[401. Binary Watch](https://leetcode-cn.com/problems/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".

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/binary-watch

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

#### [401. 二进制手表](https://leetcode-cn.com/problems/binary-watch/)

二进制手表顶部有 4 个 LED 代表小时（0-11），底部的 6 个 LED 代表分钟（0-59）。

每个 LED 代表一个 0 或 1，最低位在右侧。



例如，上面的二进制手表读取 “3:25”。

给定一个非负整数 n 代表当前 LED 亮着的数量，返回所有可能的时间。

案例:

输入: n = 1

返回: ["1:00", "2:00", "4:00", "8:00", "0:01", "0:02", "0:04", "0:08", "0:16", "0:32"]

注意事项:

输出的顺序没有要求。

小时不会以零开头，比如 “01:00” 是不允许的，应为 “1:00”。

分钟必须由两位数组成，可能会以零开头，比如 “10:2” 是无效的，应为 “10:02”

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/binary-watch

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

为了更加直观的看到回溯算法的解题步骤和思路，我们从leetcode中选择几道回溯算法相关问题进行解析。

例题1： leetcode-401-binary-watch

【思路】

1. 从问题抽象到子问题：

   问题： 在0到9这10个位置上可以点亮n盏灯

   操作： 在位置i上点亮1盏灯

   操作后得到子问题：在i+1到9的位置上可以点亮n-1盏灯

2. 递归结束条件：

   n==0,当灯都已经点亮时，得到一个最终解，递归终止，逐步返回。同时也逐步回溯，把沿途点亮的灯都灭掉。

【代码】

import java.util.ArrayList;

import java.util.List;

public class Solution {

public List<String> readBinaryWatch(int num) {

List<String> time = new ArrayList<String>();

int[] watch = new int[10];

return null;

}

public void Trial(int n, int a, int[] watch, List<String> times){

if(n==0){

String time = "";

int hour = SumOfIntArray(3, 0, watch);

time = time + hour + ":";

int minute = SumOfIntArray(9, 4, watch);

if(minute < 10) time = time + 0 + minute;

else time = time + minute;

times.add(time);

return;

}

for(; a<=9; a++){

watch[a] = 1;

if( ! ((watch[0] == 1 && watch[1] == 1) ||

(watch[4] == 1 && watch[5] == 1 && watch[6] == 1 && watch[7] == 1)))

Trial(n-1, a+1, watch, times);

watch[a] = 0;

}

return;

}

public static int SumOfIntArray(int a,int b,int[] times){

//a是低位，b是高位，a是3或9，b是0或4

int sum = 0;

int mi = 0;

for(int i=a;i>=b;i--){

sum = (int) (sum +times[i]\*Math.pow(2, mi));

mi++;

}

return sum;

}

————————————————

版权声明：本文为CSDN博主「a247027417」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/a247027417/article/details/83784036