**Bhuvaneswari P AIDS DSA Practice-5 14/11/2024**

**1.Buy and Sell Stock**

import java.util.ArrayList;

import java.util.List;

public class BuyAndSell {

public static int stockBuySell(int[] A, int N) {

List<List<Integer>> result = 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 buy = i++;

while (i < N && A[i] >= A[i - 1]) {

i++;

}

int sell = i - 1;

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

pair.add(buy);

pair.add(sell);

result.add(pair);

}

return result.isEmpty() ? 0 : 1;

}

public static void main(String[] args) {

int[] A1 = {100, 180, 260, 310, 40, 535, 695};

int N1 = A1.length;

System.out.println(stockBuySell(A1, N1));

int[] A2 = {4, 2, 2, 2, 4};

int N2 = A2.length;

System.out.println(stockBuySell(A2, N2));

int[] A3 = {5, 4, 3, 2, 1};

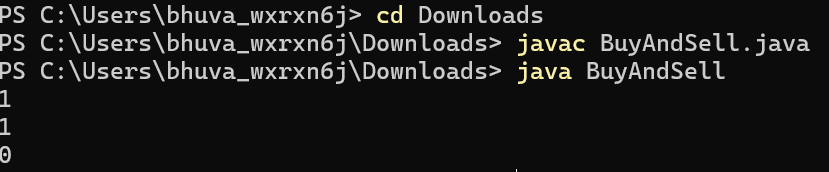
int N3 = A3.length;

System.out.println(stockBuySell(A3, N3));

}

}

**Output:**



Time Complexity: O(n)

Space Complexity: O(n)

**2.Coin Change**

public class CoinChange {

public static int countWays(int[] coins, int sum) {

int n = coins.length;

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

dp[0] = 1;

for (int coin : coins) {

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

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

}

}

return dp[sum];

}

public static void main(String[] args) {

int[] coins1 = {1, 2, 3};

int sum1 = 4;

System.out.println(countWays(coins1, sum1));

int[] coins2 = {2, 5, 3, 6};

int sum2 = 10;

System.out.println(countWays(coins2, sum2));

int[] coins3 = {5, 10};

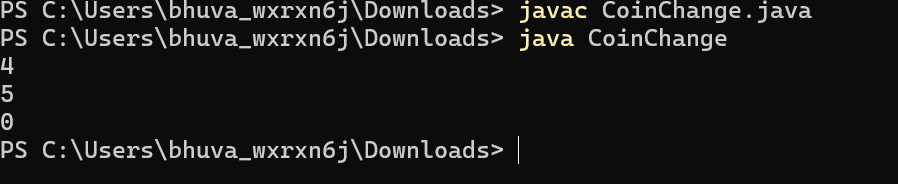
int sum3 = 3;

System.out.println(countWays(coins3, sum3));

}

}

**Output:**



Time Complexity:O(n.sum)

Space Complexity:O(sum)

**3.First and Last occurrence**

public class FirstAndLastOccurrences {

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

int first = -1, last = -1;

int low = 0, high = arr.length - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == x) {

first = mid;

high = mid - 1;

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

low = mid + 1;

} else {

high = mid - 1;

}

}

low = 0;

high = arr.length - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == x) {

last = mid;

low = mid + 1;

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

low = mid + 1;

} else {

high = mid - 1;

}

}

return new int[]{first, last};

}

public static void main(String[] args) {

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

int x1 = 5;

int[] result1 = findOccurrences(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 = findOccurrences(arr2, x2);

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

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

int x3 = 4;

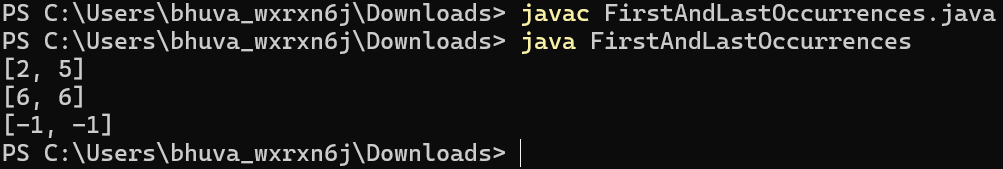
int[] result3 = findOccurrences(arr3, x3);

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

}

}

**Output:**



Time Complexity: O(logn)

Space Complexity:O(1)

**4.First Transition Point**

class TransitionPoint {

int transitionPoint(int[] arr) {

int low = 0, high = arr.length - 1;

if (arr[0] == 1) return 0;

if (arr[high] == 0) return -1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == 1 && (mid == 0 || arr[mid - 1] == 0)) {

return mid;

} else if (arr[mid] == 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1;

}

public static void main(String[] args) {

TransitionPoint sol = new TransitionPoint();

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

System.out.println("Transition Point: " + sol.transitionPoint(arr1));

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

System.out.println("Transition Point: " + sol.transitionPoint(arr2));

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

System.out.println("Transition Point: " + sol.transitionPoint(arr3));

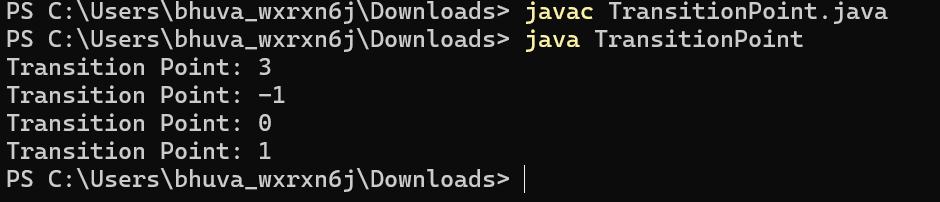
int[] arr4 = {0, 1, 1};

System.out.println("Transition Point: " + sol.transitionPoint(arr4));

}

}

**Output:**

****

Time Complexity: O(logn)

Space Complexity: O(1)

**5.First Repeating Element**

import java.util.HashMap;

class FirstRepeatingElement {

public static int firstRepeated(int[] arr) {

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

int idx = -1;

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

if (map.containsKey(arr[i])) {

if (idx == -1 || map.get(arr[i]) < idx) {

idx = map.get(arr[i]);

}

} else {

map.put(arr[i], i + 1);

}

}

return idx;

}

public static void main(String[] args) {

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

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

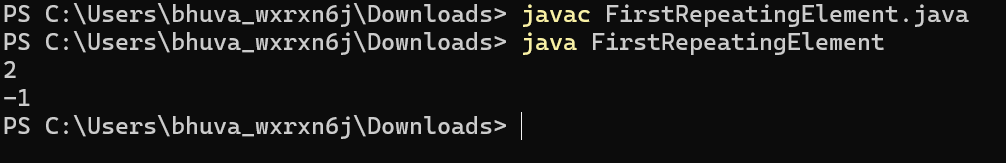
int[] arr2 = {1, 2, 3, 4};

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

}

}

**Output:**



Time Complexity:O(n)

Space Complexity:O(n)

**6.Remove Duplicates Sorted Array**

class RemoveDuplicatesSortedArray {

public static int removeDuplicates(int[] arr) {

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

int idx = 1;

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

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

arr[idx] = arr[i];

idx++;

}

}

return idx;

}

public static void main(String[] args) {

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

for (int i = 0; i < 1; i++) {

System.out.print(arr1[i] + " ");

}

System.out.println();

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

for (int i = 0; i < 3; i++) {

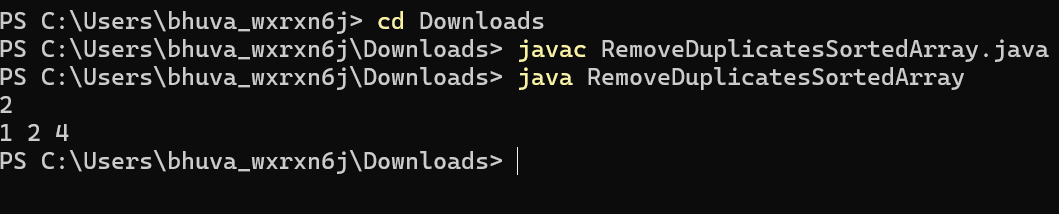
System.out.print(arr2[i] + " ");

}

}

}

**Output:**



Time Complexity:O(n)

Space Complexity:O(1)

**7.Maximum Index**

import java.util.Stack;

class MaximumIndex {

int maxIndexDiff(int[] arr) {

Stack<Integer> st = new Stack<>();

int N = arr.length;

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

if (st.isEmpty() || arr[st.peek()] > arr[i]) {

st.push(i);

}

}

int max = 0;

int dif;

int i = N - 1;

while (i >= 0) {

if (!st.isEmpty() && arr[st.peek()] <= arr[i]) {

dif = i - st.pop();

if (dif > max) {

max = dif;

}

} else {

i--;

}

}

return max;

}

public static void main(String[] args) {

MaximumIndex solution = new MaximumIndex();

int[] arr1 = {1, 10};

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

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

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

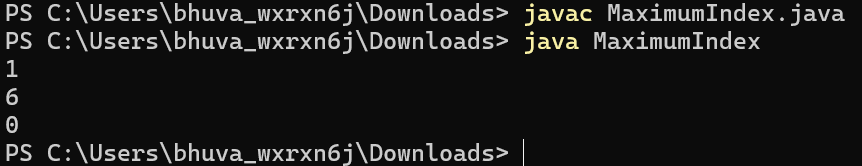
int[] arr3 = {15};

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

}

}

Output:



Time Complexity:O(n)

Space Complexity:O(n)

**8.Wave Array**

class WaveArray {

public static void waveArray(int[] arr) {

for (int i = 0; i < arr.length - 1; i += 2) {

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 i = 0; i < arr1.length; i++) {

System.out.print(arr1[i] + " ");

}

System.out.println();

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

waveArray(arr2);

for (int i = 0; i < arr2.length; i++) {

System.out.print(arr2[i] + " ");

}

System.out.println();

int[] arr3 = {1};

waveArray(arr3);

for (int i = 0; i < arr3.length; i++) {

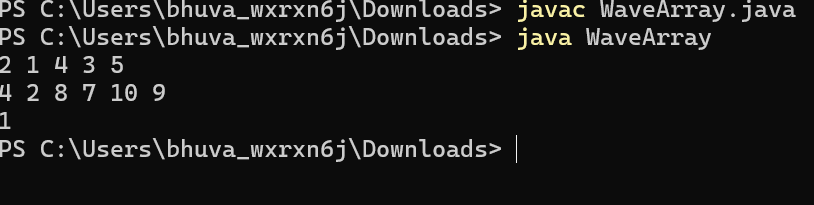
System.out.print(arr3[i] + " ");

}

}

}

**Output:**



Time Complexity: O(n)

Space Complexity: O(1)