Goldman Sachs OA

1. 给一个字符串s，找出所有Unique的长度为k的substring，并排序。比如caaab，k = 2，返回 aa, ab, ca
- 不重复子串，输入是一个string和一个substr长度，返回所有unique substrs..
返回的是String[]
打出所以这个数字长度的string，按字母排序
find distinct substr

```
public static int numberdss(String str) {
        HashSet<String> all = new HashSet<>();

        for (int i = 0; i < str.length(); i++) {
            for (int j = 0; j <= i; j++) {
                if (!all.contains(str.substring(j, i + 1))) {
                    all.add(str.substring(j, i + 1));
                }
            }
        }
        return all.size();
    }
```

```
public static int numberdss(String str) {
        HashSet<String> all = new HashSet<>();
        HashSet<StringBuilder> last = new HashSet<>();

        for (int i = 0; i < str.length(); i++) {
            for (StringBuilder sb : last) {
                sb.append(str.charAt(i));
                if (!all.contains(sb.toString())) {
                    all.add(sb.toString());
                }
            }
            if (!all.contains(str.charAt(i) + "")) {
                all.add(str.charAt(i) + "");
            }
            last.add(new StringBuilder(str.charAt(i) + ""));
        }
        return all.size();
}
```

2. 给一个数组，每一项是学生名字和分数，学生名字可能有重复，找出所有考试平均分最高的分数。比如 [['bob', '88'], ['ted', '100'], ['ted', '20']]，返回88.
第二题一开始有个case没过，有点tricky，因为分数可能有负数，然后负数需要取floor，所以不能直接用int作除法，要double做除法，然后取floor以后再转成Int. 比如 -123 / 10 = -12，但应该要-13
- Best Average Grade：input是String[][]，两列，第一列是名字，第二列是分数。要计算每一个人的平均分数，返回最高的分数。注意要用double。

4.找一个array中的第二小的数字

```
public static void print2largest(int arr[], int arr_size){
	int i, first, second;
	//there should be at least 2 elements
	if(arr_size < 2){
		return;
	}
	first = second = Integer.MIN_VALUE;
	for(i = 0; i < arr_size; i++){
		//if current element is smaller than both, update the first to it and second to first
		//if the element is in between, then just update the second
		if(arr[i] > first){
			second = first;
			first = arr[i];
		}
		else if(arr[i] > second && arr[i] != first){
			second = arr[i];
		}
	}
	if(second == Integer.MIN_VALUE){
		//NO second largest
	}
	return second;
}
```

3. 爬楼梯，多了个base，一次可以跳1、2、3步。

```
static long countSteps(int n) {
	if(n == 0){
	   return 0;
	}
	int m = 3;
	long res[] = new long[n+1];
	   res[0] = 1;
	   res[1] = 1;
```

```
        for(int i = 2; i < n + 1; i++){
            res[i] = 0;
            for(int j = 1; j <= m && j <=i; j++){
                res[i] += res[i-j];
            }
        }
        return res[n];
  }
```

notes:

```
public class Solution {
    public int climbStairs(int n) {
        if (n <= 1) {
            return n;
        }
        int last = 1, lastlast = 1;
        int now = 0;
        for (int i = 2; i <= n; i++) {
            now = last + lastlast;
            lastlast = last;
            last = now;
        }
        return now;
    }
}
```

m = 2, ways(n) = ways(n-1)+ways(n-2)

generalization: ways(n, m) = ways(n-1, m) + ways(n-2, m) + ... + ways(n-m, m)

```
//a recursive function used by countWays
static int countWaysUtil(int n, int m){
        int res[] = new int[n];
        res[0] = 1;
        res[1] = 1;
        for(int i = 2; i < n; i++){
                res[i] = 0;
                for(int j = 1; j <= m && j <=i; j++){
                        res[i] += res[i-j];
                }
        }
        return res[n-1];
}
//returns number of ways to reach s'th stair
int countWays(int s, int m){
```

```
        return countWaysUtil(s+1, m);
}
```

5.First Unique Char: input是string，找到这个string中第一个unique的cintar，返回string类型

```
static final int NO_OF_CHARS = 256
static char[] count = new char[NO_OF_CHARS];
//calculate count of char in the passed string
static void getCharCountArray(String s){
        for(int i = 0; i < s.length(); i++){
                count[[s.charAt(i)]++;
        }
        return;
}

//if all char repeated, return -1
static int firstNonRepeating(String s){
        getCharCountArray(s);
        int index = -1;
        for(int i = 0; i < s.length(); i++){
                if(count[s.charAt(i)] == 1){
                        index = i;
                        break;
                }
        }
        return index;
}
```

6.isAnagram 可以包含其他符号和空格。

```
 public boolean isAnagram(String s, String t) {
        int[] alphabet = new int[26];
        for (int i = 0; i < s.length(); i++) alphabet[s.charAt(i) - 'a']++;
        for (int i = 0; i < t.length(); i++) alphabet[t.charAt(i) - 'a']--;
        for (int i : alphabet) if (i != 0) return false;
        return true;
    }
```

7.给两个数组求点乘（dot product）

```
 static int dotProduct(int[] x, int[] y) {
        int sum = 0;
        if(x.length == 0 || y.length == 0 || x.length != y.length){
            return sum;
        }
```

```
        int size = x.length;

        for(int i = 0; i < size; i++){
          sum += x[i] * y[i];
        }
        return sum;
    }
```

8.求一个数组中和为k的pair的个数（two sum变形）.
需要注意，重复的组合，元素来自不同index也算一种。
比如 a={1,1,1} k=2，总共有3对，不是1对！！！

```
// Returns number of pairs in arr[0..n-1] with sum equal
    // to 'sum'
    static int getPairsCount(int n, int sum)
    {
        HashMap<Integer, Integer> hm = new HashMap<>();

        // Store counts of all elements in map hm
        for (int i=0; i<n; i++){

            // initializing value to 0, if key not found
            if(!hm.containsKey(arr[i]))
                hm.put(arr[i],0);

            hm.put(arr[i], hm.get(arr[i])+1);
        }
        int twice_count = 0;

        // iterate through each element and increment the
        // count (Notice that every pair is counted twice)
        for (int i=0; i<n; i++)
        {
            if(hm.get(sum-arr[i]) != null)
                twice_count += hm.get(sum-arr[i]);

            // if (arr[i], arr[i]) pair satisfies the condition,
            // then we need to ensure that the count is
            // decreased by one such that the (arr[i], arr[i])
            // pair is not considered
            if (sum-arr[i] == arr[i])
                twice_count--;
        }
```

```
        // return the half of twice_count
        return twice_count/2;
    }
```

9.Compress String(run length encoding): input: aaabbbccccc output: a3b3c5

```
public String compress(String s){
        if(s == null || s.length() == 0){
                return “”;
        }
        char[] chars = s.toCharArray();
        StringBuilder sb = new StringBuilder();

        int count = 1;
        char prev = chars[0];
        for(int i = 1; i < chars.length; i++){
                char c = chars[i];
                if(c == prev){
                        count++;
                }else{
                        sb.append(prev).append(count);
                        count = 1;
                }
                prev = c;
        }
        return sb.append(prev).append(count).toString();
}
```

10. Reverse Sentence

```
public static String reverseWord(String s){
        String words[] = s.split(“\\s”);
        String reverseWord = “”;
        for(String w:words){
                StringBuilder sb = new StringBuilder(w);
                sb.reverse();
                reverseWord += sb.toString() + “ ”;
        }
        return reverseWord.trim();
}
```

oa: