15

Muhammad Tanveer

Software Engineer

PDF generated at: 6 Aug 2024 01:05:18 UTC

View this report on HackerRank

Score

15% • 60 / 400

scored in CodePath TIP103: Unit 8 Assessment - Summer 2024 in 3 min 31 sec on 5 Aug 2024 18:00:04 PDT

Candidate Information

Email	tanveerm176@gmail.com
Test	CodePath TIP103: Unit 8 Assessment - Summer 2024
Candidate Packet	View
Taken on	5 Aug 2024 18:00:04 PDT
Time taken	3 min 31 sec/ 90 min
Invited by	CodePath

Skill Distribution

There is no associated skills data that can be shown for this assessment

Tags Distribution

SE101	100%	Big O	100%

Questions

Status	No.	Question	Time Taken	Skill	Score
✓	1	Time Complexity Multiple Choice	23 sec	-	5/5
✗	2	Greedy v. DP Multiple Choice	15 sec	-	0/5
✗	3	No DP? Multiple Choice	1 min 33 sec	-	0/5
✓	4	Min-Cost Climbing Stairs Bug Coding	24 sec	-	50/75
✗	5	Min-Cost Climbing Stairs Complexity Multiple Choice	20 sec	-	0/5
✗	6	Unique Binary Search Trees Coding	9 sec	-	0/150
✗	7	Buy and Sell Stocks Coding	10 sec	-	0/150

✓	8	Meeting Intervals Multiple Choice	12 sec	-	5/5

1. Time Complexity

✓ Correct

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's Solution

Options: (Expected answer indicated with a tick)

- ○ O(1)
- ○ O(log(n))
- ● **O(n)** ✓

O(2^n)

O(n!)

No comments.

2. Greedy v. DP

Incorrect

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's Solution

Options: (Expected answer indicated with a tick)

- ○ 0
- ○ 16
- ○ 54
- ● 80
- ○ 112 ✓

ⓘ No comments.

---

3. No DP?

✗ Incorrect

Multiple Choice

Question description

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

Candidate's Solution

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.**

---

4. Min-Cost Climbing Stairs Bug

Partially correct

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's Solution Language used: **Python 3**

```python
import fileinput

args = []
for line in fileinput.input():
    args.append(line)

nums = [int(i) for i in args[0].split()]

def min_cost(cost):
    if not cost or len(cost) == 1:
        return 0
    dp = [c for c in cost]
    for i in range(2, len(cost)):
        dp[i] = min(dp[i-1], dp[i-2]) + cost[i]
    return dp[-1]

print(min_cost(nums))
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Wrong Answer	0	0.0265 sec	10.1 KB
Testcase 1	Easy	Sample	Success	5	0.0353 sec	10 KB

Testcase 2	Easy	Hidden	Success	5	0.0243 sec	10.1 KB
Testcase 3	Easy	Hidden	Success	5	0.0254 sec	10.1 KB
Testcase 4	Easy	Hidden	Success	5	0.0312 sec	10.1 KB
Testcase 5	Easy	Hidden	Wrong Answer	0	0.0289 sec	10.2 KB
Testcase 6	Easy	Hidden	Success	5	0.023 sec	10.3 KB
Testcase 7	Easy	Hidden	Success	5	0.0253 sec	10.1 KB
Testcase 8	Easy	Hidden	Success	5	0.0257 sec	10.3 KB
Testcase 9	Easy	Hidden	Wrong Answer	0	0.0256 sec	10 KB
Testcase 10	Easy	Hidden	Success	5	0.0269 sec	10.3 KB
Testcase 11	Easy	Hidden	Wrong Answer	0	0.0372 sec	10.1 KB
Testcase 12	Easy	Hidden	Wrong Answer	0	0.0595 sec	10.1 KB

Testcase 13	Easy	Hidden	Success	5	0.0394 sec	10.3 KB
Testcase 14	Easy	Hidden	Success	5	0.0273 sec	10.3 KB

No comments.

5. Min-Cost Climbing Stairs Complexity

Incorrect

Multiple Choice

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].

What is the optimal solution memory complexity for this problem?

Candidate's Solution

Options: (Expected answer indicated with a tick)

- ○ O(1)
- ● O(n)
- ○ O(n^2)
- ○ O(2^n)

No comments.

6. Unique Binary Search Trees

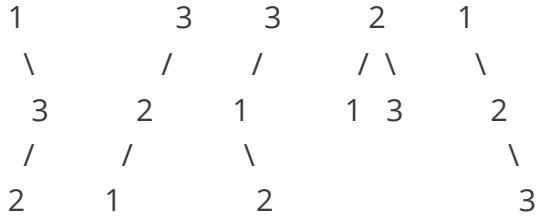
Incorrect

Coding

Question description

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

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



Candidate's Solution

Language used: **Python 3**

```python
import fileinput

args = []
for line in fileinput.input():
    args.append(line)

def numTrees(n):
    pass

print(numTrees(int(args[0])))
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample	Wrong Answer	0	0.0247 sec	10.3 KB
Testcase 1	Easy	Hidden	Wrong Answer	0	0.0276 sec	10.2 KB
Testcase 2	Easy	Hidden	Wrong Answer	0	0.0311 sec	10.3 KB
Testcase 3	Easy	Hidden	Wrong Answer	0	0.0296 sec	10 KB
Testcase 4	Easy	Hidden	Wrong Answer	0	0.0322 sec	9.87 KB
Testcase 5	Easy	Hidden	Wrong Answer	0	0.0259 sec	10.3 KB
Testcase 6	Easy	Hidden	Wrong Answer	0	0.0232 sec	10 KB

Testcase 7	Easy	Hidden	Wrong Answer	0	0.0332 sec	10.3 KB
Testcase 8	Easy	Hidden	Wrong Answer	0	0.0588 sec	10 KB
Testcase 9	Easy	Hidden	Wrong Answer	0	0.0255 sec	10.2 KB
Testcase 10	Easy	Hidden	Wrong Answer	0	0.0231 sec	10.2 KB
Testcase 11	Easy	Hidden	Wrong Answer	0	0.0266 sec	10.3 KB
Testcase 12	Easy	Hidden	Wrong Answer	0	0.0256 sec	10.3 KB
Testcase 13	Easy	Hidden	Wrong Answer	0	0.0244 sec	10.2 KB
Testcase 14	Easy	Hidden	Wrong Answer	0	0.0244 sec	10.1 KB

No comments.

7. Buy and Sell Stocks

Incorrect

Coding

Question description

Say you have an array for which the ith 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's Solution Language used: **Python 3**

```python
import fileinput

args = []
for line in fileinput.input():
    args.append(line)

nums = [int(i) for i in args[0].split()]

def maxProfit(prices):
    pass

print(maxProfit(nums))
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample	Wrong Answer	0	0.0276 sec	10.1 KB
TestCase 1	Easy	Hidden	Wrong Answer	0	0.0291 sec	10.1 KB

TestCase 2	Easy	Hidden	Wrong Answer	0	0.0357 sec	10.2 KB
TestCase 3	Easy	Hidden	Wrong Answer	0	0.0294 sec	10.1 KB
TestCase 4	Easy	Hidden	Wrong Answer	0	0.0394 sec	10.3 KB
TestCase 5	Easy	Hidden	Wrong Answer	0	0.0207 sec	10.3 KB
TestCase 6	Easy	Hidden	Wrong Answer	0	0.0296 sec	10.2 KB
TestCase 7	Easy	Hidden	Wrong Answer	0	0.0263 sec	10.3 KB
TestCase 8	Easy	Hidden	Wrong Answer	0	0.0295 sec	10.1 KB
TestCase 9	Easy	Hidden	Wrong Answer	0	0.0257 sec	10 KB
TestCase 10	Easy	Hidden	Wrong Answer	0	0.0263 sec	10.1 KB
TestCase 11	Easy	Hidden	Wrong Answer	0	0.0282 sec	10.3 KB
TestCase 12	Easy	Hidden	Wrong Answer	0	0.0208 sec	10.1 KB

TestCase 13	Easy	Hidden	Wrong Answer	0	0.0316 sec	10.1 KB
Testcase 14	Easy	Hidden	Wrong Answer	0	0.0314 sec	10.1 KB

No comments.

8. Meeting Intervals

Correct

Multiple Choice

Question description

We are given an array of meeting time intervals specified by their start and end times. For example, meeting 1 could be defined as [8-9] (8-9am) and meeting 2 could be defined as [12-13] (12pm - 1pm). We want to find the minimum number of conference rooms needed to schedule all the meetings. We decide that a greedy approach can probably work well here. In order to proceed with a greedy approach though, we must first sort the meeting intervals.

Using a **greedy approach**, what is the most efficient way to sort the meeting intervals?

Candidate's Solution

Options: (Expected answer indicated with a tick)

- [x] Sort by the meeting's start time
- [ ] Sort by the meeting's end time

Sort by the meeting's total duration time

No comments.