Task 2 Ans

18 Tháng Chín 2024 6:10 CH

HolidayTrip

```java
import java.util.Arrays;
// import java.util.Collections;
// import java.util.PriorityQueue;

public class HolidayTrip{
    public int solution(int[] p, int[] s){
    //     int totalPassengers = 0;
    //     for(int passengers : P){
    //         totalPassengers += passengers;
    //     }
    //     // Create a frequency array to count the number of cars with each seat size (1 to 9)
    //    int[] seatCount = new int[10];
    //    for(int s : S){
    //     seatCount[s]++;
    //    }
    //     int usedCars = 0;
    //     int remainingPassengers = totalPassengers;
    //    // Start filling cars from the largest seat size down to the smallest
    //     for(int i = 9; i >= 1; i--){
    //         while(seatCount[i] >0 && remainingPassengers > 0){
    //             usedCars++;
    //             remainingPassengers -= Math.min(i, remainingPassengers);
    //             seatCount[i]--;
    //         }
    //     }
    //     return usedCars;


//O(NlogN)
        // int totalPassengers = 0;
        // for (int passengers : P) {
        //     totalPassengers += passengers;
        // }
        // // Use a priority queue to store the seat sizes in descending order
        // PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Collections.reverseOrder());
        // for (int s : S) {
        //     maxHeap.add(s);
        // }
        // int usedCars = 0;
        // int remainingPassengers = totalPassengers;
        // // Use the largest available car seats first
        // while (remainingPassengers > 0) {
        //     int largestSeats = maxHeap.poll();
        //     usedCars++;
        //     remainingPassengers -= Math.min(largestSeats, remainingPassengers);
        // }
        // return usedCars;


        int tongHK = 0;
        // Tính tổng hành khách
        for (int hanhKhach : p) {
            tongHK += hanhKhach;
        }

        Arrays.sort(s);  // Sắp xếp ghế giảm dần
        //Duyệt từ cuối để sử dụng xe có nhiều ghế nhất (Đảo ngược mảng)
        int xeSuDung = 0;
        int choNgoi = 0;

        for (int i = s.length - 1; i >= 0; i--) {
            choNgoi += s[i];
            xeSuDung++;
            if (choNgoi >= tongHK) {
                break;
            }
        }
        return xeSuDung;
    }

    public static void main(String[] args) {
        HolidayTrip trip = new HolidayTrip();

        int[] P1 = {1, 4, 1};
        int[] S1 = {1, 5, 1};
        System.out.println(trip.solution(P1, S1)); // Output: 2

        int[] P2 = {4, 4, 2, 4};
        int[] S2 = {5, 5, 2, 5};
        System.out.println(trip.solution(P2, S2)); // Output: 3

        int[] P3 = {2, 3, 4, 2};
        int[] S3 = {2, 5, 7, 2};
        System.out.println(trip.solution(P3, S3)); // Output: 2
    }
}
```

Missions

```java
public class Missions {
    public static int solution(int[] d, int x){
        int days = 1; // Bắt đầu với ít nhất một ngày
        int minDiff = d[0];
        int maxDiff = d[0];

        for(int i = 1; i < d.length; i++){
            // Cập nhật độ khó tối thiểu và tối đa cho ngày hiện tại

            minDiff = Math.min(minDiff, d[i]);
            maxDiff = Math.max(maxDiff, d[i]);

            // Nếu chênh lệch giữa độ khó tối đa và tối thiểu vượt quá X, hãy bắt đầu một ngày
mới
            if(maxDiff - minDiff > x){
                days++;
                minDiff = d[i];
                maxDiff = d[i];
            }
        }
        return days;
    }

    public static void main(String[] args) {
        // Test cases
        System.out.println(solution(new int[]{5, 8, 2, 7}, 3));  // Expected output: 3
        System.out.println(solution(new int[]{2, 5, 9, 2, 1, 4}, 4));  // Expected output: 3
        System.out.println(solution(new int[]{1, 12, 10, 4, 5, 2}, 2));  // Expected output: 4
```

PlayersMovements

```java
public class PlayersMovements {
    public static int solution(String s) {
        int res = 0; //số người chơi di chuyển được
        boolean isLeftBlocked = false;
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == '^' || s.charAt(i) == 'v') {
                res++;  // Người chơi có thể di chuyển lên (^) hoặc xuống (v)
                isLeftBlocked = false;  // '^' và 'v' không ảnh hưởng đến các vị trí khác
            } else if (s.charAt(i) == '<') {
                if (!isLeftBlocked) {
                    res++;  // Nếu bên trái không bị chặn, người chơi này có thể di chuyển
                }
                isLeftBlocked = true;  // Sau khi người chơi di chuyển trái, họ chặn bên trái
            } else if (s.charAt(i) == '>') {
                if (i != s.length() - 1 && s.charAt(i + 1) != '>') {
                    // Nếu người chơi bên phải không phải là '>', người chơi này chặn bên trái
                    isLeftBlocked = true;
                } else if (i == s.length() - 1 || s.charAt(i + 1) != '<') {
                    // Người chơi ở vị trí cuối cùng có thể di chuyển, hoặc người bên phải không
di chuyển trái
                    res++;
                }
            }
        }
        return res;
    }

    public static void main(String[] args) {
        // Test cases
        System.out.println(solution("><^v"));  // Expected output: 2
        System.out.println(solution("<<^<v>>"));  // Expected output: 6
        System.out.println(solution("><><"));  // Expected output: 0
    }
}
```

SortedTwoLettersWord

```java
public class SortedTwoLettersWord {
    public static int solution(String s){
        // int n = s.length();
        // int[] countB = new int[n];
        // countB[0] = (s.charAt(0) == 'B') ? 1 : 0;
        // for(int i = 1; i < n; i++){
        //     countB[i] = countB[i - 1] +((s.charAt(i) == 'B') ? 1 : 0);
        // }
        // int[] countA = new int[n];
        // countA[n - 1] = (s.charAt(n - 1) == 'A') ? 1 : 0;
        // for(int i = n - 2; i >= 0; i--){
        //     countA[i] = countA[i + 1] + ((s.charAt(i) == 'A') ? 1 : 0);
        // }
        // int minDelete = Math.min(countA[0], countB[n - 1]); // all A's or all B's
        // for (int i = 0; i < n - 1; i++) {
        //     minDelete = Math.min(minDelete, countB[i] + countA[i + 1]);
        // }
        // return minDelete;

        int n = s.length();
        if(n == 0) return 0;
        if(n == 1) return 1;

        int countA = 0;
        int minDelete = n; // Khởi tạo biến lưu số lần xóa tối thiểu, ban đầu bằng độ dài chuỗi

        // Đếm tổng số lượng 'A' trong chuỗig
        for (char c : s.toCharArray()) {
            if (c == 'A') {
                countA++; // Tăng biến đếm khi gặp 'A'
            }
        }

        int curA = 0;
        int curB = 0;

        // Duyệt qua chuỗi và tính toán số lần xóa tối thiểu
        for (int i = 0; i < n; i++) {
            if (s.charAt(i) == 'A') {
                curA++; // Tăng biến đếm 'A' hiện tại khi gặp 'A'
            } else {
                curB++; // Tăng biến đếm 'B' hiện tại khi gặp 'B'
            }

            // Tính toán số lần xóa cần thiết để sắp xếp chuỗi
            minDelete = Math.min(minDelete, currentB + (countA - currentA));
        }
        return minDelete;
    }

    public static void main(String[] args) {
        // Test cases
        System.out.println(solution("BAAABAB"));  // Output: 2
        System.out.println(solution("BBABAA"));   // Output: 3
        System.out.println(solution("AABBBB"));   // Output: 0
    }
}
```

```
    }
}
```