问题解答

树的实现、二叉堆详解

慕课测验详解

交互编程两种范式

数据结构与算法（Python）-15/0421

陈斌 gischen@pku.edu.cn 北京大学地球与空间科学学院

目录

- 问题解答
- 树的实现
- 二叉堆详解
- 慕课测验详解
- 单片机交互编程
- 课堂练习

问题解答

- 这周会有OJ讲解吗，太难（5次）
- python的字典是怎么用散列实现的？
- 掌控版没有pygame怎么做游戏，能给点提示吗？
- 能否讲一下指针的概念，以及在不同语言下的区别？
- 发现H4传错了代码的版本（只要改动一个单词），当时的网站还检查不出错误，这种情况有救吗？

本周内容

- 601 什么是树 8m37s
- 602 树结构相关术语 8m23s
- 603 树的嵌套列表实现 11m00s
- 604 树的链表实现 6m57s
- 605 树的应用：表达式解析（上） 13m03s
- 606 树的应用：表达式解析（下） 15m15s
- 607 树的遍历 10m11s
- 608 优先队列和二叉堆 11m45s
- 609 二叉堆的Python实现 13m14s

问题解答

- **请问有什么方法得到树的高度？**

  ……遍历

- **定义binarytree的时候没有用到结点的定义，但定义链表的时候会用到，这是为什么？**

  ……class BinaryTree是递归数据结构，与class Node一样

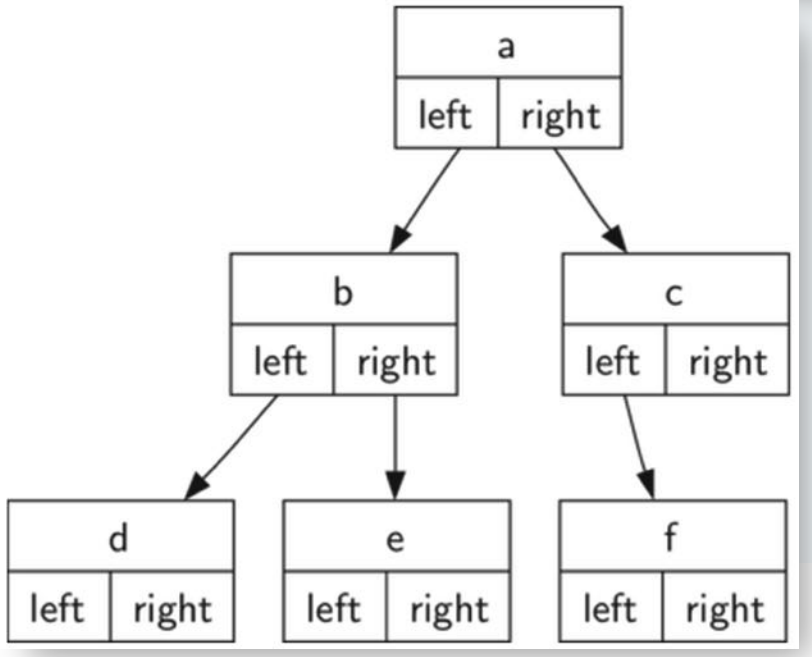
- **树的接口有什么更底层的逻辑，还是约定俗成的？**

  ……如果没有特定的性质，通用树本身的接口用处不是很大，所以就目前的来说主要是构建树结构和有限获取节点功能

树的链接实现

递归数据结构

```
class BinaryTree:
    def __init__(self,rootObj):
        self.key = rootObj
        self.leftChild = None
        self.rightChild = None
```




```
def insertLeft(self,newNode):
    if self.leftChild == None:
        self.leftChild = BinaryTree(newNode)
    else:
        t = BinaryTree(newNode)
        t.leftChild = self.leftChild
        self.leftChild = t

def insertRight(self,newNode):
    if self.rightChild == None:
        self.rightChild = BinaryTree(newNode)
    else:
        t = BinaryTree(newNode)
        t.rightChild = self.rightChild
        self.rightChild = t


def getRightChild(self):
    return self.rightChild

def getLeftChild(self):
    return self.leftChild

def setRootVal(self,obj):
    self.key = obj

def getRootVal(self):
    return self.key
```

节点class


```
class Node():
    def __init__(self, initdata=None):
        self.data = initdata
        self.next = None
        self.prev = None

    def getData(self):
        return self.data

    def getNext(self):
        return self.next

    def getPrev(self):
        return self.prev

    def setData(self, newdata):
        self.data = newdata

    def setNext(self, newnext):
        self.next = newnext

    def setPrev(self, newprev):
        self.prev = newprev
```

问题解答

- 讲解一下二叉堆的原理
- 为什么下沉法选择子节点中较小的正好就可以维护列表的大小顺序？
- 堆排序复杂度O(nlogn)怎么实现的？

二叉堆定义

- **ADT BinaryHeap的操作定义如下：**

  BinaryHeap()：创建一个空二叉堆对象；

  insert(k)：将新key加入到堆中；

  findMin()：返回堆中的最小项，最小项仍保留在堆中；

  delMin()：返回堆中的最小项，同时从堆中删除；

  isEmpty()：返回堆是否为空；

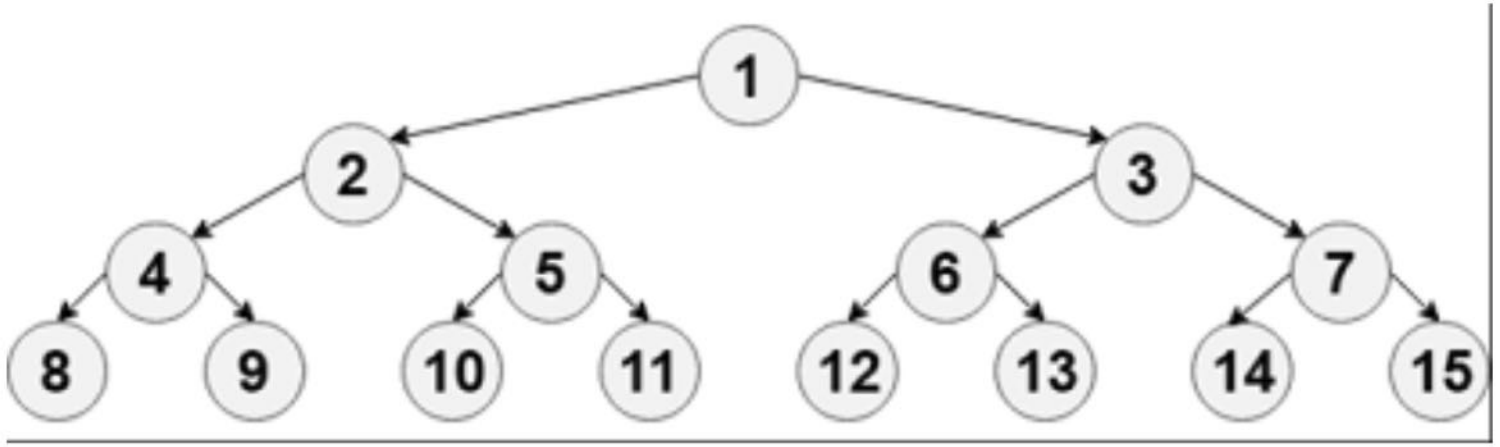
  size()：返回堆中key的个数；

  buildHeap(list)：从一个key列表创建新堆

一种性能很好的实现

- 为了使堆操作能保持在对数水平上，就必须采用二叉树结构；
- 同样，如果要使操作**始终**保持在对数数量级上，就必须始终保持二叉树的“平衡”

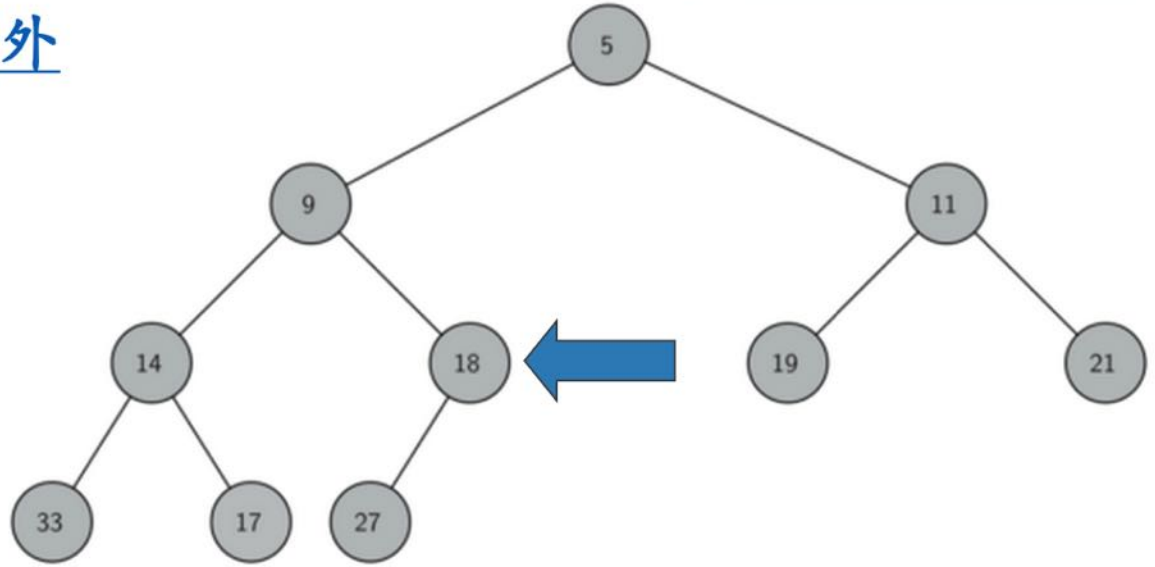
  树根左右子树拥有相同数量的节点



完全二叉树

- 我们采用“**完全二叉树**”的结构来**近似**实现“平衡”

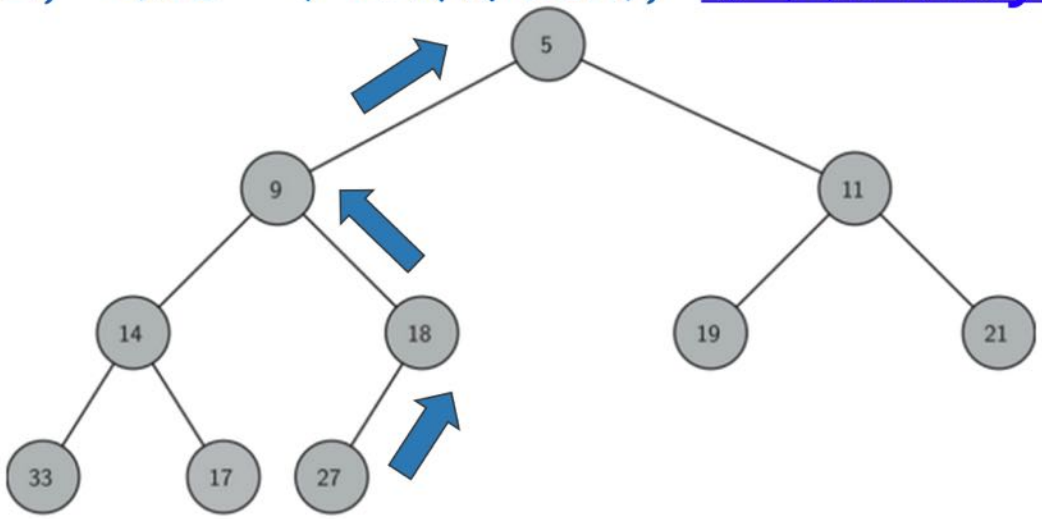
  完全二叉树，叶节点最多只出现在最底层和次底层，而且最底层的叶节点都连续集中在最左边，每个内部节点都有两个子节点，最多可有1个节点例外



堆次序Heap Order

- 任何一个节点x，其父节点p中的key均小于x中的key

这样，符合“堆”性质的二叉树，其中任何一条路径，均是一个已排序数列，根节点的key最小

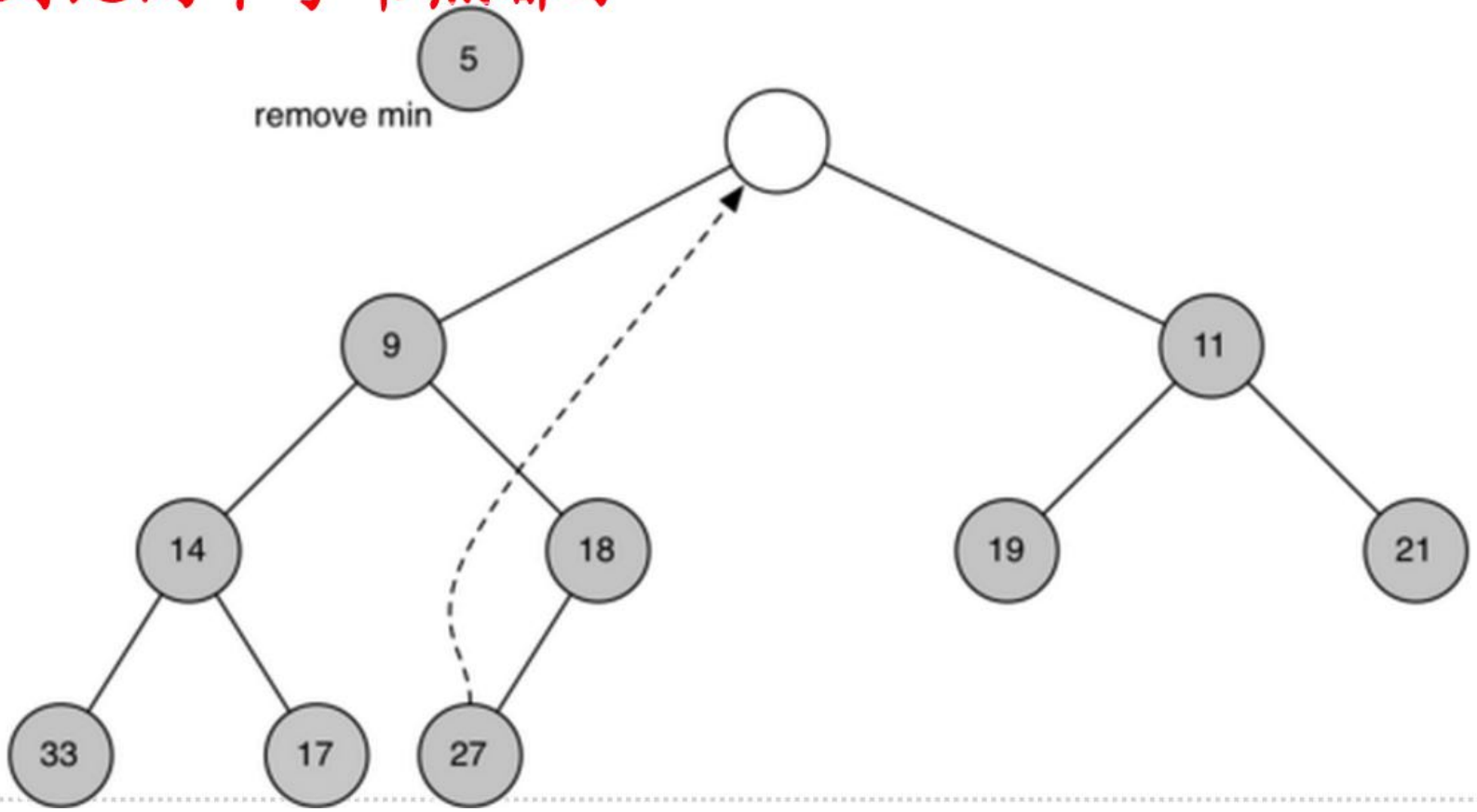


0	5	9	11	14	18	19	21	33	17	27	
0	1	2	3	4	5	6	7	8	9	10	11

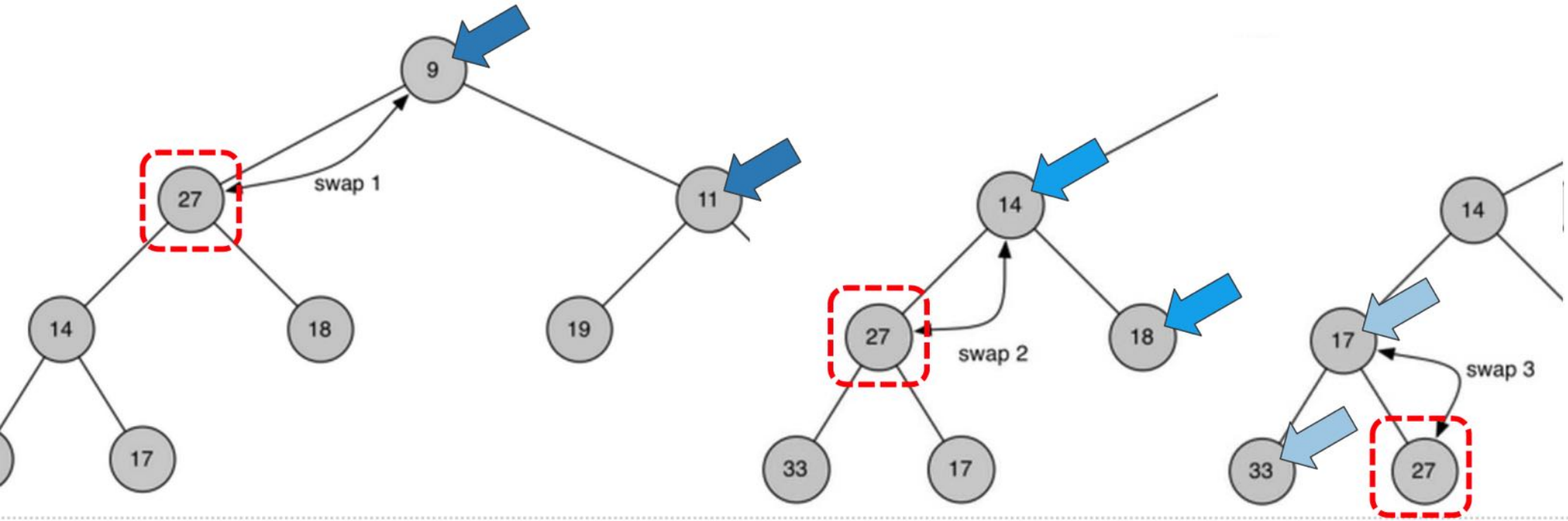
❖ delMin()方法

同样，这么简单的替换，还是破坏了“堆”次序

解决方法：将新的根节点沿着一条路径“下沉”，直到比两个子节点都小



“下沉”路径的选择：如果比子节点大，那么选择较小的子节点交换下沉



堆排序：1-建堆；2-n次delMin

- **buildHeap(lst)方法：从无序表生成“堆”**

  其实，用“下沉”法，能够将总代价控制在O(n)

```
def buildHeap(self,alist):
i = len(alist) // 2
self.currentSize = len(alist)
self.heapList = [0] + alist[:]
print(len(self.heapList), i)
while (i > 0):
    print(self.heapList, i)
    self.percDown(i)
    i = i - 1
print(self.heapList,i)
```

https://zhuanlan.zhihu.com/p/51224401

慕课第九周测验详解

交互程序的两种范式（1）

循环顺序轮询范式

特点：将处理逻辑和交互设备轮询按顺序混合为循环

优点：保持顺序控制流，结构简单，逻辑清晰

缺点：阻塞式顺序同步，交互延迟会随处理增加而上升

适合简单的交互任务

单片机小游戏：反应力升级

- 在指定时间内需要同时按下AB键可以升级
  时间从1000ms开始每级减100ms
- 显示级别（1~10）
- 显示Image.ASLEEP/音乐music.POWER_UP
- 显示3、2、1、Image.TARGET
- 等待指定时间
- 成功按钮则显示Image.HAPPY/音乐music.JUMP_UP
  升级，减时间，10级祝贺音乐，退出游戏
- 失败则显示Image.SKULL/音乐music.WAWAWAWAA
  回到显示级别，重来

单循环的轮询范式


```
1  from microbit import *
2  import music
3
4  level = 0
5  lst = ["3", "2", "1", Image.TARGET]
6  while True:
7      display.scroll(str(level))
8      sleep(100)
9      display.show(Image.ASLEEP)
10     music.play(music.POWER_UP)
11     display.show(lst, delay=200)
12     button_a.was_pressed()  # reset button status
13     button_b.was_pressed()
14     sleep(1000 - level * 100)
15     if button_a.was_pressed() and button_b.was_pressed():
16         display.show(Image.HAPPY)
17         music.play(music.JUMP_UP)
18         level += 1
19         if level >= 10:
20             music.play(music.ENTERTAINER, wait=False, loop=True)
21             display.scroll("MASTER!", loop=True)
22             break
23     else:
24         display.show(Image.SKULL)
25         music.play(music.WAWAWAWAA)
```

交互程序的两种范式（2）

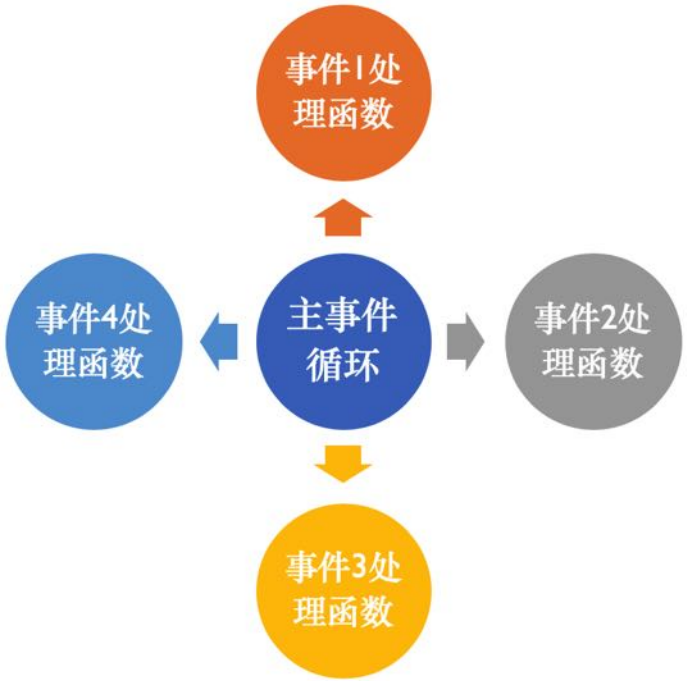
事件驱动范式

特点：将处理逻辑分散到由各种交互事件触发，事件按照发生时间和优先级排队，由事件循环统一处理

优点：交互响应及时，各种处理逻辑异步执行，可以多线索不阻塞

缺点：程序逻辑分散，多线索执行，难以整体理解和维护

适合复杂的交互任务



事件驱动范式理解交互游戏

- **游戏中的图形分为：**

  不动、或者虽然活动但没有影响的背景

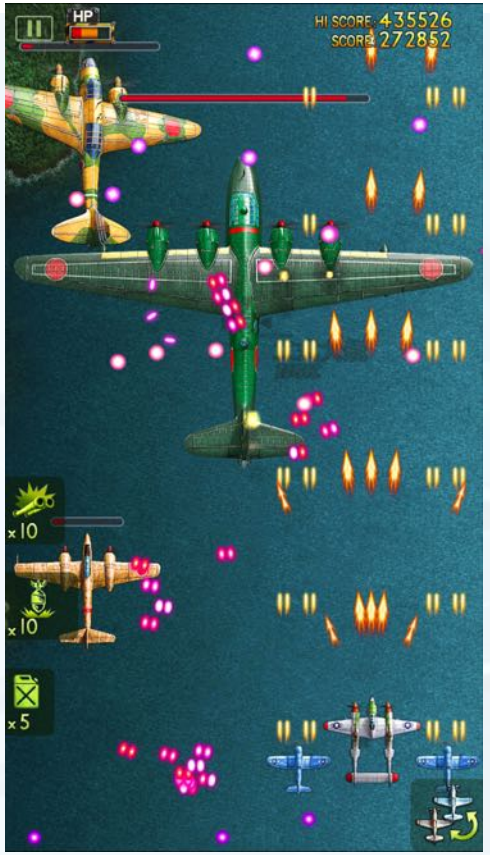
  运动、或者虽然不动但会互相碰撞、互相作用的角色

  经常更新的各种信息显示

- **要让图形动起来，就要不停地做两件事**

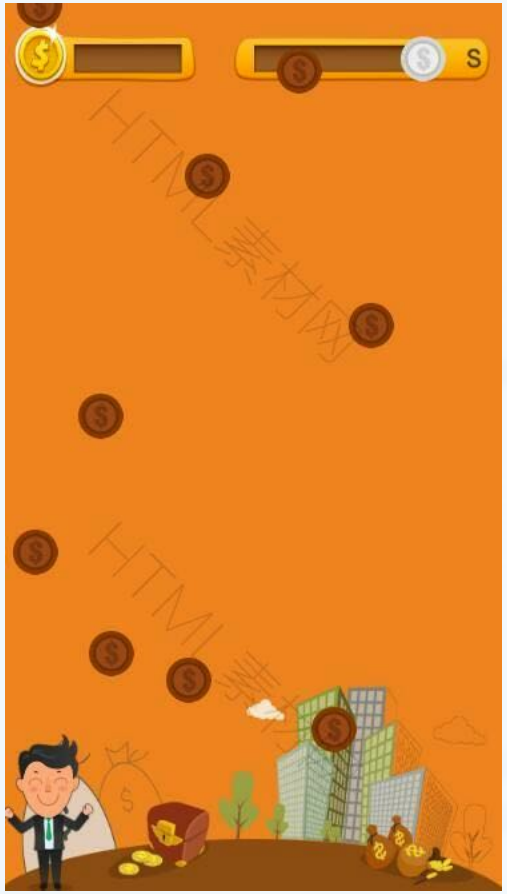
  更新：背景数据、角色位置

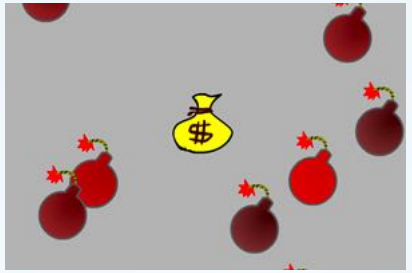
  重画：根据新的数据和位置重新绘制画面



写一个接宝物的游戏

- **角色确定**
  - 哪几种类型角色？
  - 角色的运动？控制？
  - 角色之间的碰撞？
- **画面设计**
  - 背景画面、显示信息
  - 角色的绘制位置
- **交互设计**
  - 鼠标控制？





接宝物游戏程序分析


```python
import pgzrun
import random

# 捡宝物的小游戏
# 宝物角色
# 玩家角色（5种表情）
# 背景：草原（1000*606像素）
# 1，创建一系列精灵
# 名称
names = ["peach", "cherryf", "apple", "orange", "lemon", "grape", "banana",
         "egg", "donut", "taco", "pizza", "burger", "fries", "leg", "dango",
         "cake", "cookie", "ice", "candy",
         "cherry", "bouquet", "leaf4", "sunflower", "tulip", "rose",
         "tiger", "cat", "bear"]

# 保存正在下落的宝物
things = []

# 玩家、位置和得分
player = Actor("face_sweat")
player.center = 500, 580
player.score = 0

# 2，设定窗口
WIDTH = 1000
HEIGHT = 606
```

接宝物游戏程序分析


```
29      # 3，每次需要刷新窗口的时候，会自动调用draw函数
30      def draw():
31          # 清除窗口，设置背景
32          screen.clear()
33          screen.blit("backimg", (0, 0))
34
35          # 画上宝物和玩家
36          for t in things:
37              t.draw()
38          player.draw()
39
40          # 画上分数
41          screen.draw.text("SCORE:%d" % player.score, (400, 10))
```

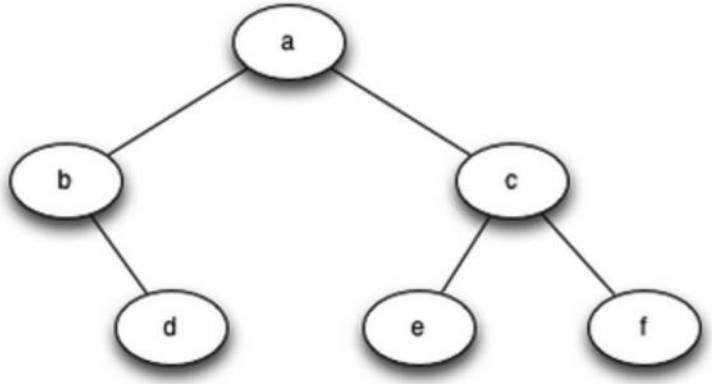
接宝物游戏程序分析


```
# 4，每一帧都会自动调用update函数
def update():
    # 平均每秒随机生成一个宝物加入到things列表里
    if random.randrange(60) == 0:
        # 随机生成宝物，和随机的顶部位置
        t = Actor(random.choice(names))
        t.center = random.randrange(1000), 0
        things.append(t)
    # 每个宝物下降，判断是不是碰到玩家
    for t in things:
        # 下降4个像素，可以调节速度
        t.y += 4
        # 如果调出底线了，就删除，玩家扣分
        if t.y >= 606:
            things.remove(t)
            player.score -= 4
            # 玩家换一个表情
            set_player_sad()
        # 判断是否被玩家接到了
        elif t.colliderect(player):
            # 碰到玩家了，被吃掉，玩家加分
            things.remove(t)
            player.score += 1
            # 玩家换一个表情
            set_player_happy()
```

接宝物游戏程序分析


```
71  # 5，鼠标移动玩家
72  def on_mouse_move(pos):
73      player.x = pos[0]
74
75
76  # 换成欢乐的表情
77  def set_player_happy():
78      player.image = "face_cool"
79      sounds.exp.play()
80      # 设置定时器，1秒后恢复
81      clock.schedule_unique(set_player_normal, 1.0)
82
83  # 换成悲伤表情
84  def set_player_sad():
85      player.image="face_cry"
86      sounds.blip.play()
87      # 设置定时器，1秒后恢复
88      clock.schedule_unique(set_player_normal, 1.0)
89
90  # 换回正常表情
91  def set_player_normal():
92      player.image = "face_sweat"
93
94
95  pgzrun.go()
```

一、写一个buildTree函数（返回一个BinaryTree对象），函数通过调用BinaryTree类方法，返回如图所示的二叉树：



课堂练习
DDL=19点

二、请为链接实现的BinaryTree类写一个__str__方法，把二叉树的内容用**嵌套列表**的方式打印输出。

编写程序：

- 扩展了的BinaryTree类定义，以及buildTree函数的Python代码；
- print(buildTree())

三、请为链接实现的BinaryTree类写一个height方法，返回树的高度。

编写程序：

- BinaryTree类的height方法；
- print(buildTree().height())（能够返回3）