# 递归与递推

计算机科学与技术学院
初炳享

2015.1.26

# 听本节课需要注意的有：

- 一切行动听指挥！
- 具体有：

  让写程序写程序

  让推公式推公式

  注意题目英文名

  没听懂的快举手！

# 开始前我们先来复习一下函数的概念

- 函数不是函数！

# 开始前我们先来复习一下函数的概念

- 函数不是函数！
- 当然，

  第一个函数代表 C 语言函数，

  第二个函数代表数学函数。

# 开始前我们先来复习一下函数的概念

- 函数不是函数！
- 当然，

  第一个函数代表 C 语言函数，

  第二个函数代表数学函数。
- 但是，

  我们今天的话题却与两种函数都有关系，

  暂且先认为它们真的有关吧...

# 现在问题来了！

- 什么是 C 语言函数？

- A function can be thought of as a user-defined operation. In general, a function is represented by a name. The operands of a function, referred to as its parameters, are specified in a comma-separated parameter list enclosed in parentheses. The result of a function is referred to as its return value, and the type of the return value is called the function return type. A function that does not yield a value has a return type of void , meaning that it returns nothing. The actions a function performs are specified in the body of the function. The function body is enclosed in braces and is sometimes referred to as the function block. The function return type followed by the name of the function, the parameter list, and the function body compose the function definition.
- int abs( int iobj )

  {

  // return the absolute value of iobj

  return( iobj < 0 ? -iobj : iobj );

  }

- 其实上面的文字摘自 C++ Primer 3rd en
- 但是同样对 C 适用
- 说了半天 C 语言函数有什么用？

- 其实上面的文字摘自 C++ Primer 3rd en
- 但是同样对 C 适用
- 说了半天 C 语言函数有什么用？
- 一会就知道了.....

# 言归正传：递归

- 递
- 汉字首尾分解：辶弟

  汉字部件分解：弟辶

  笔顺：4351523454
- 汉字意为传送，传达或顺着此序。
- 这里我们取传送的意思

# 言归正传：递归

- 归
- 外文名 return; return to, revert to
- 笔顺 : 23511
- 意思很多
- 我们取还给之意

# 言归正传：递归

- 递归
- 即为传送归还

# 言归正传：递归

- 递归
- 即为传送归还
- 从汉语角度来讲
- 这语文明显是计算机老师教的嘛！

# 言归正传：递归

- 递归
- 即为传送归还
- 从汉语角度来讲
- 这语文明显是计算机老师教的嘛！
- 谁送给谁啊！
- 谁还给谁啊！

# 言归正传：递归

为了说明这个问题

- 我们给出了递归的定义式：

定义：

- 递归：请参阅递归的定义式；

# 言归正传：递归

没看明白？再看一遍！

定义：

- 递归：请参阅递归的定义式；

# 言归正传：递归

还没明白？

- 好吧.....那么我给大家讲个故事：

# 言归正传：递归

还没明白？

- 好吧……那么我给大家讲个故事：
- 从前有座山，山里有座庙，庙里有个老和尚，正在给小和尚讲故事呢！故事是什么呢？“从前有座山，山里有座庙，庙里有个老和尚，正在给小和尚讲故事呢！故事是什么呢？‘从前有座山，山里有座庙，庙里有个老和尚，正在给小和尚讲故事呢！故事是什么呢？……

# 言归正传：递归

还没明白？

- 好吧……那么我给大家再讲个故事：

# 言归正传：递归

还没明白？

- 好吧……那么我给大家再讲个故事：
- 一只狗来到厨房，偷走一小块面包。厨子举起勺子，把那只狗打死了。于是所有的狗都跑来了，给那只狗掘了一个坟墓，还在墓碑上刻了墓志铭，让未来的狗可以看到：“一只狗来到厨房，偷走一小块面包。厨子举起勺子，把那只狗打死了。于是所有的狗都跑来了，给那只狗掘了一个坟墓，还在墓碑上刻了墓志铭，让未来的狗可以看到：‘一只狗来到厨房，偷走一小块面包。厨子举起勺子，把那只狗打死了。于是所有的狗都跑来了，给那只狗掘了一个坟墓，还在墓碑上刻了墓志铭，让未来的狗可以看到……’”

# 言归正传：递归

OK ，大家一定被我弄糊涂了.....

# 言归正传：递归

OK ，大家一定被我弄糊涂了.....

- 那我们引用一下维基百科的定义吧！

# 言归正传：递归

OK ，大家一定被我弄糊涂了.....

- 那我们引用一下维基百科的定义吧！
- 递归（英语： Recursion ），在数学与计算机科学中，是指在函数的定义中使用函数自身的方法。

# 言归正传：递归

- 还有百度百科的定义：
- 程序调用自身的编程技巧称为递归（ recursion ）。递归做为一种算法在程序设计语言中广泛应用。是指一个过程或函数在其定义或说明中有直接或间接调用自身的一种方法，它通常把一个大型复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解，递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算，大大地减少了程序的代码量。

# 言归正传：递归

- 我们刚才的定义式：

  定义：

  - 递归：请参阅递归的定义式；

    大家明白是什么意思了么？（笑）

# 说完了递归说递推

- 递推大家应该不陌生
- 一句话可以概括：

  递推（ Recursive ）是一种用若干步可重复运算来描述复杂问题的方法。
- 例如数学中的数列递推公式；
- finish

# 由递归到递推

- 今天我们的题目是递归与递推
- 那么它们有什么关系呢？

# 由递归到递推

- 今天我们的题目是递归与递推
- 那么它们有什么关系呢？
- **递归是递推的程序实现方法；**

# 回到函数

- 刚才留了个问题： C 语言函数有什么用？

# 回到函数

- 刚才留了个问题： C 语言函数有什么用？
- 那么现在答案来了：

  实现递归；

现在，我们由实际问题入手，看看递归和递推是怎么应用的？

# 问题零：上楼梯问题

- 大家都上过楼梯吧？

# 问题零：上楼梯问题

- 大家都上过楼梯吧？
- 大家一次能迈几个台阶？

# 问题零：上楼梯问题

- 大家都上过楼梯吧？
- 大家一次能迈几个台阶？
- 2 个？ 3 个？ 10 个？

# 问题零：上楼梯问题

- 现在有一 loli 大爷，一次只能迈一个或者两个台阶，现总共有 n 级台阶， loli 大爷现在在第 0 级台阶上，这个 loli 大爷想知道他（她）总共有多少种方法爬上楼？
- 思考一下！
- 时间： 1 分钟！

# 问题零：上楼梯问题

- 答案：
- 先看 n=1 的情况：

  答案当然是 1 ；
- 当 n=2 时：

  可以选择从第一级爬一层，也可以选择从第 0 级爬 2 层，则答案就是到达第一层的方案数加上到达第二层的方案数，为 2 ；
- so, when n is equal to 3/4/5/……/n……（先不用告诉我答案 ~ ~ ）

# 问题一：汉诺塔问题

The Tower of Hanoi

# 问题一：汉诺塔问题

- 没玩过的举手！

# 问题一：汉诺塔问题

- 没玩过的举手！
- 你们没有童年！

# 问题一：汉诺塔问题

- 没玩过的举手！
- 你们没有童年！
- just a joking ~ ~

# 问题一：汉诺塔问题

- 问题描述：
- 给定一个由 8 个圆盘组成的塔，这些圆盘按照大小递减的方式套在三根柱子中的一根上，目的是要将整个塔移动到另一根柱子上，每次只能移动一个圆盘，且较大的圆盘在移动的过程中不能放在较小的圆盘上面。

# 问题一：汉诺塔问题

**What is the best solution?**

# 问题一：汉诺塔问题

- 为了解决这个问题，我们将它稍微推广一下：
- 考虑：

  如果有 n 个圆盘的情形；

# 问题一：汉诺塔问题

- 我们令 Tn= 将 n 个圆盘移动到另一个柱子上所需要的最小次数；
- 则有：
- T1=1 ；
- T2=3 ；
- 显而易见： T0=0 ；

# 问题一：汉诺塔问题

- 现在我们考虑这样一个情形：
- 如何移动一个更大的塔？

# 问题一：汉诺塔问题

- 首先我们先看看三个圆盘是怎么移动的？
- A→C,
- A→B,
- C→B,
- A→C,
- B→A,
- B→C,
- A→C
- 一共需要移动 7 次；

# 问题一：汉诺塔问题

- 我们的思路是先将上面两个圆盘移动到第二个柱子上，然后移动第三个圆盘，接着再把其他两个圆盘移动到第三个柱子上。
- A→C, A→B, C→B,
- A→C,
- B→A, B→C, A→C

# 问题一：汉诺塔问题

- 这就为移动 n 个圆盘提供了一条线索：
- 先将 n-1 个圆盘移动到一个不同的柱子上；

  然后移动最大的圆盘；

  再将 n-1 个圆盘移动到最大的圆盘上面；
- 所需步骤数：

  $T_n = T_{n-1} + 1 + T_{n-1} = 2 * T_{n-1} + 1$

# 问题一：汉诺塔问题

- Tn = 2 * Tn-1 +1  (n > 0)

  我们将形如上式的式子称为递归式（递推式）；
- 它为我们提供了一个可以计算任意一个 Tn 的途径；

- 回到刚才的上楼梯问题：
- 当有 n 级台阶的时候，方法数为：

- 回到刚才的上楼梯问题：
- 当有 n 级台阶的时候，方法数为：

  Tn = Tn-1 + Tn-2;

  T1 = 1;

  T0 = 1;

- 如果 loli 大爷成年了，腿变长了，可以一次性迈 1 级 2 级 3 级台阶了呢？上 n 级台阶的方案数为多少？

- 如果 loli 大爷成年了，腿变长了，可以一次性迈 1 级 2 级 3 级台阶了呢？上 n 级台阶的方案数为多少？
- 当然是：
- Tn = Tn-1 + Tn-2 +Tn-3;
- T2 = 2;
- T1 = 1;
- T0 = 1;

# 程序实现

- 下面就到了程序实现的环节了！
- 请将以下程序的空白部分填写完整！

# 程序实现

- 你可以在 https://gitcafe.com/8cbx/Algorithm-training 找到相应示例程序，点击网页右上方向下的箭头下载
- 文件名说明：

  0-1.cpp 表示第 0 道问题的第一个程序；
- 需要你自己填写的部分为 /*  */ 之间的代码

# 程序实现

- 你可以在 https://gitcafe.com/8cbx/Algorithm-training 找到相应示例程序，点击网页右上方向下的箭头下载
- 第一个程序请使用循环与数组完成；
- 第二个程序请使用函数递归完成；
- 一共需要完成两道题目总计四个程序；
- 时限： 15 分钟

# 回到汉诺塔问题

- 大家还记得这个式子吗？
- $T_n = 2 * T_{n-1} + 1 \quad (n > 0)$
- 汉诺塔的递推式！

# 回到汉诺塔问题

- 大家还记得这个式子吗？
- $T_n = 2 * T_{n-1} + 1 \quad (n > 0)$
- 汉诺塔的递推式！
- 看没看出有点问题？？

# 回到汉诺塔问题

- 我们根本不需要用数组或者函数递归来写这个程序！！
- Tn = 2^n - 1 ！！

# 程序实现又来了.....

- 用公式实现汉诺塔问题！
- 时限： 2 分钟！
- 增加要求：多组输入数据！

# 问题二：切蛋糕问题

- 这个问题有多种描述：切饼切糕切 pizza ，我们这里就取切蛋糕这个描述吧.....
- **英文名称： Lines in the Plane**

# 问题二：切蛋糕问题

- 话说大爷有一天买了一个蛋糕，在回家的路上被一群小 loli 给拦住了，由于大爷禁不住 loli 的诱惑答应了 loli 们一个要求，她们要求大爷在蛋糕上切 n 刀 ( 只能直上直下的切，不能重复在一条线上切，每刀的两端必须交到圆周的不同点上 ) ，并将切得的蛋糕分给她们每人一块。由于大爷心地善良又吝啬无比，于是他决定用这 n 刀尽可能的将蛋糕分成更多的份数 ( 这样他送出去蛋糕的大小会少很多而不显得吝啬 ) 。你能帮助大爷计算他最多能把这块蛋糕分成多少份吗？
- 思考时间： 2 分钟！

# 问题二：切蛋糕问题

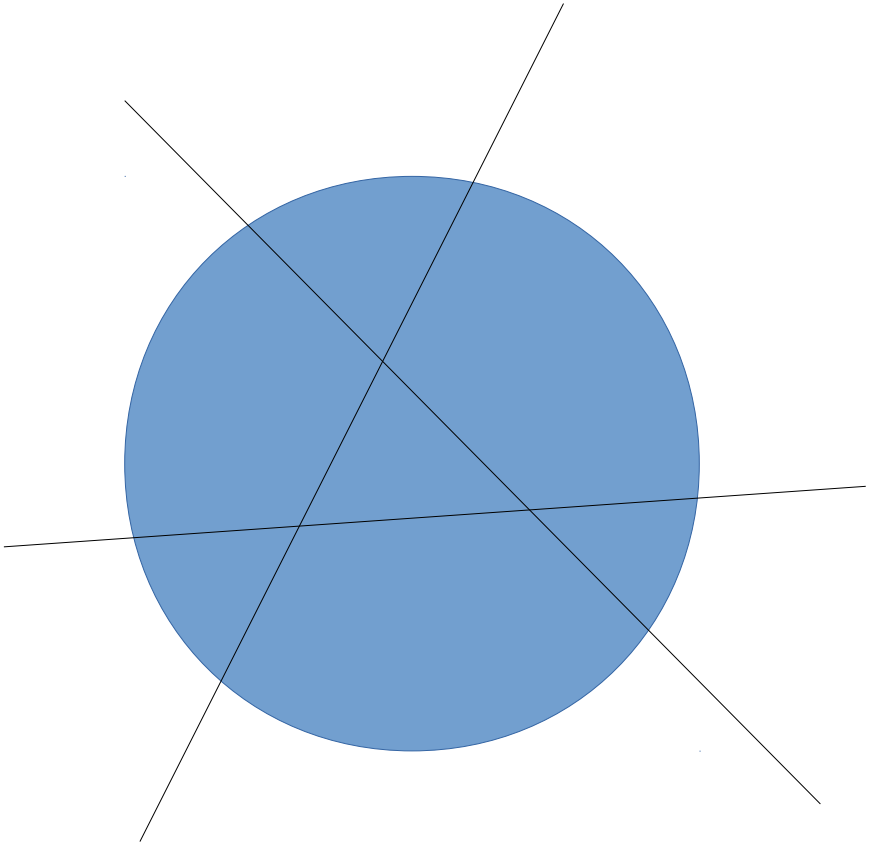
我们来分析一下这个问题：

- 当我们只需要切一刀时：

  总共能切成 2 块；
- 当我们切两刀的时候：

  总共能切 4 块；
- 切三刀的时候：

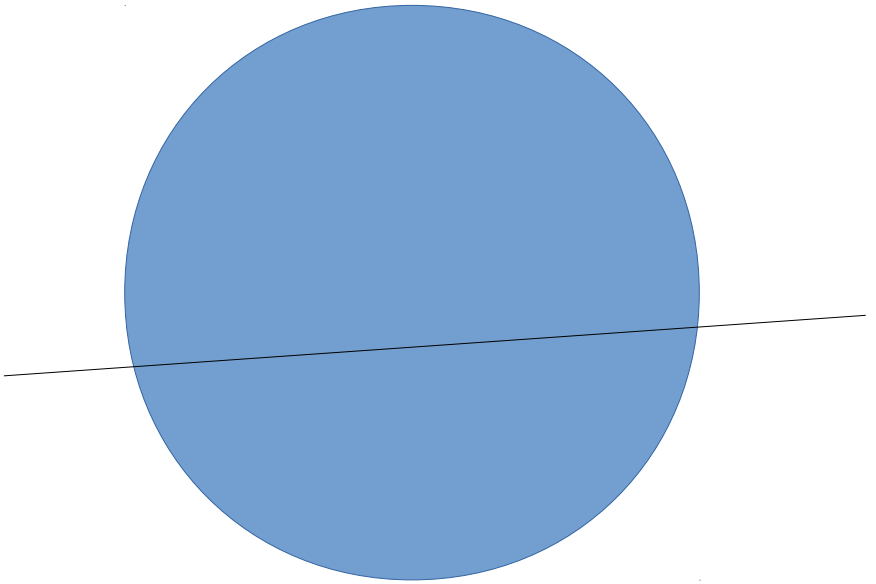
  总共能切 7 块；
- 以上均为常识！



# 问题二：切蛋糕问题

我们从 n=1 开始找规律：

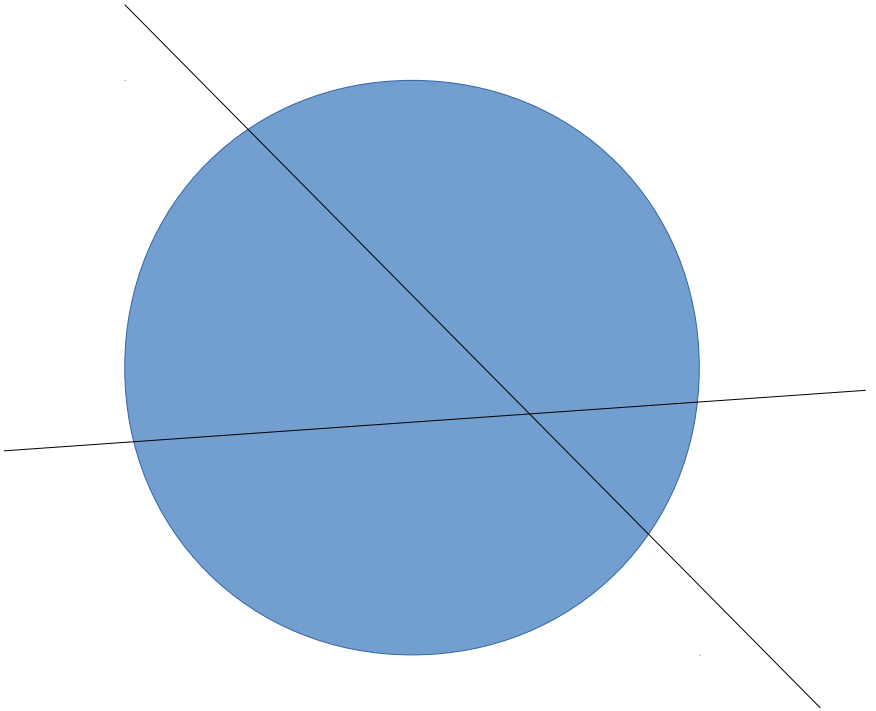
- 仅仅切一刀时，毫无疑问，可以分成两块



# 问题二：切蛋糕问题

我们从 n=1 开始找规律：

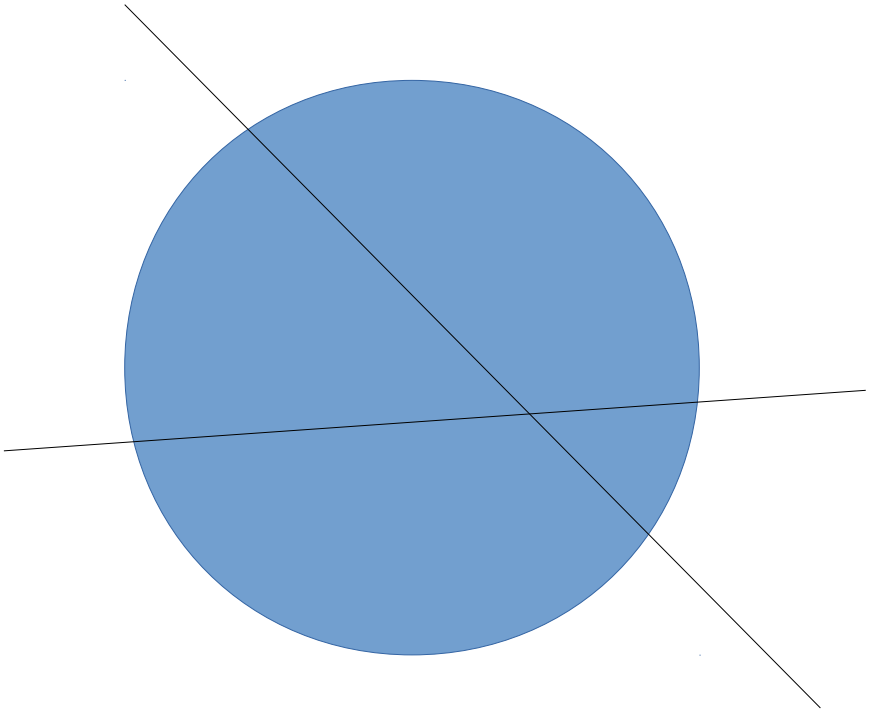
- 仅仅切一刀时，毫无疑问，可以分成两块
- 当增加一刀时，如何才能获得最优解？



与第一条直线相交！！

# 问题二：切蛋糕问题

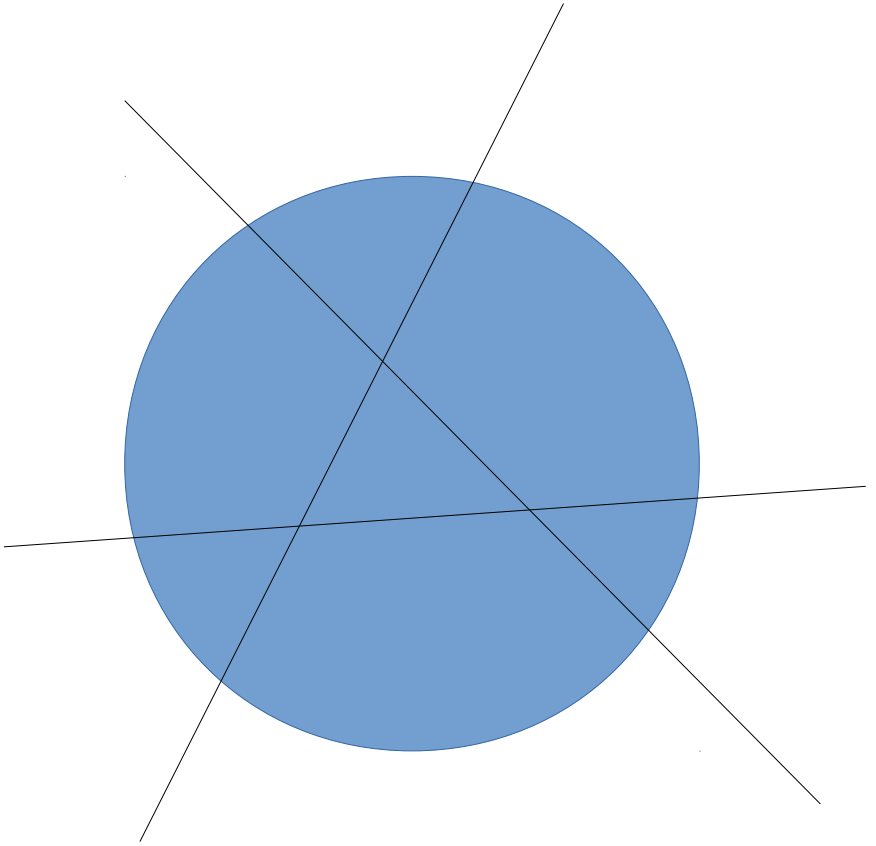
与第一条直线相交后，可以将第一条直线所分得两块均划分为新的两块！



# 问题二：切蛋糕问题

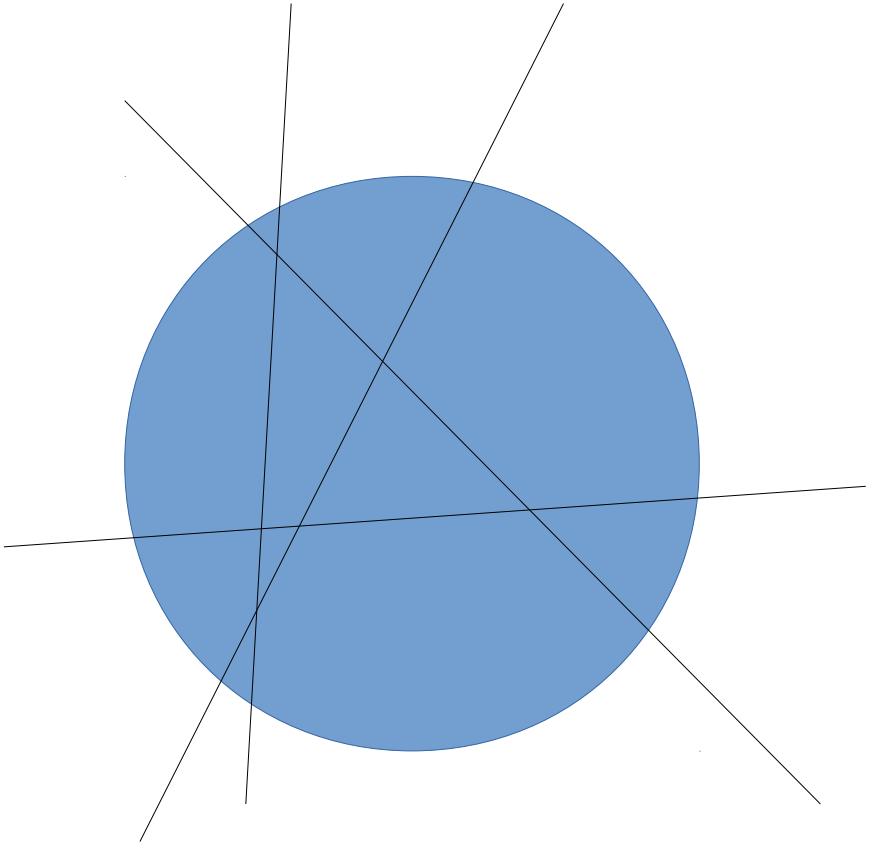
当 n=3 时，我们发现：

- 如果第三条直线与之前的两条直线相交，可以获得最优解。
- 此时，它将之前两条直线所分得四块中的三块各一分为二



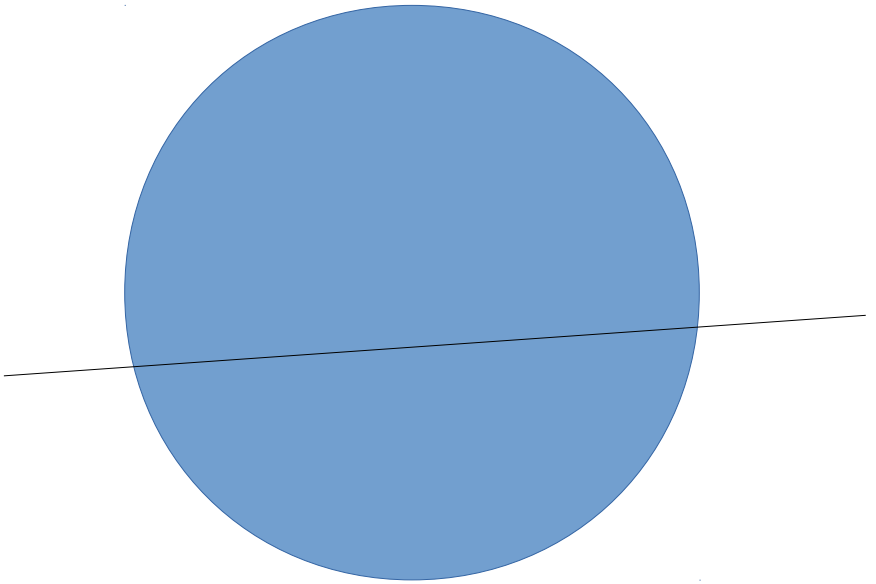
# 问题二：切蛋糕问题

- 通过 n=1 ， 2 ， 3 我们可以推测：当当前直线与之前直线均相交时，可以得到最优解。
- 我们验证一下 n=4 的情况是否正确
- 正确！
- 请思考 5 分钟找出递推公式！！



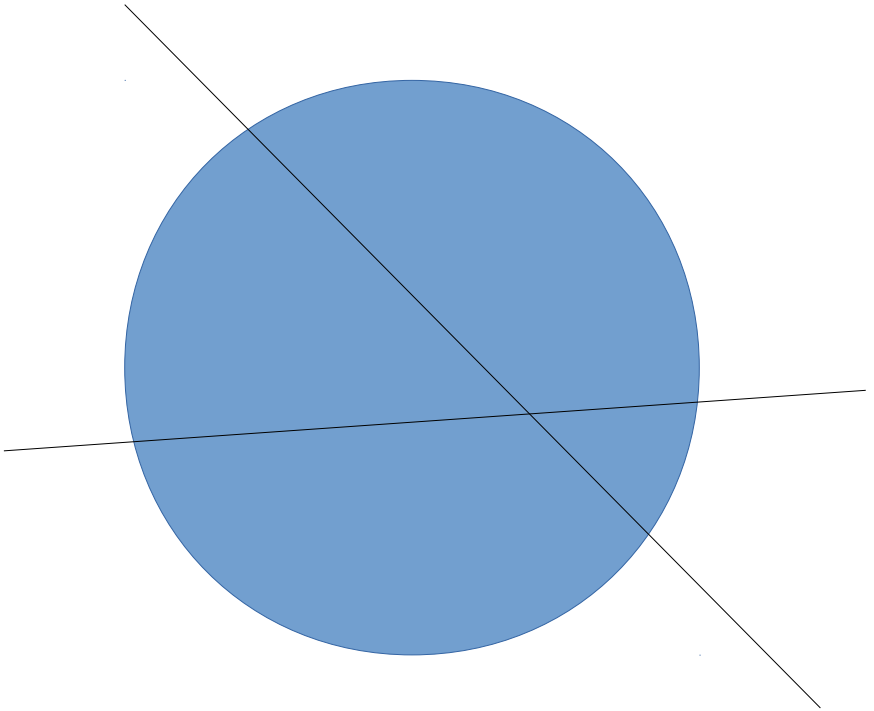
# 问题二：切蛋糕问题

- n=1 时，我们添加一条直线将平面分成 2 份；



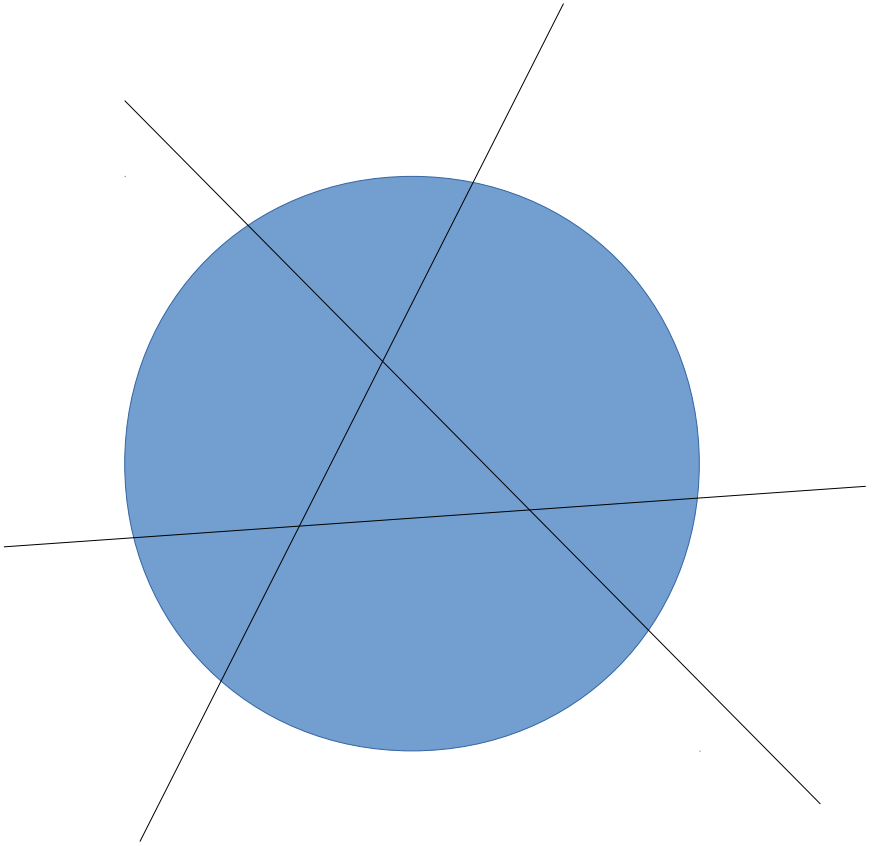
# 问题二：切蛋糕问题

- n=1 时，我们添加一条直线将平面分成 2 份；
- n=2 时，我们添加一条直线，将第一条直线两侧的蛋糕分成两部分，第一条直线两侧一共有两块蛋糕，所以总块数增加了 2



# 问题二：切蛋糕问题

- n=3 时，我们添加一条直线，它会与前两条直线相交。前两条直线两侧（ 180° 范围内）最多可以有三块蛋糕，所以把它们一分为二，总块数加 3 ；
- 想想为什么是 180° 范围内？

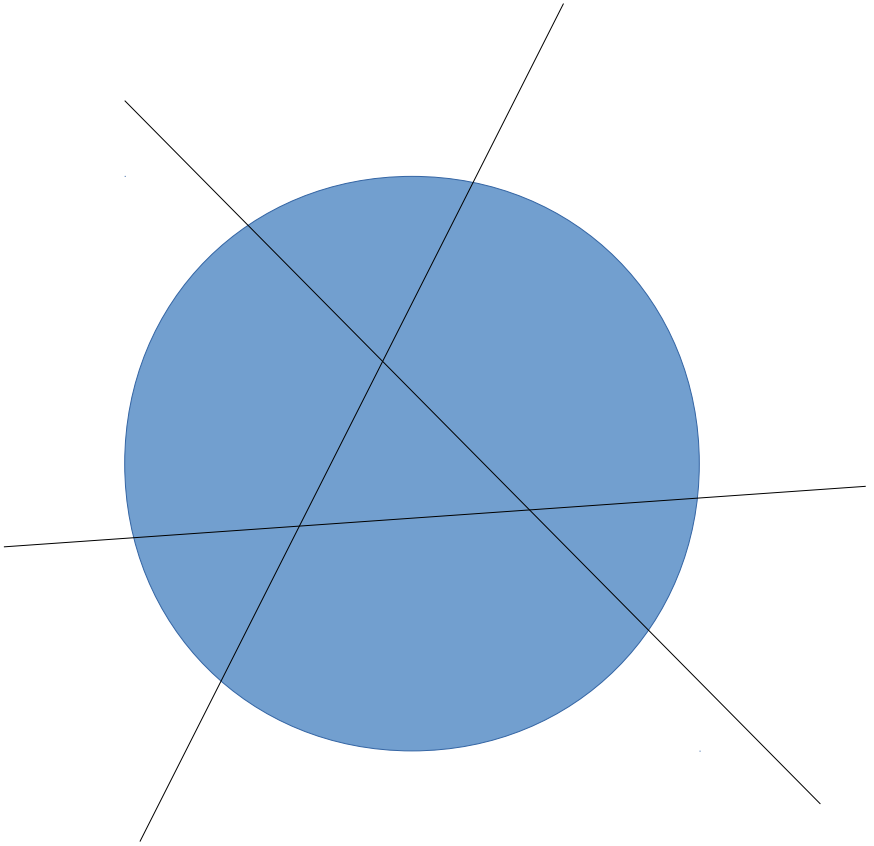


# 问题二：切蛋糕问题

- 所以递推公式为：
- Tn = Tn-1 + n ；

  T1 = 2 ；
- 公式大神可以推出：

  Tn = n(n + 1)/2 + 1 ；
- 任何一个都 OK ，只不过第二个更优 ~ ~



# 最后一个问题：约瑟夫问题

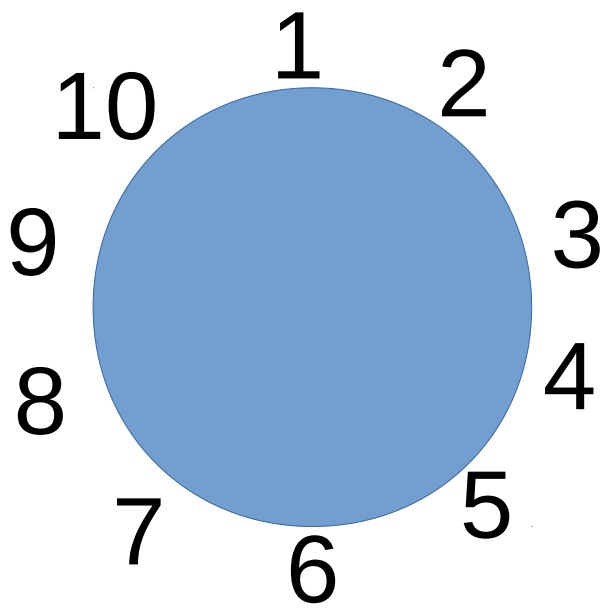
- 通常大家都叫它约瑟夫环；
- 他的确是个环；
- 下面看一下题意（ **The Josephus Problem** ）：

# 最后一个问题：约瑟夫问题

- 题意：

  n 个人围成一个圈，分别编号为 1 ~ n ，n 与 1 相邻。我们从 1 号开始，每隔 1 个人枪毙 1 个人，直到最后只有一人存活，求幸存者的号码。

- 比如此时有 10 个人：
- 那么枪毙的顺序是：
- 2,4,6,8,10,3,7,1,9
- 最终 5 幸存
- 思考 5 分钟！！



# 最后一个问题：约瑟夫问题

- 首先，可能会有人猜想 Tn = n / 2 ，但是，通过几个小范围的尝试大家可以很快发现这个是不对的！
- 现在来让我们用暴力算法来分析这个问题：
- 我们需要先写一个暴力的程序来计算 n 值很小时候的答案；
- 这个我就不代劳了～～给大家 15 分钟写出来～～

# 最后一个问题：约瑟夫问题

- 有了这个程序，我们就可以看到，当 n 取以下值时，有这么一张表：

| n | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Tn | 1 | 1 | 3 | 1 | 3 | 5 | 7 | 1 |
| n | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Tn | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 1 |

# 最后一个问题：约瑟夫问题

- 规律不难发现，但是递推公式不太好写.....：
- T(2^m + l) = 2T(2^m-1 + l / 2) – 1

  = 2(2l / 2 + 1) - 1

  = 2l + 1
- 下面的问题就是计算距离 n 最近的 m 了 ~ ~
- 我们还是可以采用打表的办法 ~ ~
- 这个就不在这里说了 ~ ~

# 结束前总结一下

- 递推与递归实际上就是需要找到找到一个递推公式，然后利用函数递归或者循环或者公式直接计算的方法来求得最终答案。
- 通常情况下，我们需要花费一定时间在递推公式的求解上，或者动用暴力的手段来获取前几个可行解来手工寻找规律。
- 一般函数递归的效率低于循环打表求解低于公式求解。

# Finish ~ ~

- 下午训练赛说明：

  有 5 道递归问题需要大家求解，难度不等，希望大家能 AC 至少 3 道题；

  另外 5 道题是之前出过的或者比较考验大家代码功力的问题，如果大家递推公式实在推不出来可以考虑这 5 道题；