

Experiment 3.1

Student Name: Yash Gupta

Branch: BE-CSE

Semester: 6

Subject Name: CC LAB

UID: 20BCS5009

Section/Group: 20BCS_DM-716 B

Date of Performance: 01/05/23

Subject Code: 20CSP_351

1. Aim:

To implement the concept of Greedy Algorithm.

2. Objective:

- The objective is to build problem solving capability and to learn the basic concepts of data structures.
- Understand the problem and find out better approach to solve particular problem

3. LeetCode code and output:

• Candy

The screenshot shows the LeetCode problem page for '135. Candy'. At the top, there are tabs for 'Description', 'Editorial', 'Solutions (1.9K)', and 'Submissions'. The problem title '135. Candy' is followed by a 'Hard' difficulty tag, a green checkmark, and statistics: 5.4K likes, 358 comments, and a star icon. Below this is a 'Companies' tag. The problem description states: 'There are n children standing in a line. Each child is assigned a rating value given in the integer array `ratings`. You are giving candies to these children subjected to the following requirements: Each child must have at least one candy. Children with a higher rating get more candies than their neighbors. Return the minimum number of candies you need to have to distribute the candies to the children.'

Example 1:

```
Input: ratings = [1,0,2]
Output: 5
Explanation: You can allocate to the first, second and third child with 2, 1, 2 candies respectively.
```

Example 2:

```
Input: ratings = [1,2,2]
Output: 4
Explanation: You can allocate to the first, second and third child with 1, 2, 1 candies respectively. The third child gets 1 candy because it satisfies the above two conditions.
```

Constraints:

- $n == ratings.length$
- $1 \leq n \leq 2 * 10^4$
- $0 \leq ratings[i] \leq 2 * 10^4$

At the bottom, the statistics are: Accepted 306.7K, Submissions 746.2K, and Acceptance Rate 41.1%.

```
class Solution:
    def candy(self, ratings: List[int]) -> int:
        length = len(ratings)
        candies = [1] * length
        for i in range(1, length):
            if ratings[i] > ratings[i-1] and candies[i] <= candies[i-1]:
                candies[i] = candies[i-1] + 1
        for i in range(length - 2, -1, -1):
            if ratings[i] > ratings[i + 1] and candies[i] <= candies[i+1]:
                candies[i] = candies[i+1] + 1
        return sum(candies)
```

Yash_Gupta202

May 13, 2023 19:55

Details

+ Solution

Python3

Runtime

154 ms

Beats

94.28%

Memory

19.3 MB

Beats

19.78%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

```
class Solution:
    def candy(self, ratings: List[int]) -> int:
        length = len(ratings)
        candies = [1] * length
        for i in range(1, length):
            if ratings[i] > ratings[i-1] and candies[i] <= candies[i-1]:
                candies[i] = candies[i-1] + 1
        for i in range(length - 2, -1, -1):
            if ratings[i] > ratings[i + 1] and candies[i] <= candies[i+1]:
                candies[i] = candies[i+1] + 1
        return sum(candies)
```

Console ^

Run

Submit

• Remove Duplicate Letters

[Description](#) [Editorial](#) [Solutions \(1.2K\)](#) [Submissions](#)

316. Remove Duplicate Letters

Medium 6.8K 433 ☆ ↻

Companies

Given a string `s`, remove duplicate letters so that every letter appears once and only once. You must make sure your result is **the smallest in lexicographical order** among all possible results.

Example 1:

Input: `s = "bcabc"`
Output: `"abc"`

Example 2:

Input: `s = "cbacdcbc"`
Output: `"acdb"`

Constraints:

- `1 <= s.length <= 104`
- `s` consists of lowercase English letters.

Note: This question is the same as 1081: <https://leetcode.com/problems/smallest-subsequence-of-distinct-characters/>

Accepted **229.9K** | Submissions **508.4K** | Acceptance Rate **45.2%**

Seen this question in a real interview before? **1/4**

Yes No

Discussion (24)

```
class Solution:
    def removeDuplicateLetters(self, s: str) -> str:
        stack = []
        seen = set()
        last_occurrence = {}
        for i in range(len(s)):
            last_occurrence[ s[i] ] = i

        # print(last_occurrence)

        for i, ch in enumerate(s):
            if( ch in seen ):
                continue
```

```

else:
    # 3
    while( stack and stack[-1] > ch and last_occurance[stack[-1]] > i ):
        removed_char = stack.pop()
        seen.remove(removed_char)
    seen.add(ch)
    stack.append(ch)
# print(stack)
return ''.join(stack)

```

Yash_Gupta202
May 13, 2023 20:01

Details
+ Solution

Python3

Runtime **57 ms**
Beats **13.50%**
Memory **16.3 MB**
Beats **21.47%**

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

```

class Solution:
    def removeDuplicateLetters(self, s: str) -> str:
        stack = []
        seen = set()
        last_occurance = {}
        for i in range(len(s)):
            last_occurance[ s[i] ] = i

        # print(last_occurance)

        for i, ch in enumerate(s):
            if( ch in seen ):
                continue
            else:
                # 3
                while( stack and stack[-1] > ch and last_occurance[stack[-1]] > i ):
                    removed_char = stack.pop()
                    seen.remove(removed_char)
                seen.add(ch)
                stack.append(ch)
            # print(stack)
        return ''.join(stack)

```

Console ^

Run
Submit