You can view this report online at : https://www.hackerrank.com/x/tests/1121393/candidates/28543915/report

Full Name:	My Nguyen
Email:	nguyen_my@yahoo.com
Test Name:	**Dynamic Programming 2 Assessment 2021**
Taken On:	9 Aug 2021 21:22:08 PDT
Time Taken:	22 min 50 sec/ 90 min
Work Experience:	3 years
Contact Number:	+14084096862
Resume:	https://hackerrank-resumes.s3.amazonaws.com/412894/JhbK9vK_4Bhc4Gvuv7s5hgcFJGeFCAThWIiNY1UGAfhwRPsrmVekT5ZtKXgX8QA2Ag/My_Nguyen_Resume.PDF
Linkedin:	https://www.linkedin.com/in/my-nguyen-87849
Invited by:	Curriculum
Skills Score:	Problem Solving (Intermediate) 0/75
Tags Score:	Algorithms 0/75 Arrays 0/75 Data Structures 0/75 Dynamic Programming 0/75 Medium 0/75 Problem Solving 0/75

20.8%

55/265

scored in **Dynamic Programming 2 Assessment 2021** in 22 min 50 sec on 9 Aug 2021 21:22:08 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**	3 min 2 sec	5/ 5	✓
Q2 Given a sequence of numbers, i.e., [2, 5, 1, 6, 3, 7], which problem would be > **Multiple Choice**	28 sec	5/ 5	✓
Q3 How many combinations will be returned by combinationSum3(3,7)with the below imp > **Multiple Choice**	1 min	0/ 5	✗
Q4 Permutations II without getCandidates > **Coding**	8 min 23 sec	0/ 60	✗
Q5 Coin Change > **Multiple Choice**	1 min 31 sec	5/ 5	✓
Q6 Beautiful Arrangement > **Coding**	3 min 28 sec	40/ 110	✓

QUESTION 1

Correct Answer

Score 5

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)

- [x] 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
- [x] 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

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

The candidate did not manually submit any code. The last compiled version has been auto-submitted and the score you see below is for the auto-submitted version.

Language used: **Java 8**

```
1  static class Tuple {
2      int c;
3      List<Integer> otherNums;
4
5      Tuple(int c, List<Integer> otherNums) {
6          this.c = c;
7          this.otherNums = otherNums;
8      }
9  }
10
```

```
static List<Tuple> getCandidates(List<Integer> nums) {
    Tuple tuple = new Tuple(1, nums);
    List<Tuple> result = new ArrayList<>();
    result.add(tuple);
    return result;
}

static void search(List<List<Integer>> solutions, List<Integer> nums,
List<Integer> s) {
    if (nums == null || nums.isEmpty()) {
        solutions.add(new ArrayList<Integer>(s));
    } else {
        for (Tuple t: getCandidates(nums)) {
            s.add(t.c);
            search(solutions, t.otherNums, s);
            s.remove(s.size()-1);
        }
    }
}

static public List<List<Integer>> permuteUnique(int[] nums) {
    List<List<Integer>> solutions = new ArrayList<>();
    List<Integer> numsList = new ArrayList<Integer>();
    for (int num: nums) {
        numsList.add(num);
    }
    search(solutions, numsList, new ArrayList<Integer>());
    return solutions;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Runtime Error	0	0.1463 sec	28.7 KB
Testcase 1	Easy	Hidden case	Runtime Error	0	0.1223 sec	28.4 KB
Testcase 2	Easy	Hidden case	Runtime Error	0	0.1264 sec	29.2 KB
Testcase 3	Easy	Hidden case	Runtime Error	0	0.1435 sec	28.8 KB
Testcase 4	Easy	Hidden case	Runtime Error	0	0.1359 sec	28.1 KB
Testcase 5	Easy	Hidden case	Runtime Error	0	0.1249 sec	28.9 KB

No Comments

QUESTION 5

Correct Answer

Score 5

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 6

Correct Answer

Beautiful Arrangement › Coding

QUESTION DESCRIPTION

Score 40

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 <= i <= 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 8**

```
1
2   // Complete the countArrangement function below.
3   static int countArrangement(int N) {
4       return N;
5   }
6
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Sample	Easy	Sample case	Success	10	0.0688 sec	23.7 KB
Testcase 1	Easy	Hidden case	Success	10	0.0677 sec	23.5 KB
Testcase 0	Easy	Hidden case	Success	10	0.0723 sec	23.4 KB
Testcase 3	Easy	Hidden case	Success	10	0.069 sec	23.4 KB
Testcase 4	Easy	Hidden case	Wrong Answer	0	0.0859 sec	23.5 KB
Testcase 5	Easy	Hidden case	Wrong Answer	0	0.0731 sec	23.4 KB
Testcase 6	Easy	Hidden case	Wrong Answer	0	0.0917 sec	23.4 KB
Testcase 7	Easy	Hidden case	Wrong Answer	0	0.0863 sec	23.6 KB
Testcase 8	Easy	Hidden case	Wrong Answer	0	0.0694 sec	23.4 KB
Testcase 9	Easy	Hidden case	Wrong Answer	0	0.0659 sec	23.4 KB
Testcase 10	Easy	Hidden case	Wrong Answer	0	0.0766 sec	23.7 KB

No Comments

QUESTION 7

Wrong Answer

Score 0

Longest increasing subsequence › Coding

Dynamic Programming | Data Structures | Medium | Algorithms | Problem Solving | Arrays

QUESTION DESCRIPTION

A sub-sequence is a sequence that can be created by deleting zero or more elements from the original sequence while maintaining order.

A sequence *S* is said to be increasing if every element in the sequence is greater than the previous element in that sequence, i.e for every element *S[i+1]*, *S[i] < S[i+1]*. Mathematically, for the sequence *S = (S[1]...S[n-1])*, *S[i] < S[i+1]* $\forall i \in [1, n-1]$.

You will be given an array of integers and must determine the length of the longest increasing subsequence.

For example, your array *s = [1, 2, 2, 3, 1, 6]*. Two examples of strictly increasing subsequences of that array are *(1,2)*, *(1, 2, 3)*. Note that the *2* cannot repeat in the second subsequence as $2 \not< 2$. The longest increasing subsequence has a length of *4*: *LIS = [1,2,3,6]*.

Function Description

Complete the function *findLIS* in the editor below. The function must return the length of the longest increasing subsequence that can be created from the array.

findLIS has the following parameter(s):

s[s[0],...s[n-1]]: an array of integers

Constraints:

- $1 \le n < 1000$
- $1 \le s[i] \le 1000000$

▼ Input Format for Custom Testing

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

The first line contains an integer *n*, the size of the array *s*.
Each of the next *n* lines contains an integer *s[i]* where $1 \le i \le n$.

▼ Sample Case 0

Sample Input 0

```
STDIN    Function Parameters
-----    -------------------
3     →  s[] Size = 3
1     →  s[] = [ 1,
4                4,
3                3 ]
```

Sample Output 0

```
2
```

Explanation 0

Inputs are *s=[1,4,3]*. Increasing subsequences are *[1,4]* and *[1,3]*.
The longest increasing sub-sequence has *2* elements.

▼ Sample Case 1

Sample Input 1

```
STDIN   Function Parameters
-----   --------------------
5    →  s[] Size = 5
1    →  s = [1,4,5,2,6]
4
5
2
6
```

Sample Output 1

```
4
```

Explanation 1

Inputs are *s=[1,4,5,2,6]*. Some increasing subsequences are *[1,4,5,6]*, *[4,5,6]*, *[5,6]* and *[2,6]*.
The longest increasing sub-sequence has *4* elements.

▼ Sample Case 2

Sample Input 2

```
STDIN    Function Parameters
-----    --------------------
4     →  s[] Size = 4
2     →  s[] = [ 2, 3, 3, 5 ]
3
3
5
```

Sample Output

```
3
```

Explanation

Inputs are *s=[2,3,3,5]*. Increasing subsequences are *[2,3,5]*, *[2,3]*, *[3,5]* and *[2,5]*.
The longest increasing sub-sequence has *3* elements.

CANDIDATE ANSWER

Language used: **Java 8**

```
1  /*
2   * Complete the function below.
3   */
4
5      static int findLIS(int[] s) {
6          return s.length;
7      }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Wrong Answer	0	0.0836 sec	25 KB
TestCase 1	Easy	Sample case	Wrong Answer	0	0.0913 sec	25 KB
TestCase 2	Easy	Sample case	Wrong Answer	0	0.0872 sec	24.8 KB
TestCase 3	Easy	Hidden case	Wrong Answer	0	0.0843 sec	24.7 KB
TestCase 4	Easy	Hidden case	Wrong Answer	0	0.0854 sec	25 KB
TestCase 5	Medium	Hidden case	Wrong Answer	0	0.0864 sec	25.1 KB
TestCase 6	Hard	Hidden case	Wrong Answer	0	0.1188 sec	26.4 KB
TestCase 7	Hard	Hidden case	Wrong Answer	0	0.1192 sec	28.8 KB

No Comments

PDF generated at: 10 Aug 2021 04:47:01 UTC