



You can view this report online at : <https://www.hackerrank.com/x/tests/1330883/candidates/43271050/report>

Full Name:

Ruchit Bhardwaj

Email:

ruchitbh@usc.edu

Test Name:

CodePath SE103: Unit 10 Assessment - Summer 2022

Taken On:

9 Aug 2022 21:07:14 PDT

Time Taken:

30 min 13 sec/ 90 min

Work Experience:

2 years

Invited by:

CodePath

Skills Score:

Problem Solving (Advanced)

75/75

Tags Score:

Algorithms

75/75

Dynamic Programming

75/75

Interviewer Guidelines

75/75

Medium

75/75

Problem Solving

75/75

94.3%

250/265


scored in **CodePath SE103: Unit 10 Assessment - Summer 2022** in 30 min 13 sec on 9 Aug 2022 21:07:14 PDT

Recruiter/Team Comments:

No Comments.

Question Description	Time Taken	Score	Status
Q1 Given the following problem, https://leetcode.com/problems/letter-combinations-o > Multiple Choice	2 min 41 sec	0/ 5	✗
Q2 Given a sequence of numbers, i.e., [2, 5, 1, 6, 3, 7], which problem would be > Multiple Choice	34 sec	5/ 5	✓
Q3 How many combinations will be returned by combinationSum3(3,7)with the below imp > Multiple Choice	1 min 52 sec	0/ 5	✗
Q4 Coin Change > Multiple Choice	3 min 20 sec	0/ 5	✗
Q5 Permutations II without getCandidates > Coding	14 min 33 sec	60/ 60	✓
Q6 Beautiful Arrangement > Coding	4 min 49 sec	110/ 110	✓
Q7 Count String Permutations > Coding	2 min 4 sec	75/ 75	✓

QUESTION 1

Wrong Answer

Score 0


Multiple Choice

QUESTION DESCRIPTION

Given the following problem, <https://leetcode.com/problems/letter-combinations-of-a-phone-number>, what is the maximum total number of elements in the result if N is the number of characters in the input string? (~2 minutes)

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

☒ 

☐ 4^N


☐ $N!$

☒ N^2

☐ $4N^2$

No Comments

QUESTION 2

Correct Answer

Score 5


Multiple Choice

QUESTION DESCRIPTION

Given a sequence of numbers, i.e., [2, 5, 1, 6, 3, 7], which problem would be best solved by a dynamic programming algorithm? (~2 minutes)

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

☒ 

☐ Find kth largest element


☐ Find where a sub-sequence, e.g., [6, 3], exists in the sequence

☒ Partition the sequence into n or fewer bins to minimize the difference between sums of partitions

☐ Shuffle the sequence to guarantee uniform probability

No Comments

QUESTION 3

Wrong Answer

Score 0

Multiple Choice

QUESTION DESCRIPTION

How many combinations will be returned by `combinationSum3(3, 7)` with the below implementation?

Python:

```
def is_valid(s, k, n):
    return len(s) == k and sum(s) == n

def is_bad(s, k, n):
    sum_s = sum(s)
    len_s = len(s)
```

```

return len_s > k or sum_s > n or (len_s == k and sum_s != n)

def search(solutions, digits, s, k, n):
    if is_valid(s, k, n):
        solutions.append(s.copy())

    for d in digits:
        s.append(d)
        if not is_bad(s, k, n):
            search(solutions, set(digits) - set([d]), s, k, n)
        s.pop()

class Solution:
    def combinationSum3(self, k, n):
        """
        :type k: int
        :type n: int
        :rtype: List[List[int]]
        """
        solutions = []
        digits = range(1, 10)
        s = []
        search(solutions, digits, s, k, n)
        return solutions

```

Java:

```

class Solution {
    boolean isValid(List<Integer> s, int k, int n) {
        int sum = 0;
        for (int num: s) {
            sum += num;
        }
        return sum == n && s.size() == k;
    }

    boolean isBad(List<Integer> s, int k, int n) {
        int size = s.size(), sum = 0;
        for (int num: s) {
            sum += num;
        }
        return size > k || sum > n || (size == k && sum != n);
    }

    void search(List<List<Integer>> solutions, Set<Integer> digits,
List<Integer> s, int k, int n) {
        if (isValid(s, k, n)) {
            solutions.add(new ArrayList<>(s));
        }
        for (int d: digits) {
            s.add(d);
            if (!isBad(s, k, n)) {
                Set<Integer> newDigits = new HashSet<>(digits);
                newDigits.remove(d);
                search(solutions, newDigits, s, k, n);
            }
            s.remove(s.size()-1);
        }
    }

    public List<List<Integer>> combinationSum3(int k, int n) {
        List<List<Integer>> solutions = new ArrayList<>();
        Set<Integer> digits = new HashSet<>();
        for (int i = 1; i < 10; i++) {
            digits.add(i);
        }
        List<Integer> s = new ArrayList<>();
        search(solutions, digits, s, k, n);
        return solutions;
    }
}

```

```
}  
}
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ☐ 0
☒ 1
☐ 3
☒ 6

No Comments

QUESTION 4



Wrong Answer

Score 0

Coin Change > Multiple Choice

QUESTION DESCRIPTION

Given the coin change solution below, add protections against edge cases:

Python:

```
def coin_change(coins, amount):  
    """  
    :type coins: List[int]  
    :param coins: int  
    :param amount:  
    :return: int  
    """  
    min_c = min(coins)  
    h = {0: 0}  
    for i in range(1, min_c):  
        h[i] = -1  
    for i in range(min_c, amount + 1):  
        t = [h[i-c] for c in coins if c <= i and h[i - c] > -1]  
        if not t:  
            h[i] = -1  
        else:  
            h[i] = -1  
    return h[amount]
```

Java:

```
int coinChange(int[] coins, int amount) {  
    int minC = Integer.MAX_VALUE;  
    for (int c : coins) {  
        minC = Math.min(c, minC);  
    }  
    HashMap<Integer, Integer> h = new HashMap<>();  
    h.put(0, 0);  
    for (int i = 1; i < minC; i++) {  
        h.put(i, -1);  
    }  
    for (int i = minC; i < amount + 1; i++) {  
        List<Integer> t = new ArrayList<>();  
        int min = Integer.MAX_VALUE;  
        for (int c : coins) {  
            if (c <= i && h.get(i - c) > -1) {
```

```

        min = Math.min(min, h.get(i - c));
        t.add(h.get(i - c));
    }
}
if (t.isEmpty()) {
    h.put(i, -1);
} else {
    h.put(i, min);
}
}
return h.get(amount);
}

```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ☐ coins = []
- ☐ Amount < 0
- ☐ Amount > 2^64
- ☐ Coins = None
- ☒ All of the above
- ☐ Some of the above

No Comments

QUESTION 5



Correct Answer

Score 60

Permutations II without getCandidates > Coding

QUESTION DESCRIPTION

Given a collection of numbers that might contain duplicates, return all possible unique permutations.

Example:

```

Input: [1,1,2]
Output:
[
  [1,1,2],
  [1,2,1],
  [2,1,1]
]

```

Fill in the missing code (in the getCandidates function) for a correct solution to this problem.

CANDIDATE ANSWER

Language used: **Java 8**

```

1 public static List<List<Integer>> permuteUnique(int[] nums) {
2     List<Integer> temp = new ArrayList<>();
3     List<List<Integer>> ans = new ArrayList<>();
4     boolean[] visited = new boolean[nums.length];
5     Arrays.sort(nums);
6     permuteUnique(nums, temp, ans, visited);

```

```

7   return ans;
8 }
9
10 private static void permuteUnique(int[] nums, List<Integer> temp,
11 List<List<Integer>> ans, boolean[] visited) {
12     if (temp.size() == nums.length) {
13         ans.add(new ArrayList<>(temp));
14         return;
15     }
16
17     for (int i = 0; i < nums.length; i++) {
18         if (visited[i]) {
19             continue;
20         }
21
22         if (i > 0 && nums[i] == nums[i - 1] && visited[i - 1]) {
23             continue;
24         }
25
26         temp.add(nums[i]);
27         visited[i] = true;
28
29         permuteUnique(nums, temp, ans, visited);
30
31         visited[i] = false;
32         temp.remove(temp.size() - 1);
33     }
34 }
35

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	10	0.0971 sec	24.7 KB
Testcase 1	Easy	Hidden case	✔ Success	10	0.1081 sec	24.8 KB
Testcase 2	Easy	Hidden case	✔ Success	10	0.1007 sec	24.9 KB
Testcase 3	Easy	Hidden case	✔ Success	10	0.121 sec	24.7 KB
Testcase 4	Easy	Hidden case	✔ Success	10	0.0966 sec	24.8 KB
Testcase 5	Easy	Hidden case	✔ Success	10	0.1017 sec	24.7 KB

No Comments

QUESTION 6



Correct Answer

Score 110

Beautiful Arrangement > Coding

QUESTION DESCRIPTION

Suppose you have N integers from 1 to N. We define a beautiful arrangement as an array that is constructed by these N numbers successfully if one of the following is true for the i_{th} position ($1 \leq i \leq N$) in this array:

1. The number at the i_{th} position is divisible by i.
2. i is divisible by the number at the i_{th} position.

Now given N, how many beautiful arrangements can you construct?

Example 1:

```

Input: 2
Output: 2
Explanation:

```

The first beautiful arrangement is [1, 2]:

Number at the 1st position (i=1) is 1, and 1 is divisible by i (i=1).

Number at the 2nd position (i=2) is 2, and 2 is divisible by i (i=2).

The second beautiful arrangement is [2, 1]:

Number at the 1st position (i=1) is 2, and 2 is divisible by i (i=1).

Number at the 2nd position (i=2) is 1, and i (i=2) is divisible by 1.

Note:

1. N is a positive integer and will not exceed 15.

CANDIDATE ANSWER

Language used: Java 15

```
1 class Result {
2
3     /*
4      * Complete the 'countArrangement' function below.
5      *
6      * The function is expected to return an INTEGER.
7      * The function accepts INTEGER N as parameter.
8      */
9
10    public static int count = 0;
11    public static int countArrangement(int N) {
12        if (N == 0) return 0;
13        helper(N, 1, new int[N + 1]);
14        return count;
15    }
16
17    private static void helper(int N, int pos, int[] used) {
18        if (pos > N) {
19            count++;
20            return;
21        }
22
23        for (int i = 1; i <= N; i++) {
24            if (used[i] == 0 && (i % pos == 0 || pos % i == 0)) {
25                used[i] = 1;
26                helper(N, pos + 1, used);
27                used[i] = 0;
28            }
29        }
30    }
31 }
32 }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Sample	Easy	Sample case	✔ Success	10	0.0593 sec	31 KB
Testcase 1	Easy	Hidden case	✔ Success	10	0.0568 sec	31.2 KB
Testcase 0	Easy	Hidden case	✔ Success	10	0.079 sec	31.6 KB
Testcase 3	Easy	Hidden case	✔ Success	10	0.0692 sec	31.1 KB

Testcase 4	Easy	Hidden case	✓ Success	10	0.0702 sec	31.5 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.0598 sec	31.2 KB
Testcase 6	Easy	Hidden case	✓ Success	10	0.0585 sec	31.1 KB
Testcase 7	Easy	Hidden case	✓ Success	10	0.0852 sec	31.4 KB
Testcase 8	Easy	Hidden case	✓ Success	10	0.0886 sec	31.2 KB
Testcase 9	Easy	Hidden case	✓ Success	10	0.0813 sec	31.4 KB
Testcase 10	Easy	Hidden case	✓ Success	10	0.1438 sec	31.5 KB

No Comments

QUESTION 7



Correct Answer

Score 75

Count String Permutations > Coding

Dynamic Programming

Medium

Algorithms

Problem Solving

Interviewer Guidelines

QUESTION DESCRIPTION

Find the number of strings of a given length that can be formed under the following rules:

- Each letter is a vowel, that is, it is in the set $\{a, e, i, o, u\}$.
- The letter a may only be followed by the letter e .
- An e may only be followed by an a or an i .
- An i may not be next to another i .
- The letter o may only be followed by an i or a u .
- The letter u may only be followed by an a .

Example

To illustrate some of the rules, start with the string $s = 'a'$ and build to the right.

1. ' a ' may only be followed by ' e ', so the new string can be ' ae '.
2. ' ae ' may only be followed by ' a ' or ' i ', so the new string can be ' aea ' or ' aei '.
3. ' aea ' must be ' $aeae$ ' next, and ' aei ' can be ' $aeia$ ', ' $aeie$ ', ' $aeio$ ', or ' $aeiu$ ' because an ' i ' cannot follow another ' i '.

Analyses of lengths of strings up to 3 are in the samples below. Since the number of permutations might be very large, return the value modulo $(10^9 + 7)$.

Function Description

Complete the `countPerms` function in the editor below.

`countPerms` has the following parameter(s):

int n: the length of string to analyze

Returns:

int: the number of permutations, modulo $(10^9 + 7)$

Constraints

- $0 < n < 10^5$

▼ Input Format For Custom Testing

Input from stdin will be processed as follows and passed to the function.

The only line contains an integer, n , the length of the string to analyze.

▼ Sample Case 0

Sample Input

STDIN	Function
-----	-----
1	→ length of string to analyze n = 1

Sample Output

5

Explanation

There are 5 strings of length 1, each containing a single vowel.

$5 \bmod (10^9+7) = 5$.

▼ Sample Case 1

Sample Input

STDIN	Function
-----	-----
2	→ length of string to analyze n = 2

Sample Output

10

Explanation

There are 10 strings of length 2: {'io', 'iu', 'oi', 'ou', 'ua', 'ae', 'ea', 'ei', 'ia', 'ie'}.

$10 \bmod (10^9+7) = 10$

▼ Sample Case 2

Sample Input

STDIN	Function
-----	-----
3	→ length of string to analyze n = 3

Sample Output

19

Explanation

There are 19 strings of length 3: {'iua', 'oia', 'oie', 'oio', 'oiu', 'oua', 'uae', 'aea', 'aei', 'eae', 'eia', 'eie', 'eio', 'eiu', 'iae', 'iea', 'iei', 'ioi', 'iou'}.

$19 \bmod (10^9+7) = 19$

INTERVIEWER GUIDELINES

▼ Solution

Skills: Dynamic Programming, Algorithms, Problem Solving

Optimal Solution:

The problem can be solved using dynamic programming. We first refactor the rules to obtain the characters that each vowel can succeed. The modified rules would be as follows:-

'a' can follow characters 'e', 'i' and 'u'

'e' can follow characters 'a' and 'i'

'i' can follow characters 'e' and 'o'

'o' can follow character 'i'

'u' can follow characters 'i' and 'o'

Using these rules, we create a dp array of size 5, with each element corresponding to the possible number of strings ending at a vowel. The final answer is the sum of the possible number of strings ending at each of the 5 vowels.

```
def countPerms(n):
    mod = int(1e9+7)
    vowels = "aeiou"
    #Creating a dp hashmap corresponding to the total possible number of
    strings ending at a,e,i,o,u
    dp={}
    for i in vowels:
        dp[i]=1

    #Refactored the rules to get characters that a particular vowel can
    come after
    for i in range(1,n):
        temp={}
        temp['a'] = dp['e'] + dp['i'] + dp['u']
        temp['e'] = dp['a'] + dp['i']
        temp['i'] = dp['e'] + dp['o']
        temp['o'] = dp['i']
        temp['u'] = dp['i'] + dp['o']
        dp=temp
    ret = 0
    #Taking sum of n length strings ending at each vowel
    for i in dp:
        ret += dp[i]

    return ret%mod
```

▼ Complexity Analysis

Time Complexity - $O(n)$

Space Complexity - $O(1)$

CANDIDATE ANSWER










Language used: **Java 15**

```
1  class Result {
2
3      /*
4       * Complete the 'countPerms' function below.
5       *
6       * The function is expected to return an INTEGER.
7       * The function accepts INTEGER n as parameter.
8       */
9
10     public static int countPerms(int n) {
11         int a = 0, e = 1, i = 2, o = 3, u = 4, MOD = (int) (1e9 + 7);
12         long[] dp = new long[5], prevDP = new long[5];
13         Arrays.fill(prevDP, 1L);
14         while (n-- > 1) {
15             dp[a] = (prevDP[e] + prevDP[i] + prevDP[u]) % MOD;
16             dp[e] = (prevDP[a] + prevDP[i]) % MOD;
17             dp[i] = (prevDP[e] + prevDP[o]) % MOD;
18             dp[o] = prevDP[i];
19             dp[u] = (prevDP[i] + prevDP[o]) % MOD;
20             long[] tmp = dp; dp = prevDP; prevDP = tmp;
21         }
```

```

21         }
22         return (int) ((prevDP[a] + prevDP[e] + prevDP[i] + prevDP[o] +
23 prevDP[u]) % MOD);
24     }
25 }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Sample Input	Easy	Sample case	 Success	1	0.061 sec	31 KB
Sample Input	Easy	Sample case	 Success	1	0.0703 sec	31.4 KB
Sample Input	Easy	Sample case	 Success	1	0.1109 sec	31.5 KB
Dynamic Programming Without Integer Overflow	Easy	Sample case	 Success	8	0.0852 sec	31.1 KB
Dynamic Programming Without Integer Overflow	Easy	Hidden case	 Success	8	0.0619 sec	31.2 KB
Dynamic Programming Without Integer Overflow	Easy	Hidden case	 Success	8	0.0666 sec	31.1 KB
Dynamic Programming With Integer Overflow	Medium	Sample case	 Success	8	0.0604 sec	31 KB
Dynamic Programming With Integer Overflow	Medium	Hidden case	 Success	8	0.06 sec	31.1 KB
Dynamic Programming With Integer Overflow	Medium	Hidden case	 Success	8	0.0693 sec	31.1 KB
Dynamic Programming With Integer Overflow	Hard	Hidden case	 Success	8	0.0847 sec	31.3 KB
Dynamic Programming With Integer Overflow	Hard	Hidden case	 Success	8	0.0742 sec	31.4 KB
Dynamic Programming With Integer Overflow	Hard	Hidden case	 Success	8	0.0837 sec	31.2 KB

No Comments