# 算法分析与设计作业2

1.

def f(day):

if day == 10:

return 1

n = f(day + 1)

return n \* 2 + 1

print(f(1))

思路是：

递归函数输入当前的天数，如果day==10，那么就返回1；否则迭代计算下一天会剩多少桃子，直到day=10，然后逐层返回最初的桃子

2.

并归排序算法：  
def GenerateArray(size):  
 array = []  
 for i in range(size):  
 array.append(random.randint(0, 20))  
 print(f'未排序前：', array)  
 return array  
  
  
def Sort\_bottom\_up(arr):  
 size = 2  
 while size <= len(arr):  
 left, right = 0, size - 1  
 while right <= len(arr) - 1:  
 merge(arr, left, right)  
 left += size  
 right += size  
 size \*= 2  
  
  
def merge(arr, left, right):  
 n1 = []  
 n2 = []  
 if right - left == 1:  
 n1.append(arr[left])  
 n2.append(arr[right])  
 else:  
 mid = (left + right - 1) // 2  
 n1 = arr[left:mid + 1]  
 n2 = arr[mid + 1:right + 1]  
 i, j, k = 0, 0, left  
 while i < len(n1) and j < len(n2):  
 if n1[i] <= n2[j]:  
 arr[k] = n1[i]  
 i += 1  
 else:  
 arr[k] = n2[j]  
 j += 1  
 k += 1  
 while i < len(n1):  
 arr[k] = n1[i]  
 i += 1  
 k += 1  
 while j < len(n2):  
 arr[k] = n2[j]  
 j += 1  
 k += 1  
  
  
def main():  
 size = 8  
 array = GenerateArray(size)  
 array2 = array  
 start = time.time()  
 Sort\_bottom\_up(array)  
 print("排序好了：", array)  
 end = time.time()  
 print("所需时间", end - start)  
 start = time.time()  
 array2.sort()  
 end = time.time()  
 print("不使用算法进行排序时间", end - start)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

* 并归排序的执行过程是先两两排序，在四个四个排序，在八个八个排序，自下而上。他的时间复杂度是nlogn，因为并归排序一共有log层，每一层的时间复杂度为n。
* 递归的第一层，将**n**个数划分为**2**个子区间，每个子区间的数字个数为**n/2**；
* 递归的第二层，将**n**个数划分为**4**个子区间，每个子区间的数字个数为**n/4**；
* 递归的第三层，将**n**个数划分为**8**个子区间，每个子区间的数字个数为**n/8**;
* 递归的第**logn**层，将**n**个数划分为**n**个子区间，每个子区间的数字个数为**1**；

快速排序算法：

def generateArray():  
 array = []  
 for i in range(8):  
 array.append(random.randint(0, 100))  
 print(f'未排序的数组： {array}')  
 return array  
  
  
def QuickSort(arr, i, j):  
 if i < j:  
 jizhun = arr[i]  
 left, right = i, j  
 while i < j:  
 # j的停止条件：找到第一个比基准值小的元素  
 while arr[j]>jizhun and i < j:  
 j -= 1  
 # i的停止条件：找到第一个比基准值大的元素  
 while arr[i]<jizhun and i < j:  
 i += 1  
 arr[i], arr[j] = arr[j], arr[i]  
 k = i  
 # 交换基准值与位置为k的值  
 arr[k], jizhun = jizhun, arr[k]  
 # k左边  
 QuickSort(arr, left, k - 1)  
 # k右边  
 QuickSort(arr, k + 1, right)  
  
  
def main():  
 array = generateArray()  
 QuickSort(array, 0, len(array)-1)  
 print(f'排序后的数组 {array}')  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

快速排序的实现步骤：选取一个基准值，一般选择数组的第一个元素。然后设置两个索引i,j，一个指向数第一个元素，一个指向数组的最后一个元素。  
然后i向右移动，找到第一个比基准值大的元素，j向左移动，找到第一个比基准值小的元素，让i,j元素位置交换，  
直到i=j。此时，i和j的位置为k，交换k与基准值的位置。此时，基准值左边的值比基准值小，基准值右边的值比基准值大，但顺序不一定是正确的。然后需要对基准值  
两边的元素做同样的排序，直到每个部分只有一个值。他的时间复杂度在nlogn到n平方之间