



You can view this report online at : <https://www.hackerrank.com/x/tests/1121392/candidates/28245807/report>

Full Name:

My Nguyen

Email:

nguyen_my@yahoo.com

Test Name:

Dynamic Programming 1 Assessment 2021

Taken On:

1 Aug 2021 17:54:10 PDT

Time Taken:

84 min 59 sec/ 90 min

Contact Number:

+14084096862

Resume:

https://hackerrank-resumes.s3.amazonaws.com/412894/JhbK9vK_4Bhc4Gvuv7s5hgcFJGeFCATHWliNY1UGAfhwRPsrnVekT5ZtKXgX8QA2Ag/My_Nguyen_Resume.PDF

Linkedin:

<https://www.linkedin.com/in/my-nguyen-87849>

Invited by:

Curriculum

Skills Score:

Tags Score:

Big O

5/5

SE101

5/5

45.9%

250/545

scored in **Dynamic Programming 1 Assessment 2021** in 84 min 59 sec on 1 Aug 2021 17:54:10 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Time Complexity > Multiple Choice	43 sec	5/ 5	✓
Q2	Greedy v. DP > Multiple Choice	5 min 54 sec	0/ 5	✗
Q3	No DP? > Multiple Choice	26 sec	5/ 5	✓
Q4	Min-Cost Climbing Stairs Bug > Coding	13 min 33 sec	50/ 75	✓
Q5	Min-Cost Climbing Stairs Complexity > Multiple Choice	2 min 28 sec	0/ 5	✗
Q6	Unique Binary Search Trees > Coding	19 min 9 sec	30/ 150	✓
Q7	Buy and Sell Stocks > Coding	14 min 42 sec	90/ 150	✓
Q8	Burst Balloons > Coding	24 min 29 sec	70/ 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 > 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

Correct Answer

Score 5

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 50

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     if (cost == null || cost.length == 1) {
3         return 0;
4     }
5     ...
```

```

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

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✘ Wrong Answer	0	0.0946 sec	24.8 KB
Testcase 1	Easy	Sample case	✔ Success	5	0.0857 sec	24.9 KB
Testcase 2	Easy	Hidden case	✔ Success	5	0.086 sec	24.8 KB
Testcase 3	Easy	Hidden case	✔ Success	5	0.0918 sec	24.8 KB
Testcase 4	Easy	Hidden case	✔ Success	5	0.0998 sec	24.8 KB
Testcase 5	Easy	Hidden case	✘ Wrong Answer	0	0.0915 sec	24.8 KB
Testcase 6	Easy	Hidden case	✔ Success	5	0.0942 sec	24.9 KB
Testcase 7	Easy	Hidden case	✔ Success	5	0.0867 sec	24.9 KB
Testcase 8	Easy	Hidden case	✔ Success	5	0.1129 sec	24.9 KB
Testcase 9	Easy	Hidden case	✘ Wrong Answer	0	0.0896 sec	24.8 KB
Testcase 10	Easy	Hidden case	✔ Success	5	0.0897 sec	24.9 KB
Testcase 11	Easy	Hidden case	✘ Wrong Answer	0	0.0922 sec	25 KB
Testcase 12	Easy	Hidden case	✘ Wrong Answer	0	0.0904 sec	24.9 KB
Testcase 13	Easy	Hidden case	✔ Success	5	0.0847 sec	24.9 KB
Testcase 14	Easy	Hidden case	✔ Success	5	0.092 sec	24.9 KB

No Comments

QUESTION 5



Wrong Answer

Score 0

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 30

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:



CANDIDATE ANSWER

Language used: **Java 8**

```

1 public static int numTrees(int n) {
2     // System.out.println("n: " + n);
3     if (n <= 2)
4         return n;
5     return 3 + numTrees(n-1);
6 }
  
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.0846 sec	25 KB
Testcase 1	Easy	Hidden case	✓ Success	10	0.0891 sec	24.8 KB
Testcase 2	Easy	Hidden case	✗ Wrong Answer	0	0.093 sec	24.7 KB
Testcase 3	Easy	Hidden case	✗ Wrong Answer	0	0.0874 sec	25.1 KB
Testcase 4	Easy	Hidden case	✗ Wrong Answer	0	0.0968 sec	24.9 KB
Testcase 5	Easy	Hidden case	✗ Wrong Answer	0	0.0982 sec	24.9 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.0931 sec	25 KB
Testcase 7	Easy	Hidden case	✗ Wrong Answer	0	0.087 sec	24.9 KB
Testcase 8	Easy	Hidden case	✗ Wrong Answer	0	0.0882 sec	24.8 KB
Testcase 9	Easy	Hidden case	✗ Wrong Answer	0	0.0871 sec	24.9 KB
Testcase 10	Easy	Hidden case	✗ Wrong Answer	0	0.0981 sec	24.8 KB
Testcase 11	Easy	Hidden case	✗ Wrong Answer	0	0.0916 sec	24.8 KB
Testcase 12	Easy	Hidden case	✗ Wrong Answer	0	0.0863 sec	24.9 KB
Testcase 13	Easy	Hidden case	✗ Wrong Answer	0	0.0963 sec	24.8 KB
Testcase 14	Easy	Hidden case	✗ Wrong Answer	0	0.0864 sec	24.7 KB

No Comments

QUESTION 7



Correct Answer

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)

```

[[]]
[[]]
[[]]

```

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     // System.out.println("prices: " + Arrays.toString(prices));
3     int profit = 0;
4     if (prices == null || prices.length == 0)
5         ;
6     else {
7         int i = 0;
8         while (i < prices.length-1) {
9             if (prices[i] < prices[i+1]) {
10                 profit += prices[i+1] - prices[i];
11                 i += 2;
12             } else {
13                 i += 1;
14             }
15             /*int j = i + 1;
16             while (prices[j] > prices[j-1])
17                 j++;*/
18         }
19     }
20     // System.out.println("profit: " + profit);
21     return profit;
22 }
23
24

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✔ Success	10	0.0955 sec	25.1 KB
TestCase 1	Easy	Hidden case	✘ Wrong Answer	0	0.0915 sec	24.8 KB
TestCase 2	Easy	Hidden case	✔ Success	10	0.0833 sec	24.8 KB
TestCase 3	Easy	Hidden case	✔ Success	10	0.0925 sec	24.9 KB
TestCase 4	Easy	Hidden case	✘ Wrong Answer	0	0.097 sec	25 KB
TestCase 5	Easy	Hidden case	✔ Success	10	0.0913 sec	24.9 KB
TestCase 6	Easy	Hidden case	✘ Wrong Answer	0	0.092 sec	24.9 KB
TestCase 7	Easy	Hidden case	✔ Success	10	0.1034 sec	25 KB
TestCase 8	Easy	Hidden case	✔ Success	10	0.0933 sec	25.1 KB
TestCase 9	Easy	Hidden case	✘ Wrong Answer	0	0.097 sec	24.8 KB
TestCase 10	Easy	Hidden case	✔ Success	10	0.0954 sec	24.9 KB

TestCase 11	Easy	Hidden case	✔ Success	10	0.1038 sec	24.9 KB
TestCase 12	Easy	Hidden case	✘ Wrong Answer	0	0.0993 sec	25 KB
TestCase 13	Easy	Hidden case	✔ Success	10	0.0862 sec	24.9 KB
Testcase 14	Easy	Hidden case	✘ Wrong Answer	0	0.0974 sec	24.9 KB

No Comments

QUESTION 8



Correct Answer

Score 70

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 `nums[-1] = 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     if (nums == null || nums.length == 0)
3         return 0;
4     if (nums.length == 1)
5         return nums[0];
6
7     List<Integer> list = new ArrayList<>();
8     for (int elem : nums)
9         list.add(elem);
10
11     return recur(list);
12 }
13
14 private static int recur(List<Integer> list) {
15     if (list.size() == 2) {
16         int l0 = list.get(0);
17         int l1 = list.get(1);
18         return l0*l1 + Math.max(l0, l1);
19     }
20
21     int product = 1;
22     for (int i = 0; i < 3; i++)
23         product *= list.get(i);
24     list.remove(1);
```

```
25     return product + recur(list);
26 }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	10	0.1008 sec	25 KB
Testcase 1	Easy	Hidden case	✘ Wrong Answer	0	0.1011 sec	25 KB
Testcase 2	Easy	Hidden case	✔ Success	10	0.0979 sec	25 KB
Testcase 3	Easy	Hidden case	✘ Wrong Answer	0	0.0937 sec	24.9 KB
Testcase 4	Easy	Hidden case	✔ Success	10	0.0863 sec	25 KB
Testcase 5	Easy	Hidden case	✔ Success	10	0.0943 sec	25 KB
Testcase 6	Easy	Hidden case	✘ Wrong Answer	0	0.1106 sec	24.8 KB
Testcase 7	Easy	Hidden case	✔ Success	10	0.0903 sec	25 KB
Testcase 8	Easy	Hidden case	✘ Wrong Answer	0	0.1034 sec	24.8 KB
Testcase 9	Easy	Hidden case	✔ Success	10	0.0922 sec	24.9 KB
Testcase 10	Easy	Hidden case	✔ Success	10	0.1032 sec	24.9 KB
Testcase 11	Easy	Hidden case	✘ Wrong Answer	0	0.1092 sec	24.9 KB
Testcase 12	Easy	Hidden case	✘ Wrong Answer	0	0.0828 sec	25 KB
Testcase 13	Easy	Hidden case	✘ Wrong Answer	0	0.1023 sec	25.1 KB
Testcase 14	Easy	Hidden case	✘ Wrong Answer	0	0.0849 sec	24.8 KB

No Comments