实验6 复杂算法及其实现

实验报告

# 1. 实验目的

（1）了解复杂算法的设计；

（2）掌握用 Python语言进行算法实现。

# 2. 实验步骤（详述每一个子实验的关键语法及其实现）

（上机课未成功提交的可另附课下Accepted提交评测序号）

2.1 实验任务6-1

主要语法：

1学会使用列表结构将学生的信息数据进行储存。

2学会采用for循环，在其中采用“a,b,c=input().split()”语句，按行输入每个学生三门课的成绩，按照空格将输入的多个数据分开，分别赋给等号左边的变量

3学会采用列表的append方法，将输入添加到存储所有学生三门课成绩的列表中

4学会定义函数，采用选择排序算法实现多个关键字的排序

代码：

/\*

Author: 杨耕铭

Result: AC Submission\_id: 843749

Created at: Wed May 23 2018 22:52:39 GMT+0800 (CST)

Problem\_id: 1363 Time: 300 Memory: 8936

\*/

n=int(input())

a=list()

b=list()

for i in range(0,n):

y1,y2,y3=input().split(' ')

y1=int(y1)

y2=int(y2)

y3=int(y3)

Y=y1+y2+y3

c=[i+1,Y,y1]

a.append(c)

b.append(a[0])

for j in range(1,n):

d=len(b)

e=0

for k in range(0,d):

if a[j][1]>b[k][1]:

e=1

b.insert(k,a[j])

break

elif a[j][1]==b[k][1]:

if a[j][2]>b[k][2]:

e=1

b.insert(k,a[j])

break

else:

continue

if e==0:

b.append(a[j])

for i in range(0,5):

f=b[i][0]

g=b[i][1]

print(str(f)+' '+str(g))

运行结果：

输入：6

90 67 80

87 66 91

78 89 91

88 99 77

67 89 64

78 89 98

输出：

6 265

4 264

3 258

2 244

1 237

2.2 实验任务6-2

主要语法：

1学会类比解决动态规划问题

2学会使用和编写最长上升子序列 来解决本问题如

2,3,4,7和2,3,4,6就是序列2 5 3 4 1 7 6的两种选取方案。最长的长度是4。

设原序列为a。

当aj<ai(j<i)且numj+1>numi时，numi=numj+1。最后的答案就是numn。

对于每一个数，都是在“可以接下去”的中，从前面的最优值+1转移而来。

代码：

/\*

Author: 杨耕铭

Result: AC Submission\_id: 844357

Created at: Wed May 23 2018 23:43:56 GMT+0800 (CST)

Problem\_id: 1364 Time: 201 Memory: 8500

\*/

N=int(input())

a=input().split()

list1=[]

list2=[]

for i in range(N):

a[i]=int(a[i])

list1.append(0)

list2.append(0)

for i in range(N):

list1[i]=1

for j in range(i):

if a[i]>a[j]:

list1[i]=max(list1[i],list1[j]+1)

for i in range(N):

list2[N-1-i]=1

for j in range(N-i,N):

if a[N-1-i]>a[N-1+N-i-j]:

list2[N-1-i]=max(list2[N-1-i],list2[N-1+N-i-j]+1)

m=1

for i in range(N):

if m<list1[i]+list2[i]:

m=list1[i]+list2[i]

print(N-m+1)

运行结果：

输入：8

186 186 150 200 160 130 197 220

输出；4

2.3 实验任务6-3（本题课后提交，通过序列号为：872875）

主要语法：

1学会使用贪心算法

代码：

n=int(input())

ai=[]

bi=[]

a,b=map(int,input().split())

for i in range(n):

c,d=map(int,input().split())

ai.append(c)

bi.append(d)

for i in range(n):

e=0

for j in range(i+1,n):

if ai[i]\*bi[i]>ai[j]\*bi[j]:

c=ai[i]

ai[i]=ai[j]

ai[j]=c

c=bi[i]

bi[i]=bi[j]

bi[j]=c

e=1

if e==0:

break

m=int(0)

for i in range(n):

if int(a/bi[i])>m:

m=int(a/bi[i])

a=a\*ai[i]

print(m)

运行结果：

输入：3

1 1

2 3

7 4

4 6

输出：2

# 3. 实验总结（实验心得及体会，以及对本课程的建议等）

这次实验相比上次难度再次大大提高，使得我做起来十分吃力。在课堂上虽然大致明白了什么贪心算法和冒泡排序，但在做起题目使用起来仍然不熟练，难以将其转化为python中的程序语句。总的来说希望课程能适当降低点难度，使得我们这些基础差的同学能跟上老师的步伐，将题目得心应手的做出来。

4. 思考题

动态规划和搜索有什么不同点和相同点？

**动态规划**

**1. 动机**：消除递归过程中产生的大量***重叠子问题***

**2.** 多阶段策略问题利用递归的思想, 把规模为n的问题转化为规模为n-1的问题, 直到转化为可以直接求解的原子问题. 一般情况下, 这样的递归方法的时间复杂度是指数级别的, 但是如果所有不同的子问题的数目是多项式级别的, 那么只需要多项式时间就可以解决这个问题, 这就是动态规划的本质.

**3.** **算法四个步骤**：（1）描述最优解结构（2）状态转移方程（3）bottom-up求解（4）构造最优解(最优分割，最优路径)

在动态规划算法中，每步所作的选择往往依赖于相关子问题的解。因而只有在解出相关子问题后，才能作出选择。

而搜索则只是直接将问题输入，在已知的被解决的问题中进行搜索查找，目标明确，不依赖与相关子问题。

相同点则是两者都是为了搜索与查找问题，但搜索与查找的方式不同。