



You can view this report online at : <https://www.hackerrank.com/x/tests/1330882/candidates/42910037/report>

Full Name:	Ruchit Bhardwaj
Email:	ruchitbh@usc.edu
Test Name:	CodePath SE103: Unit 9 Assessment - Summer 2022
Taken On:	2 Aug 2022 18:33:23 PDT
Time Taken:	30 min 13 sec/ 90 min
Invited by:	CodePath
Skills Score:	
Tags Score:	<div>Big O5/5</div> <div>SE1015/5</div>

98.2%

535/545

scored in **CodePath SE103: Unit 9 Assessment - Summer 2022** in 30 min 13 sec on 2 Aug 2022 18:33:23 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Time Complexity > Multiple Choice	37 sec	5/ 5	✓
Q2	Greedy v. DP > Multiple Choice	7 min 22 sec	0/ 5	✗
Q3	No DP? > Multiple Choice	38 sec	0/ 5	✗
Q4	Min-Cost Climbing Stairs Bug > Coding	10 min 5 sec	75/ 75	✓
Q5	Min-Cost Climbing Stairs Complexity > Multiple Choice	25 sec	5/ 5	✓
Q6	Unique Binary Search Trees > Coding	2 min 8 sec	150/ 150	✓
Q7	Buy and Sell Stocks > Coding	5 min 44 sec	150/ 150	✓
Q8	Burst Balloons > Coding	2 min 57 sec	150/ 150	✓

**QUESTION 1**

Correct Answer

Score 5

**Time Complexity** > Multiple Choice SE101 Big O**QUESTION DESCRIPTION**

What is the Big-O complexity of the following Python function, which takes in an array (n) and returns a count of the number of positive numbers in it (see implementation below)?

```
def count_the_positives(n):  
    positives = 0  
    for i in n:  
        if i > 0:  
            positives +=1  
    return positives
```

**CANDIDATE ANSWER**

**Options:** (Expected answer indicated with a tick)

- ☐ O(1)
- ☐ O(log(n))
- ☒ O(n)
- ☐ O(2^n)
- ☐ O(n!)

No Comments

## QUESTION 2



Wrong Answer

Score 0

## Greedy v. DP &gt; Multiple Choice

## QUESTION DESCRIPTION

## Given the following problem:

Given  $n$  balloons, indexed from 0 to  $n-1$ . Each balloon is painted with a number on it represented by array `nums`. You are asked to burst all the balloons. If the you burst balloon  $i$  you will get `nums[left] * nums[i] * nums[right]` coins. Here `left` and `right` are adjacent indices of  $i$ . After the burst, the `left` and `right` then becomes adjacent.

Find the maximum coins you can collect by bursting the balloons wisely.

Imagine someone as already built both a greedy algorithm solution and a dynamic programming solution. The greedy algorithm simply chooses the optimal solution available at each iteration of the algorithm. What would be the difference in output between the greedy algorithm and DP for the array `[4, 1, 8, 6, 2]`?

## Example:

Given `[3, 1, 5, 8]`

Return 167

```
nums = [3,1,5,8] --> [3,5,8] --> [3,8] --> [8] --> []  
coins = 3*1*5 + 3*5*8 + 1*3*8 + 1*8*1 = 167
```


## CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ☐ 0
- ☐ 16
- ☐ 54
- ☒ 80
- ☐ 112

No Comments

QUESTION 3

Wrong Answer

Score 0

No DP? > Multiple Choice

QUESTION DESCRIPTION

Which one of these problems is not amenable to dynamic programming?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)


☐ Find shortest distance between S and T in a DAG.

☒ Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand. Find the minimum element.

☐ Given n, count the number of structurally distinct binary trees that store all values 1 - n.

No Comments

QUESTION 4

Correct Answer

Score 75

Min-Cost Climbing Stairs Bug > Coding

QUESTION DESCRIPTION

On a staircase, the  $i$ -th step has some non-negative cost `cost[i]` assigned (0 indexed).

Once you pay the cost, you can either climb one or two steps. You need to find minimum cost to reach the top of the floor, and you can either start from the step with index 0, or the step with index 1.

**Example 1:**  
**Input:** `cost = [10, 15, 20]`  
**Output:** 15  
**Explanation:** Cheapest is start on `cost[1]`, pay that cost and go to the top.

**Example 2:**  
**Input:** `cost = [1, 100, 1, 1, 1, 100, 1, 1, 100, 1]`  
**Output:** 6  
**Explanation:** Cheapest is start on `cost[0]`, and only step on 1s, skipping `cost[3]`.

**Note:**  
`cost` will have a length in the range `[2, 1000]`.  
Every `cost[i]` will be an integer in the range `[0, 999]`.

Consider the code below that's been written to solve this problem. Currently it does not solve the solution correctly. Can you fix the bug?

CANDIDATE ANSWER

Language used: Java 8

```
1 public static int min_cost(int[] cost) {
2     int n = cost.length;
3     if (cost == null || cost.length == 1) {
4         return 0;
5     }
```

```

6
7     int[] dp = new int[cost.length];
8     for (int i = 0; i < dp.length; i++) {
9         dp[i] = cost[i];
10    }
11
12    for (int i = 2; i < cost.length; i++) {
13        dp[i] = Integer.min(dp[i-1], dp[i-2]) + cost[i];
14    }
15
16    return Math.min(dp[n - 1], dp[n - 2]);
17 }
18
19

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	5	0.1011 sec	24.6 KB
Testcase 1	Easy	Sample case	✔ Success	5	0.1068 sec	24.5 KB
Testcase 2	Easy	Hidden case	✔ Success	5	0.0866 sec	24.8 KB
Testcase 3	Easy	Hidden case	✔ Success	5	0.0865 sec	24.7 KB
Testcase 4	Easy	Hidden case	✔ Success	5	0.109 sec	24.6 KB
Testcase 5	Easy	Hidden case	✔ Success	5	0.1703 sec	24.9 KB
Testcase 6	Easy	Hidden case	✔ Success	5	0.0965 sec	24.6 KB
Testcase 7	Easy	Hidden case	✔ Success	5	0.1516 sec	24.7 KB
Testcase 8	Easy	Hidden case	✔ Success	5	0.1214 sec	24.7 KB
Testcase 9	Easy	Hidden case	✔ Success	5	0.1156 sec	24.6 KB
Testcase 10	Easy	Hidden case	✔ Success	5	0.1208 sec	24.8 KB
Testcase 11	Easy	Hidden case	✔ Success	5	0.1061 sec	24.8 KB
Testcase 12	Easy	Hidden case	✔ Success	5	0.1066 sec	24.6 KB
Testcase 13	Easy	Hidden case	✔ Success	5	0.103 sec	24.8 KB
Testcase 14	Easy	Hidden case	✔ Success	5	0.107 sec	25 KB

No Comments

QUESTION 5

Correct Answer

Score 5

Min-Cost Climbing Stairs Complexity > Multiple Choice

QUESTION DESCRIPTION

On a staircase, the  $i$ -th step has some non-negative cost  $\text{cost}[i]$  assigned (0 indexed).

Once you pay the cost, you can either climb one or two steps. You need to find minimum cost to reach the top of the floor, and you can either start from the step with index 0, or the step with index 1.

**Example 1:**  
**Input:**  $\text{cost} = [10, 15, 20]$   
**Output:** 15  
**Explanation:** Cheapest is start on  $\text{cost}[1]$ , pay that cost and go to the top.

**Example 2:**  
**Input:**  $\text{cost} = [1, 100, 1, 1, 1, 100, 1, 1, 100, 1]$   
**Output:** 6  
**Explanation:** Cheapest is start on  $\text{cost}[0]$ , and only step on 1s, skipping  $\text{cost}[3]$ .

What is the optimal solution memory complexity for this problem?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

☒  $O(1)$

☐  $O(n)$

☐  $O(n^2)$

☐  $O(2^n)$

No Comments

QUESTION 6

Correct Answer

Score 150

Unique Binary Search Trees > Coding

QUESTION DESCRIPTION

Given  $n$ , how many structurally unique **BST's** (binary search trees) that store values  $1 \dots n$ ?

**Example:**  
**Input:** 3  
**Output:** 5  
**Explanation:** Given  $n = 3$ , there are a total of 5 unique BST's:

13321

1

3

2

3

1

3

2

1

2

3

2

1

1

3

2

2

1

2

1

3

2

CANDIDATE ANSWER

Language used: **Java 8**

```
1 public static int numTrees(int n) {  
2     int[] C = new int[n+1];  
3     C[0] = 1; C[1] = 1;  
4     for (int i = 2; i <= n; i++) {  
5         for (int j = 1; j <= i; j++) {  
6             C[i] += C[j-1]*C[i-j];  
7         }  
8     }  
9     return C[n];  
10 }
```

```

2     int[] G = new int[n+1];
3     G[0] = G[1] = 1;
4
5     for(int i=2; i<=n; ++i) {
6         for(int j=1; j<=i; ++j) {
7             G[i] += G[j-1] * G[i-j];
8         }
9     }
10    return G[n];
11 }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	10	0.1007 sec	24.7 KB
Testcase 1	Easy	Hidden case	✔ Success	10	0.1231 sec	24.8 KB
Testcase 2	Easy	Hidden case	✔ Success	10	0.1046 sec	24.6 KB
Testcase 3	Easy	Hidden case	✔ Success	10	0.1203 sec	24.6 KB
Testcase 4	Easy	Hidden case	✔ Success	10	0.0917 sec	24.5 KB
Testcase 5	Easy	Hidden case	✔ Success	10	0.0971 sec	24.7 KB
Testcase 6	Easy	Hidden case	✔ Success	10	0.1037 sec	24.7 KB
Testcase 7	Easy	Hidden case	✔ Success	10	0.1219 sec	24.6 KB
Testcase 8	Easy	Hidden case	✔ Success	10	0.1117 sec	24.7 KB
Testcase 9	Easy	Hidden case	✔ Success	10	0.1195 sec	24.6 KB
Testcase 10	Easy	Hidden case	✔ Success	10	0.1379 sec	24.8 KB
Testcase 11	Easy	Hidden case	✔ Success	10	0.1054 sec	24.8 KB
Testcase 12	Easy	Hidden case	✔ Success	10	0.0997 sec	24.8 KB
Testcase 13	Easy	Hidden case	✔ Success	10	0.1025 sec	24.8 KB
Testcase 14	Easy	Hidden case	✔ Success	10	0.0974 sec	24.7 KB

No Comments

#### QUESTION 7



Correct Answer

Score 150

### Buy and Sell Stocks > Coding

#### QUESTION DESCRIPTION

Say you have an array for which the  $i$ th element is the price of a given stock on day  $i$ . Design an algorithm to find the maximum profit. You may complete as many transactions as you like (ie, buy one and sell one share of the stock multiple times) with the following restrictions:

- You may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).
- After you sell your stock, you cannot buy stock on next day. (ie, cooldown 1 day)

[LS]

**Example:**

prices = [1, 2, 3, 0, 2]

maxProfit = 3

transactions = [buy, sell, cooldown, buy, sell]

#### CANDIDATE ANSWER

Language used: **Java 8**

```

1 public static int maxProfit(int[] prices) {
2     if (prices.length <= 1) return 0;
3     int[] s0 = new int[prices.length];
4     Arrays.fill(s0, 0);
5
6     int[] s1 = new int[prices.length];
7     Arrays.fill(s1, 0);
8
9     int[] s2 = new int[prices.length];
10    Arrays.fill(s2, 0);
11
12    s1[0] = -prices[0];
13    s0[0] = 0;
14    s2[0] = Integer.MIN_VALUE;
15    for (int i = 1; i < prices.length; i++) {
16        s0[i] = Math.max(s0[i - 1], s2[i - 1]);
17        s1[i] = Math.max(s1[i - 1], s0[i - 1] - prices[i]);
18        s2[i] = s1[i - 1] + prices[i];
19    }
20    return Math.max(s0[prices.length - 1], s2[prices.length - 1]);
21 }
22
23

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✔ Success	10	0.1022 sec	24.8 KB
TestCase 1	Easy	Hidden case	✔ Success	10	0.0959 sec	24.8 KB
TestCase 2	Easy	Hidden case	✔ Success	10	0.1017 sec	24.8 KB
TestCase 3	Easy	Hidden case	✔ Success	10	0.0824 sec	24.5 KB
TestCase 4	Easy	Hidden case	✔ Success	10	0.0945 sec	24.7 KB
TestCase 5	Easy	Hidden case	✔ Success	10	0.1251 sec	24.9 KB
TestCase 6	Easy	Hidden case	✔ Success	10	0.1008 sec	24.8 KB
TestCase 7	Easy	Hidden case	✔ Success	10	0.1 sec	24.6 KB
TestCase 8	Easy	Hidden case	✔ Success	10	0.0902 sec	24.7 KB
TestCase 9	Easy	Hidden case	✔ Success	10	0.0976 sec	24.7 KB
TestCase 10	Easy	Hidden case	✔ Success	10	0.1105 sec	24.6 KB
TestCase 11	Easy	Hidden case	✔ Success	10	0.0907 sec	24.9 KB
TestCase 12	Easy	Hidden case	✔ Success	10	0.1165 sec	24.9 KB
TestCase 13	Easy	Hidden case	✔ Success	10	0.1042 sec	24.6 KB
TestCase 14	Easy	Hidden case	✔ Success	10	0.1032 sec	24.7 KB

No Comments

#### QUESTION 8



Correct Answer

Score 150

#### Burst Balloons > Coding

##### QUESTION DESCRIPTION

Given  $n$  balloons, indexed from  $0$  to  $n-1$ . Each balloon is painted with a number on it represented by array `nums`. You are asked to burst all the balloons. If the you burst balloon  $i$  you will get `nums[left] *`



nums[i] \* nums[right] coins. Here left and right are adjacent indices of i. After the burst, the left and right then becomes adjacent.

Find the maximum coins you can collect by bursting the balloons wisely.

**Note:**

(1) You may imagine  $\text{nums}[-1] = \text{nums}[n] = 1$ . They are not real therefore you can not burst them.

(2)  $0 \leq n \leq 500, 0 \leq \text{nums}[i] \leq 100$

Example:

Given [3, 1, 5, 8]



Return 167

nums = [3,1,5,8] --> [3,5,8] --> [3,8] --> [8] --> []  
coins =  $3*1*5$  +  $3*5*8$  +  $1*3*8$  +  $1*8*1$  = 167

**CANDIDATE ANSWER**

Language used: **Java 8**

```
1 public static int maxCoins(int[] nums) {
2     int[][] dp = new int[nums.length][nums.length];
3     return maxCoins(nums, 0, nums.length - 1, dp);
4 }
5
6 public static int maxCoins(int[] nums, int start, int end, int[][] dp) {
7     if (start > end) {
8         return 0;
9     }
10    if (dp[start][end] != 0) {
11        return dp[start][end];
12    }
13    int max = nums[start];
14    for (int i = start; i <= end; i++) {
15        int val = maxCoins(nums, start, i - 1, dp) +
16                get(nums, i) * get(nums, start - 1) * get(nums, end + 1) +
17                maxCoins(nums, i + 1, end, dp);
18
19        max = Math.max(max, val);
20    }
21    dp[start][end] = max;
22    return max;
23 }
24
25 public static int get(int[] nums, int i) {
26     if (i == -1 || i == nums.length) {
27         return 1;
28     }
29     return nums[i];
30 }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	 Success	10	0.1029 sec	24.6 KB
Testcase 1	Easy	Hidden case	 Success	10	0.1004 sec	24.7 KB
Testcase 2	Easy	Hidden case	 Success	10	0.0963 sec	24.5 KB
Testcase 3	Easy	Hidden case	 Success	10	0.1034 sec	24.7 KB
Testcase 4	Easy	Hidden case	 Success	10	0.0971 sec	24.7 KB
Testcase 5	Easy	Hidden case	 Success	10	0.0904 sec	24.7 KB

Testcase 6	Easy	Hidden case	✔ Success	10	0.0975 sec	24.9 KB
Testcase 7	Easy	Hidden case	✔ Success	10	0.0861 sec	24.8 KB
Testcase 8	Easy	Hidden case	✔ Success	10	0.0845 sec	24.7 KB
Testcase 9	Easy	Hidden case	✔ Success	10	0.1006 sec	24.7 KB
Testcase 10	Easy	Hidden case	✔ Success	10	0.1062 sec	24.8 KB
Testcase 11	Easy	Hidden case	✔ Success	10	0.107 sec	24.8 KB
Testcase 12	Easy	Hidden case	✔ Success	10	0.0984 sec	24.6 KB
Testcase 13	Easy	Hidden case	✔ Success	10	0.1029 sec	24.9 KB
Testcase 14	Easy	Hidden case	✔ Success	10	0.0937 sec	24.7 KB

No Comments

---

PDF generated at: 3 Aug 2022 02:05:24 UTC