

Experiment 3.2

Student Name: Yash Gupta
Branch: BE-CSE
Semester: 6
Subject Name: CC LAB

UID: 20BCS5009
Section/Group: 20BCS_DM-716 B
Date of Performance: 08/05/23
Subject Code: 20CSP_351

1. Aim:

To implement the concept of backtracking.

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:

• Binary Watch

DescriptionEditorialSolutions (1.1K)Submissions


401. Binary Watch

Easy1.2K2.2K☆

Companies

A binary watch has 4 LEDs on the top to represent the hours (0-11), and 6 LEDs on the bottom to represent the minutes (0-59). Each LED represents a zero or one, with the least significant bit on the right.

- For example, the below binary watch reads "4:51".



Given an integer `turnedOn` which represents the number of LEDs that are currently on (ignoring the PM), return *all possible times the watch could represent*. You may return the answer in **any order**.

The hour must not contain a leading zero.

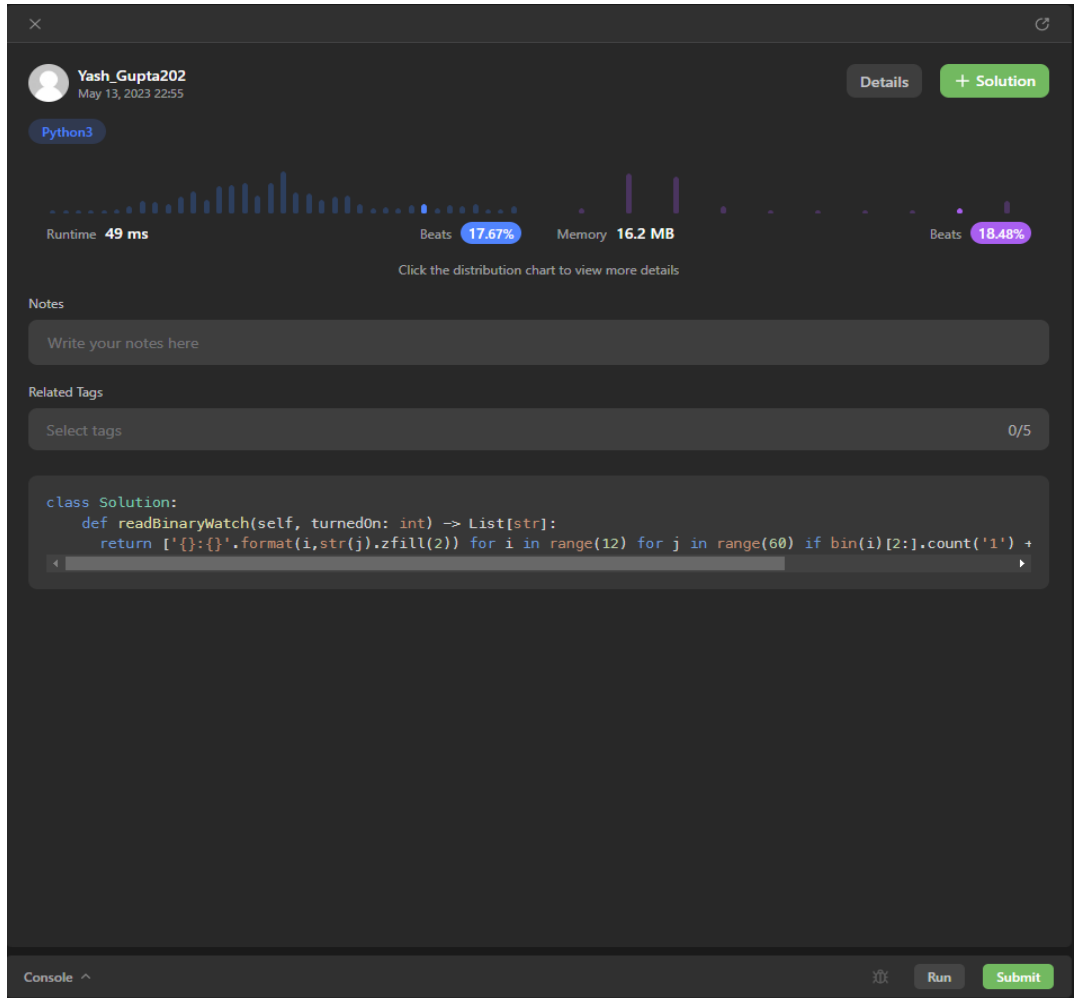
- For example, "01:00" is not valid. It should be "1:00".

The minute must be consist of two digits and may contain a leading zero.

```

class Solution:
    def readBinaryWatch(self, turnedOn: int) -> List[str]:
        return ['{}:{}'.format(i,str(j).zfill(2)) for i in range(12) for j in range(60) if bin(i)[2:].count('1') +
bin(j)[2:].count('1') == turnedOn]

```



• Subsets

```

class Solution:
    def subsets(self, nums: List[int]) -> List[List[int]]:
        res = []
        def dfs(i,subset):
            if i>=len(nums):
                res.append(subset)
                return
            dfs(i+1,subset+[nums[i]])
            dfs(i+1,subset)
        dfs(0,[])
        return res

```

Description
Editorial
Solutions (7.8K)
Submissions

78. Subsets

Medium
14.3K
209

Companies

Given an integer array `nums` of **unique** elements, return *all possible subsets (the power set)*.

The solution set **must not** contain duplicate subsets. Return the solution in **any order**.

Example 1:

Input: `nums = [1,2,3]`
Output: `[[], [1], [2], [1,2], [3], [1,3], [2,3], [1,2,3]]`

Example 2:

Input: `nums = [0]`
Output: `[[], [0]]`

Constraints:

- `1 <= nums.length <= 10`
- `-10 <= nums[i] <= 10`
- All the numbers of `nums` are **unique**.

Accepted **1.4M** | Submissions **1.9M** | Acceptance Rate **75.1%**

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

Yes No

Discussion (25)

Similar Questions

Yash_Gupta202
May 13, 2023 23:02

Details
+ Solution

Python3

Runtime **55 ms**
Beats **7.35%**
Memory **16.5 MB**
Beats **18.38%**

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

```

class Solution:
    def subsets(self, nums: List[int]) -> List[List[int]]:
        res = []
        def dfs(i, subset):
            if i >= len(nums):
                res.append(subset)
                return
            dfs(i+1, subset+[nums[i]])
            dfs(i+1, subset)
        dfs(0, [])
        return res

```