少儿编程

21工作室出品

本次兴趣课分两个项目

- 第一个项目
  - 二叉树与二进制
- 第二个项目
  - 老师如何计算当前时间到放假时间是第几周的

二分查找

- 二分查找也称折半查找（Binary Search），它是一种效率较高的查找方法。但是，折半查找要求线性表必须采用顺序存储结构，而且表中元素按关键字有序排列。
- 首先，假设表中元素是按升序排列，将表中间位置记录的关键字与查找关键字比较，如果两者相等，则查找成功；否则利用中间位置记录将表分成前、后两个子表，如果中间位置记录的关键字大于查找关键字，则进一步查找前一子表，否则进一步查找后一子表。重复以上过程，直到找到满足条件的记录，使查找成功，或直到子表不存在为止，此时查找不成功。

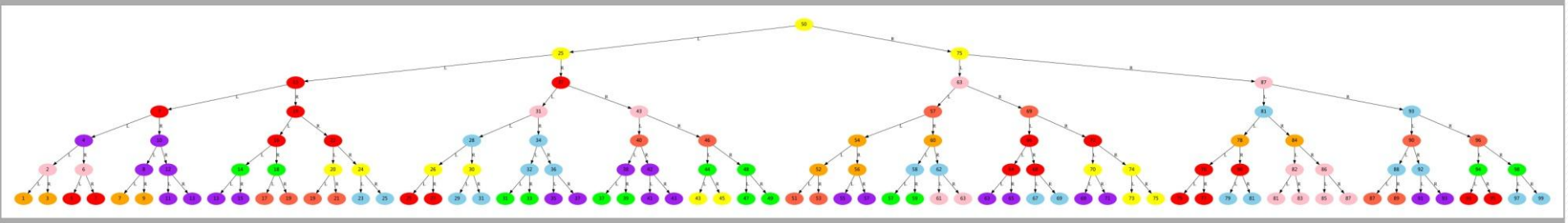
猜数游戏源代码

NumberGuessChinies.py - D:\tmp\yiyi学编程\NumberGuessChinies.py (3.7.4)

File Edit Format Run Options Window Help

```
import random

secret = random.randint(1, 100)
guess = 0
tries = 0

print ("你好！我是一个猜数机器人，我有一个秘密！")
print ("这个数字在1到99之间。我会给你六次机会。")

while guess != secret and tries < 6:
    guess = int(input("你猜的数是多少？"))
    if guess < secret:
        print ("太低了，像个小狗！")
    elif guess > secret:
        print ("太高了，像个长颈鹿！")
    tries = tries + 1

if guess == secret:
    print ("呕天呐，你找到它了，你知道了我的秘密，你做到了！")
else:
    print ("没机会了，祝你下次好运！")
    print ("这个秘密数字是", secret)
```

Ln: 1 Col: 0



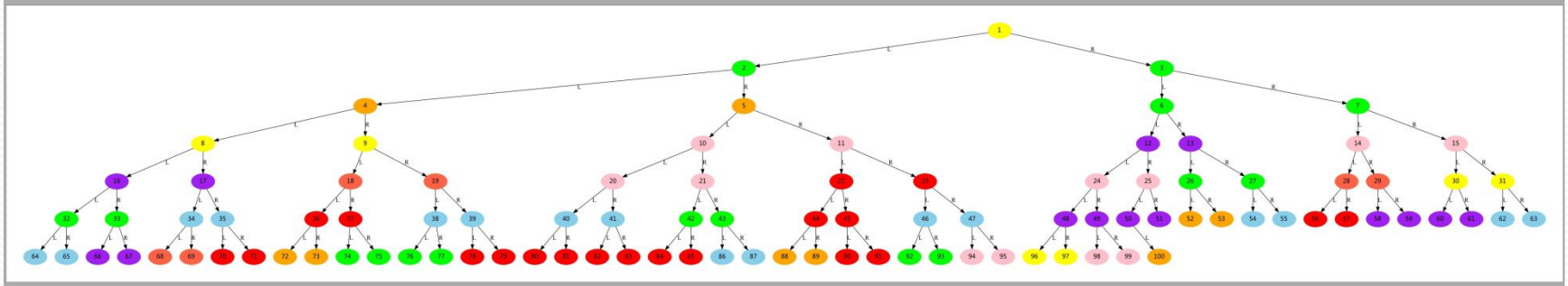
树状图

- 树状图是一种数据结构，它是由n（n>=1）个有限节点组成一个具有层次关系的集合。把它叫做“树”是因为它看起来像一棵倒挂的树，也就是说它是根朝上，而叶朝下的。它具有以下的特点：每个节点有零个或多个子节点；没有父节点的节点称为根节点；每一个非根节点有且只有一个父节点；除了根节点外，每个子节点可以分为多个不相交的子树；
- 叶子结点是离散数学中的概念。一棵树当中没有子结点（即度为o）的结点称为叶子结点，简称“叶子”。叶子是指出度为o的结点，又称为终端结点。

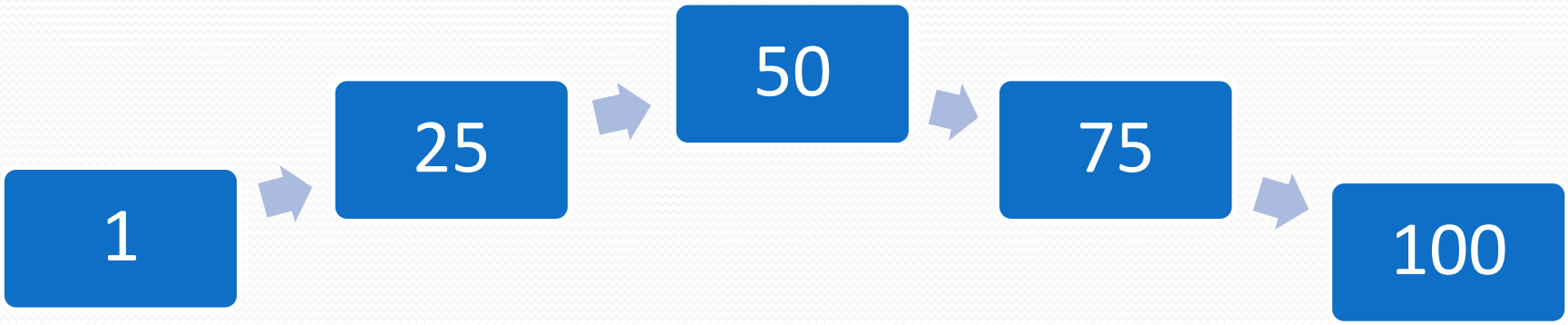
二叉树

- 二叉树的递归定义为：二叉树是一棵空树，或者是一棵由一个根节点和两棵互不相交的，分别称作根的左子树和右子树组成的非空树；左子树和右子树又同样都是二叉树

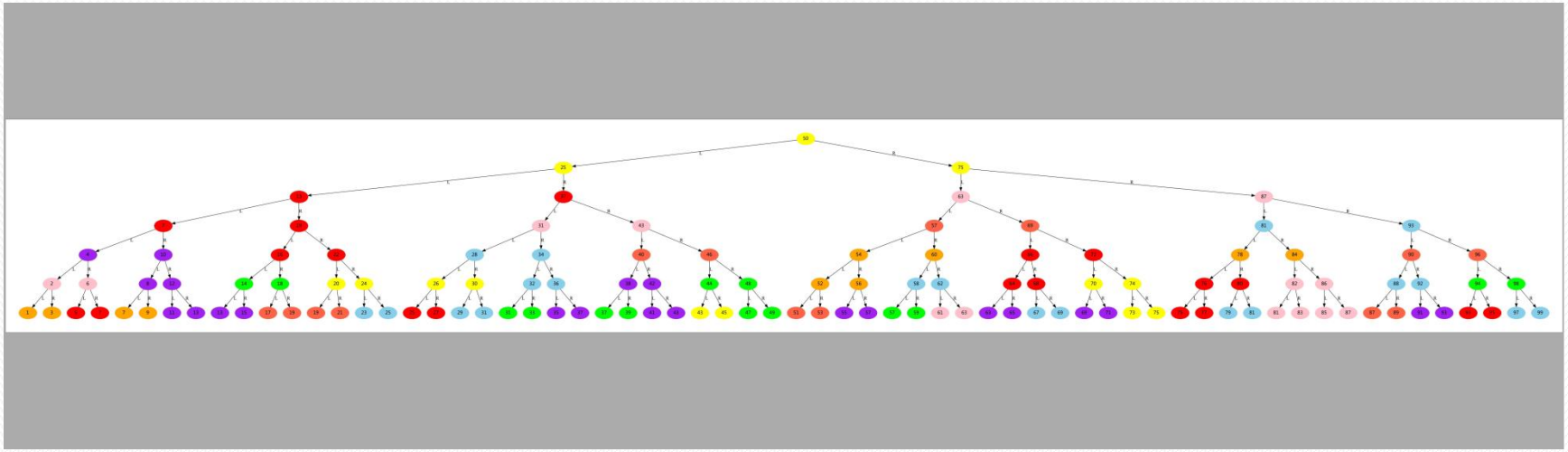
完全二叉树



二分法与二叉树



二分法与二叉树



二叉树与二进制

- 完全二叉树

完全二叉树

一棵深度为k的有n个结点的二叉树，对树中的结点按从上至下、从左到右的顺序进行编号，如果编号为i（1≤i≤n）的结点与满二叉树中编号为i的结点在二叉树中的位置相同，则这棵二叉树称为完全二叉树。

1、具有n个结点的完全二叉树的深度$\lfloor \log_2 k \rfloor + 1$（注：⌊⌋表示向下取整）

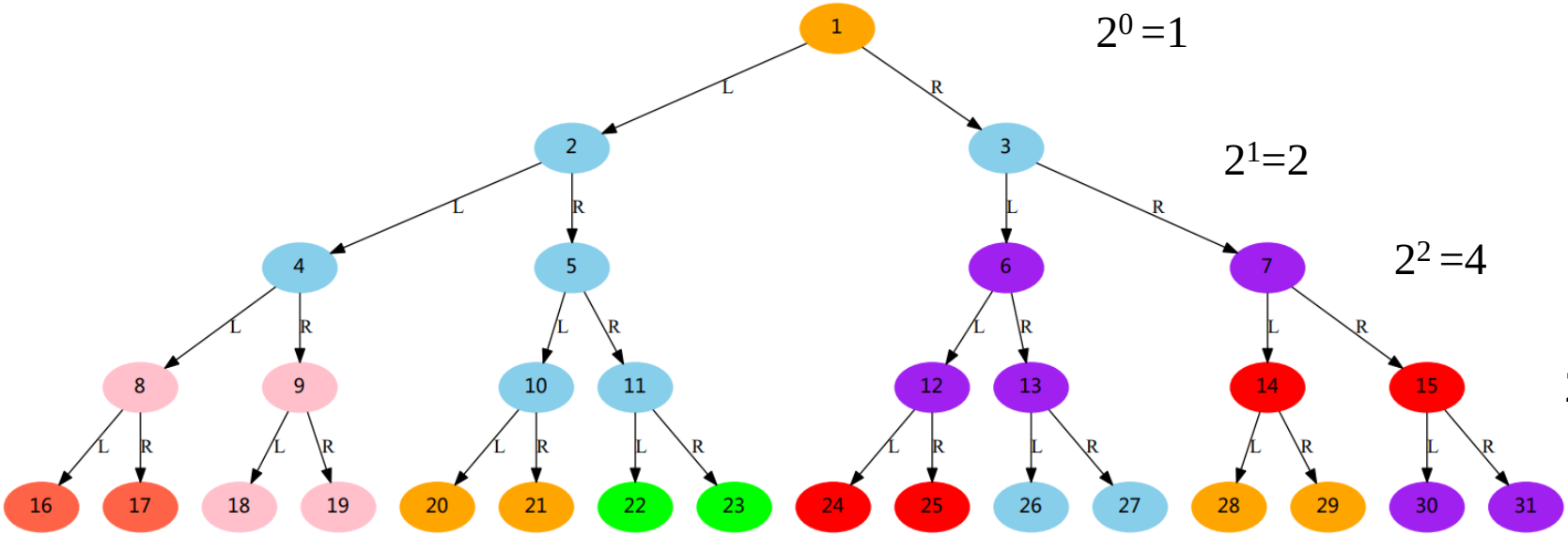
2、如果对一棵有n个结点的完全二叉树的结点按层序编号，则对任一结点i(1≤i≤n)有: [2]

如果i=1，则结点i是二叉树的根，无双亲;如果i>1，则其双亲parent(i)是结点[i/2]. [2]

如果2i>n，则结点i无左孩子，否则其左孩子lchild(i)是结点2i; [2]

如果2i+1>n，则结点i无右孩子，否则其右孩子rchild(i)是结点2i+1. [2]

每层几个球


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
L
R
$2^0=1$
$2^1=2$
$2^2=4$
$2^3=8$
$2^4=16$

$$2^4 = 2^{4-1} + 2^{4-2} + \cdots + 2^1 + 2^0 + 1$$

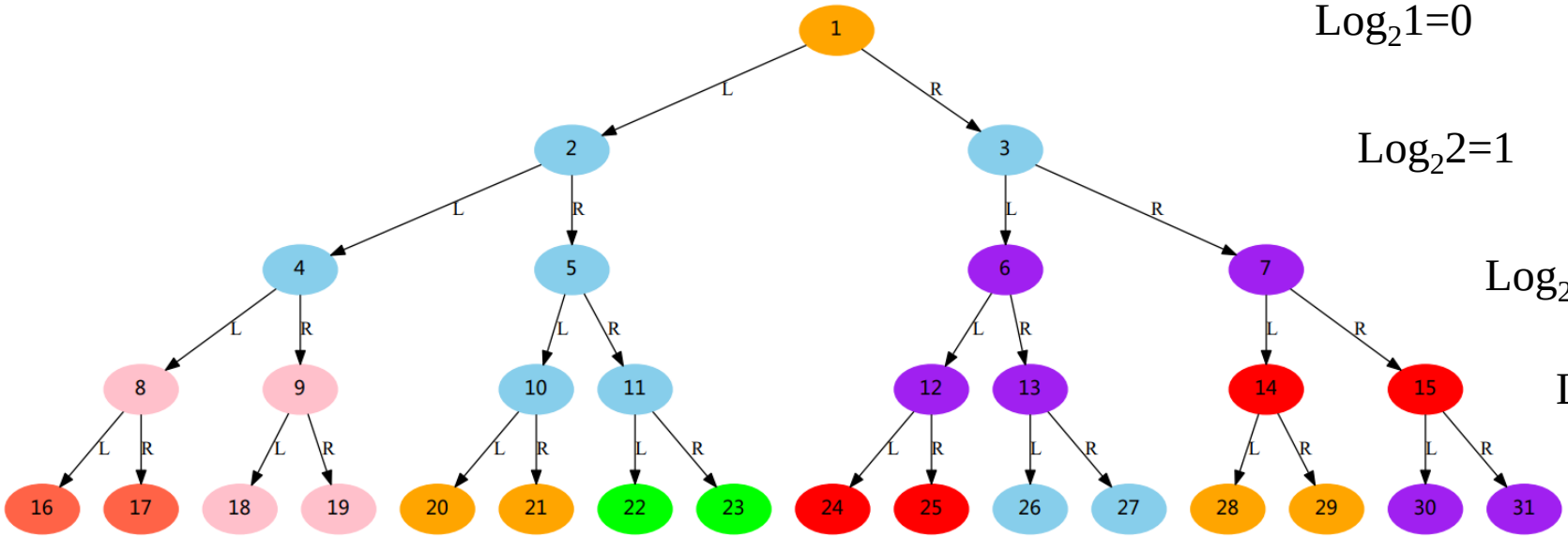
$$16=8+4+2+1+1$$

最下层的球的个数等于所有上面层的球的个数的总和+1

- 1,2,4,8,16,32
- 1+2+4+8+16+32=64-1

- 1+2+4+8+16=32-1
- 1+2+4+8=16-1

多少个球反求层数



$Log_2 1=0$

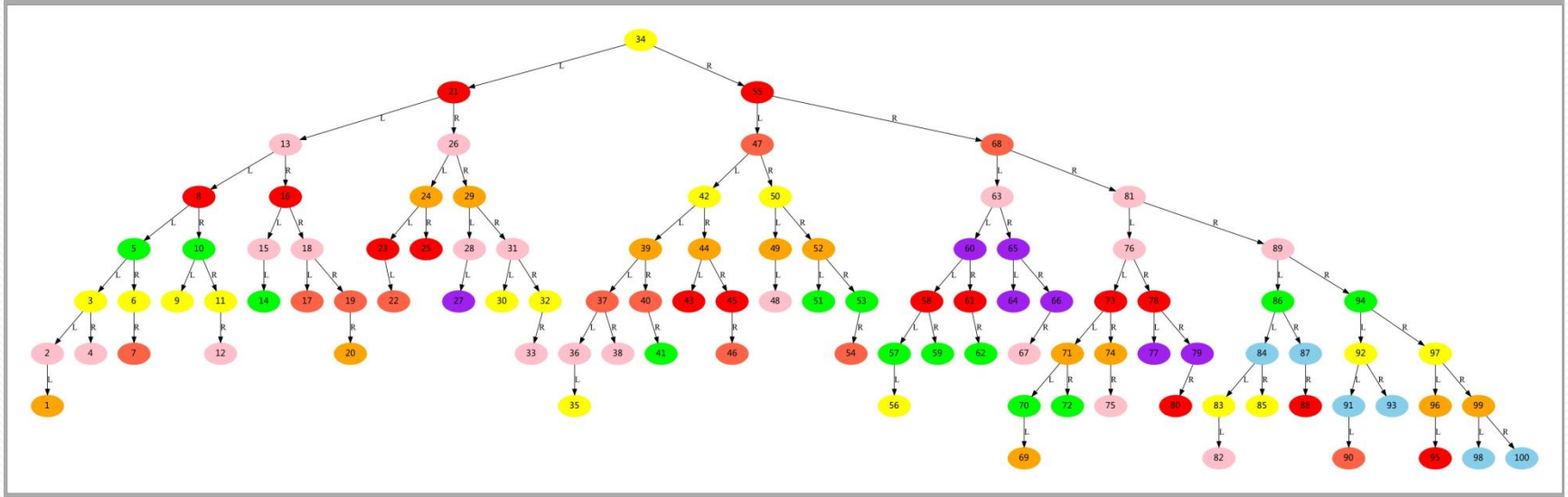
$Log_2 2=1$

$Log_2 4=2$

$Log_2 8=3$

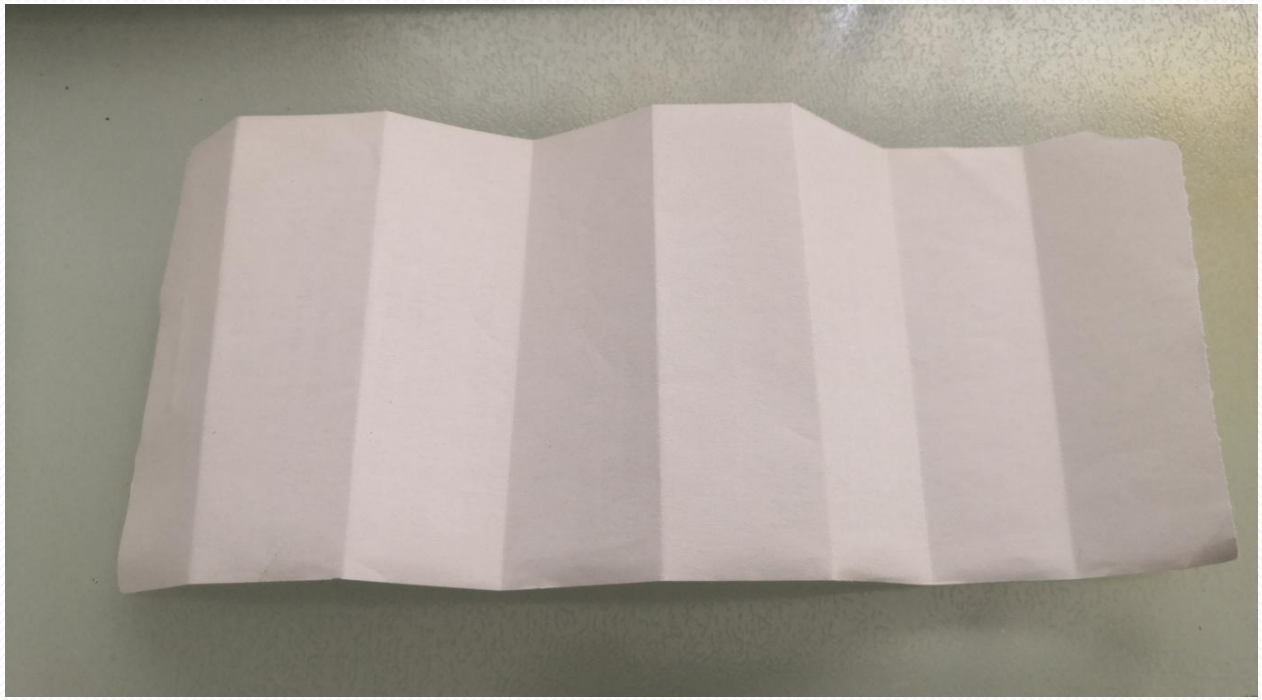
$Log_2 16=4$

平衡二叉树


34
21
55
13
26
47
68
8
16
24
29
42
50
63
81
5
10
15
18
23
25
28
31
39
44
49
52
60
65
76
89
3
6
9
11
14
17
19
22
27
30
32
37
40
43
45
48
51
53
58
61
64
66
73
75
86
94
2
4
7
12
20
33
36
38
41
46
54
57
59
62
67
71
74
77
79
84
87
92
97
1
35
56
70
72
75
80
83
85
88
91
93
96
99
69
82
90
95
98
100

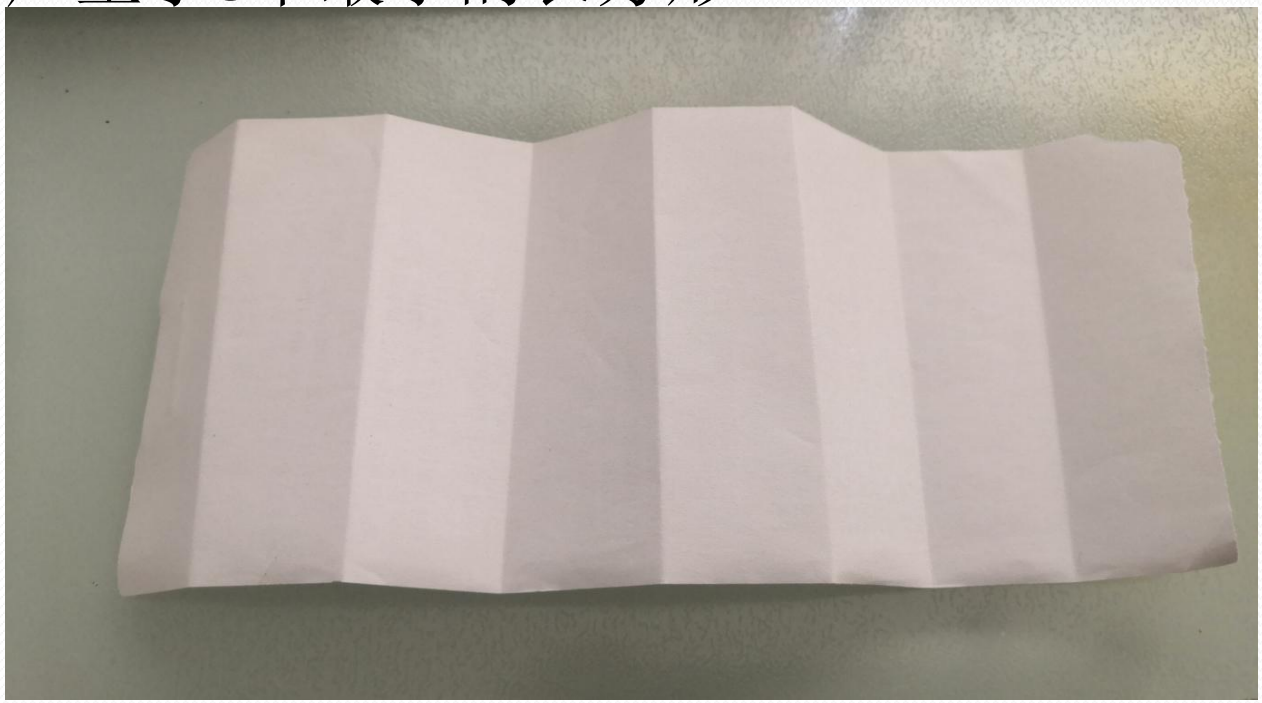
折纸与二进制

- 将一张长方形的纸对折，可以得到一条折痕，继续对折，对折时每次折痕与上次折痕保持平行。问对折N次后，可以得到几条折痕？



折纸与二进制

- 我对折了3次
- 产生了7条折痕
- 产生了8个最小的长方形



- 对折的次数相当于从0开始的层数，3次，相当于第四层
- 最小的长方形块，就是第四层的球的个数
- 折痕相当于上面三层球数的加和

二进制的多项式公式

二进制公式：

$$2^n = 2^{n-1} + 2^{n-2} + \cdots + 2^1 + 2^0 + 1$$

$$10^n = 9 \times 10^{n-1} + 9 \times 10^{n-2} + \cdots + 9 \times 10^1 + 9 \times 10^0 + 1$$

$$N^n = (N-1) \times N^{n-1} + (N-1) \times N^{n-2} + \cdots + (N-1) \times N^1 + (N-1) \times N^0 + 1$$

$$2^4 = 2^{4-1} + 2^{4-2} + \cdots + 2^1 + 2^0 + 1$$

$$16=8+4+2+1+1$$

二进制

- 0, 1, 10, 11, 100, 101, 110, 111
- 0, 1, 2, 3, 4, 5, 6, 7

- 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111
- 8, 9, 10, 11, 12, 13, 14, 15
- A, B, C, D, E, F

二进制

- 0,1　　一位数可以表达二种情况
- 00,01,10,11　　二位数可以表达四种情况
- 000,001,010,011,100,101,110,111
  三位数可以表达八种情况
- 0000,0001,0010,0011,0100,0101,0110,0111
  1000,1001,1010,1011,1100,1101,1110,1111
  四位数可以表达十六种情况
- 是不是很像二叉树

十进制

- 0,1,2,3,4,5,6,7,8,9　　一位数
- 10,11,12,13,14,15,16,17,18,19　　两位数
- ……
- 90,91,92,93,94,95,96,97,98,99　　两位数
- 100,101,102,103,104,105,106,107,108,109　　三位数

老师是怎么计算第几周课表的方法一

```python
from datetime import datetime
import time


def week_num(start_time, end_time):
    week_start = datetime.strptime(start_time, '%Y-%m-%d')
    week_end = datetime.strptime(end_time, '%Y-%m-%d')

    # 方法1:
    print(week_end.isocalendar()[1])
    print(week_start.isocalendar()[1])
    return week_end.isocalendar()[1] - week_start.isocalendar()[1]
    # 方法2:
    # return int(datetime.strftime(week_end, "%W")) - int(datetime.strftime(week_start, "%W"))


# print(week_num("2020-05-31", "2020-06-03"))
print(week_num("2020-07-11", time.strftime("%Y-%m-%d", time.localtime(time.time()))))
```

```
32

28

4
```

老师是怎么计算第几周课表的方法二

```python
from datetime import datetime, date
# 不同天的时间差
time_1 = '2020-07-11 15:00:00'
time_2 = '2020-08-06 16:00:00'

time_1_struct = datetime.strptime(time_1, "%Y-%m-%d %H:%M:%S")
time_2_struct = datetime.strptime(time_2, "%Y-%m-%d %H:%M:%S")

# 来获取时间差中的秒数。注意，seconds获得的秒只是时间差中的小时、分钟和秒部分，没有包含天数差，total_seconds包含天数差
# 所以total_seconds两种情况都是可以用的
total_seconds = (time_2_struct - time_1_struct).total_seconds()
print('不同天的秒数为: ')
print(int(total_seconds))

min_sub = total_seconds / 60
print('不同天的分钟数为: ')
print(int(min_sub))

min_sub = total_seconds / (24*60*60*7)
print('不同天的周数为: ')
print(int(min_sub)+1)
```

```
不同天的秒数为:
2250000
不同天的分钟数为:
37500
不同天的周数为:
4
```

这里是得数+1，因为7月11号星期六。7月13号星期一的时候就已经是第一周了。但是时间间隔小于1周