

实验二排序算法

2023 年 3 月 9 日

1 实验二 · Lab 2 · 排序算法

1.1 实验目的

通过求解实际排序问题，掌握算法设计和分析的流程和方法。

1.2 实验内容

学校正在选举学生会成员，有 n ($n \leq 999$) 名候选人，每名候选人编号分别从 1 到 n ，现在收集到了 m ($m \leq 2000000$) 张选票，每张选票都写了一个候选人编号。现在想把这些堆积如山的选票按照投票数字从小到大排序。

- 输入格式：输入 n 和 m 以及 m 个选票上的数字。
- 输出格式：求出排序后的选票编号。

输入输出样例：

输入：

5 10

2 5 2 2 5 2 2 2 1 2

输出：

1 2 2 2 2 2 2 5 5

(注意：请用归并排序和插入排序，两者的时间复杂度应分别为 $n \log n$ 和 n^2)

1.3 实验设计思路

由题意知，该问题是一个经典的排序问题。

首先对题目进行抽象，为：给出一个长度为 0-2000000 的列表序列，里面元素为 1-999 的整型随机值。

然后对该列表采用归并和插入排序，使其变成一个递增的序列。

扩展该题，可以得出 1~999 号候选人的具体得票数，以及票数排序。同时尽可能确保列表在排序过程中稳定，不破坏相同元素的先后顺序。

因此我们首先设计整体框架，搭出一个测试框架。具体流程如下：

1. 实现接收候选票函数，可以根据给出的 n, m ，接收一个长度为 m ，随机值为 $0 \sim n$ 整数的序列 $list$ ，作为排序算法应用主体。
2. 分别实现归并排序算法和插入排序算法函数
3. 利用归并和插入函数进行实验

我们可以搭出一个更通用的测试框架。具体流程如下：

1. 实现生成随机候选票函数，可以根据给出的 n, m ，生成一个长度为 m ，随机值为 $0 \sim n$ 整数的序列 $list$ ，作为排序算法应用主体。（务必确保随机性）
2. 利用官方库 `sort` 函数，先对 $list$ 进行排序，得到排序后序列 $list_sorted$ 作为成功组对照。
3. 分别实现归并排序算法和插入排序算法函数
4. 利用归并和插入函数进行实验，并与成功组对照，判断是否正确。即同时完成实验

1.4 伪代码实现

```
[ ]: getVotes (int n , int m):  
    list = OS.in()  
    return list  
  
generateVotes (int n , int m):  
    for 1 to m:  
        list.insert(Random.int(1,n))  
    return list  
  
insertion_sort(A)  
for j = 2 to length[A]  
    key = A[j]  
    i = j - 1  
    while i > 0 and A[i] > key  
        A[i + 1] = A[i]  
        i = i - 1  
    A[i + 1] = key
```

```

merge_sort(A, p, r)
if p < r
    q = (p + r) / 2
    merge_sort(A, p, q)
    merge_sort(A, q + 1, r)
    merge(A, p, q, r)

merge(A, p, q, r)
n1 = q - p + 1
n2 = r - q
let L[1..n1 + 1] and R[1..n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i - 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = infinity
R[n2 + 1] = infinity
i = 1
j = 1
for k = p to r
    if L[i] <= R[j]
        A[k] = L[i]
        i = i + 1
    else
        A[k] = R[j]
        j = j + 1

main:
    n,m = OS.in()
    list = getVotes(n,m)
    --
    list = yourFunction(list) 此处为插入排序或者归并排序函数调用
    --
    print(list)

```

```

main: 通用
    n,m = OS.in()
    list0 = generateVotes(n,m)
    list = list0
    list_sorted = sort(list)
    --
    list = yourFunction(list) 此处为插入排序或者归并排序函数调用
    --
    if list == list_sorted:
        print("success!")
        print(list)

```

1.5 python 代码实现

1.5.1 接收候选票函数实现

```

[1]: def getVotes(n,m): # 其实并没有利用循环来使用 n,m, 因为 python 的方便性
    lst = []
    lst = list(map(int, input("Enter numbers separated by spaces.").split()))
    return lst

```

1.5.2 生成随机候选票函数实现

```

[2]: import random

def generateVotes(n,m):
    lst = []
    i = 1
    while i<=m:
        lst.append(random.randint(1,n))
        i += 1
    return lst

```

1.5.3 归并排序和插入排序函数实现

```
[3]: # @param    arr        需要排序的序列数组
# @param    start    排序序列的初始 index, 一般为 0
# @param    end        排序序列的结尾 index, 注意不是数组长度
def sort_insertion(arr,start,end): # 插入排序算法
    for j in range(start+1,end+1):
        key = arr[j]
        i = j - 1
        while (i >= 0) and (arr[i] > key):
            arr[i+1] = arr[i]
            i -= 1
        arr[i+1] = key

# @param    arr        需要合并的序列数组
# @param    p        合并序列的初始 index, 一般为 0
# @param    q        合并序列的中间 index
# @param    r        合并序列的末尾 index
def merge(arr,p,q,r): # 归并函数 (加入哨兵, 简化代码)
    leftArr = arr[p:q+1]
    rightArr = arr[q+1:r+1]

    leftArr.append(float('inf'))
    rightArr.append(float('inf'))

    i = 0
    j = 0
    for k in range(p,r+1):
        if leftArr[i] <= rightArr[j]:
            arr[k] = leftArr[i]
            i += 1
        else:
            arr[k] = rightArr[j]
            j += 1
```

```

# @param    arr    需要排序的序列数组
# @param    p      排序序列的初始 index, 一般为 0
# @param    r      排序序列的结尾 index, 注意不是数组长度
def sort_merge(arr,p,r):    # 归并排序算法
    if p < r:
        q = (p+r)//2
        sort_merge(arr,p,q)
        sort_merge(arr,q+1,r)
        merge(arr,p,q,r)
    else:
        return

```

1.5.4 主体运行代码

```

[5]: try:
    n, m = map(int, input("Please enter n and m, then press Enter:\n").split())
except ValueError:
    print("Please enter two integers separated by a space.")
    exit()

# list0 = generateVotes(n,m)
list0 = getVotes(n,m)
# 在此可以调整是否是自输入或者生成候选票

for i in list0:
    print(i,end=" ")

lst = list0.copy()

list0.sort()

# sort_insertion(lst,0,len(lst)-1)
sort_merge(lst,0,len(lst)-1)
# 在此可以调整排序函数的执行为哪个算法

print("\n")
if lst == list0:

```

```

# print("success!\n\n")
for i in lst:
    print(i,end=" ")
else:
    print("error")

```

2 5 2 2 5 2 2 2 1 2

1 2 2 2 2 2 2 2 5 5

1.6 代码运行截图

```

1 try:
2     n, m = map(int, input("Please enter n and m, then press Enter:\n").split())
3 except ValueError:
4     print("Please enter two integers separated by a space.")
5     exit()
6
7 # list0 = generateVotes(n,m)
8 list0 = getVotes(n,m)
9 # 在此可以调整是否是自输入或者生成候选票
10
11 for i in list0:
12     print(i,end=" ")
13
14 lst = list0.copy()
15
16 list0.sort()
17
18 # sort_insertion(lst,0,len(lst)-1)
19 sort_merge(lst,0,len(lst)-1)
20 # 在此可以调整排序函数的执行为哪个算法
21
22 print("\n")
23 if lst == list0:
24     # print("success!\n\n")
25     for i in lst:
26         print(i,end=" ")
27 else:
28     print("error")

```

Output: 2 5 2 2 5 2 2 2 1 2

Output: 1 2 2 2 2 2 2 2 5 5

https://github.com/yuxiio/ItA_lab.git 代码仓库地址

1.7 总结

设计算法问题时，应该考虑效率的同时，一定要保证可测试性和通用性。

因此我们可以设计一个通用测试框架，使用官方库运行出一个成功实例来对照。

当然本次实验中我也遇到了很多问题，比如 python 的变量实质其实就是名字标签加内存空间的绑定。因此 list 的复制和操作一定要注意。

同时我还遇到了 TypeError: 'list' object is not callable 的错误。

通过 debug 我了解到了，在 python 中，尽量不要将变量命名为特定包的名字，不然变量名覆盖冲突会导致代码运行错误。这是一个值得记住的编程经验。