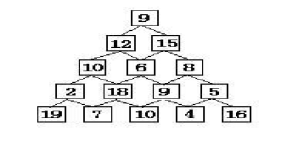
**第1关：数塔问题**

**本关任务：编写用动态规划解决数塔问题。**

**求上图从顶层到顶层的一个路径，使路径上的数字和最大。要求输出最大的数字和max和数值和最大的路径。**

**解题思路：原始信息有层数和数塔中的数据，层数用一个整型变量n存储，数塔中的数据用二维数组data，存储成如下的下三角阵:**

**9**

**12 15**

**10 6 8**

**2 18 9 5**

**19 7 10 4 16**

**必需用二维数组d存储各阶段的决策结果。二维数组d的存储内容如下：**

**d[n][j]=data[n][j]， j=1,2,……,n；**

**d[i][j]=max(d[i+1][j]，d[i+1][j+1])+data[i][j]， i=n-1,n-2,……1，j=1,2,……,i**

**最后d[1][1]存储的就是问题的结果。**

**测试输入：**

**5**

**9**

**12 15**

**10 6 8**

**2 18 9 5**

**19 7 10 4 16**

**输出示例：**

**max=59**

**数值和最大的路径是：9->12->10->18->10**

**#include <stdio.h>**

**#define N 5 //问题规模**

**int main() {**

**int a[50][50];**

**a[1][1] = 9;**

**a[2][1] = 12, a[2][2] = 15;**

**a[3][1] = 10, a[3][2] = 6, a[3][3] = 8;**

**a[4][1] = 2, a[4][2] = 18, a[4][3] = 9, a[4][4] = 5;**

**a[5][1] = 19, a[5][2] = 7, a[5][3] = 10, a[5][4] = 4, a[5][5] = 16;**

**int i, j, dp[50][50] = { 0 }, path[50][50] = { 0 };**

**for (j = 1; j <= N; j++)       //初始子问题 ，倒数第二层（第i-1层）开始**

**dp[N][j] = a[N][j];**

**for (i = N - 1; i >= 1; i--)                       //进行第 i+1 层的决策，从i 到 1 向上**

**for (j = 1; j <= i+1; j++) {                     //每一层有 i+1 个**

**if (dp[i + 1][j] > dp[i + 1][j + 1]) {**

**dp[i][j] = a[i][j] + dp[i + 1][j];**

**path[i][j] = j;                        //本次决策选择下标j的元素**

**}**

**else {**

**dp[i][j] = a[i][j] + dp[i + 1][j + 1];**

**path[i][j] = j + 1;                     //本次决策选择下标j+1的元素**

**}**

**}**

**printf("max=%d\n", dp[1][1]);**

**printf("数值和最大的路径是：");**

**j = path[1][1];                          //计算dp[1][1]的选择**

**for (i = 1; i < N; i++)**

**{**

**printf("%d->", a[i][j]);**

**j = path[i][j];                         //计算dp[i][j]的选择**

**}**

**printf("%d\n", a[i][j]);**

**}**

**/\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*/**

**第2关：最长公共子序列**

**本关任务：编写用动态规划解决最长公共子序列问题。**

**求字符串序列“ABCDBAB”和“BDCABA”的最长公共子序列**

**测试输入：**

**a=“ABCDBAB”**

**b=“BDCABA”**

**输出示例：**

**BCBA**

**#include "stdio.h"**

**#include "string.h"**

**char a[1000]="ABCDBAB";**

**char b[1000]="BDCABA";**

**char str[100];**

**int c[100][100];**

**int lcs\_len()**

**{**

**int m,n,i,j,lcs;**

**m = strlen(a);**

**n = strlen(b);**

**for(i=0;i<=m;i++) c[i][0]=0;**

**for(i=0;i<=n;i++) c[0][i]=0;**

**for(i=1;i<=m;i++)**

**for(j=1;j<=n;j++)**

**{**

**if(a[i-1]==b[j-1])**

**c[i][j] = c[i-1][j-1]+1;**

**else if(c[i-1][j]>=c[i][j-1])**

**c[i][j] = c[i-1][j];**

**else**

**c[i][j] = c[i][j-1];**

**}**

**lcs = c[m][n];**

**return lcs;**

**}**

**void build\_lcs()**

**{**

**int k,i=strlen(a),j=strlen(b);**

**k = lcs\_len();**

**str[k]=' ';**

**while(k>0)**

**if(c[i][j]==c[i-1][j])**

**i=i-1;**

**else if(c[i][j]==c[i][j-1])**

**j=j-1;**

**else**

**{**

**k=k-1;**

**str[k]=a[i-1];**

**j=j-1;**

**}**

**}**

**int main()**

**{**

**build\_lcs();**

**printf("%s",str);**

**return 0;**

**}**

**第3关：求序列-2 11 -4 13 -5 -2的最大子段和**

**本关任务：编写用动态规划解决最大子段和问题。**

**给定由n个整数（可能为负数）组成的序列：a1,a2,……,an, 求该序列的最大子段和。当所有整数均为负数，定义其最大子段和为0。**

**解题思路：**

**定义b[j]=max(a[i]+a[i+1]+…+a[j])，其中1<=i<=j，并且1<=j<=n。那么所求的最大子段和可以表示为max b[j]，1<=j<=n。 由b[j]的定义可知，当b[j−1]>0时b[j]=b[j−1]+a[j]，否则b[j]=a[j]。故b[j]的动态规划递归表达式为: b[j]=max(b[j−1]+a[j],a[j])，1<=j<=n。**

**测试输入：**

**6**

**-2 11 -4 13 -5 -2**

**输出示例：**

**20**

**#include <stdio.h>**

**/\*\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*\*/**

**int main(){**

**int n;**

**scanf("%d",&n);**

**int a[n][2];**

**int max=0;**

**for(int i=0;i<n;i++){**

**scanf("%d",&a[i][0]);**

**if(i==0){**

**a[i][1]=a[i][0];**

**}**

**else{**

**a[i][1]=a[i-1][1]+a[i][0]>a[i][0]?a[i-1][1]+a[i][0]:a[i][0];**

**}**

**max=max>a[i][1]?max:a[i][1];**

**}**

**printf("%d",max);**

**return 0;**

**}**

**/\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*/**

**第4关：求最长的单调递增子序列长度**

**本关任务：编写用动态规划解决求最长的单调递增子序列长度问题。**

**给定一个长度为n的数组，找出一个最长的单调递增子序列（不一定连续，但是顺序不能乱）。例如：给定一个长度为7的数组A5，6，7，1，2，8,9，则其最长的单调递增子序列为5，6，7，8，9，长度为5。求318714101223411624的最长的单调递增子序列长度。**

**解题思路：**

**设长度为n的数组为(a[0]，a[1],a[2],...，a[n−1])，则假定以a[j]结尾的数组序列的最长递增子序列长度为L(j)，则L(j)=max(L(i))+1,i<j且a[i]<a[j]。也就是说，我们需要遍历在j之前的所有位置i(从0到j−1)，找出满足条件a[i]<a[j]的L(i)，求出max(L(i))+1即为L(j)的值。最后，我们遍历所有的L(j)（从0到n−1），找出最大值即为最大递增子序列。**

**测试输入：**

**10**

**3 18 7 14 10 12 23 41 16 24**

**输出示例：**

**6**

**#include <stdio.h>**

**/\*\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*\*/**

**int main(){**

**int n;**

**scanf("%d",&n);**

**int m[n][3];**

**m[0][1]=1;**

**m[0][2]=0;**

**for(int i=0;i<n;i++){**

**scanf("%d",&m[i][0]);**

**if(i!=0){**

**m[i][1]=0;**

**int k=i-1;**

**while(k>=0){**

**if(m[i][0]>m[k][0]){**

**if(k==i-1){**

**m[i][1]=m[k][1]+1;**

**m[i][2]=k;**

**}**

**else{**

**int max=m[k][1]+1;**

**if(max>m[i][1]){**

**m[i][1]=max;**

**m[i][2]=k;**

**}**

**}**

**}**

**k--;**

**}**

**if(k<0&&m[i][1]==0){**

**m[i][1]=1;**

**m[i][2]=i;**

**}**

**}**

**}**

**int max=m[0][1],j=0;**

**for(int i=0;i<n;i++){**

**if(m[i][1]>=max){**

**max=m[i][1];**

**j=i;**

**}**

**}**

**printf("%d\n",max);**

**}**

**/\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*/**

**第5关：矩阵连乘问题**

**将矩阵连乘积AiAi+1…Aj简记为A[i:j]，其中i<=j。设在矩阵Ak和Ak+1之间将矩阵链断开，则其相应加括号为(AiAi+1…Ak) (Ak+1Ak+2…Aj)。A[i:j]的计算量等于三部分计算量之和： （1）A[i:k]的计算量， （2）A[k+1:j]的计算量， （3）A[i:k]与A[k+1:j]相乘的计算量。 设计算A[i:j]所需最少乘积数目为，则原问题的最优值为。 当i=j时,a[i:j]=Ai​，因此，m[i][j]=0,i=1,⋅⋅⋅,n 当i<j时，m[i][j]=i<k<jmin​{m[i][k]+m[k+1][j]+pi−1​pk​pj​} 其中，矩阵Ai​的矩阵数为pi−1​×pi​ 矩阵A1​的维度：p0​p1​=3035 矩阵A2​的维度：p1​p2​=3515 矩阵A3​的维度：p2​p3​=155 矩阵A4​的维度：p3​p4​=510 矩阵A5​的维度：p4​p5​=1020 矩阵A6​的维度：p5​p6​=2025 求这6个矩阵连乘的最小相乘次数。**

**测试输入：**

**6**

**30 35**

**35 15**

**15 5**

**5 10**

**10 20**

**20 25**

**输出示例：**

**m[1][6]=15125**

**#include <stdio.h>**

**#include <stdlib.h>**

**/\*\*\*\*\*\*\*\*\*\* Begin \*\*\*\*\*\*\*\*\*\*/**

**int main(){**

**int n;**

**scanf("%d",&n);**

**int a[n][2];**

**int b[n][n]={0};**

**for(int i=0;i<n;i++){**

**scanf("%d %d",&a[i][0],&a[i][1]);**

**}**

**for(int i=1;i<n;i++){**

**for(int j=0;j<n-i;j++){**

**b[j][j+i]=b[j][j]+b[j+1][j+i]+a[j][0]\*a[j][1]\*a[j+i][1];**

**int k=j+1;**

**for(;k<j+i;k++){**

**int t=b[j][k]+b[k+1][j+i]+a[j][0]\*a[k][1]\*a[j+i][1];**

**if(t<b[j][j+i]) {**

**b[j][j+i]=t;**

**}**

**}**

**}**

**}**

**printf("m[%d][%d]=%d",1,n,b[0][n-1]);**

**return 0;**

**}**

**/\*\*\*\*\*\*\*\*\*\* End \*\*\*\*\*\*\*\*\*\*/**