Binary Watch

A binary watch has 4 LEDs on the top which represent the hours (0-11), and the 6 LEDs on the bottom represent the minutes (0-59).

Each LED represents a zero or one, with the least significant bit on the right.

For example, the above binary watch reads "3:25".

Given a non-negative integer *n* which represents the number of LEDs that are currently on, return all possible times the watch could represent.

Example:

```
Input: n = 1
Return: ["1:00", "2:00", "4:00", "8:00", "0:01", "0:02", "0:04", "0:08", "0:16", "0
:32"]
```

Note:

- The order of output does not matter.
- 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, for example "10:2" is not valid, it should be "10:02".

Subscribe to see which companies asked this question

Solution 1

Just go through the possible times and collect those with the correct number of one-bits.

Python:

```
def readBinaryWatch(self, num):
    return ['%d:%02d' % (h, m)
            for h in range(12) for m in range(60)
            if (bin(h) + bin(m)).count('1') == num]
```

Java:

```
public List<String> readBinaryWatch(int num) {
    List<String> times = new ArrayList<>();
    for (int h=0; h<12; h++)
        for (int m=0; m<60; m++)
            if (Integer.bitCount(h * 64 + m) == num)
                times.add(String.format("%d:%02d", h, m));
    return times;
}
```

written by StefanPochmann original link here

Solution 2

```
vector<string> readBinaryWatch(int num) {
    vector<string> rs;
    for (int h = 0; h < 12; h++)
    for (int m = 0; m < 60; m++)
        if (bitset<10>(h << 6 | m).count() == num)
            rs.emplace_back(to_string(h) + (m < 10 ? ":0" : ":") + to_string(m));
    return rs;
}
```

written by mzchen original link here

Solution 3

```
public class Solution {
    public List<String> readBinaryWatch(int num) {
        List<String> res = new ArrayList<>();
        int[] nums1 = new int[]{8, 4, 2, 1}, nums2 = new int[]{32, 16, 8, 4, 2,
1};
        for(int i = 0; i <= num; i++) {
            List<Integer> list1 = generateDigit(nums1, i);
            List<Integer> list2 = generateDigit(nums2, num - i);
            for(int num1: list1) {
                if(num1 >= 12) continue;
                for(int num2: list2) {
                    if(num2 >= 60) continue;
                    res.add(num1 + ":" + (num2 < 10 ? "0" + num2 : num2));
                }
            }
        }
        return res;
    }

    private List<Integer> generateDigit(int[] nums, int count) {
        List<Integer> res = new ArrayList<>();
        generateDigitHelper(nums, count, 0, 0, res);
        return res;
    }

    private void generateDigitHelper(int[] nums, int count, int pos, int sum, Lis
t<Integer> res) {
        if(count == 0) {
            res.add(sum);
            return;
        }

        for(int i = pos; i < nums.length; i++) {
            generateDigitHelper(nums, count - 1, i + 1, sum + nums[i], res);
        }
    }
}
```

written by xieta00221 original link here