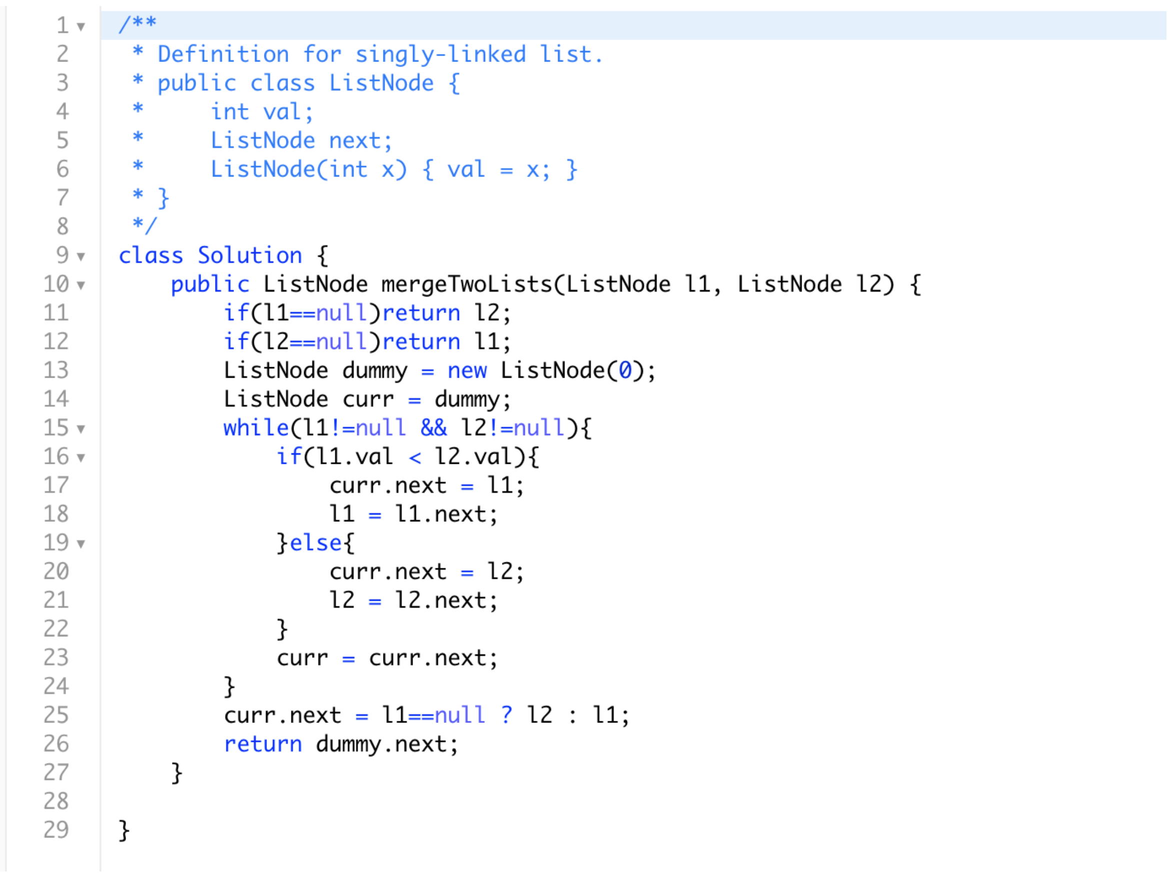
Flight movies

```java
// https://leetcode.com/discuss/interview-question/356960
public class Main {

    public static List<Integer> findPair(List<Integer> nums, int target) {
        target -= 30;
        Map<Integer, Integer> map = new HashMap<>();
        List<Integer> result = Arrays.asList(-1, -1);
        int largest = 0;
        for (int i = 0; i < nums.size(); i++) {
            int complement = target - nums.get(i);
            if ((nums.get(i) > largest || complement > largest) && map.containsKey(complement)) {
                result.set(0, map.get(complement));
                result.set(1, i);
                largest = Math.max(nums.get(i), complement);
            }
            map.put(nums.get(i), i);
        }
        return result;
    }

    public static void main(String[] args) {
        test(Arrays.asList(1, 10, 25, 35, 60), 90);
        test(Arrays.asList(20, 50, 40, 25, 30, 10), 90);
        test(Arrays.asList(5, 55, 40, 20, 30, 30), 90);
    }

    private static void test(List<Integer> nums, int target) {
        System.out.println(findPair(nums, target));
    }
```

```java
public static void main(String[] args) {
    int[] nums1 = {1, 10, 25, 35, 60};
    int target1 = 90;
    System.out.println(Arrays.toString(Find2Sum(nums1, target1-30)));
    int[] nums2 = {20, 50, 40, 25, 30, 10};
    int target2 = 90;
    System.out.println(Arrays.toString(Find2Sum(nums2, target2-30)));
    int[] nums3 = {50, 20, 10, 40, 25, 30};
    int target3 = 90;
    System.out.println(Arrays.toString(Find2Sum(nums3, target3-30)));
}

private static int[] Find2Sum(int[] nums, int target) {
    Map<Integer, Integer> map = new HashMap<>();
    int max = Integer.MIN_VALUE;
    int[] res = new int[2];
    for(int i=0;i<nums.length;i++) {
        if(map.containsKey(nums[i])) {
            if(nums[i] > max || nums[map.get(nums[i])] > max) {
                res[0] = map.get(nums[i]);
                res[1] = i;
                max = Math.max(nums[i], nums[map.get(nums[i])]);
            }
        }
        map.put(target - nums[i], i);
    }
    return res;
}
```

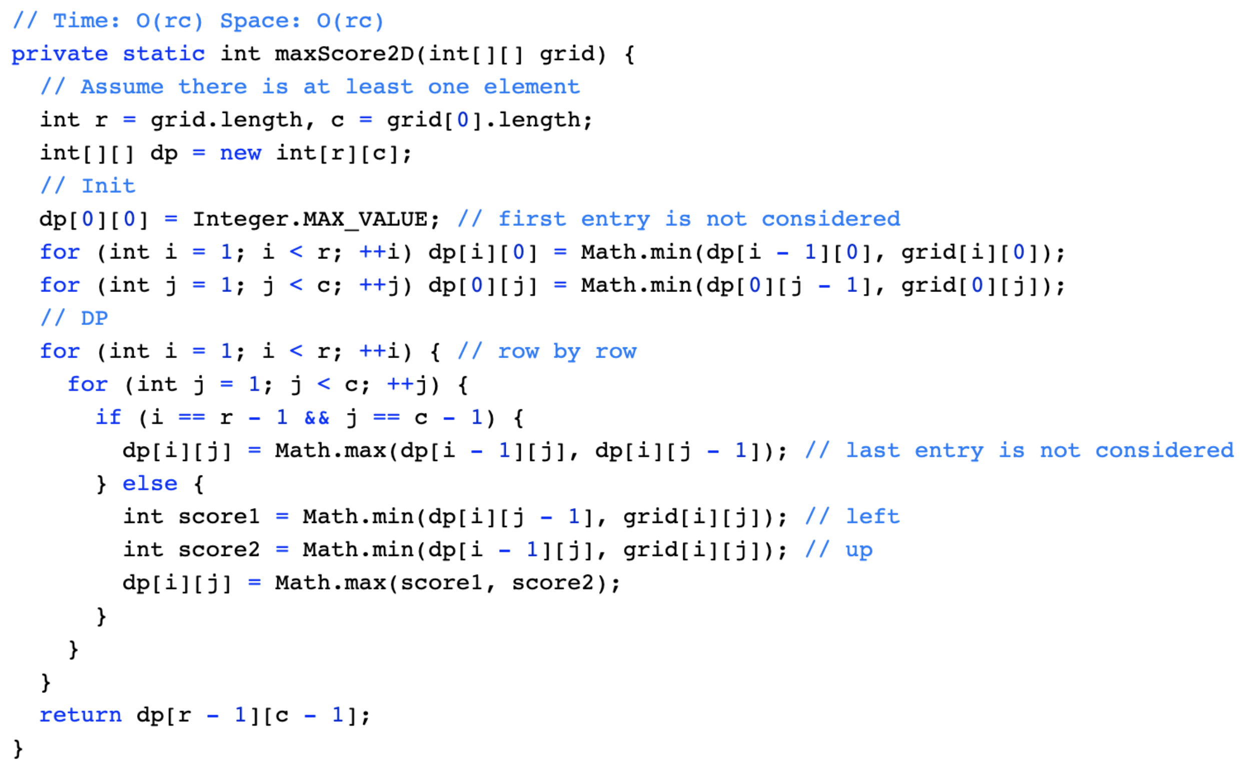
Merge two sorted list



```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
    public ListNode mergeTwoLists(ListNode l1, ListNode l2) {
        if(l1==null)return l2;
        if(l2==null)return l1;
        ListNode dummy = new ListNode(0);
        ListNode curr = dummy;
        while(l1!=null && l2!=null){
            if(l1.val < l2.val){
                curr.next = l1;
                l1 = l1.next;
            }else{
                curr.next = l2;
                l2 = l2.next;
            }
            curr = curr.next;
        }
        curr.next = l1==null ? l2 : l1;
        return dummy.next;
    }

}
```

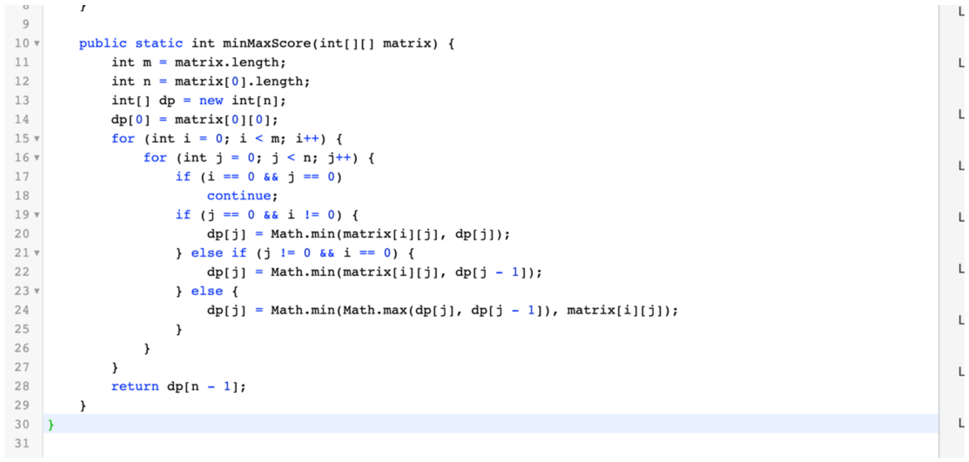
Maximum minum path

二维数组



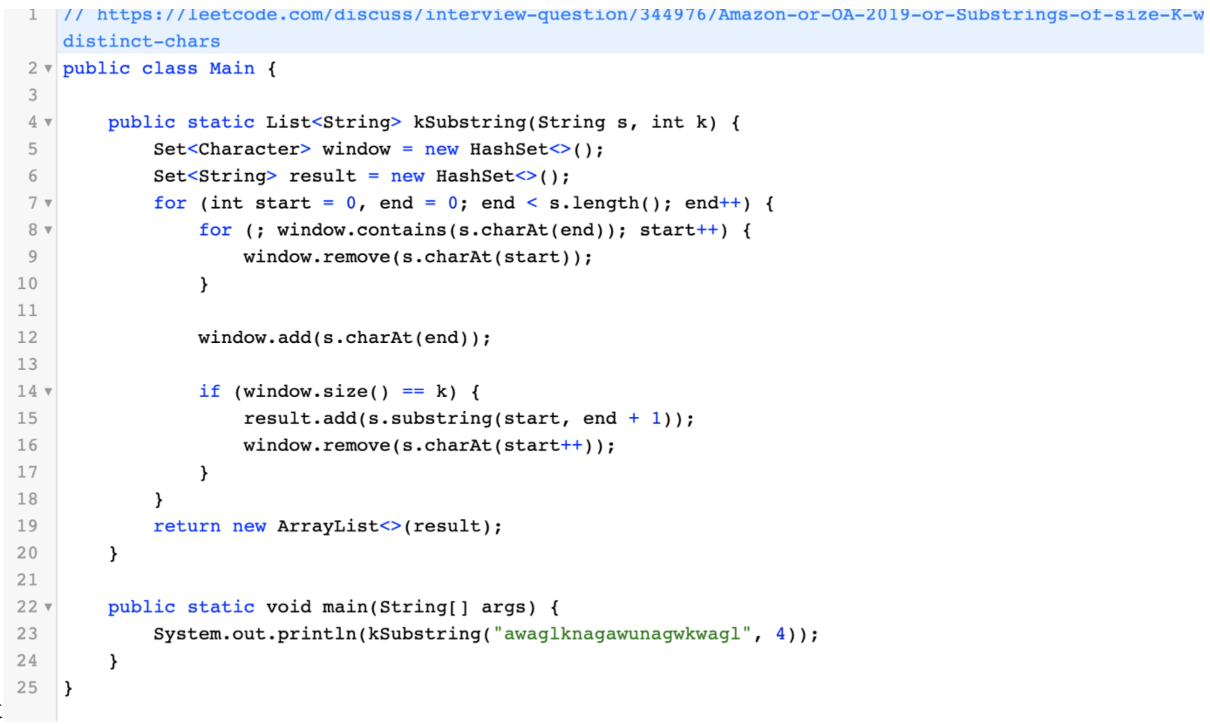
```
// Time: O(rc) Space: O(rc)
private static int maxScore2D(int[][] grid) {
  // Assume there is at least one element
  int r = grid.length, c = grid[0].length;
  int[][] dp = new int[r][c];
  // Init
  dp[0][0] = Integer.MAX_VALUE; // first entry is not considered
  for (int i = 1; i < r; ++i) dp[i][0] = Math.min(dp[i - 1][0], grid[i][0]);
  for (int j = 1; j < c; ++j) dp[0][j] = Math.min(dp[0][j - 1], grid[0][j]);
  // DP
  for (int i = 1; i < r; ++i) { // row by row
    for (int j = 1; j < c; ++j) {
      if (i == r - 1 && j == c - 1) {
        dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]); // last entry is not considered
      } else {
        int score1 = Math.min(dp[i][j - 1], grid[i][j]); // left
        int score2 = Math.min(dp[i - 1][j], grid[i][j]); // up
        dp[i][j] = Math.max(score1, score2);
      }
    }
  }
  return dp[r - 1][c - 1];
}
```

一维数组



```java
    ;

    public static int minMaxScore(int[][] matrix) {
        int m = matrix.length;
        int n = matrix[0].length;
        int[] dp = new int[n];
        dp[0] = matrix[0][0];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (i == 0 && j == 0)
                    continue;
                if (j == 0 && i != 0) {
                    dp[j] = Math.min(matrix[i][j], dp[j]);
                } else if (j != 0 && i == 0) {
                    dp[j] = Math.min(matrix[i][j], dp[j - 1]);
                } else {
                    dp[j] = Math.min(Math.max(dp[j], dp[j - 1]), matrix[i][j]);
                }
            }
        }
        return dp[n - 1];
    }
}
```

Substrings of size K

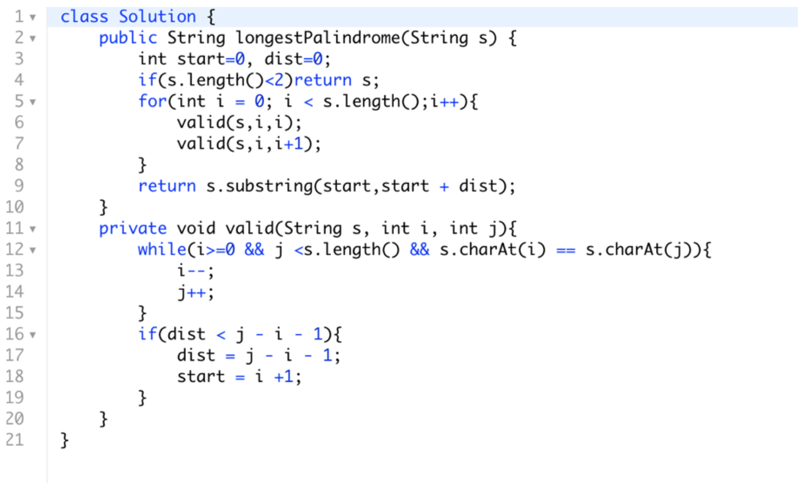


```java
// https://leetcode.com/discuss/interview-question/344976/Amazon-or-OA-2019-or-Substrings-of-size-K-w
distinct-chars
public class Main {

    public static List<String> kSubstring(String s, int k) {
        Set<Character> window = new HashSet<>();
        Set<String> result = new HashSet<>();
        for (int start = 0, end = 0; end < s.length(); end++) {
            for (; window.contains(s.charAt(end)); start++) {
                window.remove(s.charAt(start));
            }

            window.add(s.charAt(end));

            if (window.size() == k) {
                result.add(s.substring(start, end + 1));
                window.remove(s.charAt(start++));
            }
        }
        return new ArrayList<>(result);
    }

    public static void main(String[] args) {
        System.out.println(kSubstring("awaglknagawunagwkwagl", 4));
    }
}
```

返回 list 版本

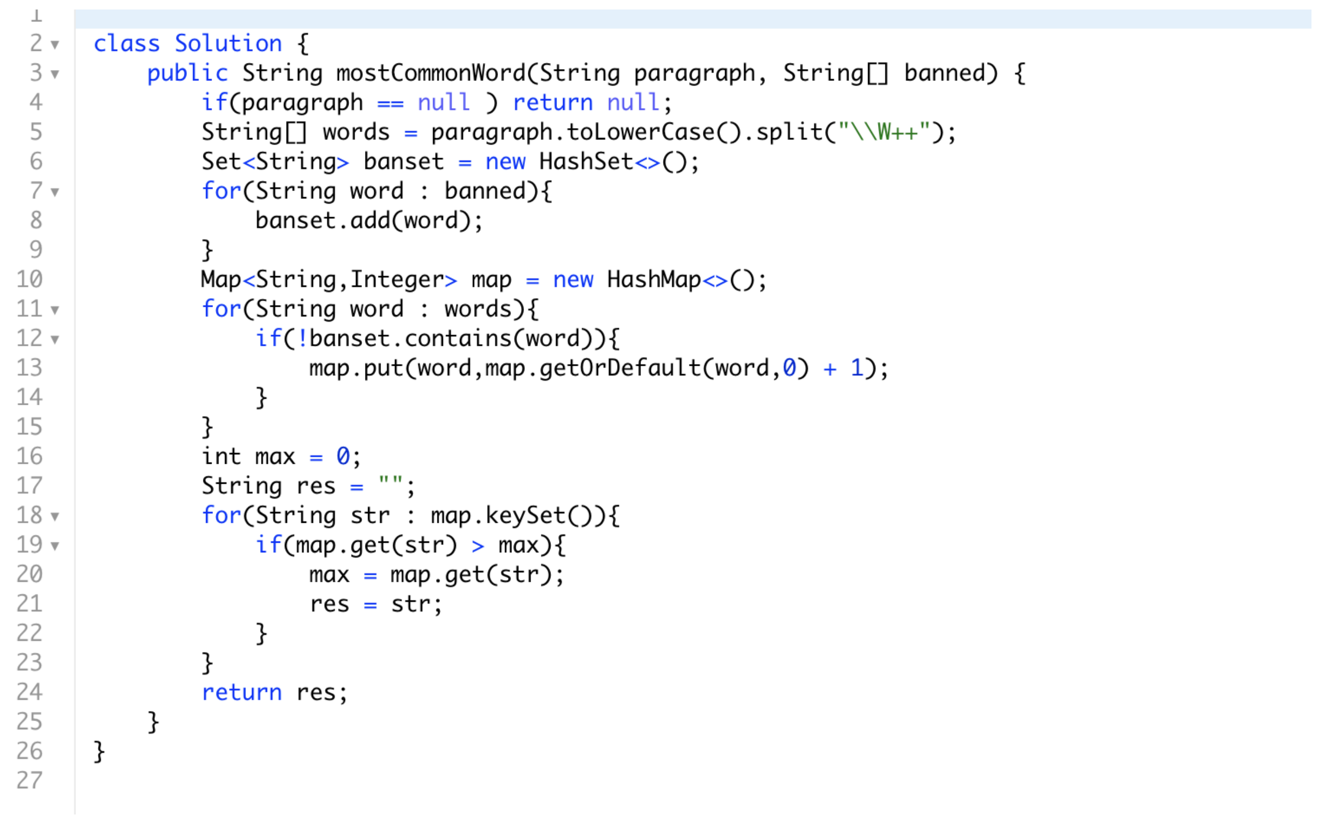
如果返回 count 则 return 后面改成 return result.size(); public 后面应该是 int 而不是 list

Longest Palindromic Substring



```java
class Solution {
    public String longestPalindrome(String s) {
        int start=0, dist=0;
        if(s.length()<2)return s;
        for(int i = 0; i < s.length();i++){
            valid(s,i,i);
            valid(s,i,i+1);
        }
        return s.substring(start,start + dist);
    }
    private void valid(String s, int i, int j){
        while(i>=0 && j <s.length() && s.charAt(i) == s.charAt(j)){
            i--;
            j++;
        }
        if(dist < j - i - 1){
            dist = j - i - 1;
            start = i +1;
        }
    }
}
```

Most common word

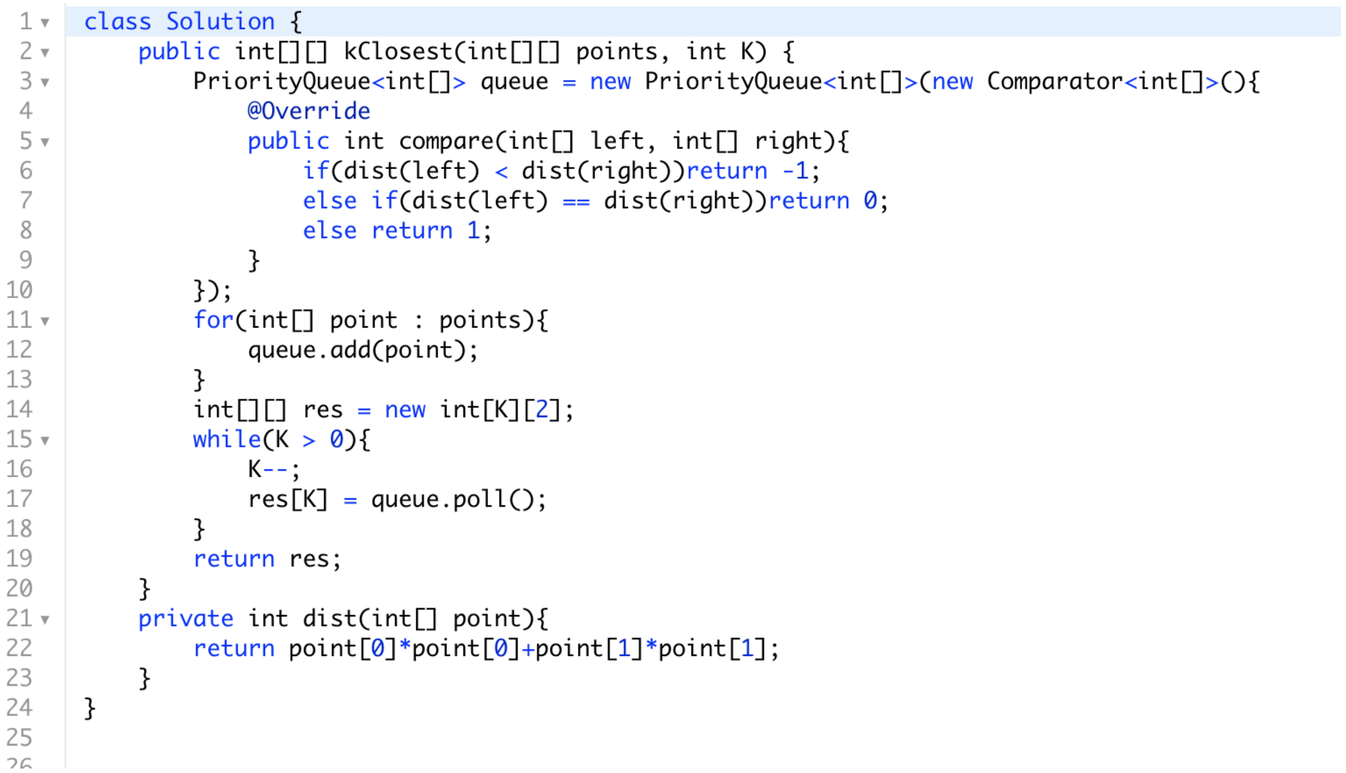


```java

class Solution {
    public String mostCommonWord(String paragraph, String[] banned) {
        if(paragraph == null ) return null;
        String[] words = paragraph.toLowerCase().split("\\W++");
        Set<String> banset = new HashSet<>();
        for(String word : banned){
            banset.add(word);
        }
        Map<String,Integer> map = new HashMap<>();
        for(String word : words){
            if(!banset.contains(word)){
                map.put(word,map.getOrDefault(word,0) + 1);
            }
        }
        int max = 0;
        String res = "";
        for(String str : map.keySet()){
            if(map.get(str) > max){
                max = map.get(str);
                res = str;
            }
        }
        return res;
    }
}

```

注意 case 为 null 的情况

return list 情况怎么做？？？

K closet point to origin



```java
class Solution {
    public int[][] kClosest(int[][] points, int K) {
        PriorityQueue<int[]> queue = new PriorityQueue<int[]>(new Comparator<int[]>(){
            @Override
            public int compare(int[] left, int[] right){
                if(dist(left) < dist(right))return -1;
                else if(dist(left) == dist(right))return 0;
                else return 1;
            }
        });
        for(int[] point : points){
            queue.add(point);
        }
        int[][] res = new int[K][2];
        while(K > 0){
            K--;
            res[K] = queue.poll();
        }
        return res;
    }
    private int dist(int[] point){
        return point[0]*point[0]+point[1]*point[1];
    }
}

```

返回 list 时候怎么做？？？？