数据结构与算法（Python）-06/第7周

北京大学 陈斌

2021.04.20

线下课堂

- 本周内容小结
- 贪心策略与动态规划
- 慕课作业讲解
- 课堂练习

W06：递归（下）

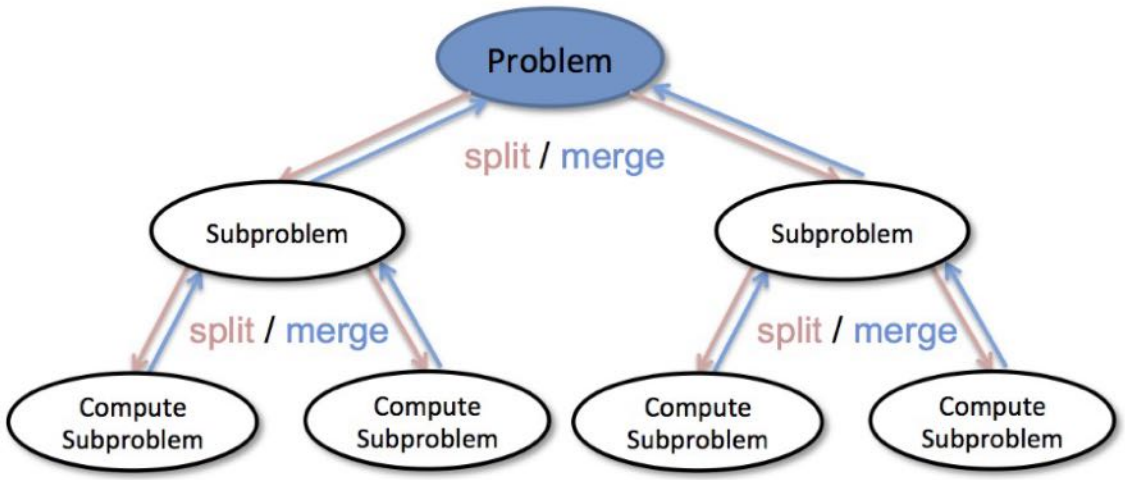
- 408 分治策略 5m34s
- 409 优化问题和贪心策略 7m49s
- 410 找零兑换问题的递归解法 12m32s
- 411 找零兑换问题的动态规划解法 13m26s
- 412 动态规划案例分析 17m53s
- 413 递归小结 6m51s

分治策略

❖ 解决问题的典型策略：分而治之

将问题分为若干更小规模的部分

通过解决每一个小规模部分问题，并将结果汇总得到原问题的解



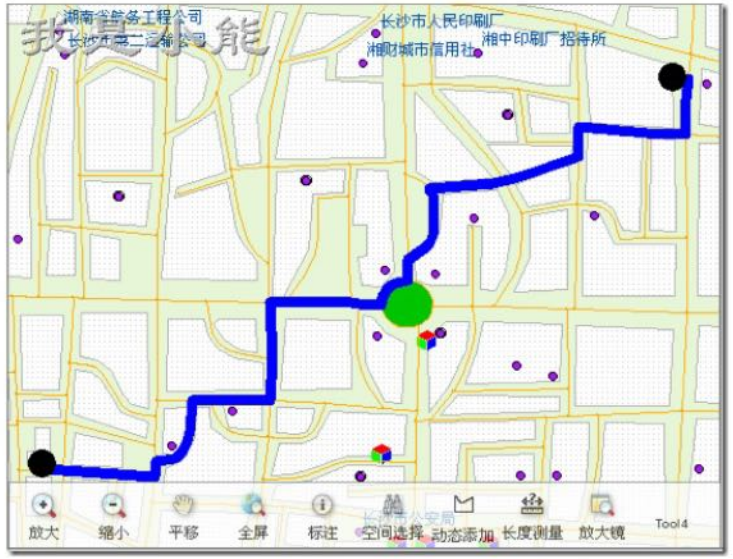
优化问题

- 计算机科学中许多算法都是为了找到某些问题的最优解

例如，两个点之间的最短路径；

能最好匹配一系列点的直线；

或者满足一定条件的最小集合



贪心策略Greedy Method

- **贪心策略**

  因为我们每次都试图解决问题的尽量大的一部分

  对应到兑换硬币问题，就是每次以最多数量的最大面值硬币来迅速减少找零面值

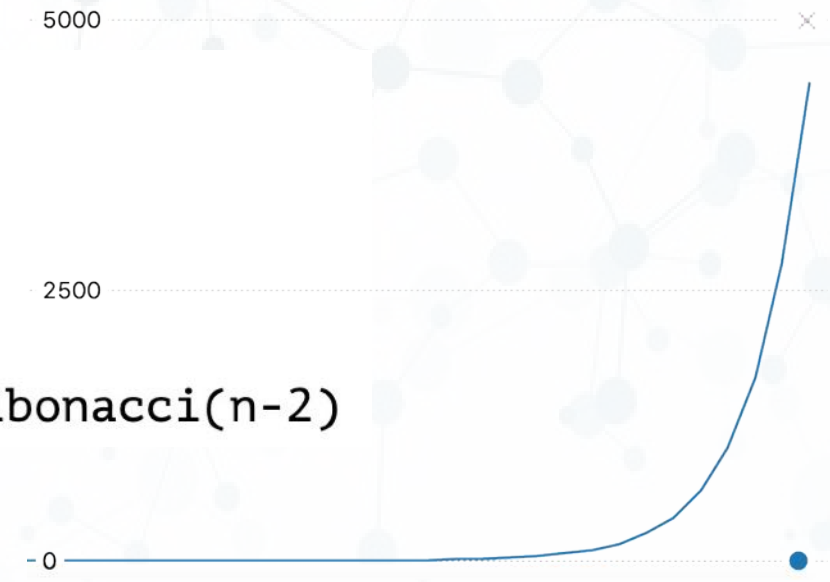
- **“贪心策略”解决找零兑换问题，在美元或其他货币的硬币体系下表现尚好**

递归的问题

- **用分治策略来设计算法，用递归最直接**
  算法简洁易懂

- **但递归方法的问题在于重复计算，导致计算量爆炸**
  你猜猜用这个能计算到斐波那契数列的第几个？

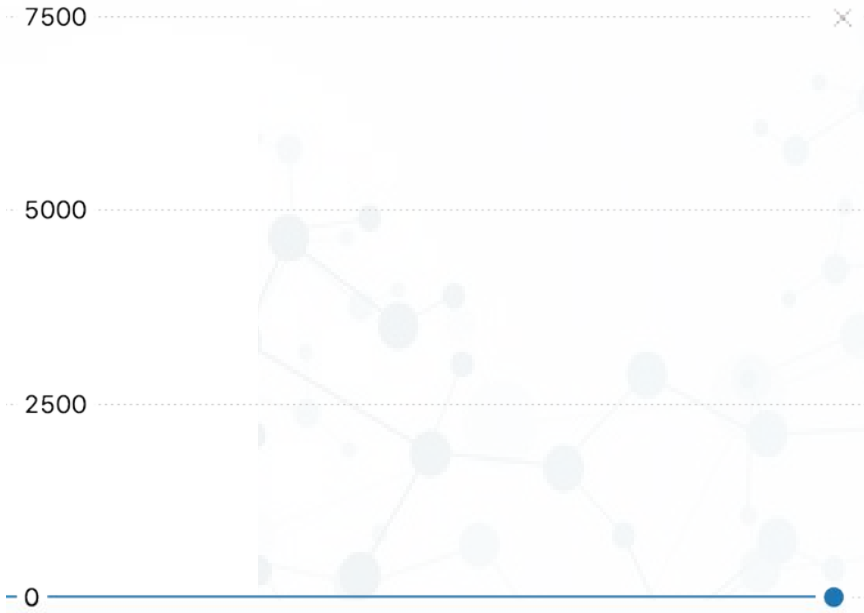
```python
def fibonacci(n):
    if n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)
```



函数值缓存/备忘录技术解决递归重复计算


```python
fib = [-1]*100

def fibonacci(n):
    if n == 1:
        return 0
    elif n == 2:
        return 1
    elif fib[n] >= 0:
        return fib[n]
    else:
        fib[n-1]= fibonacci(n-1)
        fib[n-2]= fibonacci(n-2)
        return fib[n-1] + fib[n-2]
```

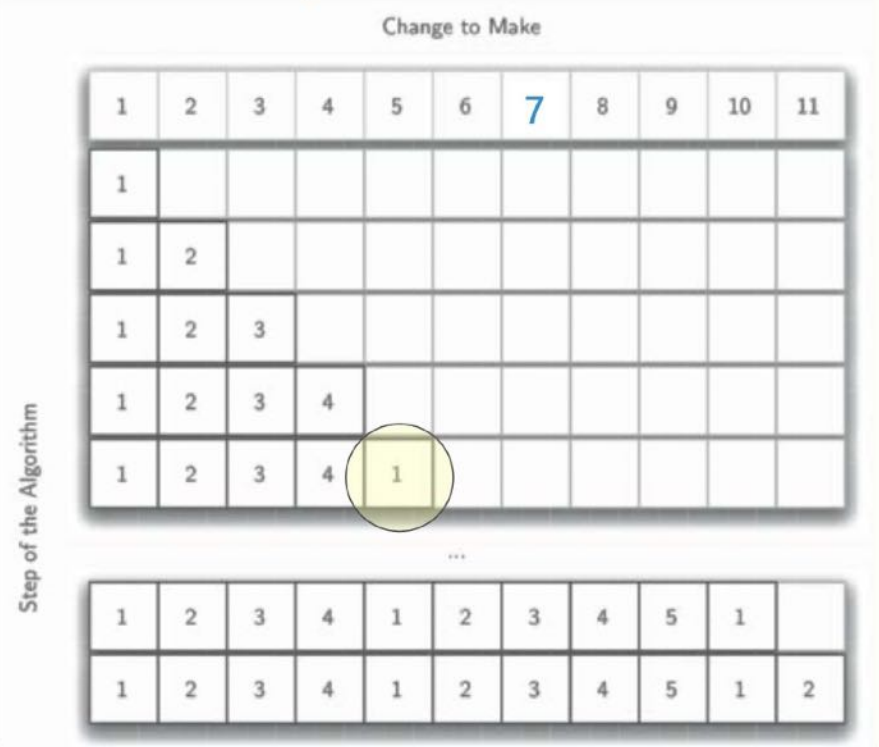


从函数值缓存到动态规划

找零兑换：动态规划算法

- 采用动态规划来解决11分钱的兑换问题

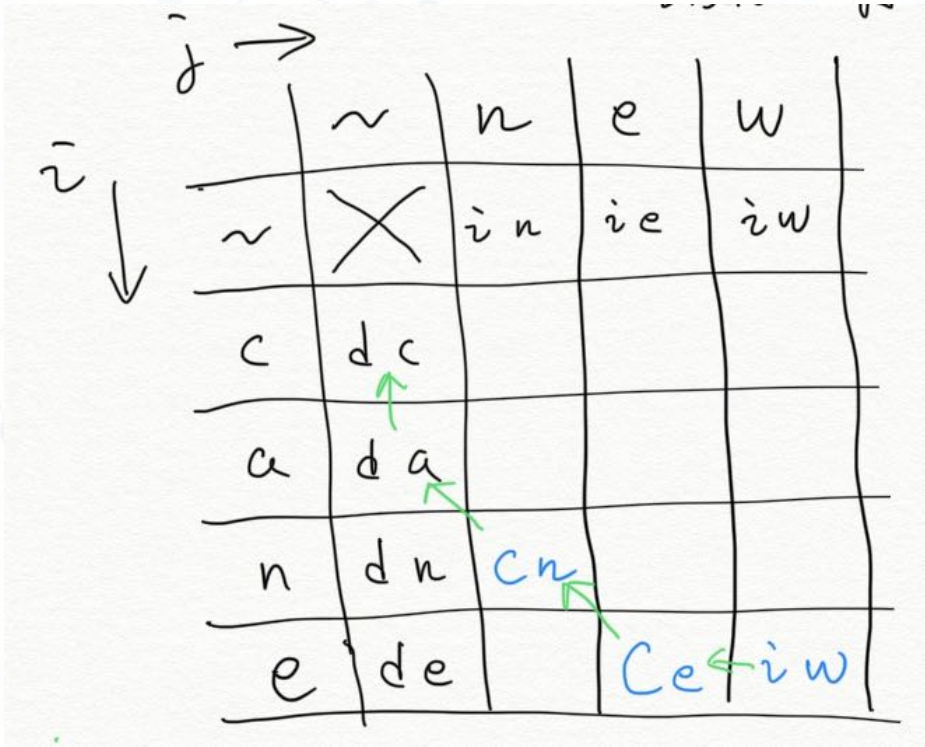
  从1分钱兑换开始，逐步建立一个兑换表



贪心策略和动态规划的区别

- 贪心算法要求的“局部最优等同于总体最优”，
- 和动态规划要求的“问题最优解包括规模更小相同问题的最优解”，
- 二者有什么不同？
- 包括：仅依赖于规模小的最优解

尽量大步 → 尽量大步 → 到达最优



动态规划的适用范围

- **最优 子结构**

  问题的最优解包含子问题的最优解

- **无后效性**

  当前阶段的状态仅由以前阶段决定；

  后续阶段状态的变化不会影响当前阶段的状态。

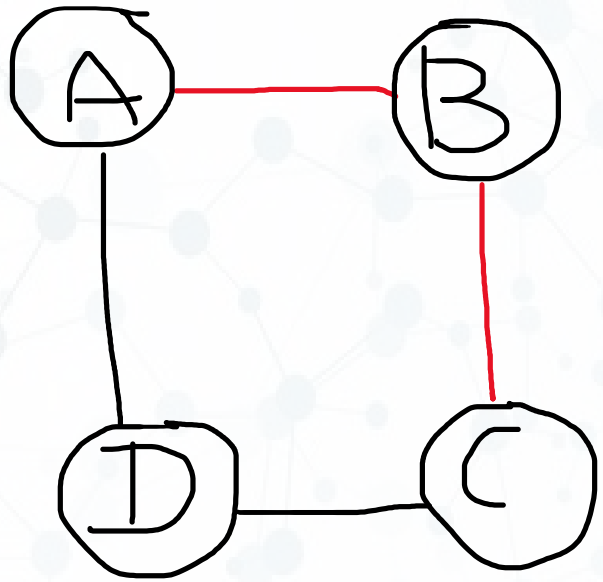
- **重复 子问题**

  解决问题过程中的子问题存在大量重叠（冗余计算）

- **不适用动态规划的反例：**

  最长路径问题

  A-C的最长路径经过B，但这条最长路径并不一定包含A-B的最长路径



慕课W06-1:铺瓷砖递归思路

- **长度为N的区域，铺装方法total(N)=**

  第1块铺灰（1），total(N-1)

  + 第1块铺红（2），total(N-2)
  + 第1块铺绿（3），total(N-3)
  + 第1块铺蓝（4），total(N-4)

$$total(N)=\begin{cases}1, N=0\\1, N=1\\2, N=2\\4, N=3\\8, N=4\\\sum_{i=1}^{4} total(N-i)\end{cases}$$

慕课W06-1:铺瓷砖递归版本


```python
# W06-1, 铺瓷砖递归版本
n = int(input())
cache = {0: 1, 1: 1, 2: 2, 3: 4, 4: 8}

# 第1块是1, total(n-1)
# 第1块是2, total(n-2)
# ..第1块是4, total(n-4)
def total(n):
    if n <= 4:
        return cache[n]
    else:
        cache[n] = total(n - 1) + total(n - 2) \
                   + total(n - 3) + total(n - 4)
        return cache[n]

print(total(n))
```

即使用了函数值缓存，
还有超时用例

慕课W06-1:铺瓷砖迭代版本


```python
# W06-1-2, 铺瓷砖迭代版本
n = int(input())
cache = {0: 1, 1: 1, 2: 2, 3: 4, 4: 8}

def total(n):
    if n <= 4:
        return cache[n]
    else:
        for i in range(5, n + 1):
            cache[i] = cache[i - 1] + cache[i - 2] \
                       + cache[i - 3] + cache[i - 4]
        return cache[n]

print(total(n))
```

慕课W06-2：分发糖果，贪心策略思路

- **记录评分ratings，和对应的糖果数cds**
- **将小朋友的评分按照次序加入**

第一个，无论多少分，只有1颗糖果；

后续的：

如果分数高于前一个，则比前面多1颗糖果；

如果分数与前一个相同，则给最低标准1颗糖果；

如果分数低于前一个，则给最低标准1颗糖果……

……但是，如果前一个仅有1颗糖果，

……就需要加1颗糖，并向前继续加，

……直到评分不递增，或者糖果有多。

慕课W06-2:分发糖果


```python
def candy(ratings):
    cds = [1] * len(ratings)  # 糖果列表
    for i in range(1, len(ratings)):
        if ratings[i - 1] < ratings[i]:
            cds[i] = cds[i - 1] + 1  # 分数高，比前面多1颗糖
        elif ratings[i - 1] == ratings[i]:
            cds[i] = 1  # 分数相同，给最低标准1颗糖
        else:
            cds[i] = 1  # 分数低，给最低标准1颗糖
            if cds[i - 1] == 1:  # 但如果前面仅有1颗糖，需要加糖
                for k in range(i - 1, -1, -1):
                    cds[k] += 1
                    # 评分向前递增，而糖果数没有递增的话，才加糖
                    if k > 0 and (ratings[k] >= ratings[k - 1] \
                                  or cds[k] < cds[k - 1]):
                        break
    return sum(cds)
```

思路2:两次扫描（左-右；右-左）

- 分发糖果是一个贪心策略可解的问题
- 从左到右，从右到左，两次扫描即可
- 每趟扫描的规则：

  至少1颗糖；

  如果下一人评分变高加1颗；不高就不加
- 取每人两趟的最大值累加，就是最少糖果数量
- https://leetcode-cn.com/problems/candy/solution/

慕课W06-3:表达式按不同顺序求值递归思路

- 以**后缀表达式求值**作为基础
- 对nums，ops的运算次序组合进行穷举
- calc(nums, ops, numstk, opstk)

  nums：剩余操作数列表；ops：剩余运算符列表；

  numstk：当前的操作数栈；opstk：当前的运算符栈。
- 递归结束条件：nums中没有操作数了，连续计算得到值

  结果加入结合results，自动去重复。
- 递归：对于栈中的子表达式

  不计算，再入一个操作数和一个操作符，递归调用；

  计算1次、2次……，分别再入一个操作数和一个操作符，递归调用。

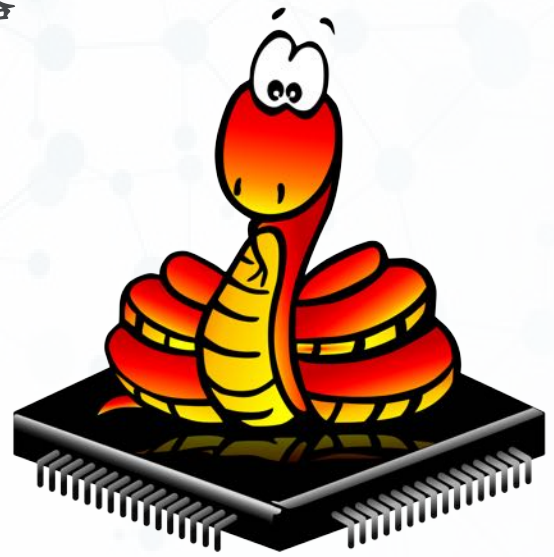
```
27      def calc(nums, ops, numstk, opstk):
28          if len(nums) == 0:  # 递归结束，计算值返回。
29              while len(opstk) > 0:
30                  op = opstk.pop()
31                  op2 = numstk.pop()
32                  op1 = numstk.pop()
33                  numstk.append(doMath(op, op1, op2))
34              results.add(numstk.pop())
35              return
36          else:
37              # 不计算栈中的子表达式，入栈，递归调用
38              calc(nums[1:], ops[1:], \
39                   numstk + [nums[0]], opstk + [ops[0]])
40              while len(opstk) > 0:
41                  # 计算一到多次子表达式，入栈，递归调用
42                  op = opstk.pop()
43                  op2 = numstk.pop()
44                  op1 = numstk.pop()
45                  numstk.append(doMath(op, op1, op2))
46                  calc(nums[1:], ops[1:], \
47                       numstk + [nums[0]], opstk + [ops[0]])
```

```
4  def findWays(expr):
5      # 用于将字符串转为数字与运算符，供参考
6      nums, ops = [], []
7      num = 0
8      for c in expr:
9          if "0" <= c <= "9":
10             num = num * 10 + ord(c) - 48
11         else:
12             ops.append(c)
13             nums.append(num)
14             num = 0
15     else:
16         nums.append(num)
17     results = set()
```

```
49     # 头两个数入栈，头一个运算符入栈，递归调用
50     calc(nums[2:], ops[1:], [nums[0], nums[1]], [ops[0]])
51     # 结果排序、输出
52     ret = sorted(list(results))
53     return ",".join(map(str, ret))
```

基于Python的开源硬件

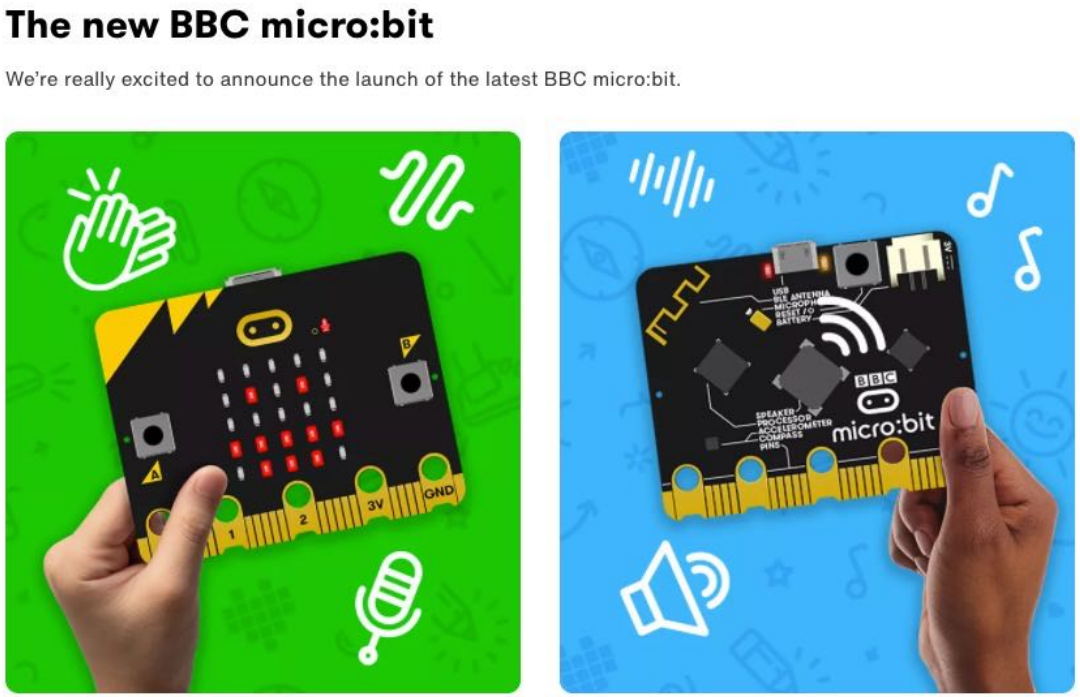
- **Python语言的应用不仅限于web，爬虫，数据处理**
  甚至可以和C语言一样驱动硬件
- **这就是从2014年开始的micropython项目**
  基于python3.4语法，python3的子集
  这是以STM32微控制器为核心的pyboard，还扩展到ESP32系列微控制器
- **Micropython被移植到了更多微控制器**
- **甚至功能更为强大的神经网络加速芯片K210（MaixPy）**
  可以进行计算机视觉处理、深度神经网络图像分类、人脸识别等
  以后再为大家介绍



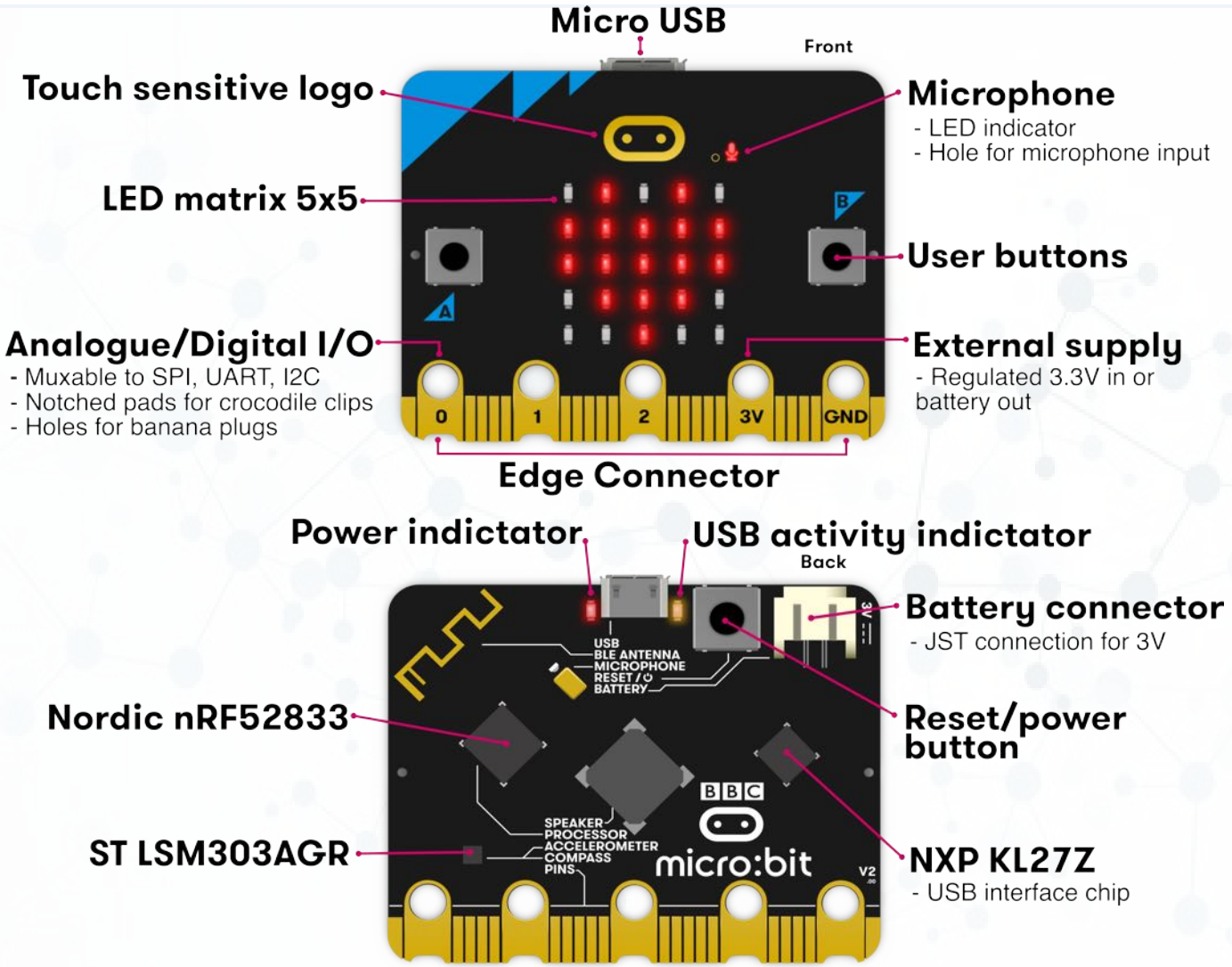
micro:bit

- BBC micro:bit教育基金会研发
- 专注于青少年编程教育
- 从2017年开始拥有了micropython移植固件
- 我们的同学也在micro:bit上创作了许多精彩的创意作品

  当然，大家印象最深的就是“MEMORY ERROR”内存不足
- 2020年下半年，micro:bit教育基金会发布了第二代micro:bit



micro:bit v2


Micro USB
Front
Touch sensitive logo
Microphone
- LED indicator
- Hole for microphone input
LED matrix 5x5
User buttons
Analogue/Digital I/O
- Muxable to SPI, UART, I2C
- Notched pads for crocodile clips
- Holes for banana plugs
External supply
- Regulated 3.3V in or battery out
0
1
2
3V
GND
Edge Connector
Power indictator
USB activity indictator
Back
Battery connector
- JST connection for 3V
USB
BLE ANTENNA
MICROPHONE
RESET / ⏻
BATTERY
Nordic nRF52833
Reset/power button
SPEAKER
PROCESSOR
ACCELEROMETER
COMPASS
PINS
ST LSM303AGR
NXP KL27Z
- USB interface chip
micro:bit
V2

功能-性能参数

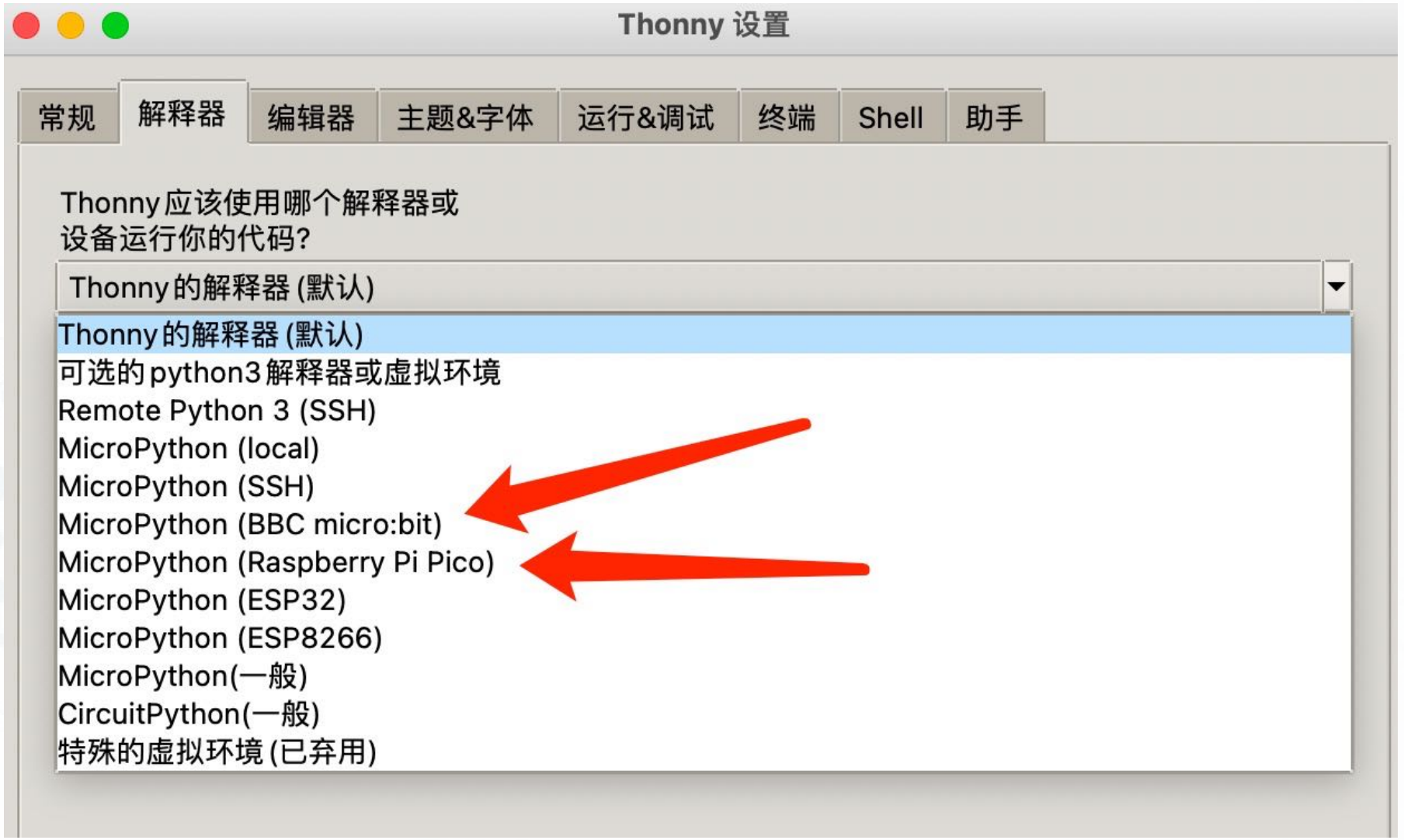
项目	micro:bit v1.5	micro:bit v2
MCU	Nordic nRF51822	Nordic nRF52833
ARM类型	Cortex-M0 32bit,16MHz	Cortex-M4 32bit+FPU,64MHz
内存/闪存	16KB RAM/256KB Flash	128KB RAM/512KB Flash
IO端口	19GPIO;SPI/I2C/UART 6个10-bit ADC 3个PWM通道	19GPIO; SPI/I2C/UART 6个10-bit ADC 3个PWM通道
传感器	时钟和定时器 温度传感器 5*5 LED/光线传感器 2个按钮 蓝牙/无线 磁力计；加速计	时钟和定时器 温度传感器 5*5 LED/光线传感器 2个按钮 蓝牙/无线 磁力计；加速计 麦克风；喇叭；可触摸LOGO
价格	$15；RMB100	$15；RMB100

开发文档

- micropython文档
  http://docs.micropython.org/en/latest/

- micro:bit v2
  https://github.com/microbit-foundation/micropython-microbit-v2（micropython固件）
  https://microbit-micropython.readthedocs.io/en/v2-docs/（文档）

Python集成开发环境：thonny



micro:bit v2编程演示

› 摇骰子，摇到6点就变笑容和播放音乐

```
1  from microbit import *
2  from random import randint
3  import music
4
5  while True:
6      display.show("*")
7      if accelerometer.was_gesture("shake"):
8          display.clear()
9          d = randint(1, 6)
10         display.show(str(d))
11         sleep(1000)
12         if d == 6:
13             display.show(Image.HAPPY)
14             music.play(music.POWER_UP)
15     sleep(100)
```

课程活动预告：Python开源硬件

chbpku 动态 投稿 577 频道 8 收藏 14 订阅 设置 搜索视频

关注数	粉丝数	获赞数	播放数	阅读数
35	1.9万	3.3万	42.9万	7330

【sessdsa18】第13组：宇宙飞船系列游戏

173 2018-5-7

【sessdsa18】第14组：遥控向日葵花车

348 2018-5-7

【sessdsa18】第15组：深蹲计数&猜旋律

119 2018-5-7

【sessdsa18】第16组：paper.io多人对抗游戏

464 2018-5-7

【sessdsa18】第17组：迷宫

185 2018-5-7

【sessdsa18】第18组：复古小游戏机

242 2018-5-7

【sessdsa18】第19组：点灯游戏

134 2018-5-7

【sessdsa18】第20组：野外探险工具箱

196 2018-5-7

【sessdsa18】第21组：奔跑计步器

404 2018-5-7

【sessdsa18】第22组：疯狂炸弹人（超长3分钟）

330 2018-5-7

【sessdsa18】第23组：捕鱼达人

364 2018-5-7

【sessdsa18】第24组：皮坎旋律（音乐游戏）

165 2018-5-7

【sessdsa18】第25组：双人射击对战游戏

【sessdsa18】第26组：俄罗斯方块

【sessdsa18】第27组 音乐合奏—天空之城

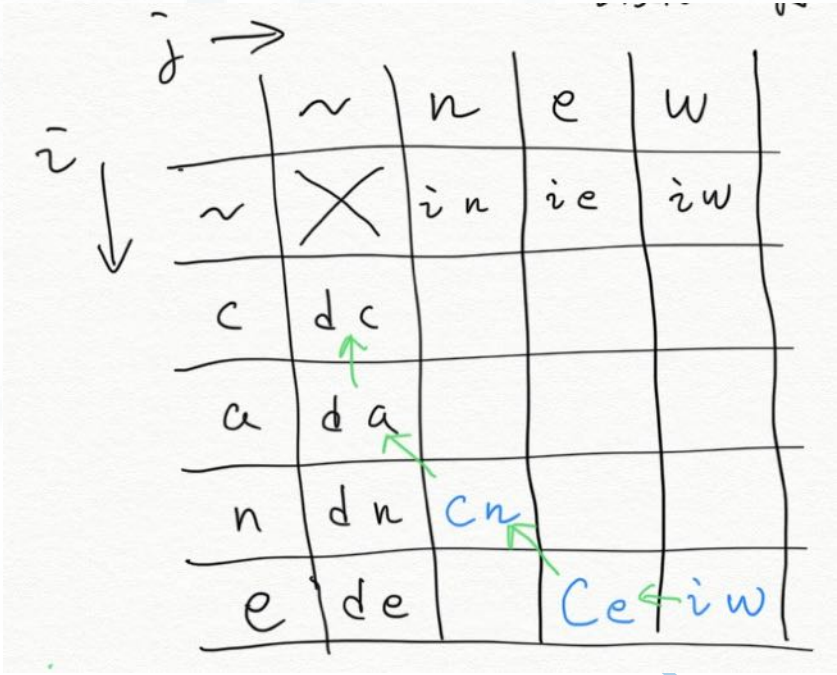
【sessdsa18】第28组：Micro:musicbox

【sessdsa18】第29组：数算课的生死时速

【sessdsa18】第30组：无敌乒乓球

【K06】单词最小编辑距离的递归解法

$$m(i,j)=\begin{cases} \text{insert} * j & \text{当 } i==0 \\ \text{delete} * i & \text{当 } j==0 \\ \min\begin{cases} \text{copy } o[i] + m(i-1,j-1) & \text{"有条件"\ } o[i]==t[j] \\ \text{delete } o[i] + m(i-1,j) \\ \text{insert } t[j] + m(i,j-1) \end{cases} \end{cases}$$



下课！