**PRACTICE SET - 4**

Meena Rhoshini C

Nov14, 24

1. **Stock buy and sell**

import java.util.\*;

public class S {

public static List<int[]> sbs(int[] a, int n) {

List<int[]> r = new ArrayList<>();

int i = 0;

while (i < n - 1) {

while (i < n - 1 && a[i + 1] <= a[i]) i++;

if (i == n - 1) break;

int b = i++;

while (i < n && a[i] >= a[i - 1]) i++;

r.add(new int[]{b, i - 1});

}

return r;

}

public static void main(String[] args) {

int n1 = 7;

int[] a1 = {100, 180, 260, 310, 40, 535, 695};

List<int[]> res1 = sbs(a1, n1);

if (res1.isEmpty()) System.out.println("No Profit");

else System.out.println(1);

int n2 = 5;

int[] a2 = {4, 2, 2, 2, 4};

List<int[]> res2 = sbs(a2, n2);

if (res2.isEmpty()) System.out.println("No Profit");

else System.out.println(1);

int n4 = 4;

int[] a4 = {90, 80, 70, 60};

List<int[]> res4 = sbs(a4, n4);

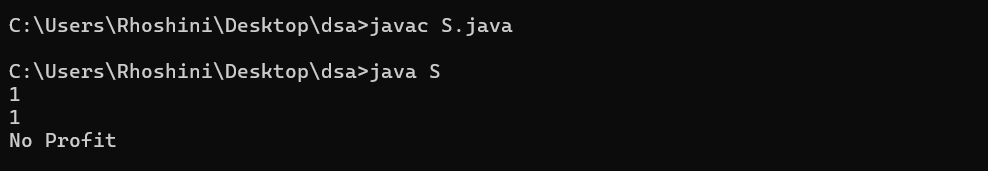
if (res4.isEmpty()) System.out.println("No Profit");

else System.out.println(1);

}

}

OUTPUT:



Time complexity: O(n)

Space complexity: O(n)

1. **Coin Change (Count Ways)**

import java.util.\*;

public class C {

public static int wc(int[] c, int s) {

int[] dp = new int[s + 1];

dp[0] = 1;

for (int coin : c) {

for (int j = coin; j <= s; j++) {

dp[j] += dp[j - coin];

}

}

return dp[s];

}

public static void main(String[] args) {

int[] c1 = {1, 2, 3};

int s1 = 4;

System.out.println(wc(c1, s1));

int[] c2 = {2, 5, 3, 6};

int s2 = 10;

System.out.println(wc(c2, s2));

int[] c3 = {5, 10};

int s3 = 3;

System.out.println(wc(c3, s3));

int[] c4 = {1, 2};

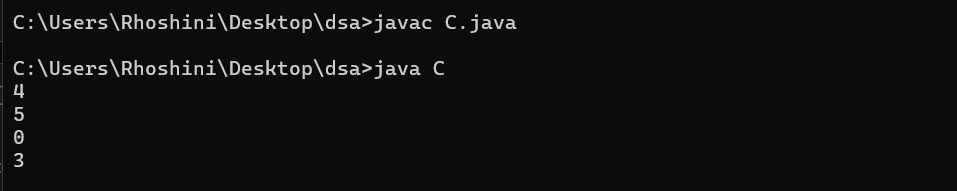
int s4 = 5;

System.out.println(wc(c4, s4));

}

}

OUTPUT:



Time complexity: O(sum × coins.length)

Space complexity: O(sum)

1. **First and Last Occurrences**

public class F {

public static int[] fo(int[] arr, int x) {

int[] result = {-1, -1};

result[0] = binarySearch(arr, x, true);

if (result[0] != -1) {

result[1] = binarySearch(arr, x, false);

}

return result;

}

public static int binarySearch(int[] arr, int x, boolean findFirst) {

int l = 0, r = arr.length - 1, ans = -1;

while (l <= r) {

int m = l + (r - l) / 2;

if (arr[m] == x) {

ans = m;

if (findFirst) {

r = m - 1;

} else {

l = m + 1;

}

} else if (arr[m] < x) {

l = m + 1;

} else {

r = m - 1;

}

}

return ans;

}

public static void main(String[] args) {

int[] arr1 = {1, 3, 5, 5, 5, 5, 67, 123, 125};

int x1 = 5;

int[] result1 = fo(arr1, x1);

System.out.println(result1[0] + " " + result1[1]);

int[] arr2 = {1, 3, 5, 5, 5, 5, 7, 123, 125};

int x2 = 7;

int[] result2 = fo(arr2, x2);

System.out.println(result2[0] + " " + result2[1]);

int[] arr3 = {1, 2, 3};

int x3 = 4;

int[] result3 = fo(arr3, x3);

System.out.println(result3[0] + " " + result3[1]);

int[] arr4 = {1, 1, 1, 1, 1, 1};

int x4 = 1;

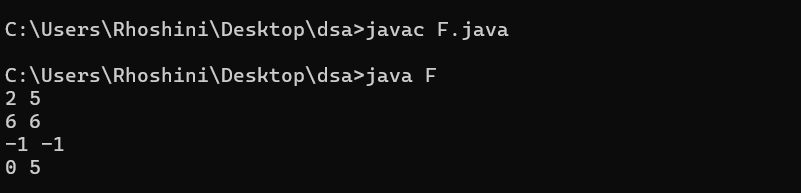
int[] result4 = fo(arr4, x4);

System.out.println(result4[0] + " " + result4[1]);

}

}

OUTPUT:



Time complexity: O(log n)

Space complexity: O(1)

1. **Find Transition Point**

public class T {

public static int tp(int[] arr) {

int l = 0, r = arr.length - 1, ans = -1;

while (l <= r) {

int m = l + (r - l) / 2;

if (arr[m] == 1) {

ans = m;

r = m - 1;

} else {

l = m + 1;

}

}

return ans == -1 ? -1 : ans;

}

public static void main(String[] args) {

int[] arr1 = {0, 0, 0, 1, 1};

System.out.println(tp(arr1));

int[] arr2 = {0, 0, 0, 0};

System.out.println(tp(arr2));

int[] arr3 = {1, 1, 1};

System.out.println(tp(arr3));

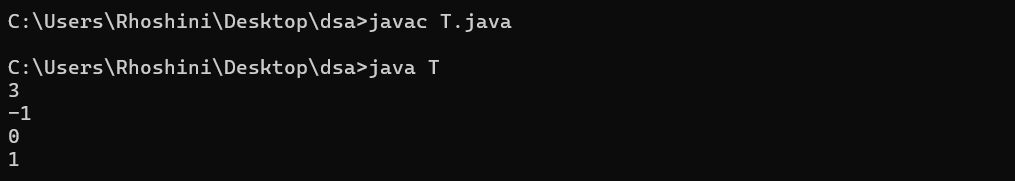
int[] arr4 = {0, 1, 1};

System.out.println(tp(arr4));

}

}

OUTPUT:



Time complexity: O(log n)

Space complexity: O(1)

1. **First Repeating Element**

import java.util.\*;

public class FirstR {

public static int fr(int[] arr) {

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

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

if (seen.contains(arr[i])) {

return i + 1;

}

seen.add(arr[i]);

}

return -1;

}

public static void main(String[] args) {

int[] arr1 = {1, 5, 3, 4, 3, 5, 6};

System.out.println(fr(arr1));

int[] arr2 = {1, 2, 3, 4};

System.out.println(fr(arr2));

int[] arr3 = {10, 20, 30, 20, 40};

System.out.println(fr(arr3));

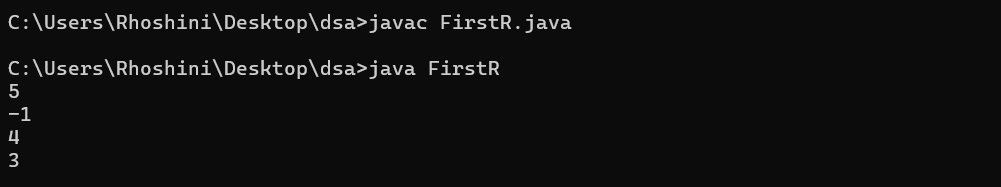
int[] arr4 = {7, 8, 7, 6, 5, 4};

System.out.println(fr(arr4));

}

}

OUTPUT:



Time complexity: O(n)

Space complexity: O(n)

1. **Remove Duplicates Sorted Array**

public class R {

public static int rd(int[] arr) {

if (arr.length == 0) return 0;

int index = 1;

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

if (arr[i] != arr[i - 1]) {

arr[index++] = arr[i];

}

}

return index;

}

public static void main(String[] args) {

int[] arr1 = {2, 2, 2, 2, 2};

int size1 = rd(arr1);

System.out.println(size1);

int[] arr2 = {1, 2, 4};

int size2 = rd(arr2);

System.out.println(size2);

int[] arr3 = {1, 1, 2, 3, 3, 3, 4};

int size3 = rd(arr3);

System.out.println(size3);

int[] arr4 = {10, 10, 10, 20, 30};

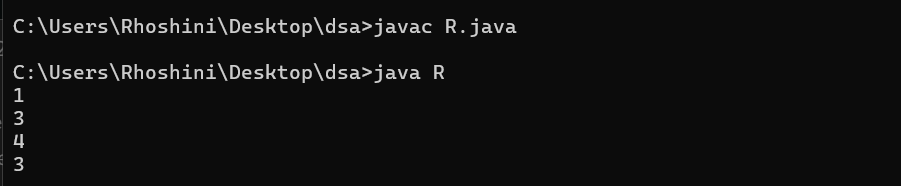
int size4 = rd(arr4);

System.out.println(size4);

}

}

OUTPUT:



Time complexity: O(n)

Space complexity: O(1)

1. **Maximum Index**

public class M {

public static int maxIndexDiff(int[] arr) {

int n = arr.length;

int[] leftMin = new int[n];

int[] rightMax = new int[n];

leftMin[0] = arr[0];

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

leftMin[i] = Math.min(leftMin[i - 1], arr[i]);

}

rightMax[n - 1] = arr[n - 1];

for (int i = n - 2; i >= 0; i--) {

rightMax[i] = Math.max(rightMax[i + 1], arr[i]);

}

int i = 0, j = 0, maxDiff = -1;

while (i < n && j < n) {

if (leftMin[i] < rightMax[j]) {

maxDiff = Math.max(maxDiff, j - i);

j++;

} else {

i++;

}

}

return maxDiff;

}

public static void main(String[] args) {

int[] arr1 = {1, 10};

System.out.println(maxIndexDiff(arr1));

int[] arr2 = {34, 8, 10, 3, 2, 80, 30, 33, 1};

System.out.println(maxIndexDiff(arr2));

int[] arr3 = {5, 10, 15, 20, 25};

System.out.println(maxIndexDiff(arr3));

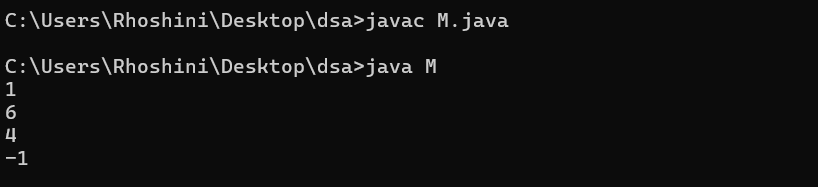
int[] arr4 = {3, 2, 1};

System.out.println(maxIndexDiff(arr4));

}

}

OUTPUT:



Time complexity:O(n)

Space complexity: O(n)

1. **Wave Array**

public class W {

public static void waveArray(int[] arr) {

for (int i = 1; i < arr.length; i += 2) {

if (i - 1 >= 0 && arr[i - 1] > arr[i]) {

int temp = arr[i];

arr[i] = arr[i - 1];

arr[i - 1] = temp;

}

if (i + 1 < arr.length && arr[i] < arr[i + 1]) {

int temp = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = temp;

}

}

}

public static void main(String[] args) {

int[] arr1 = {1, 2, 3, 4, 5};

waveArray(arr1);

for (int num : arr1) System.out.print(num + " ");

System.out.println();

int[] arr2 = {2, 4, 7, 8, 9, 10};

waveArray(arr2);

for (int num : arr2) System.out.print(num + " ");

System.out.println();

int[] arr3 = {1};

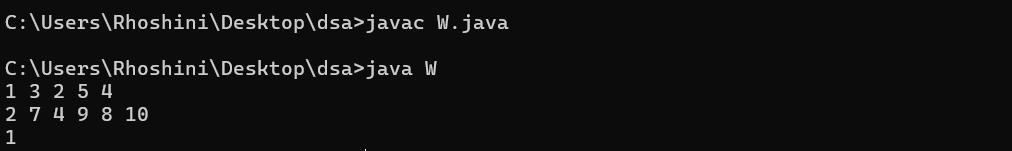
waveArray(arr3);

for (int num : arr3) System.out.print(num + " ");

}

}

OUTPUT:

****

Time complexity: O(n)

Space complexity: O(1)