Test 1

* **（1）给出*N*个1-9的数字 (*v*1,*v*2,…,*vN*)，不改变它们的相对位置，在中间加入*K*个乘号和*N-K*-1个加号，（括号随便加）使最终结果尽量大。因为乘号和加号一共就是*N*-1个了，所以恰好每两个相邻数字之间都有一个符号。并说明其具有优化子结构性质及子问题重叠性质。**
  + **例如： *N*=5, *K*=2，5个数字分别为1、2、3、4、5，可以加成：**
  + **1\*2\*(3+4+5)=24**
  + **1\*(2+3)\*(4+5)=45**
  + **(1\*2+3)\*(4+5)=45**

**分析：**

设目标结果值为其中n表示前n个数，c表示前n个数具有的乘号个数。子问题为对1~n的子序列具有最优解。

状态转移方程：

最优子结构性质：

假设不是最大值则存在>则存在>则。则并非最大所以求解最优解包括求解子问题最优解。

**代码：**

from typing import List

def getMax(nums:List[int],C:int)->int:

lenth=len(nums)

dp=[[0]\*(C+1) for i in range(lenth)]

for i in range(lenth):

for j in range(C+1):

if j==0:

dp[i][j]=sum(nums[:i+1])

else:

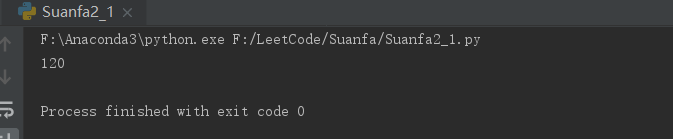
for k in range(j,i+1):

dp[i][j] = max(dp[k-1][j-1]\*sum(nums[k:i+1]), dp[i][j])

return dp[lenth-1][C]

print(getMax([1,2,3,4,5],2))

**实验截图**：



* **给定一长度为N的整数序列(a1,*a*2,…,*aN*) ，将其划分成多个子序列（此问题中子序列是连续的一段整数），满足每个子序列中整数的和不大于一个数B，设计一种划分方法，最小化所有子序列中最大值的和。说明其具有优化子结构及子问题重叠性质**
  + **例如： 序列长度为8的整数序列(2,2,2,8,1,8,2,1)，B=17，可将其划分成三个子序列(2,2,2)，(8,1,8)以及(2,1)，则可满足每个子序列中整数和不大于17，所有子序列中最大值的和12为最终结果。**

**分析：**

要求子序列中最大值和最小，则要尽可能将子序列扩大。设要得到i到j中要得到的最小子序列最大值和为minsmax(i，j)。则可以将此问题分为子问题，minsmax(i，k)+max(k,j) (其中sum(k,j)<B)。可列出状态转移方程

最优子结构和子问题重叠：

由题当求解minsmax(0,n)是需要先求的minsmax(0,k)所以该问题有最优子结构。由题该问题中同一个子序列必然会得到多重利用所以具有子问题重叠。

**代码：**

def minsmax(nums:List[int],B:int)->int:

lenth=len(nums)

if lenth==1:

return nums[0]

sum=0

minmax=float('inf')

for i in range(lenth-1,-1,-1):

sum=sum+nums[i]

if sum<=B:

minmax=min(max(nums[i:])+minsmax(nums[:i],B),minmax)

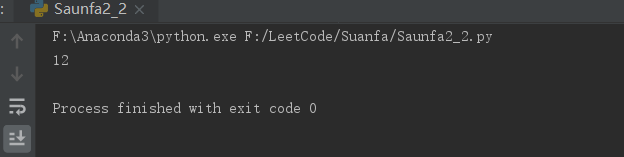
else:

break

return minmax

print(minsmax([2,2,2,8,1,8,2,1],17))

**实验截图：**



* **对一棵树进行着色，每个结点可着黑色或白色，相邻结点不能着相同黑色，但可着相同白色。令树的根为r，请设计一种算法对树中尽量多的节点着黑色。**

（我对题目中相邻节点的理解是两个点有边存在）

由于题目未描述树为二叉树，假设树的储存采用邻接矩阵方法，其中表示i为j父节点。所以根据父节点的颜色可以确定子节点的颜色，当父节点为黑色，则子节点只可为白色，当父节点为白色，子节点有两个选择白或黑。

当算法遍历至节点i时，若i节点涂黑，则遍历所有i的子节点并且设置为白色取。若i涂白，则取。状态转移方程为：

**代码：**

#树涂色

from typing import List

class getmaxblack():

nodes=[]

def \_\_init\_\_(self,nodes:List[List]):

self.nodes=nodes

def init\_situation(self):

initwhite,wlist=self.maxblack(0,0) #根节点为白

initblack,blist=self.maxblack(1,0) #根节点为黑

initblack+=1

blist+=[0] if initblack>initwhite:

return initblack,blist

else:

return initwhite,wlist

def maxblack(self,flag:int,i:int):

maxsum=0

leaveflag=1

blacklist=[]

for k in range(len(self.nodes[i])):

if self.nodes[i][k]==1:

leaveflag=0

if flag==0:

if self.maxblack(0,k)>self.maxblack(1,k):

tmp,tmplist=self.maxblack(0,k)

maxsum+=tmp

blacklist+=tmplist

else:

tmp, tmplist = self.maxblack(1, k)

maxsum += tmp

blacklist += tmplist

else:

tmp, tmplist = self.maxblack(1, k)

maxsum += tmp

blacklist += tmplist

if leaveflag==0:

return maxsum,blacklist

else:

if flag==1:

return 1,[i]

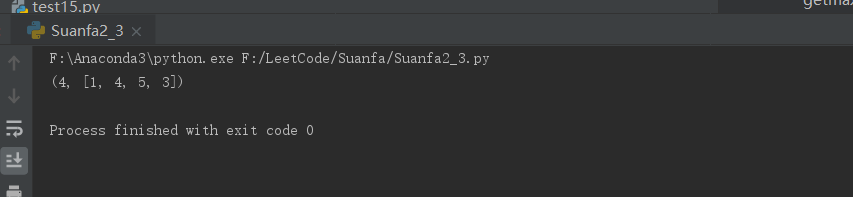
else:

return 0,[]

gb=getmaxblack([[0,1,1,1,0,0],[0,0,0,0,0,0],[0,0,0,0,1,1],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0]])

print(gb.init\_situation())

**实验截图：**



**代表最多四个黑点，其中涂黑的为下标1，4，5，3的节点**

* **（4）在自然语言处理中一个重要的问题是分词，例如句子“他说的确实在理”中“的确”“确实”“实在”“在理”都是常见的词汇，但是计算机必须为给定的句子准确判断出正确分词方法。一个简化的分词问题如下：给定一个长字符串*y*=*y*1*y*2…*yn*，分词是把*y*切分成若干连续部分，每部分都单独成为词汇。我们用函数quality(x)判断切分后的某词汇*x*=*x*1*x*2…*xk*的质量，函数值越高表示该词汇的正确性越高。分词的好坏用所有词汇的质量的和来表示。例如对句子“确实在理”分词，quality(确实) + quality(在理) > quality(确)+quality(实在)+quality(理)。请设计一个动态规划算法对字符串*y*分词，要求最大化所有词汇的质量和。（假定你可以调用quality(x)函数在一步内得到任何长度的词汇的质量）**

**题目分析：**

题目要求获得最大词总权值，类似于第二题，假设i到j下标的字符串总权值最大为。那么可以建立状态转移方程

**代码：**

from typing import List

#分词权值

def quality(String: List):

return 0

def dividewords(String:List):

divided=[]

length=len(String)

if length==1:

divided.append(String)

return quality(String),divided

maxquality=0

for i in range(length-1,-1,-1):

nquality,tmplist=dividewords(String[:i])

if maxquality<nquality+quality(String[i:]):

maxquality=nquality+quality(String[i:])

divided=tmplist.append(String[i:])

return maxquality,divided

**实验截图**：

由于该实验中quality函数未给出，所以未进行确切验证，但是算法实现逻辑应该没有问题

* **（5）给定个活动，活动表示为一个三元组，其中表示活动开始时间，表示活动的结束时间，表示活动的权重。带权活动选择问题是选择一些活动，使得任意被选择的两个活动和执行时间互不相交，即区间与互不重叠，并且被选择的活动的权重和最大。请设计一种方法求解带权活动选择问题。**

**题目分析：**

该题目是一个带权活动安排问题，和以前学习的活动安排问题类似，先将活动按照结束时间排序。遍历整个活动表，保持一个活动的最大权值值maxweight[i]。遍历至第i个活动时，查找它之前可兼容的活动下标记为last。可建立状态转移方程如下：

**代码：**

#带权活动安排

from typing import List

def maxweight(activity:List[List])->int:

nact=sorted(activity,key=lambda x:x[1])

lenth=len(nact)

maxweight=[0]\*lenth

maxweight[0]=nact[0][2]

for i in range(1,lenth):

last=-1

for j in range(0,lenth):

if nact[j][1]>=nact[i][0]:

break

else:

last=j

if last==-1:

maxweight[i]=max(nact[i][2],maxweight[i-1])

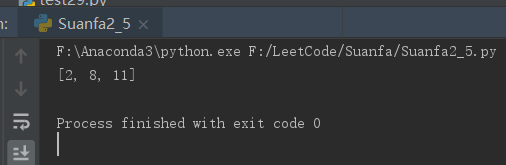
else:

maxweight[i]=max(maxweight[last]+nact[i][2],maxweight[i-1])

return maxweight

print(maxweight([[1,4,2],[7,10,3],[2,6,8]]))

**实验截图：**



权值最大为11