



You can view this report online at : <https://www.hackerrank.com/x/tests/1374201/candidates/43510086/report>

Full Name: Ruchit Bhardwaj
Email: ruchitbh@usc.edu
Test Name: CodePath SE103: Unit 11 Assessment - Summer 2022
Taken On: 16 Aug 2022 05:10:25 PDT
Time Taken: 65 min 32 sec/ 90 min
Invited by: CodePath
Skills Score:
Tags Score:

- Arrays 30/35
- BFS/DFS 5/5
- Binary Search Trees 30/30
- Binary Trees 35/35
- Greedy Algorithms 0/5
- Linked Lists 30/30
- Recursion 3.75/5
- Strings 30/30
- Time Complexity 5/5

91.3%
118/130

scored in **CodePath SE103: Unit 11 Assessment - Summer 2022** in 65 min 32 sec on 16 Aug 2022 05:10:25 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Delete Tree Nodes > Multiple Choice	1 min 12 sec	5/ 5	✔
Q2	Stack: Insert at Bottom > Multiple Choice	3 min 9 sec	5/ 5	✔
Q3	Design Front Middle Back Queue > Multiple Choice	1 min 40 sec	0/ 5	✘
Q4	Count number of trees in a forest > Multiple Choice	1 min 5 sec	5/ 5	✔
Q5	Meeting Intervals > Multiple Choice	12 sec	0/ 5	✘
Q6	Search Comparison > Multiple Choice	45 sec	5/ 5	✔
Q7	Recursion and Iteration > Multiple Choice	39 sec	3.75/ 5	⚠
Q8	Solution Assessment I > Multiple Choice	1 min 21 sec	5/ 5	✔
Q9	Recover Binary Search Tree > Coding	14 min 52 sec	30/ 30	✔
Q10	Reverse Nodes in k-Group > Coding	6 min 24 sec	30/ 30	✔



QUESTION 1



Correct Answer

Score 5

Delete Tree Nodes > Multiple Choice Binary Trees

QUESTION DESCRIPTION

Suppose we want to write a method to delete all nodes in a tree. We don't want simply set the root node to be null, because this would still leave the rest of the tree nodes hanging around. What type of traversal would be best for this type of tree deletion?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ☐ In-order
- ☐ Pre-order
- ☒ Post-order

No Comments

QUESTION 2



Correct Answer

Score 5

Stack: Insert at Bottom > Multiple Choice

QUESTION DESCRIPTION

Given the following snippet of code, what does the function solve() do?

In Python:

```
def insertAtBottom(stack, item):  
    if isEmpty(stack):  
        push(stack, item)  
    else:  
        temp = pop(stack)  
        insertAtBottom(stack, item)  
        push(stack, temp)  
  
def solve(stack):  
    if not isEmpty(stack):  
        temp = pop(stack)  
        solve(stack)  
        insertAtBottom(stack, temp)
```

In Java:

```
static void insert_at_bottom(char x){  
  
    if(st.isEmpty())  
        st.push(x);  
  
    else {  
        st.pop();  
        insert_at_bottom(x);  
        st.push(a);  
    }  
}  
  
static void solve(){  
    if(st.size() > 0){  
        char x = st.peek();  
        st.pop();  
        reverse();  
  
        insert_at_bottom(x);  
    }  
}
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ☒ ☐ reverse a stack
- ☐ place the last element at the top
- ☐ place the top element at the end
- ☐ none of the above

No Comments

QUESTION 3



Wrong Answer

Score 0

Design Front Middle Back Queue > Multiple Choice

QUESTION DESCRIPTION

Design a queue that supports `push` and `pop` operations in the front, middle, and back. We currently have the following functions:

- `FrontMiddleBack()` Initializes the queue.
- `void pushFront(int val)` Adds `val` to the **front** of the queue.
- `void pushMiddle(int val)` Adds `val` to the **middle** of the queue.
- `void pushBack(int val)` Adds `val` to the **back** of the queue.
- `int popFront()` Removes the **front** element of the queue and returns it. If the queue is empty, return `-1`.
- `int popMiddle()` Removes the **middle** element of the queue and returns it. If the queue is empty, return `-1`.
- `int popBack()` Removes the **back** element of the queue and returns it. If the queue is empty, return `-1`.

In Python:

```
class FrontMiddleBackQueue:

    def __init__(self):
        self.arr = deque()

    def pushFront(self, val: int) -> None:
        self.arr.appendleft(val)

    def pushMiddle(self, val: int) -> None:
        size = len(self.arr)
        mid = size // 2
        self.arr.insert(mid, val)

    def pushBack(self, val: int) -> None:
        // insert missing code here

    def popFront(self) -> int:
        if self.arr:
            return self.arr.popleft()
        else:
            return -1

    def popMiddle(self) -> int:
        size = len(self.arr)
        if size:
            if size - 1:
                mid = (size - 1) // 2
                val = self.arr[mid]
                del self.arr[mid]
                return val
            else:
                return self.arr.pop()
        else:
            return -1

    def popBack(self) -> int:
        if self.arr:
            return self.arr.pop()
        else:
            return -1
```

What is the missing code for the function `pushBack`?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ☐ self.arr.appendleft(val)
- ☒ self.arr.appendRight(val)
- ☒ self.arr.append(val)
- ☐ none of the above

No Comments

QUESTION 4



Correct Answer

Score 5

Count number of trees in a forest > Multiple Choice

QUESTION DESCRIPTION

Given n nodes of a forest (collection of trees), find the number of trees in the forest. What is missing in this incomplete implementation?

```
def addEdge(adj, u, v):
    adj[u].append(v)
    adj[v].append(u)

def DFSUtil(u, adj, visited):
    visited[u] = True
    for i in range(len(adj[u])):
        if (visited[adj[u][i]] == False):
            # insert code here

def countTrees(adj, V):
    visited = [False] * V
    res = 0
    for u in range(V):
        if (visited[u] == False):
            DFSUtil(u, adj, visited)
            res += 1
    return res
```


CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ☒ ☒ DFSUtil(adj[u][i], adj, visited)
- ☐ DFSUtil(adj[i][0], adj, visited)
- ☐ DFSUtil(adj[u][i], visited, adj)
- ☐ none of the above

No Comments

QUESTION 5

Wrong Answer

Score 0

Meeting Intervals > Multiple Choice Greedy Algorithms Arrays


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.

What is the most efficient way to sort the meeting intervals so that we can approach it with the greedy mindset?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)



☐ Sort by the meeting's start time


☐ Sort by the meeting's end time

☒ Sort by the meeting's total duration time

☐ We can't go with a greedy approach for this problem

No Comments

QUESTION 6

Correct Answer

Score 5


Search Comparison > Multiple Choice BFS/DFS

QUESTION DESCRIPTION

When is it better to use a **breadth first search** on a graph versus a depth first search?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)



☐ BFS explores vertices in the order of their distance from the source vertex, where distance is the minimum length of a path from the source vertex to the node.


☐ If our tree is broad, use DFS as BFS will take too much memory.

☐ If we know the solution lies somewhere deep in a tree or far from the source vertex in the graph, use DFS. If we know the solution is not that far from the source vertex, use BFS. Similarly, if our tree is very deep, choose BFS over DFS.

☒ All are true.

No Comments

QUESTION 7


Correct Answer

Score 3.75

Recursion and Iteration > Multiple Choice

Recursion

QUESTION DESCRIPTION

What are the advantages of recursion over iteration?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

☒ ☐ To reduce unnecessary calling of a function.

☒ ☐ Extremely useful when applying the same solution.


☒ ☐ Recursion reduce the length of code.

☒ ☐ Stacks evolutions and infix, prefix, postfix evaluations etc.

☐ ☐ It is not more efficient in terms of space and time complexity.

No Comments

QUESTION 8


Correct Answer

Score 5

Solution Assessment I > Multiple Choice

Time Complexity

QUESTION DESCRIPTION

Below is a function that produces all possible generalized abbreviations of a word. The ordering of the output does not matter.

Input: "done"
Output: ['4', '3e', '2n1', '2ne', '1o2', '1o1e', '1on1', '1one', 'd3', 'd2e', 'd1n1', 'd1ne', 'do2', 'do1e', 'don1', 'done']

Solution (Python):

```
def generate_all_abbreviations(word):  
    answer = []  
    abbreviation_helper(answer, '', word, 0, 0)  
    return answer  
  
def abbreviation_helper(answer, word_so_far, word, curr_position,  
consecutive_count):  
    if curr_position == len(word):  
        if consecutive_count != 0:  
            answer.append(word_so_far + str(consecutive_count))  
        else:  
            answer.append(word_so_far)  
    else:  
        abbreviation_helper(answer, word_so_far, word, curr_position + 1,  
consecutive_count + 1)  
        new_word_so_far = word_so_far  
        if consecutive_count != 0:  
            new_word_so_far += str(consecutive_count)  
        new_word_so_far += word[curr_position]  
        abbreviation_helper(answer, new_word_so_far, word, curr_position +  
1, 0)
```

Solution (Java)

```

ArrayList<String> generate_all_abbreviations(String word) {
    ArrayList<String> answer = new ArrayList<String>();
    abbreviation_helper(answer, "", word, 0, 0);
    return answer;
}

void abbreviation_helper(ArrayList<String> answer, String word_so_far,
String word, int curr_position, int consecutive_count) {

    if (curr_position == word.length()) {
        if (consecutive_count != 0) {
            answer.add(word_so_far + consecutive_count);
        } else {
            answer.add(word_so_far);
        }
    } else {
        abbreviation_helper(answer, word_so_far, word, curr_position + 1,
consecutive_count + 1);
        String new_word_so_far = word_so_far;
        if (consecutive_count != 0) {
            new_word_so_far += consecutive_count;
        }
        new_word_so_far += word.charAt(curr_position);
        abbreviation_helper(answer, new_word_so_far, word, curr_position +
1, 0);
    }
}

```

What is the time complexity for this solution?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

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

No Comments

QUESTION 9



Correct Answer

Score 30

Recover Binary Search Tree > Coding

Binary Search Trees

Binary Trees

QUESTION DESCRIPTION

Two elements of a binary search tree (BST) are swapped by mistake. Recover the tree without changing its structure.

Example 1:

Input: [1,3,null,null,2]

```

  1
 /
3
 \
 2

```

Output: [3,1,null,null,2]

Output: [3,1,null,null,2]

```

    3
   /
  1
   \
    2

```

Example 2:

Input: [3,1,4,null,null,2]

```

    3
   / \
  1   4
   /
  2

```

Output: [2,1,4,null,null,3]

```

    2
   / \
  1   4
   /
  3

```

CANDIDATE ANSWER

Language used: Java 8

```

1  /**
2   * public class TreeNode {
3   *     int val;
4   *     TreeNode left;
5   *     TreeNode right;
6   *     TreeNode(int x) { val = x; }
7   * }
8   */
9
10
11 static TreeNode first = null;
12 static TreeNode second = null;
13 static TreeNode prev = new TreeNode(Integer.MIN_VALUE);
14
15 public static void recoverTree(TreeNode root) {
16     traverse(root);
17
18     int temp = first.val;
19     first.val = second.val;
20     second.val = temp;
21 }
22
23 public static void traverse(TreeNode root) {
24     if (root == null) {
25         return;
26     }
27
28     traverse(root.left);
29
30     if (first == null && prev.val >= root.val) {
31         first = prev;
32     }

```

```

33
34     if (first != null && prev.val >= root.val) {
35         second = root;
36     }
37     prev = root;
38
39     traverse(root.right);
40 }
41

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	2	0.1133 sec	25.1 KB
Testcase 1	Easy	Sample case	✔ Success	2	0.1047 sec	24.9 KB
Testcase 2	Easy	Sample case	✔ Success	2	0.1361 sec	24.8 KB
Testcase 3	Easy	Sample case	✔ Success	2	0.1111 sec	25 KB
Testcase 4	Easy	Sample case	✔ Success	2	0.1184 sec	24.9 KB
Testcase 5	Easy	Sample case	✔ Success	2	0.0983 sec	25.1 KB
Testcase 6	Easy	Sample case	✔ Success	2	0.1292 sec	24.9 KB
Testcase 7	Easy	Sample case	✔ Success	2	0.1343 sec	25 KB
Testcase 8	Easy	Sample case	✔ Success	2	0.1575 sec	24.9 KB
Testcase 9	Easy	Sample case	✔ Success	2	0.1878 sec	25 KB
Testcase 10	Easy	Sample case	✔ Success	2	0.186 sec	24.8 KB
Testcase 11	Easy	Sample case	✔ Success	2	0.1319 sec	25 KB
Testcase 12	Easy	Sample case	✔ Success	2	0.0999 sec	24.9 KB
Testcase 13	Easy	Sample case	✔ Success	2	0.1128 sec	25 KB
Testcase 14	Easy	Sample case	✔ Success	2	0.1171 sec	24.9 KB

No Comments

QUESTION 10



Correct Answer

Score 30

Reverse Nodes in k-Group > Coding Linked Lists

QUESTION DESCRIPTION

Given a linked list, reverse the nodes of a linked list k at a time and return its modified list. k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of k then left-out nodes in the end should remain as it is.

Example:

Given this linked list: 1->2->3->4->5

For $k = 2$, you should return: 2->1->4->3->5

For $k = 3$, you should return: 3->2->1->4->5

Note:

- Only constant extra memory is allowed.
- You may not alter the values in the list's nodes, only nodes itself may be changed.

CANDIDATE ANSWER

Language used: **Java 8**

```

1      /**
2       * Definition for singly-linked list.
3       * public class ListNode {
4       *     int val;
5       *     ListNode next;
6       *     ListNode(int x) { val = x; }
7       * }
8       */
9      public static ListNode reverseKGroup(ListNode head, int k) {
10         ListNode curr = head;
11         int count = 0;
12         while (curr != null && count != k) { // find the k+1 node
13             curr = curr.next;
14             count++;
15         }
16         if (count == k) { // if k+1 node is found
17             curr = reverseKGroup(curr, k); // reverse list with k+1 node as
18 head
19             // head - head-pointer to direct part,
20             // curr - head-pointer to reversed part;
21             while (count-- > 0) { // reverse current k-group:
22                 ListNode tmp = head.next; // tmp - next head in direct part
23                 head.next = curr; // preappending "direct" head to the
24 reversed list
25                 curr = head; // move head of reversed part to a new node
26                 head = tmp; // move "direct" head to the next node in direct
27 part
28             }
29             head = curr;
30         }
31         return head;
32     }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	2	0.1168 sec	24.9 KB
Testcase 1	Easy	Sample case	✔ Success	2	0.1014 sec	24.9 KB
Testcase 2	Easy	Hidden case	✔ Success	2	0.1002 sec	25 KB
Testcase 3	Easy	Hidden case	✔ Success	2	0.0952 sec	24.8 KB
Testcase 4	Easy	Hidden case	✔ Success	2	0.0928 sec	24.9 KB
Testcase 5	Easy	Hidden case	✔ Success	2	0.1071 sec	24.9 KB
Testcase 6	Easy	Hidden case	✔ Success	2	0.1176 sec	24.9 KB
Testcase 7	Easy	Hidden case	✔ Success	2	0.1684 sec	25.1 KB
Testcase 8	Easy	Hidden case	✔ Success	2	0.1022 sec	24.9 KB
Testcase 9	Easy	Hidden case	✔ Success	2	0.1335 sec	25 KB
Testcase 10	Easy	Hidden case	✔ Success	2	0.1156 sec	24.9 KB
Testcase 11	Easy	Hidden case	✔ Success	2	0.1774 sec	24.7 KB
Testcase 12	Easy	Hidden case	✔ Success	2	0.1018 sec	24.9 KB
Testcase 13	Easy	Hidden case	✔ Success	2	0.1372 sec	24.8 KB
Testcase 14	Easy	Hidden case	✔ Success	2	0.11 sec	24.7 KB

No Comments

QUESTION 11



Correct Answer

Score 30

Expression Add Operators > Coding

Strings

Arrays

QUESTION DESCRIPTION

Given a string that contains only digits `0-9` and a target value, return all possibilities to add binary operators (not unary) `+`, `-`, or `*` between the digits so they evaluate to the target value.

Note that operands in the returned expressions should not contain leading zeros.

Example 1:

```
Input: num = "123", target = 6
Output: ["1*2*3", "1+2+3"]
```

Example 2:

```
Input: num = "232", target = 8
Output: ["2*3+2", "2+3*2"]
```

Example 3:

```
Input: num = "105", target = 5
Output: ["10-5", "1*0+5"]
```

Example 5:

```
Input: num = "3456237490", target = 9191
Output: []
```

Constraints:

- `1 <= num.length <= 10`
- `num` consists of only digits.
- `-231 <= target <= 231 - 1`

CANDIDATE ANSWER

Language used: Java 8

```
1      public static ArrayList<String> addOperators(String num, int target) {
2          ArrayList<String> res = new ArrayList<>();
3          StringBuilder sb = new StringBuilder();
4          dfs(res, sb, num, 0, target, 0, 0);
5          return res;
6      }
7
8      public static void dfs(List<String> res, StringBuilder sb, String num,
9 int pos, int target, long prev, long multi) {
10         if(pos == num.length()) {
11             if(target == prev) res.add(sb.toString());
12             return;
13         }
14         for(int i = pos; i < num.length(); i++) {
15             if(num.charAt(pos) == '0' && i != pos) break;
16             long curr = Long.parseLong(num.substring(pos, i + 1));
17             int len = sb.length();
18             if(pos == 0) {
19                 dfs(res, sb.append(curr), num, i + 1, target, curr, curr);
20                 sb.setLength(len);
```

```

21     } else {
22         dfs(res, sb.append("+").append(curr), num, i + 1, target, prev +
23 curr, curr);
24         sb.setLength(len);
25         dfs(res, sb.append("-").append(curr), num, i + 1, target, prev -
26 curr, -curr);
27         sb.setLength(len);
28         dfs(res, sb.append("*").append(curr), num, i + 1, target, prev -
29 multi + multi * curr, multi * curr);
30         sb.setLength(len);
31     }
32 }
33 }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	2	0.1604 sec	24.9 KB
Testcase 1	Easy	Sample case	✔ Success	2	0.0921 sec	24.6 KB
Testcase 2	Easy	Sample case	✔ Success	2	0.1103 sec	24.8 KB
Testcase 3	Easy	Sample case	✔ Success	2	0.1011 sec	24.6 KB
Testcase 4	Easy	Sample case	✔ Success	2	0.1636 sec	40.7 KB
Testcase 5	Easy	Hidden case	✔ Success	2	0.1159 sec	24.9 KB
Testcase 6	Easy	Hidden case	✔ Success	2	0.0943 sec	24.7 KB
Testcase 7	Easy	Hidden case	✔ Success	2	0.1146 sec	24.9 KB
Testcase 8	Easy	Hidden case	✔ Success	2	0.102 sec	24.6 KB
Testcase 9	Easy	Hidden case	✔ Success	2	0.1125 sec	24.4 KB
Testcase 10	Easy	Hidden case	✔ Success	2	0.0916 sec	24.6 KB
Testcase 11	Easy	Hidden case	✔ Success	2	0.098 sec	24.7 KB
Testcase 12	Easy	Hidden case	✔ Success	2	0.092 sec	24.6 KB
Testcase 13	Easy	Hidden case	✔ Success	2	0.0861 sec	25 KB
Testcase 14	Easy	Hidden case	✔ Success	2	0.1063 sec	24.9 KB

No Comments