] , function should return 5, resulting string S could be codil , dilco, coity,ityco

A = ["abc","kkk","def","csv"] , returns 6 , resulting Strings S could be abcdef , defabc, defcsv , csvdef

A = ["abc","ade","akl"] , return 0 , impossible to concatenate as letters wont be unique.

N is [1..8] ; A consists of lowercase English letters ; sum of length of strings in A does not exceed 100.

Should be NlogN.

Intuition – Start with a longest string , if it has all letters unique that means its length should definitely be a part of answer !!

```
static void findConcatenated(String[] input) {

                Arrays.sort(input,new Comparator<String>(){

                        public int compare(String s1, String s2) {

                       return s2.length()- s1.length();

                    }

                });

                HashMap <Character,Integer> characterInputAddress = new HashMap<>();

                for (int i = 0; i < input.length; i++) {

                        String curr = input[i];
                        if(curr.length() != findUniqueWordLength(curr)) continue;
                        boolean charaterExists = false;
                        for(int j = 0 ; j < curr.length() ; j++) {

        if(characterInputAddress.containsKey(curr.charAt(j))) {
                                        charaterExists = true;
                                        break;
                                }
                        }
                        if(!charaterExists) {
                                for(int j = 0 ; j < curr.length() ; j++) {

        characterInputAddress.put(curr.charAt(j),i);
                                }
                        }
                }
                System.out.println(characterInputAddress.size());
        }


static int findUniqueWordLength(String s) {
                HashSet<Character> currWithoutDuplicates = new HashSet<>();
                for (int i = 0; i < s.length(); i++) {
                        currWithoutDuplicates.add(s.charAt(i));
                }
                return currWithoutDuplicates.size();
        }
```

15. https://leetcode.com/discuss/interview-question/406031/

Write a function that, given an array A of N integers, returns the lagest integer K > 0 such that both values K and –K exisit in array A. If there is no such integer, the function should return 0.

Example 1:

```
Input: [3, 2, -2, 5, -3]

Output: 3
```

Example 2:

```
Input: [1, 2, 3, -4]

Output: 0
```

ava solution:

```
public static void main(String[] args) {

        int[] nums1 = { 3, 2, -2, 5, -3 };

        int[] nums2 = { 1, 2, 3, -4 };

        System.out.println(largestNum(nums1));

        System.out.println(largestNum(nums2));

        System.out.println("-------------------------------------");

        System.out.println(largestNum2(nums1));

        System.out.println(largestNum2(nums2));

}


private static int largestNum(int[] nums) {
        int res = 0;

        Set<Integer> set = new HashSet<>();

        for(int i=0;i<nums.length;i++) {

                set.add(-nums[i]);

                if(set.contains(nums[i])) {

                        res = Math.max(res, Math.abs(nums[i]));

                }

        }

        return res;

}


private static int largestNum2(int[] nums) {

        int res = 0;

        Arrays.sort(nums);

        int l = 0, r = nums.length -1;

        while(l < r) {

                int sum = nums[l] + nums[r];

                if(sum == 0) {

                        res = Math.max(res, Math.max(nums[l], nums[r]));

                        l++;

                        r--;

                }

                else if(sum < 0) {

                        l++;

                }else {

                        r--;

                }
        }

        return res;

}
```

3

0

3

0

16. https://leetcode.com/discuss/interview-question/414660/

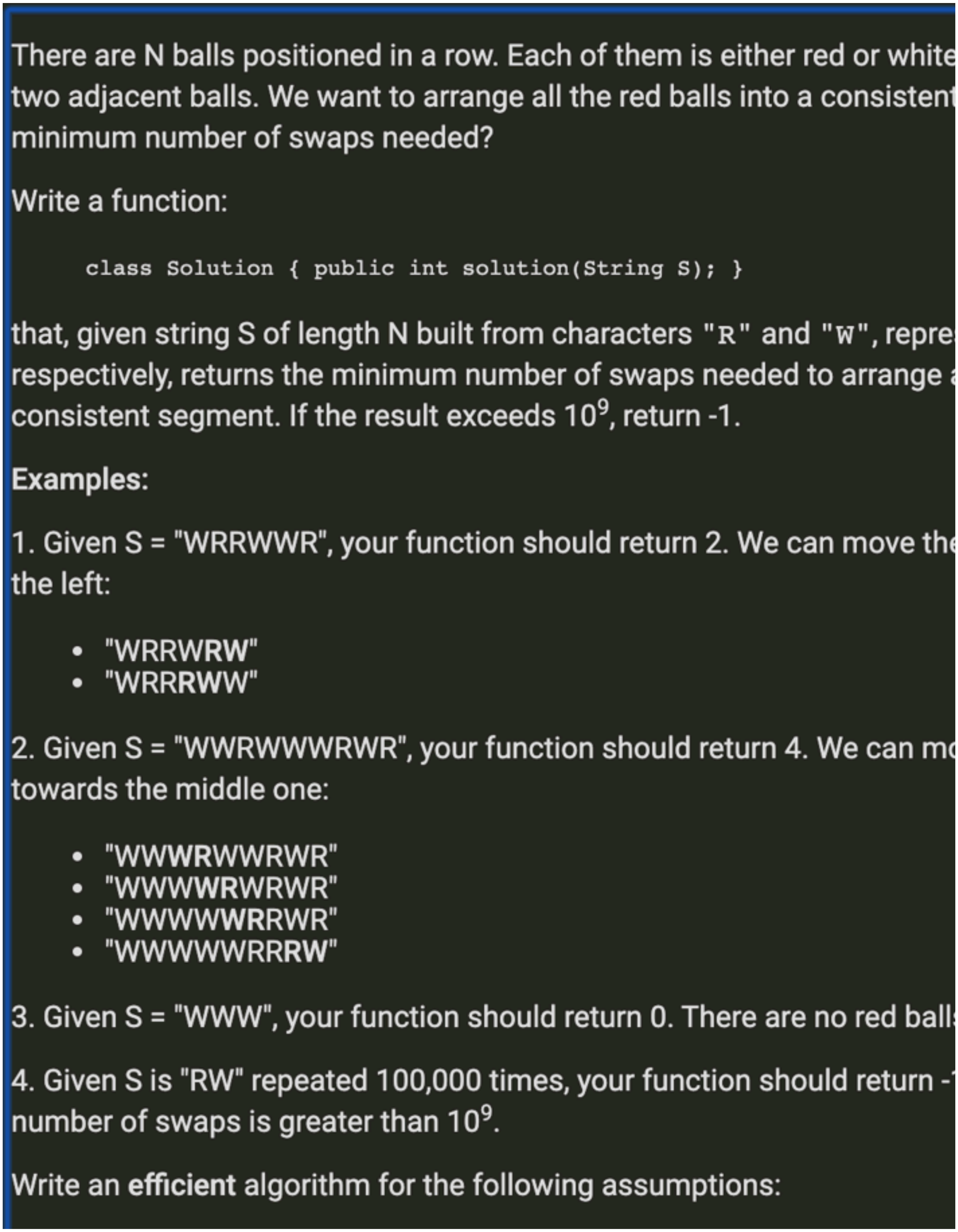
Microsoft | OA 2019 | Min Swaps to Group Red Balls

1

Anonymous User

Last Edit: 14 hours ago

597 VIEWS



There are N balls positioned in a row. Each of them is either red or white
two adjacent balls. We want to arrange all the red balls into a consistent
minimum number of swaps needed?

Write a function:

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

that, given string S of length N built from characters `"R"` and `"W"`, repre
respectively, returns the minimum number of swaps needed to arrange
consistent segment. If the result exceeds 10^9, return -1.

Examples:

1. Given S = "WRRWWR", your function should return 2. We can move the
the left:

- "WRRW**RW**"
- "WRR**RW**W"

2. Given S = "WWRWWWRWR", your function should return 4. We can mo
towards the middle one:

- "WW**WR**WWRWR"
- "WWW**WR**WRWR"
- "WWWW**WR**RWR"
- "WWWWWRR**RW**"

3. Given S = "WWW", your function should return 0. There are no red ball

4. Given S is "RW" repeated 100,000 times, your function should return -
number of swaps is greater than 10^9.

Write an **efficient** algorithm for the following assumptions:

Example:

```
Input: "RRRWRR"

Output: 2
```

Related problems:

- Minimum swaps need to make K girls sitting together

```java
public static int solution(String s) {
    List<Integer> redIndices = getRedIndices(s);
    int mid = redIndices.size() / 2;
    int minSwaps = 0;
    for (int i = 0; i < redIndices.size(); i++) {
        // number of swaps for each R is the distance to mid, minus the number of R's between them
        minSwaps += Math.abs(redIndices.get(mid) - redIndices.get(i)) - Math.abs(mid - i);
    }
    return minSwaps;
}

private static List<Integer> getRedIndices(String s) {
    List<Integer> indices = new ArrayList<>(s.length());
    for (int i = 0; i < s.length(); i++) {
        if (s.charAt(i) == 'R') {
            indices.add(i);
        }
    }
    return indices;
}
```

17. https://leetcode.com/problems/maximum-length-of-a-concatenated-string-with-unique-characters/

1239. Maximum Length of a Concatenated String with Unique Characters

Medium

7414FavoriteShare

Given an array of strings `arr`. String `s` is a concatenation of a sub-sequence of `arr` which have **unique characters**.

Return *the maximum possible length* of `s`.

Example 1:

```
Input: arr = ["un","iq","ue"]

Output: 4

Explanation: All possible concatenations are "","un","iq","ue","uniq" and "ique".

Maximum length is 4.
```

Example 2:

```
Input: arr = ["cha","r","act","ers"]

Output: 6

Explanation: Possible solutions are "chaers" and "acters".
```

Example 3:

```
Input: arr = ["abcdefghijklmnopqrstuvwxyz"]

Output: 26
```

Constraints:

- `1 <= arr.length <= 16`
- `1 <= arr[i].length <= 26`
- `arr[i]` contains only lower case English letters.

- I think you can optimize a bit by adding memory to it. Time was limited during the contest, I didn't have time to optimized it.

```
class Solution {
    private int result = 0;

    public int maxLength(List<String> arr) {
        if (arr == null || arr.size() == 0) {
            return 0;
        }

        dfs(arr, "", 0);

        return result;
    }

    private void dfs(List<String> arr, String path, int idx) {
        boolean isUniqueChar = isUniqueChars(path);

        if (isUniqueChar) {
            result = Math.max(path.length(), result);
        }

        if (idx == arr.size() || !isUniqueChar) {
            return;
        }

        for (int i = idx; i < arr.size(); i++) {
            dfs(arr, path + arr.get(i), i + 1);
        }
    }

    private boolean isUniqueChars(String s) {
        Set<Character> set = new HashSet<>();

        for (char c : s.toCharArray()) {
            if (set.contains(c)) {
                return false;
            }
            set.add(c);
        }
        return true;
    }
}
```

18. https://leetcode.com/discuss/interview-question/414880/

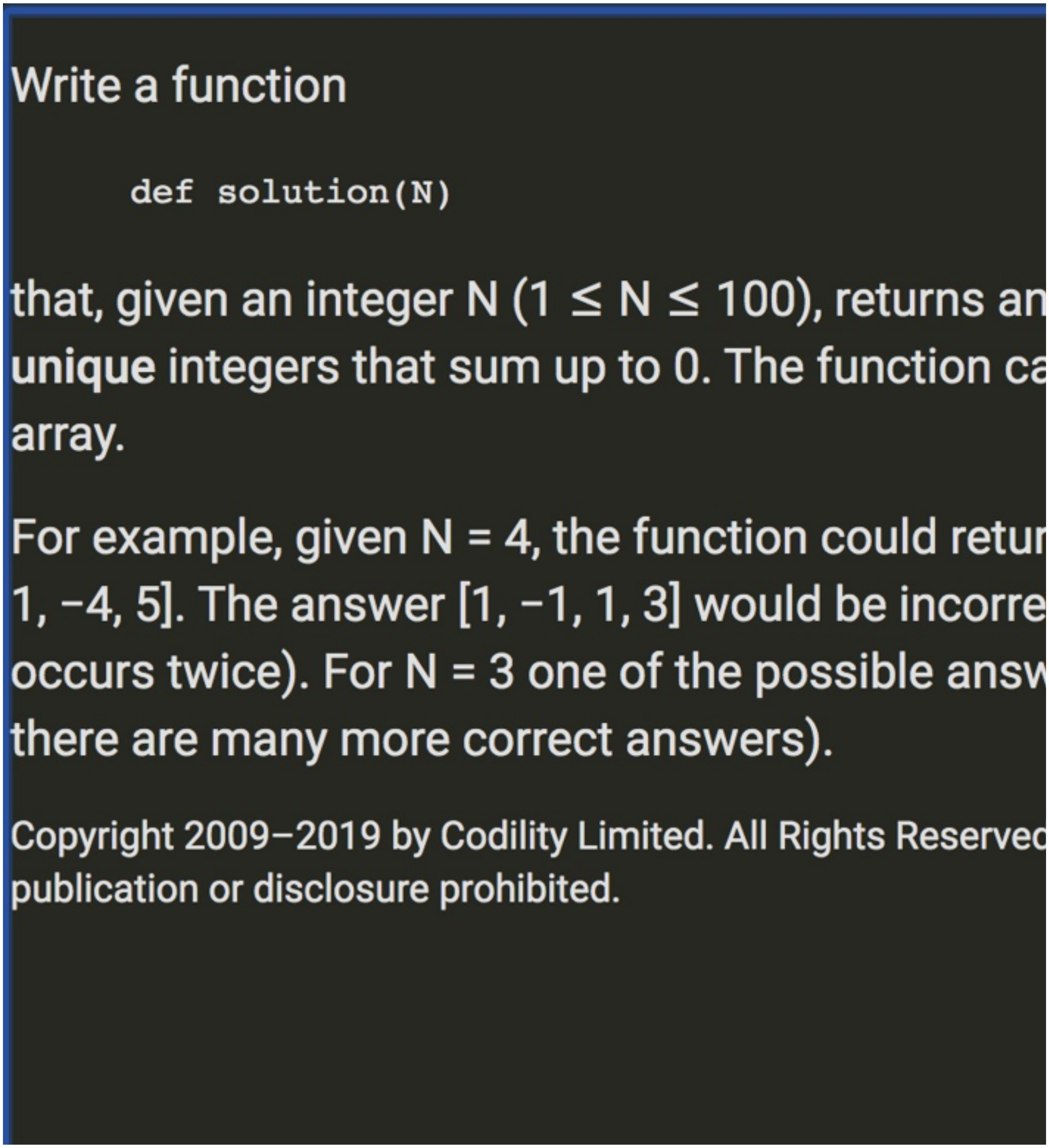
Microsoft | OA 2019 | Unique Integers That Sum Up To 0

0

SithisModerator5581

Last Edit: October 27, 2019 7:20 PM

517 VIEWS


Write a function

```
def solution(N)
```

that, given an integer N (1 ≤ N ≤ 100), returns an
unique integers that sum up to 0. The function ca
array.

For example, given N = 4, the function could retur
1, −4, 5]. The answer [1, −1, 1, 3] would be incorre
occurs twice). For N = 3 one of the possible answ
there are many more correct answers).

Java O(N)

```java
public static List<Integer> solution(int n)

{

    int remaining = n;

    int start = 1;

    List<Integer> list = new ArrayList<Integer>();


    // if odd we need a 0

    if(n % 2 != 0)

    {

        list.add(0);

        remaining--;

    }


    while(remaining > 0)

        {

            list.add(start);

            list.add(-start);

            start++;

            remaining -= 2;

        }

    return list;

}
```

19. https://leetcode.com/discuss/interview-question/421975/

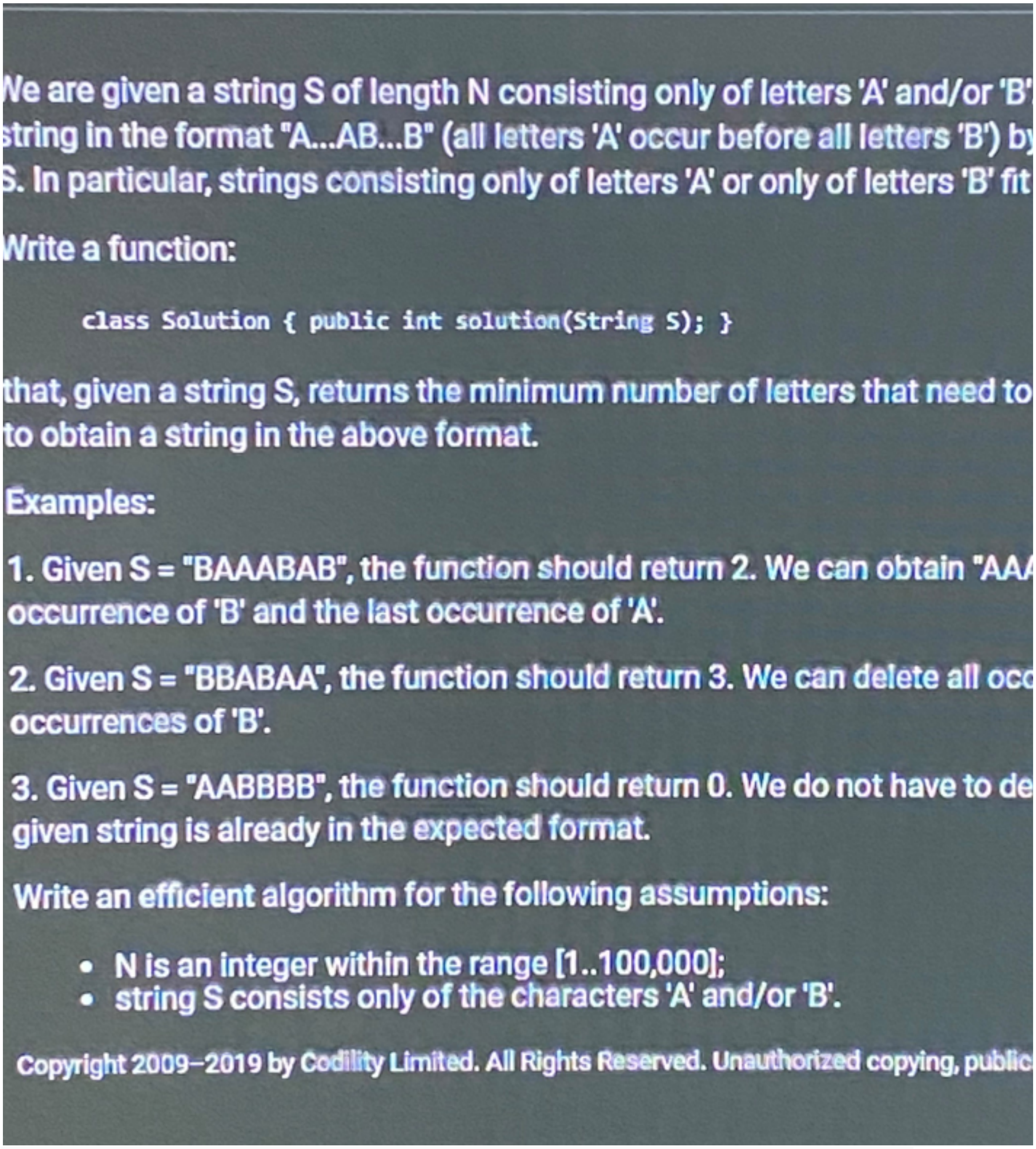
Microsoft | OA 2019 | Min Deletions To Obtain String in Right Format

8

Anonymous User

Last Edit: 14 hours ago

1.8K VIEWS


Ve are given a string S of length N consisting only of letters 'A' and/or 'B'
string in the format "A...AB...B" (all letters 'A' occur before all letters 'B') by
S. In particular, strings consisting only of letters 'A' or only of letters 'B' fit

Write a function:

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

that, given a string S, returns the minimum number of letters that need to
to obtain a string in the above format.

Examples:

1. Given S = "BAAABAB", the function should return 2. We can obtain "AA
occurrence of 'B' and the last occurrence of 'A'.

2. Given S = "BBABAA", the function should return 3. We can delete all occ
occurrences of 'B'.

3. Given S = "AABBBB", the function should return 0. We do not have to de
given string is already in the expected format.

Write an efficient algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- string S consists only of the characters 'A' and/or 'B'.

Comments: 7

Notice we can partition the original string in half, deleting all B's in the left-side and all A's on the right side. Therefore, the trick is to find a constant time computation of the number of Bs in the left partition, and number of As in the right partition. Then we can iterate through all n + 1 partitions, sum the B's and A's, and track the minimum deletions.

For example take the string BAABAB, the partitions are as follows:

| BAABAB –> BBB
B | AABAB –> BB
BA | ABAB –> ABB
BAA | BAB –> AABB
BAAB | AB –> AAB
BAABA | B –> AAAB
BAABAB | –> AAA

Define f(i) = number of Bs up–to index i [exclusive]+ number As after index i [inclusive]
This function is the number of deletion required to transform the string to A...AB...B where index i corresponds to the partition before the first B.

Re–stating the problem: Find f(k) s.t. f(k) <= f(i) for all indices i.

```
int sol(string s) {

  int rhs = 0, lhs = 0;

  for (int i = 0; i < s.size(); ++i) if (s[i] == 'A') ++rhs;

  // rhs equals number of A's after index i [inclusive]

  // lhs equals number of B's before index i [exclusive]

  int ans = rhs;

  for (int i = 0; i < s.size(); ++i) {

    if (s[i] == 'A') --rhs;

    else ++lhs;

    ans = min(ans, rhs + lhs);

  }

  return ans;

}
```