**Coding Practice day 5**

**1.** **Bull and Sell Stock**

**Code**

import java.util.Arrays;

class Main {

static int maxProfit(int[] prices, int start, int end) {

int maxProfit = 0;

for (int i = start; i < end; i++) {

for (int j = i + 1; j <= end; j++) {

if (prices[j] > prices[i]) {

int currentProfit = (prices[j] - prices[i]) + maxProfit(prices, start, i - 1) + maxProfit(prices, j + 1, end);

maxProfit = Math.max(maxProfit, currentProfit);

}

}

}

return maxProfit;

}

static int findMaxProfit(int[] prices) {

return maxProfit(prices, 0, prices.length - 1);

}

public static void main(String[] args) {

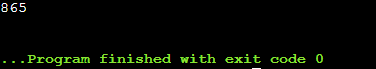
int[] prices = {100, 180, 260, 310, 40, 535, 695};

System.out.println(findMaxProfit(prices));

}

}

**Output**

****

Time Complexity:O(n)  
Space Complexity:O(1)

**2.** **Coin Exchange**

**Code**

import java.io.\*;

public class Main {

static int minCoins(int[] coins, int numCoins, int amount) {

if (amount == 0) return 0;

int minCoins = Integer.MAX\_VALUE;

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

if (coins[i] <= amount) {

int result = minCoins(coins, numCoins, amount - coins[i]);

if (result != Integer.MAX\_VALUE && result + 1 < minCoins)

minCoins = result + 1;

}

}

return minCoins;

}

public static void main(String[] args) {

int[] coins = {9, 6, 5, 1};

int numCoins = coins.length;

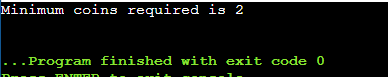
int amount = 11;

System.out.println("Minimum coins required is " + minCoins(coins, numCoins, amount));

}

}

**Output**

****

Time Complexity: O(m^sum)  
Space Complexity:O(sum)

**3.** **First and Last Occurance  
Code**

import java.io.\*;

class Main {

public static void findFirstAndLast(int[] arr, int target) {

int length = arr.length;

int first = -1, last = -1;

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

if (target != arr[i]) continue;

if (first == -1) first = i;

last = i;

}

if (first != -1) {

System.out.println(first);

System.out.println(last);

} else {

System.out.println("Not Found");

}

}

public static void main(String[] args) {

int[] arr = { 1, 2, 2, 2, 2, 3, 4, 7, 8, 8 };

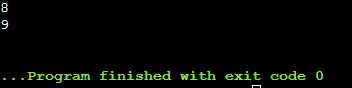
int target = 8;

findFirstAndLast(arr, target);

}

}

**Output**

****

Time Complexity: O(n)  
Space Complexity:O(1)

**4.** **Find Transition Point**

**Code**

import java.util.\*;

class Main {

static int findTransitionPoint(int[] arr, int length) {

for (int i = 0; i < length; i++)

if (arr[i] == 1)

return i;

return -1;

}

public static void main(String[] args) {

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

int length = arr.length;

int transitionPoint = findTransitionPoint(arr, length);

if (transitionPoint >= 0)

System.out.print("Transition point is " + transitionPoint);

else

System.out.print("There is no transition point");

}

}

**Output**

****

Time Complexity: O(n)  
Space Complexity:O(1)

**5.** **First Repeating element**

**Code**

import java.util.\*;

public class Main {

public static int a(int[] arr, int n) {

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

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

if (arr[i] == arr[j]) {

return i;

}

}

}

return -1;

}

public static void main(String[] args) {

int[] arr = { 10, 5, 3, 4, 3, 5, 6 };

int n = arr.length;

int idx = a(arr, n);

if (idx == -1) {

System.out.println("No repeating element found!");

} else {

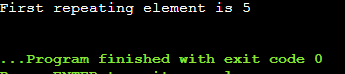
System.out.println("First repeating element is " + arr[idx]);

}

}

}

**Output**

****

Time Complexity: O(n)  
Space Complexity:O(n)

**6. Remove Duplicates from sorted array**

**Code**import java.util.HashSet;

class Main {

static int removeDuplicates(int[] arr) {

HashSet<Integer> uniqueElements = new HashSet<>();

int index = 0;

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

if (!uniqueElements.contains(arr[i])) {

uniqueElements.add(arr[i]);

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

}

}

return index;

}

public static void main(String[] args) {

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

int size = removeDuplicates(arr);

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

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

}

}

}

**Output**

****

Time Complexity: O(n)  
Space Complexity:O(1)

**7.Maximum Index**

**Code**

public class Main {

int findMaxDistance(int[] arr, int length) {

int maxDistance = -1;

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

for (int j = length - 1; j > i; --j) {

if (arr[j] > arr[i] && maxDistance < (j - i)) {

maxDistance = j - i;

}

}

}

return maxDistance;

}

public static void main(String[] args) {

Main max = new Main();

int[] arr = { 9, 2, 3, 4, 5, 6, 7, 8, 18, 0 };

int length = arr.length;

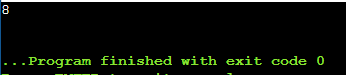
int maxDist = max.findMaxDistance(arr, length);

System.out.println(maxDist);

}

}

**Output**

****

Time Complexity: O(n^2)  
Space Complexity:O(1)

**8.Wave Array**

**Code**

import java.util.\*;

public class Main {

void swap(int[] arr, int x, int y) {

int temp = arr[x];

arr[x] = arr[y];

arr[y] = temp;

}

void rearrange(int[] arr, int size) {

Arrays.sort(arr);

for (int i = 0; i < size - 1; i += 2)

swap(arr, i, i + 1);

}

public static void main(String[] args) {

Main obj = new Main();

int[] arr = {10, 90, 49, 2, 1, 5, 23};

int size = arr.length;

obj.rearrange(arr, size);

for (int i : arr)

System.out.print(i + " ");

}

}

**Output**

****

Time Complexity: O(n)  
Space Complexity:O(1)