

递归， 字典

（下载幻灯片和 .py 文件并跟随！）

6.0001 讲座 6

测验准备

§ a paper and online component

§ 打开书/笔记

§ 不开放互联网, 不开放电脑

§ start print out whatever you may want to bring

上次

§元组不可变

§列表-可变

§别名、克隆

§可变性副作用

今天

§ 递归 除法/减法

§ dicSonaries-anothermutableobjecttype

递归

递归是自我重复的过程
类似的。

什么是递归？

§ 算法: 远离设计soluSonstopproblemsby分而治之或减少和征服

- 将问题简化为相同的简化版本问题

§ SemanScally: a programming technique where a func0on calls itself

- 在编程中, goalisto没有无限递归 ◦ 必须有 1个或多个易于解决的基本案例 ◦ 必须解决相同的问题, 并且目标的其他输入

简化大问题输入

到目前为止的迭代算法


§循环构造 (while and for loops) 导致迭代算法

§可以捕获计算状态变量的 Soninaset
that update one each iteration through loop

乘法 迭代解决方案

§ “mulSplya”等价于“addtoitselfbSmes” §捕获状态。一个

iteration数字 (i)starts at biβi-1 and stop when 0 0a 1a 2a ... + 一个
computation (结果)结果β结果+一个



```
def mult_iter(a, b):  
    结果 = 0  
    当b > 0 时:  
        结果 += 一个  
        b -= 1  
    返回结果
```

iteration
current value of computation,
a running sum
current value of iteration variable

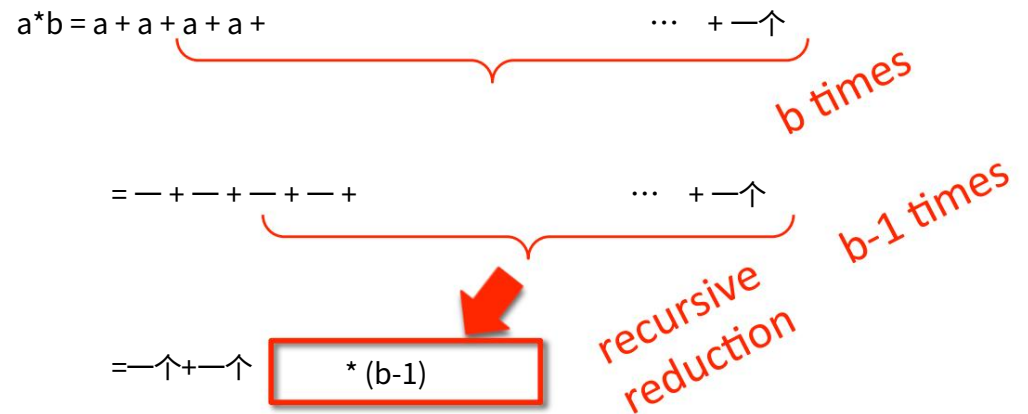
乘法

递归解决方案

§ 递归步骤 ·
think how to reduce
问题更简单/
相同问题的小版本

§ 基本情况

· 保持减少问题, 达
到一个可以直接解决的
简单情况 · 当 $b=1, a*b=a$



def mult(a, b):

如果 $b == 1$:

返回一个

别的:

返回 $a + \text{mult}(a, b-1)$

阶乘

嗯! $= n * (n-1) * (n-2) * (n-3) * \dots * 1$

§ whatndowknowthefactorial ?

$n=1$

如果 $n == 1$:

返回 1

base case

§如何减少问题?重写

intermsofsomethingsimplertoreachbasecase

$n * (n-1)!$ à其他:

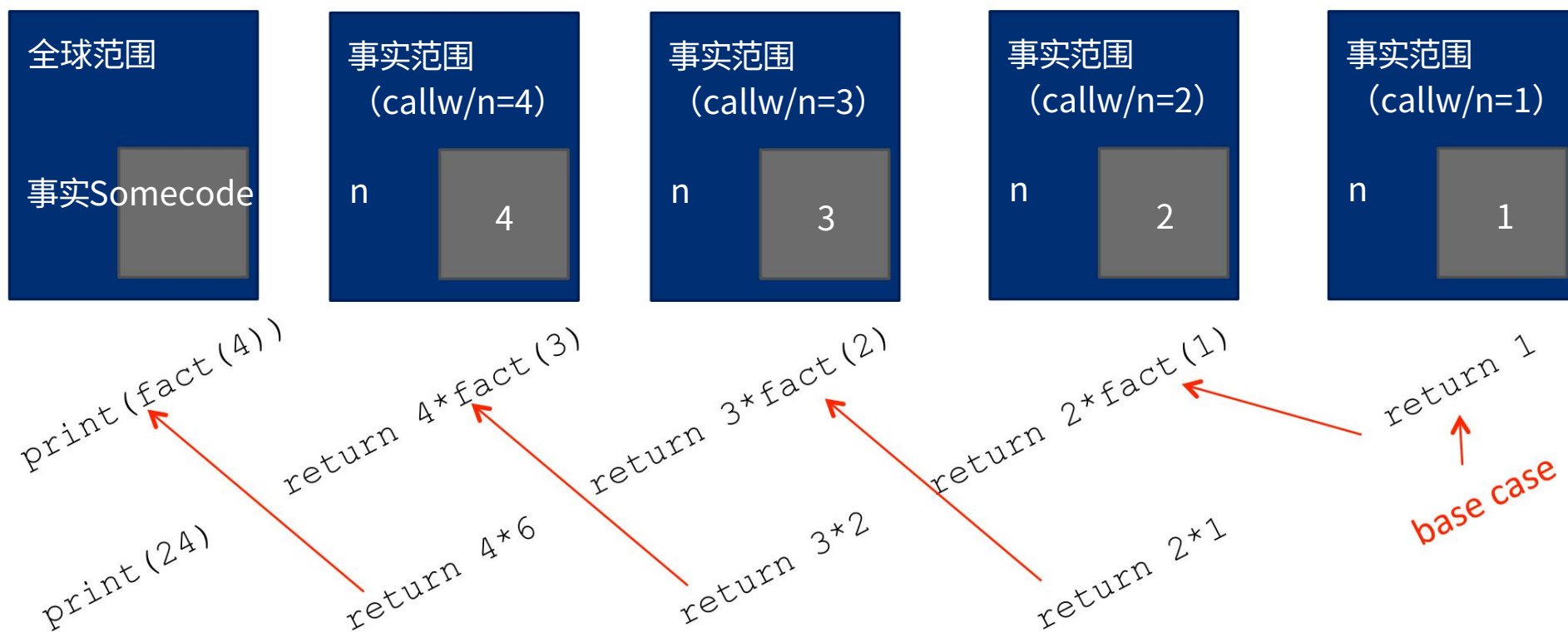
返回 $n * \text{阶乘}(n-1)$

recursive step

递归的 功能 范围 例子

```
事实 (n) :
    如果 n == 1:
        返回 1
    别的:
        返回 n * 事实 (n-1)
```

打印 (事实 (4))



一些观察

§ 每个递归调用一个funcSon创建自己的范围/环境

§ 范围内的变量绑定不会被递归调用改变

§ flow of control passes back to previous scope once funcSon call returns value

using the same variable names but they are different objects in separate scopes

迭代与递归

def 阶乘_iter(n):

 产品 = 1

 对于范围内的i (1,n+1):

 产品 *= i

 回报产品

默认阶乘 (n) :

如果n == 1:

 返回1

别的:

 返回n*阶乘 (n-1)

§ 递归可能更简单,更直观 § 递归可能对程序员
POV 有效 § 递归对计算机 POV 可能无效

归纳推理

§ 如何知道我们的递归代码将起作用？

§ `mult_iter` 终止, 因为
 1. `b` 最终会变为 0, 并减少
 2. 每次循环, `b` 必须最终变得少于 1

```
def mult_iter(a, b):
    结果 = 0

    当 b > 0 时:
        结果 += 一个
        b -= 1

    返回 结果
```

§

`mult` 当 `b=1` 时没有递归调用并停止

```
def mult(a, b):
```

§

```
    如果 b == 1:
```

```
        返回 一个
```

`mult` 当 `b>1` 时做一个递归调用, 使用 `b` 的更小版本; 最终必须达到调用

```
    如果
```

```
        返回 a + mult(a, b-1)
```

数学归纳法

§ To prove a statement indexed on integers is true for all values of n :

- 证明最小值为真时 ($n=0$ or $n=1$)
- 然后证明如果 n 的任意值为真, 可以证明它必须为 $n+1$ 为真

归纳示例

$$\S \ 0+1+2+3+\cdots+n=(n(n+1))/2$$

§证明:

- If $n=0$, then LHS is 0 and RHS is $0*1/2=0$, so true ◦ 假设
- some k 为真, 则需要证明 $k+(k+1)=((k+1)(k+2))/2$
- $$0+1+2+\cdots$$
 - LHS is $k(k+1)/2+(k+1)$ 由
 - assump S on that property holds for problem of size k ◦ 这变
 - 成, 通过代数, $((k+1)(k+2))/2$ ◦ 因此表达式对 all $n \geq 0$ 成立

与代码相关？

§相同的逻辑适用

```
def mult(a, b):
```

```
    如果 b == 1:
```

```
        返回一个
```

```
    别的:
```

```
        返回 a + mult(a, b-1)
```

§基本情况,我们可以证明 mult 必须返回正确答案

§对于递归情况,我们可以假设 mult 正确返回小于 b 的问题的答案,然后通过 headdiSonstep,它必须为大小 b 的问题返回正确的答案

§因此通过inducSon,代码正确返回答案

河内塔

§故事： ◦

3tallspikes ◦

Stackof64different sizeddiscs-startononespike ◦

Needtomovestacktosecondspike（此时宇宙结束）

◦只能移动一个磁盘,较大的磁盘永远无法覆盖小磁盘

河内塔

§看到一组不同大小的堆栈示例,您将如何编写一个程序来打印出正确的移动集?

§**递归思考!** ◦求解
更小的问题 ◦求解基本问题
题 ◦ 求解更小的问题

```
def 打印范围(开始, 结束):  
    + str(fr) +  
    至 + str(到))
```

多重递归

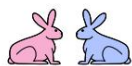
基本案例

§斐波那契数列

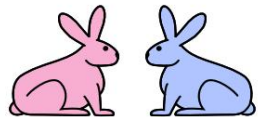
- Leonardo of Pisa (又名斐波那契)模拟了以下挑战。新生的一对兔子 (一个雌性,一个雄性)被放置在一个月内。兔子有一个月的妊娠期。假设兔子从来没有死过,雌性从第二个月开始每个月都会产生一对新的兔子 (一个雄性,一个雌性)

上。

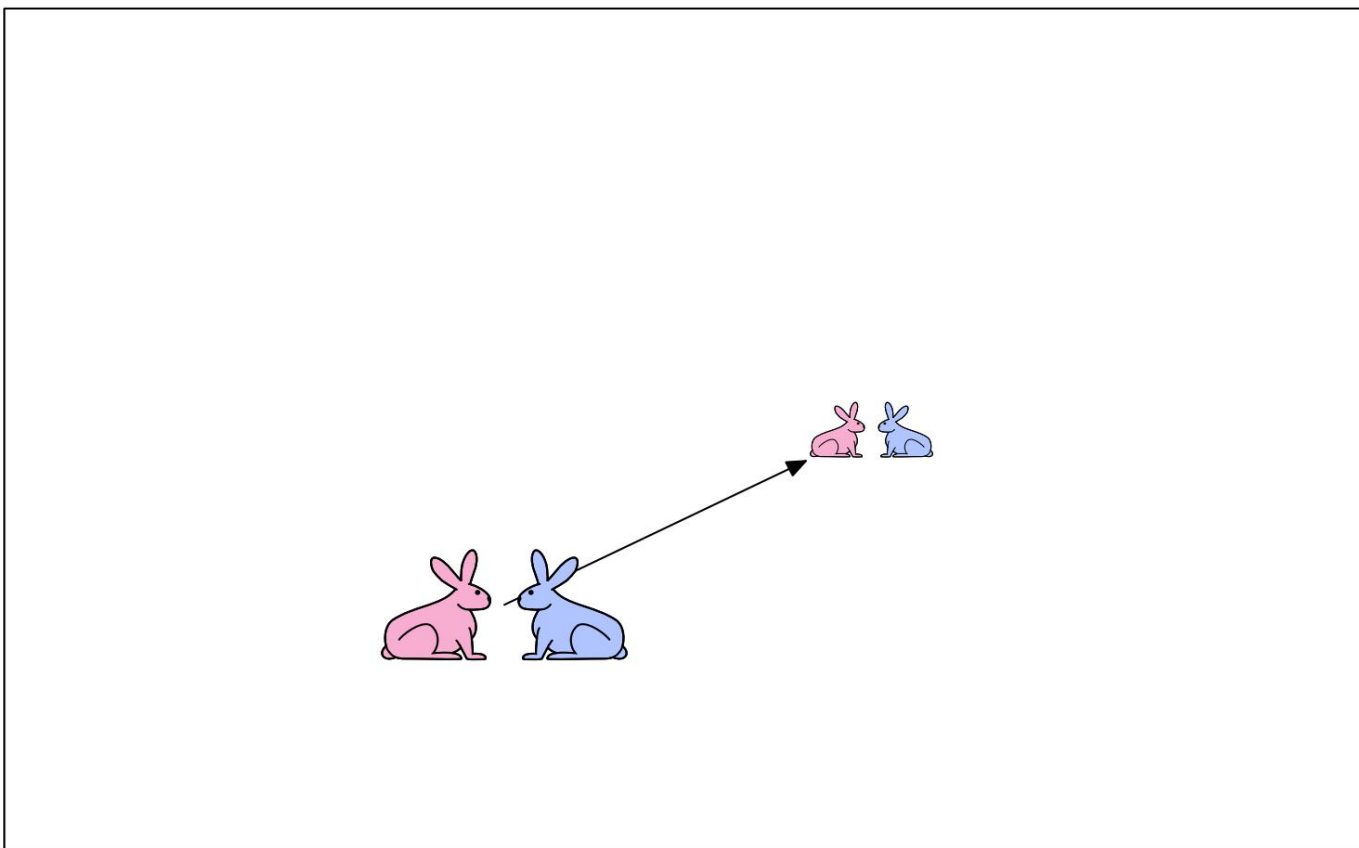
- 年底有多少只雌性兔子在吃?



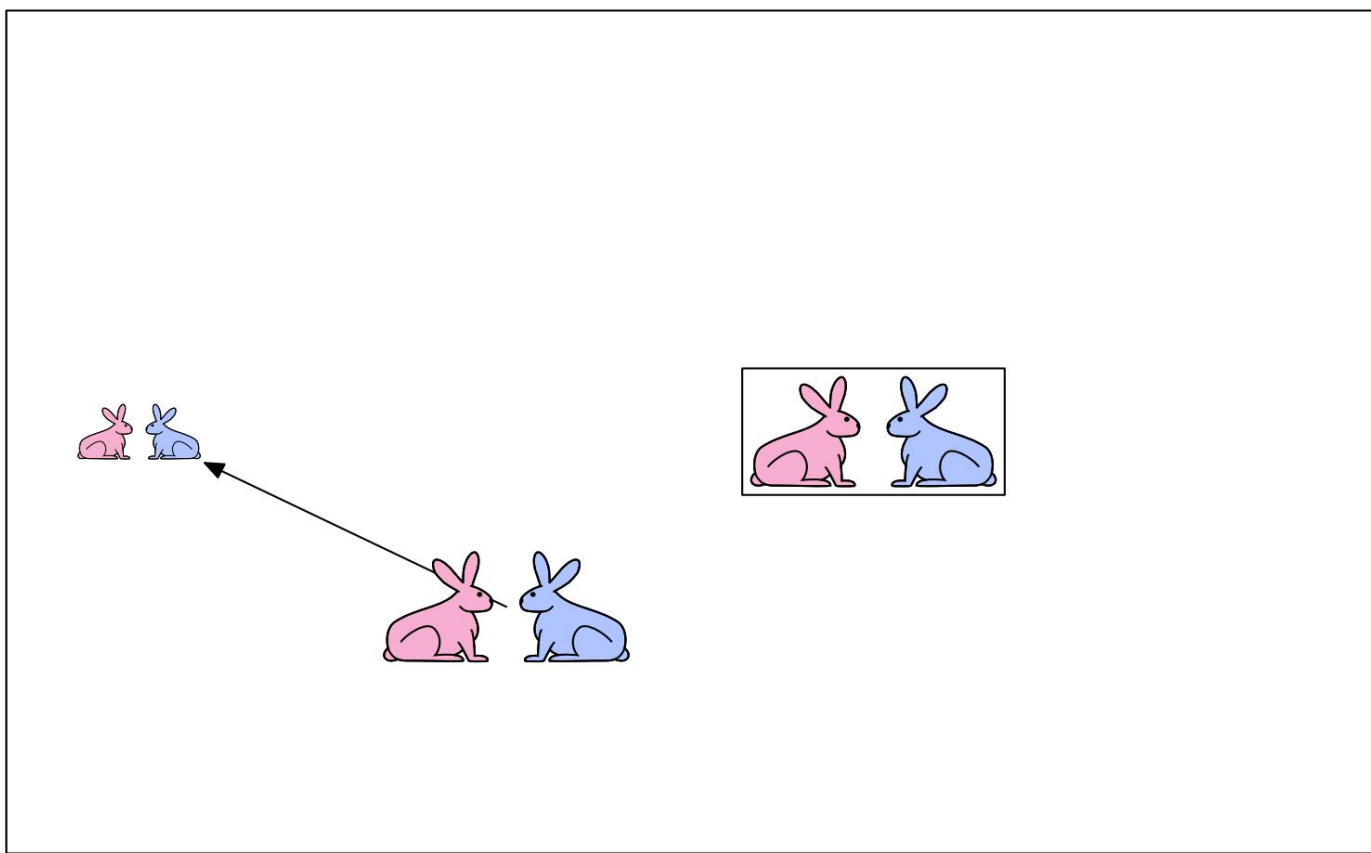
民主党人丹尼弗里曼和亚当哈茨教授



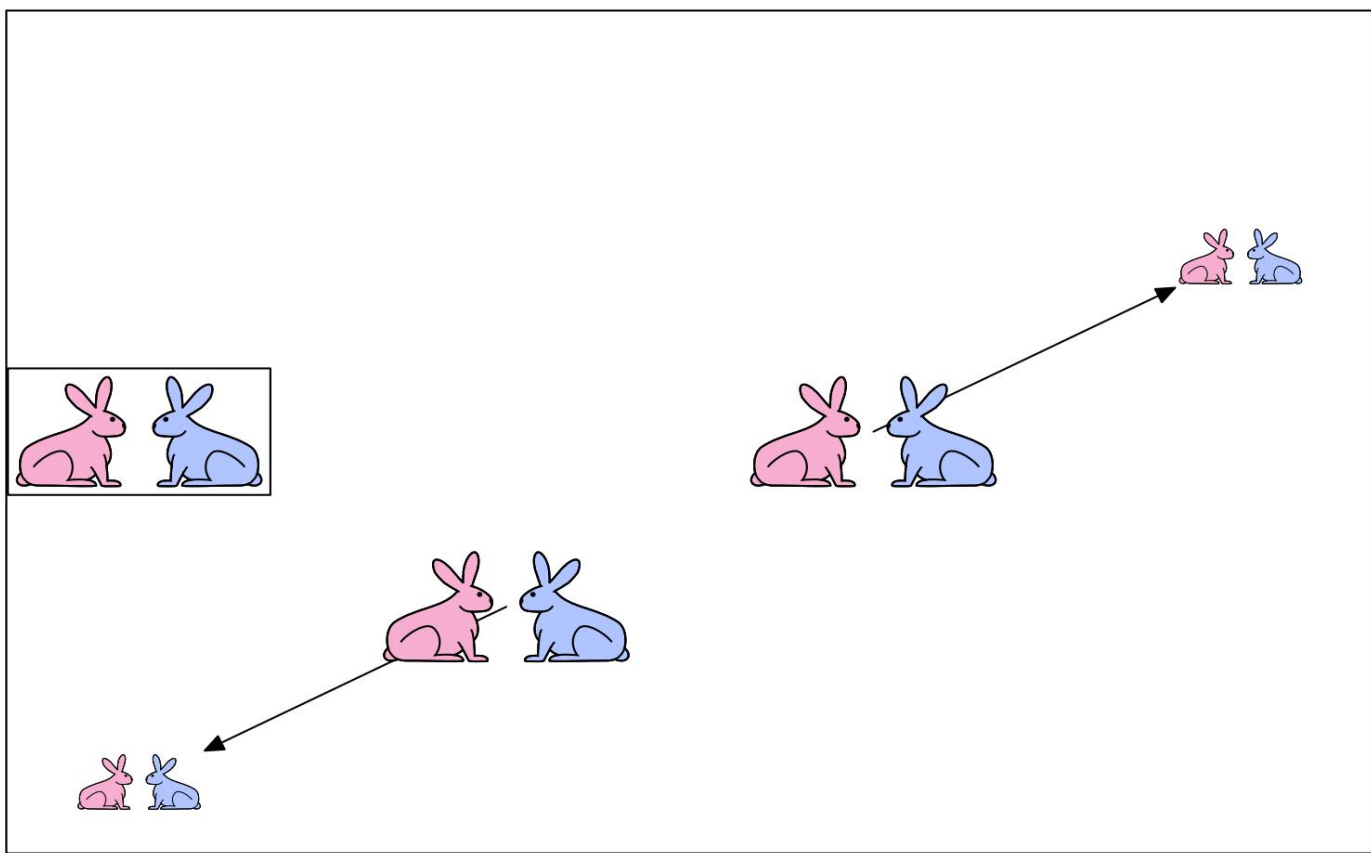
民主党人丹尼弗里曼和亚当哈茨教授



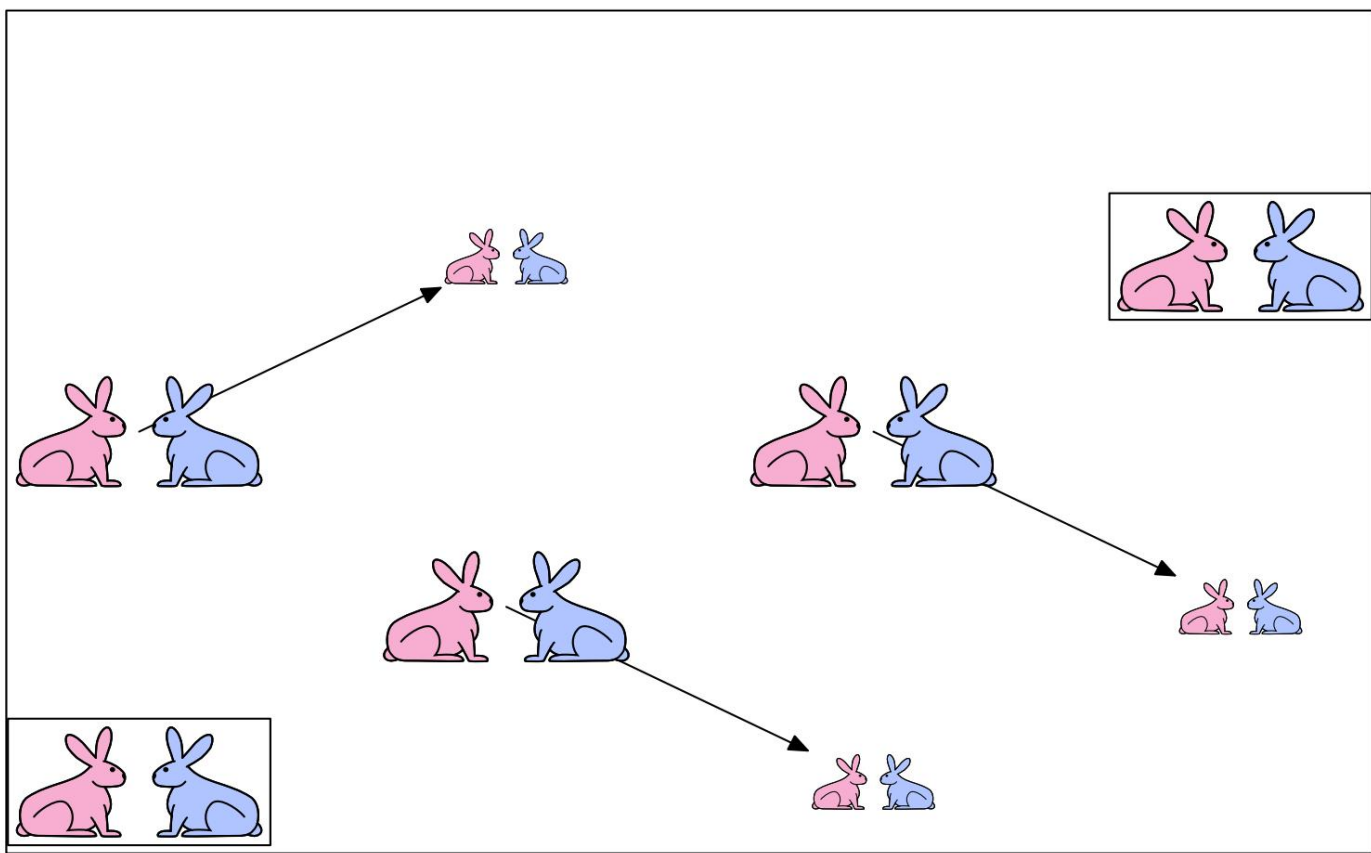
民主党人丹尼弗里曼和亚当哈茨教授



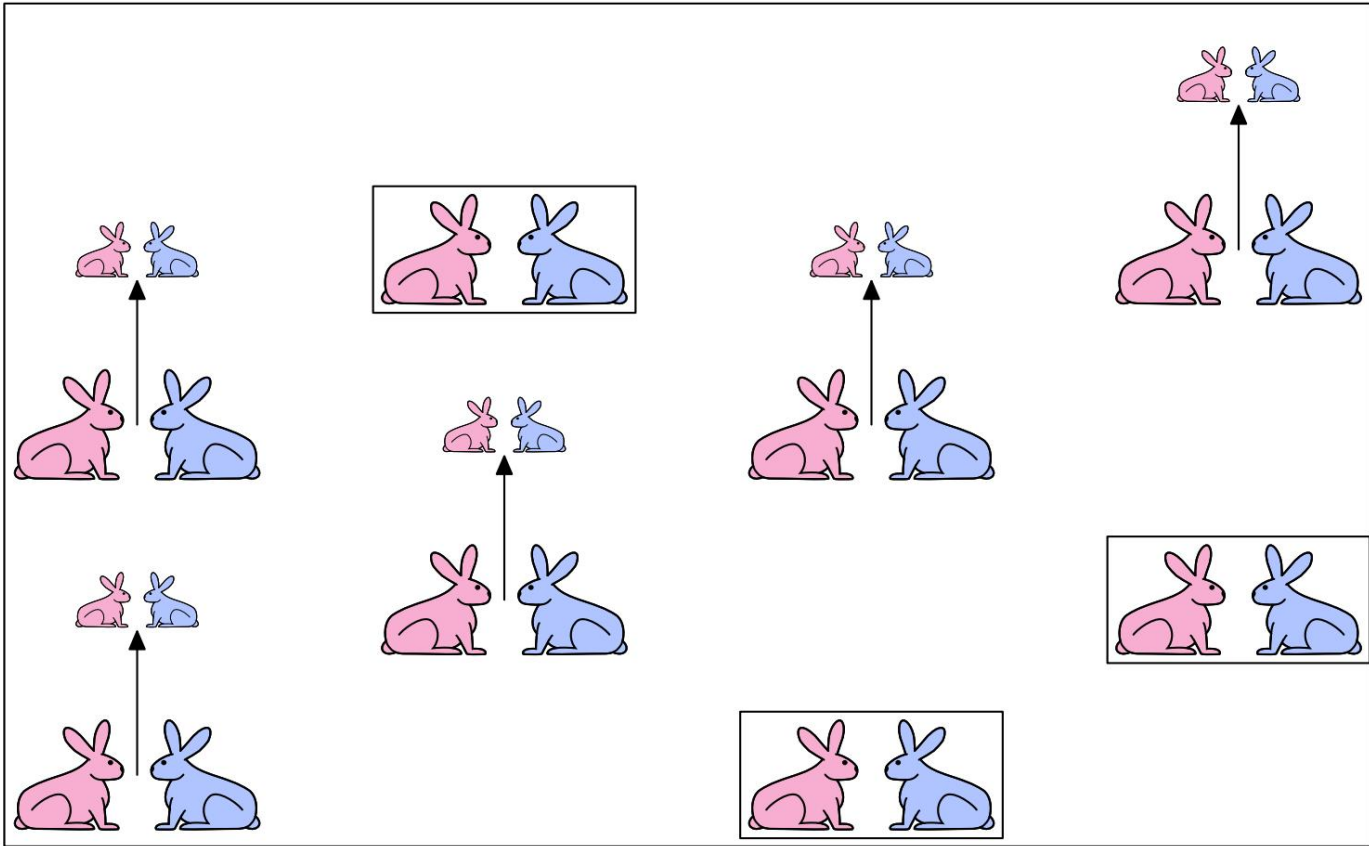
民主党人丹尼弗里曼和亚当哈茨教授



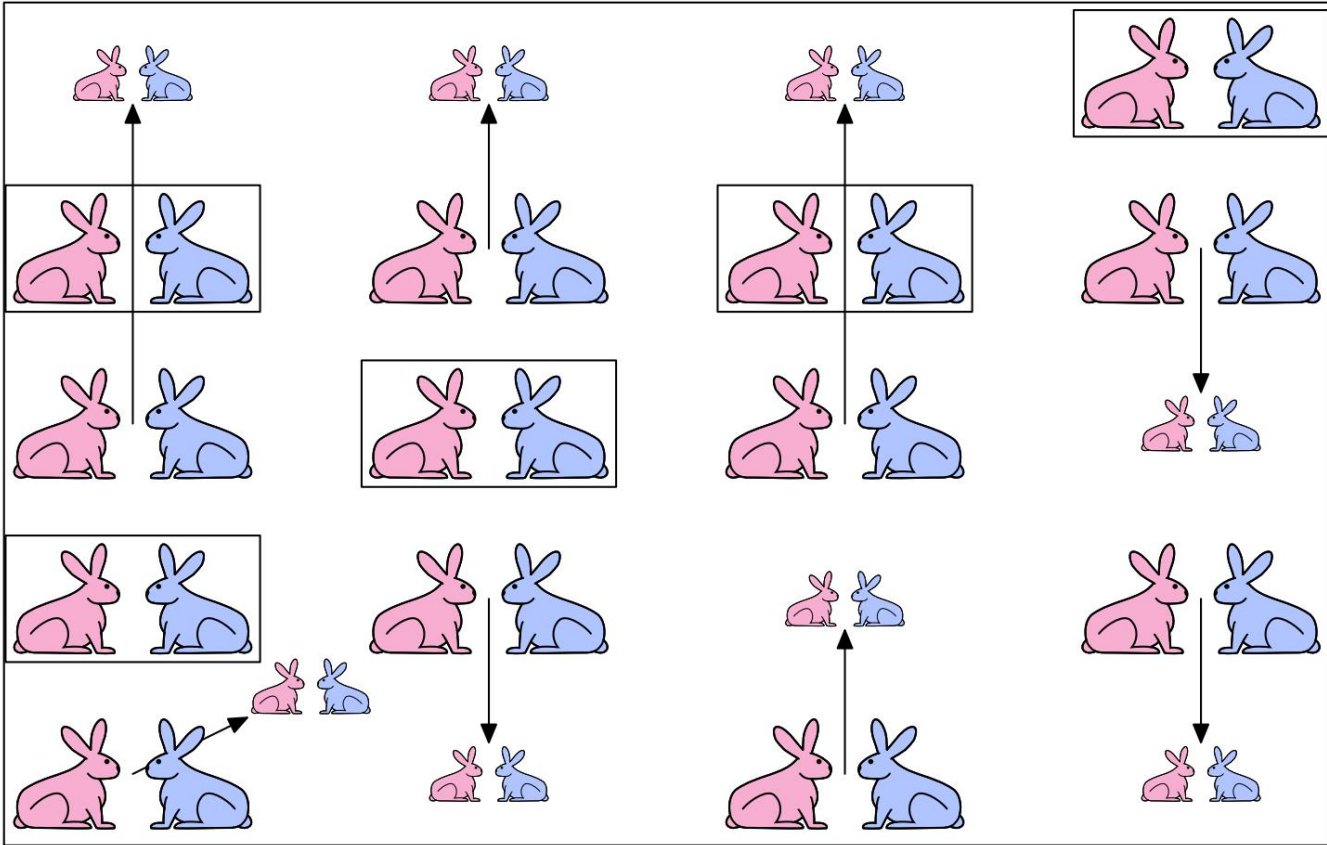
民主党人丹尼弗里曼和亚当哈茨教授

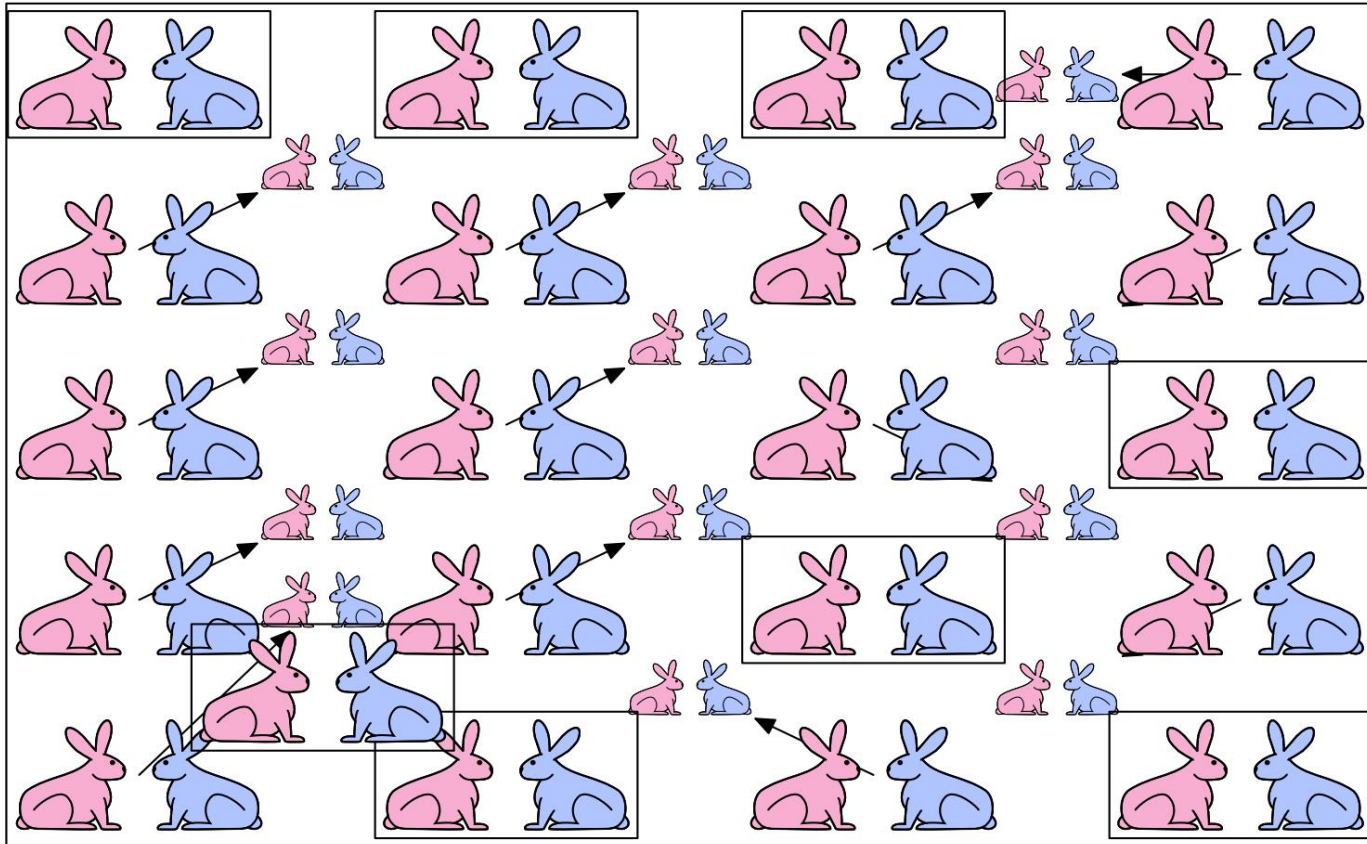


民主党人丹尼弗里曼和亚当哈茨教授

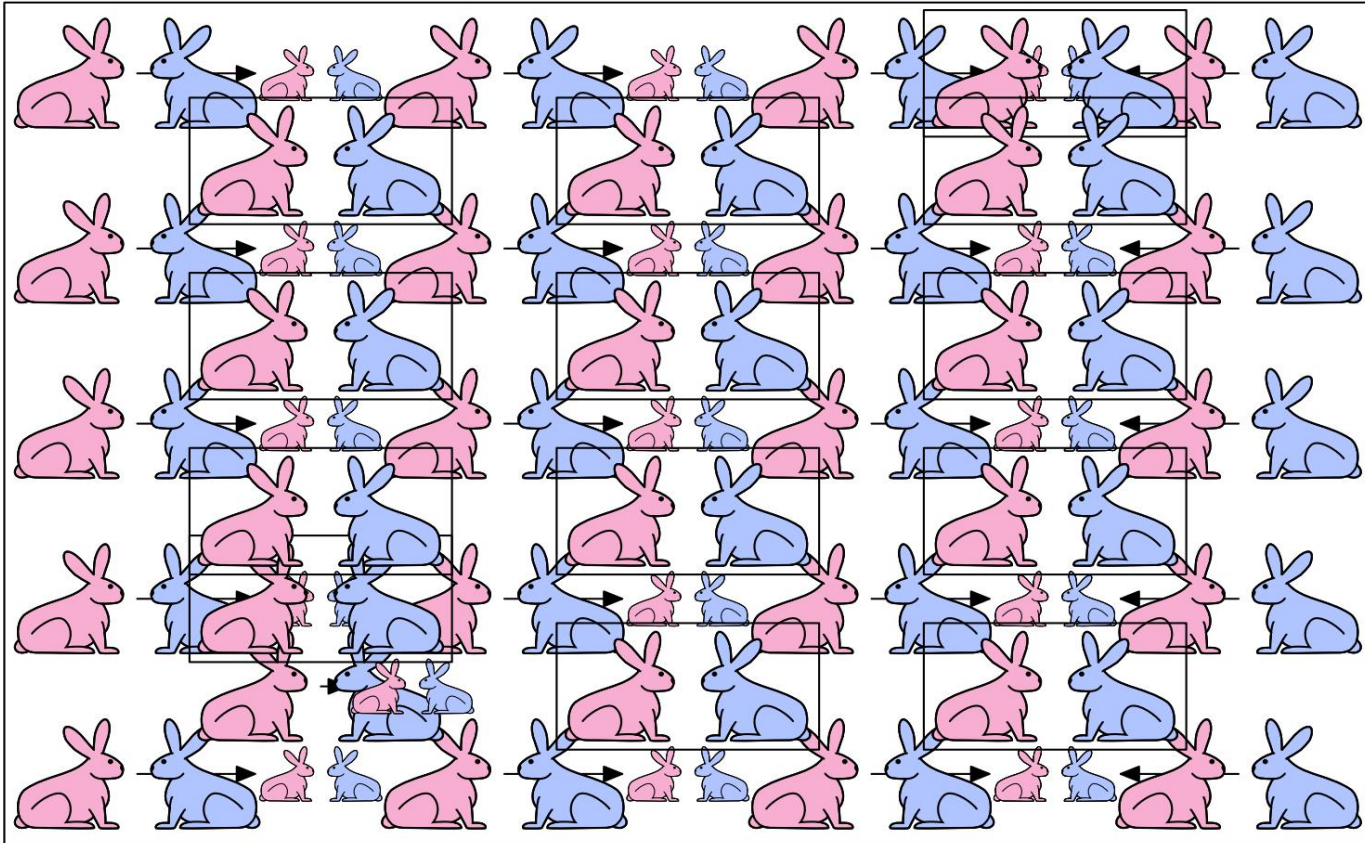


民主党人丹尼弗里曼和亚当哈茨教授





民主党人丹尼弗里曼和亚当哈茨教授



民主党人丹尼弗里曼和亚当哈茨教授

斐波那契

Ayeronemonth(callit0)–1female

Ayersecondmonth–sSll1female（现在怀孕）

Ayerthirdmonth – 两个女性,一个怀孕,一个没有

一般来说, $females(n)=females(n-1)+females(n-2)$

月	女性
0	1
1	1
2	2
3	3
4	5
5	8
6	13

- 这些可以添加到第 $n-1$ 个月的活动中的
gettotaliveinmonthn

斐波那契

§基本情况：

女性(0)=1
女性(1)=1

§递归案例

女性(n)=女性(n-1)+女性(n-2)

斐波那契

定义纤维 (x) :

假设 x 是一个 $\text{int} \geq 0$

返回 x 的斐波那契

如果 $x == 0$ 或 $x == 1$:

返回 1

别的:

返回 $\text{fib}(x-1) + \text{fib}(x-2)$

非递归 数字

§ howtocheckifastringofcharactersisapalindrome,即,
向前和向后读相同的

- “AblewasI,erelsawElba” 被授予拿破仑。
- “Arewenotdrawnonward,wefew,drawnonwardtonewera?”
归属于安妮迈克尔斯



图片由**维基百科**提供,在公共领域。



作者:Larth_Rasnal (自己的作品)[GFDL (<https://www.gnu.org/licenses/fdl-1.3.en.html>)或抄送 3.0 (<https://creativecommons.org/licenses/by/3.0/>)]通过**维基共享资源**。

递归求解？

§首先,将字符串转换为字符,去掉标点符号,然后将大写字母转换为小写

案子

§然后。

基本情况: `asStringoflength0or1isapalindrome` 递归情况:

- 如果第一个字符匹配最后一个字符,则
 `isisapalindromeifmiddlesecSonisapalindrome`

例子

§ AblewasI,erelsawElba to ablewasiereisawleba

§ isPalindrome(ablewasiereisawleba) issameas



◦ a == a 和

isPalindrome(blewasiereisawleb)



```
def isPalindrome(s):
```

```
    def toChars(s): s = s.lower()
```

```
        年 =
```

```
        对于s中的c :
```

```
            如果 abcdefghijklmnopqrstuvwxyz 中的c :
```

```
                年 = 年 + c
```

```
        返回答案
```

```
def isPal(s):
```

```
    如果 len(s) <= 1 :返回True 否则 :返回s[0]
```

```
    == s[-1]和isPal(s[1:-1])
```

```
    返回isPal(toChars(s))
```

分而治之

§ 一个“分而治之”算法的例子

§ 通过将其分解为子问题的集合来解决难题,例如: ○子问题比原始问题更容易解决

字典

如何储存 学生信息

§到目前为止,canstoreusingseparatelistseveryinfo

名称= [安娜 , 约翰 , 丹妮丝 , 凯蒂]

等级 = [B , A+ , A , A]

课程 = [2.00, 6.0001, 20.002, 9.01]

§每个项目的**单独列表** §每个列表必须具有**相同的长度**

如何更新/检索 学生信息

```
def get_grade (学生,name_list,grade_list,course_list) :
```

```
    i = name_list.index (学生)
```

```
    等级=等级列表[i]
```

```
    课程 = course_list[i]
```

```
    返回 (课程,成绩)
```

§ 如果有很多不同的信息来跟踪

§ 必须维护许多列表并传递参数

§ 必须始终使用整数索引

§ 必须记住要更改mulSplelists

一种更好、更清洁的方法

一本字典

§ niceto **index item of interest directly** (not always int)

§ nicetouse **one data structure**, no separate lists

一个列表

0	元素1
1	元素2
2	元素3
3	元素4
...	...

index

element

嗜好

Key1	选择1
键2	VAL2
键3	VAL3
Key4	Val4
...	...

custom
index by
label

element

Python字典

§ store pairsofdata ·
键 · 值

安娜 key1	选择1 val1
丹尼斯 key2	VAL2
约翰 key3	VAL3
凯蒂

custom
index by
label

element

我的字典 = {}



empty
dictionary

成绩 = { Ana : B , John : A+ , Denise : A , Katy : A }

键1val1

key2val2

关键3

val3

key4val4

字典查找

§类似于索引intoalist

§查找密钥_

§返回与键关联的值

安娜	乙
丹尼斯	一个
约翰	A+
凯蒂	一个

§如果key isn't found, done error

成绩 = { Ana : B , John : A+ , Denise : A , Katy : A }

成绩[约翰] 评估这个 “A+”

Give a KeyError 的等级[Sylvan]

字典 运营

安娜	乙
丹尼斯	一个
约翰	A+
凯蒂	一个
森林	一个

成绩 = { Ana : B , John : A+ , Denise : A , Katy : A }

§添加条目

成绩[Sylvan] = A

§测试ifkeyindicSonary

“约翰”的成绩

丹尼尔 的成绩

返回 True

返回 False

§删除条目

德尔 (度[安娜])

字典 运营

安娜	乙
丹尼斯	一个
约翰	A+
凯蒂	一个

成绩 = { Ana : B , John : A+ , Denise : A , Katy : A }

§ get an **iterable that acts like a tuple of all keys**

Grades.keys() → 返回 [Denise , Katy , John , Ana]

§ get an **iterable that acts like a tuple of all values**

grades.values() → 返回 [A , A , A+ , B]

no guaranteed
order

no guaranteed
order

字典键和值

§值 ·任何

类型 (不可变和可变) ·可以重复 ·
dictionary values can be lists, even
other dictionaries!

§键 ·必

须是唯一的 ·不可变类

型 (int、float、string、tuple、bool)

·实际上需要一个可散列的对象, 但将可变类型视为可散列的对象 ·小心将浮点类型视为密钥

§~~北~~to keys or values!

`d = {4:{1:0}, (1,3): 12 , const : [3.14, 2.7, 8.44]}`

列表

对比

听写

§ **有序** 的元素序列

§ 查找元素按整数索引

§ 索引有**顺序**

§ indexisan**整数**

§ **匹配** “键”到 “值”

§ lookuponeitemanotheritem _

§ 保证**北方**

§ keycanbeany**不**
可变类型

示例:3 个函数 分析歌曲歌词

1)创建频率字典映射str:int

2)找到出现最多的单词和多少个Smes

· usealist, incase there is more than oneword ·

returnatuple(list,int)for(words_list,highest_freq)

3) 找到最少出现 X0 次的单词

·让用户选择 “atleastXSmes” ,因此允许作为参数 ·

returnalistoftuples,eachtupleisa(list, int)

包含按频率排序的单词列表 · IDEA:FromsongdicSonary,查找最频

繁的单词。删除最常用的单词。重复。它之所以有效,是因为您是 MutaSngtheson

gdic 声纳。

创建字典

`def` Lyrics_to_frequencies(歌词):

我的字典 = {}

歌词中的单词:

如果myDict中的单词:

myDict[单词] += 1

别的:

我的字典[单词] = 1

返回我的字典

can iterate over list
can iterate over keys
in dictionary
update value
associated with key

使用字典

`def` most_common_words (频率) :

值 = freqs.values()

最佳 = 最大值 (值)

话 = []

对于频率中的k :

如果freqs[k] == 最好:

words.append(k)

返回 (单词,最好的)

this is an iterable, so can
apply built-in function
can iterate over keys
in dictionary

利用字典 特性

```
def words Often(freqs, minTimes):
```

```
    结果 = []
```

```
    完成 = 假
```

```
    虽然没有完成:
```

```
        temp = most_common_words(freqs) if temp[1]
        >= minTimes: result.append(temp) for w in
            temp[0]: del(freqs[w])
```

```
    别的:
```

```
        完成=真
```

```
    返回结果
```

can directly mutate
dictionary; makes it
easier to iterate

打印 (words Often (披头士乐队,5))

斐波那契递归代码

定义纤维 (n) :

如果 $n == 1$:

返回 1

elif $n == 2$:

返回 2

别的:

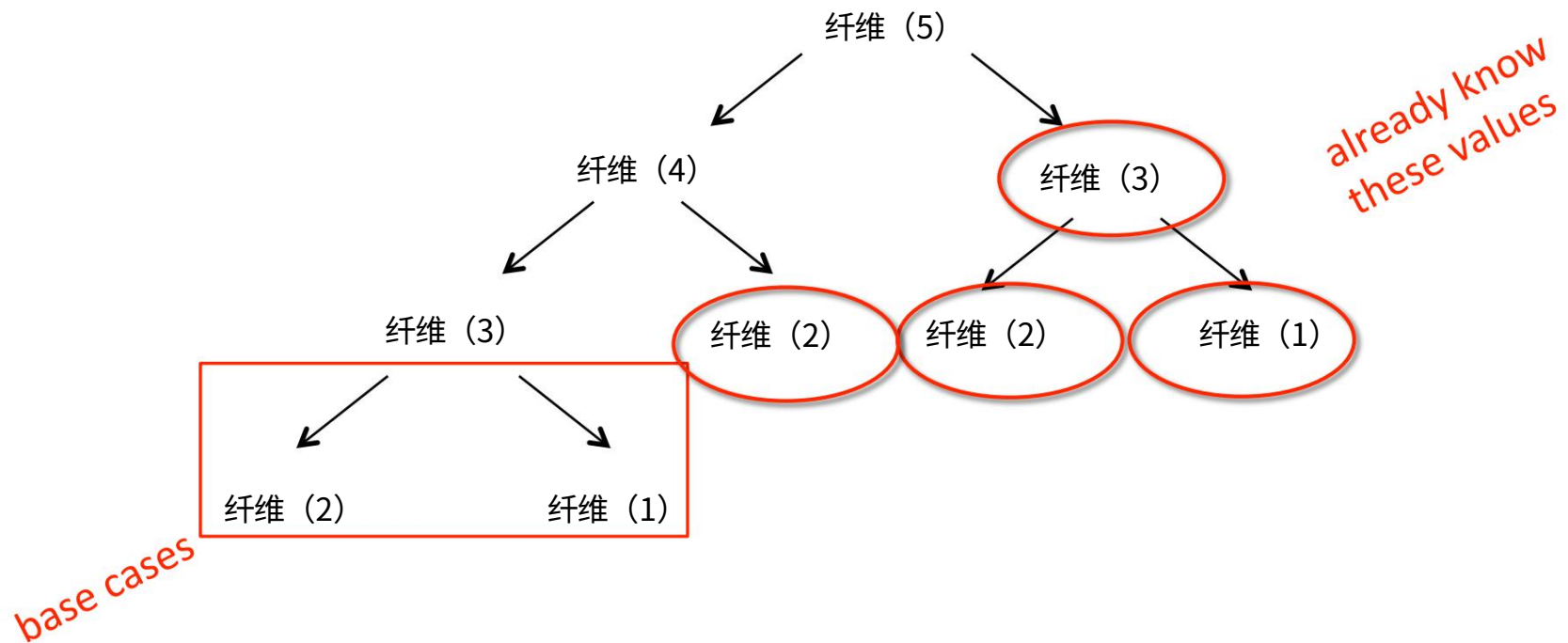
返回 $\text{fib}(n-1) + \text{fib}(n-2)$

§两个基本情况

§自称两次

§此代码效率低下

低效斐波那契 $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$



§重新计算 same values many times! §可以跟踪已经计算的值

斐波那契字典

```
def fib_efficient(n, d):
    如果n在d 中:
        返回d[n]
    别的:
        ans = fib_efficient(n-1, d) + fib_efficient(n-2, d)
        d[n] = 年
        返回答案
```

```
d = {1:1, 2:2}
```

```
打印 (fib_efficient (6,d) )
```

Method sometimes called "memoization"

Initialize dictionary with base cases

§ do a lookup first in case already calculated the value § 修改
字典作为通过funcSoncalls的进度

效率提升

§调用 fib(34) results in 11,405,773 recursive calls to
程序 §调用

fib_efficient (34) results in 65 recursive calls to
步骤

§使用 dicSonaries 捕获中间结果可能非常有效

麻省理工学院开放课件
<https://ocw.mit.edu>

6.0001 计算机科学和 Python 编程简介
2016 年秋季

有关引用这些材料或我们的使用条款的信息,请访问: <https://ocw.mit.edu/terms>。