动态规划经典问题

(2012-03-16 10:30:18) 转 载▼

标签： 动态规划 c it 分类： 2012.3

1、三角数塔问题
设有一个三角形的数塔，顶点为根结点，每个结点有一个整数值。从顶点出发，可以向左走或向右走，如图所

示：

要求从根结点开始，请找出一条路径，使路径之和最大，只要输出路径的和。
【代码】

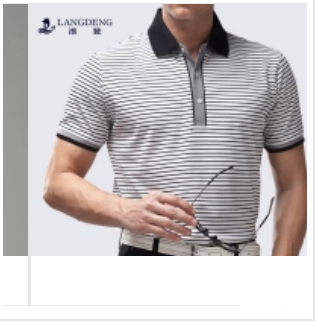
```
//
//                例题1 三角数字塔问题                          //
//
#include <stdio.h>
#include <stdlib.h>

#define MAXN 101

int n,d[MAXN][MAXN];
int a[MAXN][MAXN];

void fnRecursive(int,int);
//递推方法函数声明
int fnMemorySearch(int,int);
//记忆化搜索函数声明

int main()
{
    int i,j;
    printf("输入三角形的行数n(n=1-100):\n");
    scanf("%d",&n);
    printf("按行输入数字三角形上的数(1-100):\n");
```






```
    for(i=1; i<=n; i++)
        for(j=1; j<=i; j++)
            scanf("%d",&a[i][j]);
    for(i=1; i<=n; i++)
        for(j=1; j<=i; j++)
            d[i][j]=-1;//初始化指标数组
    printf("递推方法：1\n记忆化搜索方法：2\n");
    int select;
    scanf("%d",&select);
    if(select==1)
    {
        fnRecursive(i,j);//调用递推方法
        printf("\n%d\n",d[1][1]);
    }
    if(select==2)
    {
        printf("\n%d\n",fnMemorySearch(1,1));//调用记忆化搜索方法
    }
    else
        printf("输入错误！");

        return 0;
}

void fnRecursive(int i,int j)
//递推方法实现过程
{
    for(j=1; j<=n; j++)
        d[n][j]=a[n][j];
    for(i=n-1; i>=1; i--)
        for(j=1; j<=i; j++)
            d[i][j]=a[i][j]+(d[i+1][j]>d[i+1][j+1]?d[i+1][j]:d[i+1][j+1]);
}

int fnMemorySearch(int i,int j)
//记忆化搜索实现过程
{
    if(d[i][j]>=0) return d[i][j];
    if(i==n) return(d[i][j]=a[i][j]);
    if(fnMemorySearch(i+1,j)>fnMemorySearch(i+1,j+1))
        return(d[i][j]=(a[i][j]+fnMemorySearch(i+1,j)));
    else
        return(d[i][j]=(a[i][j]+fnMemorySearch(i+1,j+1)));
}
```

2、硬币问题

问题描述：有n种硬币，面值分别为V1,V2,…,Vn, 每种有无限多。给定非负整数S，可以选用多少硬币，使得面值之和恰好为S，输出硬币数目的最小值和最大值。（1<=n<=100,0<=S<=10000,1<=Vi<=S）

【代码】

```
//
//            例题1 硬币问题                          //
//
#include <stdio.h>
#include <stdlib.h>

#define INF 100000000
#define MAXNUM 10000
#define MONEYKIND 100

int n,S;
int V[MONEYKIND];
int min[MAXNUM],max[MAXNUM];

void dp(int*,int*);
//递推方法函数声明
```

```
void print_ans(int*,int);
//输出函数声明

int main()
{
    int i;
    printf("输入硬币的种数n(1-100):\n");
    scanf("%d",&n);
    printf("输入要组合的钱数S(0-10000):\n");
    scanf("%d",&S);
    printf("输入硬币种类：\n");
    for(i=1; i<=n; i++)
    {
        scanf("%d",&V[i]);
    }
    dp(min,max);
    printf("最小组合方案：\n");
    print_ans(min,S);
    printf("\n");
    printf("最大组合方案：\n");
    print_ans(max,S);

    return 0;
}

void dp(int *min,int *max)
//递推过程实现
{
    int i,j;
    min[0] = max[0] = 0;
    for(i=1; i<=S; i++)//初始化数组
    {
        min[i]=INF;
        max[i]=-INF;
    }
    for(i=1; i<=S; i++)
        for(j=1; j<=n; j++)
            if(i>=V[j])
            {
                if(min[i-V[j]]+1<min[i]){
                    min[i]=min[i-V[j]]+1;//最小组合过程
                    //printf("%d\n",min[i]);
                }
                if(max[i-V[j]]+1>max[i])
                    max[i]=max[i-V[j]]+1;//最大组合过程
            }
}

void print_ans(int *d,int S)
//输出函数实现
{
    int i;
    for(i=1; i<=n; i++)
        if(S>=V[i]&&d[S]==d[S-V[i]]+1)
        {
            printf("%d-",V[i]);
            print_ans(d,S-V[i]);
            break;
        }
}
```

3、矩形嵌套问题

问题描述：有n个矩形，每个矩形可以用两个整数a,b描述，表示它的长和宽。矩形X(a,b)可以嵌套在矩形Y(c,d)中，当且仅当a<c,b<d或者b<c,a<d（相当于把矩形X旋转90度）。例如(1,5)可以嵌套在(6,2)内，但不能嵌套在(3,4)内。你的任务是选出尽量多的矩形排成一行，使得除了最后一个之外，每一个矩形都可以嵌套在下一个矩

形内。

【代码】

```
//
//              例题2 矩形嵌套问题                          //
//
#include <stdio.h>
#include <stdlib.h>

#define MAXNUM 100//矩形的最大个数
struct Rect
{
    int x;
    int y;
};

int n;//矩形个数
struct Rect rect[MAXNUM];
int G[MAXNUM][MAXNUM];//邻接有向图
int d[MAXNUM];//过程数组

int dp(int i,int G[MAXNUM][MAXNUM]);

int main()
{
    int i,j,z;
    printf("输入矩形个数n(1-100):\n");
    scanf("%d",&n);
    printf("输入矩形的长和宽：\n");
    for(i=1; i<=n; i++)
    {
        scanf("%d",&rect[i].x);
        scanf("%d",&rect[i].y);
    }
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
        {
            if(rect[i].x<rect[j].x&&rect[i].y<rect[j].y)
                G[i][j]=1;
        }
    printf("输入要开始的矩形z：\n");
    scanf("%d",&z);
    //printf("%d-",z);
    int temp = dp(z,G);
    printf("最大可嵌套个数：%d\n",temp);
    return 0;
}

int dp(int i,int G[MAXNUM][MAXNUM])
{
    int j;
    if(d[i]>0)
        return d[i];
    d[i]=1;
    for(j=1; j<=n; j++)
        if(G[i][j])
            if(dp(j,G)+1>d[i])
            {
                d[i]=dp(j,G)+1;
            }
    return d[i];
}
```

4、求一组数列的最长的不下降序列。

问题描述：设有一个正整数序列b1，b2，b3，……，bn，若下标为i1<i2<i3，……<ik且有bi1≤bi2≤……bik则

称存在一个长度为k的不下降序列。可能有多个不下降序列，输出最长序列的长度。
样例输入：13 7 9 16 38 24 37 18
样例输出：5 (7 9 16 24 37)
【代码】

```
//
//              例题2 数组的最长不下降序列问题          //
//
#include <stdio.h>
#include <stdlib.h>

#define MAXNUM 100//最大数字个数

int b[MAXNUM][2];
int n;
int len;

int main()
{
    int i,j;
    printf("输入数列中数字的个数n(1-100)：\n");
    scanf("%d",&n);
    printf("输入数字：\n");
    for(i=1; i<=n; i++)
    {
        scanf("%d",&b[i][0]);
        b[i][1]=1;
    }
    for(i=2; i<=n; i++)
    {
        len = 0;
        for(j=1; j<i; j++)
            if((b[i][0]>=b[j][0])&&(b[j][1]>len))
                len=b[j][1];
        if(len>0)
            b[i][1]=len+1;
    }
    len = 1;
    for(j=2; j<=n; j++)
        if(b[j][1]>len)
            len = b[j][1];
    printf("最长为：%d\n",len);
    return 0;
}
```

5、0-1背包问题
给定n种物品和一背包。物品i的重量是wi，其价值为vi，背包的容量为C。问应如何选择装入背包的物品，使得装入背包中物品的总价值最大?
【代码】

```
//
//              0-1背包问题                              //
//
#include <stdio.h>
#include <stdlib.h>

#define MAXN 100//物品种类最大数量

int w[MAXN],V[MAXN];
int C;//最大容量
int n;//物品种类
int d[MAXN][MAXN];
int jMax;

int min(int,int);
//两数之间的最小值
```

```
int max(int,int);
//两数之间的最大值
void print_ans(int d[][MAXN],int,int);
//构造最优解并输出

int main()
{
    int i,j;
    printf("输入物品种类数n(1-100):\n");
    scanf("%d",&n);
    printf("输入每个种类的重量：\n");
    for(i=1; i<=n; i++)
    {
        scanf("%d",&w[i]);
    }
    printf("输入每个种类的价值：\n");
    for(i=1; i<=n; i++)
    {
        scanf("%d",&V[i]);
    }
    printf("输入背包的容量C：\n");
    scanf("%d",&C);


    jMax=min(C,w[n]-1);
    for(j=0; j<=jMax; j++)
        d[n][j]=0;
    for(j=w[n]; j<=C; j++)
        d[n][j]=V[n];

    for(i=n-1; i>1; i--)
    {
        jMax=min(C,w[i]-1);
        for(j=0; j<=jMax; j++)
            d[i][j]=d[i+1][j];
        for(j=w[i]; j<=C; j++)
            d[i][j]=max(d[i+1][j],d[i+1][j-w[i]]+V[i]);
    }
    d[1][C]=d[2][C];
    if(C>=w[i])
        d[1][C]=max(d[1][C],d[2][C-w[i]]+V[1]);



    printf("最大可容纳：%d\n",d[1][C]);
    return 0;
}

int min(int x,int y)
{
    if(x>y)
        return y;
    else
        return x;
}

int max(int x,int y)
{
    if(x<y)
        return y;
    else
        return x;
}
```

6、数字游戏问题
小W发明了一个游戏，他在黑板上写出了一行数字a1，a2，a3，……，an，然后给你M个回合的机会，每回合你可以从中选择一个数字擦去它，接着剩下来的每个数字ai都要递减一个值bi。如此重复m个回合，所有你擦去的数字之和就是你所得的分数。
编程算算，对于每个a和b序列，可以得到的最大得分是多少。
输入样例：
```
3               {数字个数n}
3               {回合数m}
10 20 30        {n个原始序列}
4 5 6           {n个每回合每个数字递减的值}
```
输出样例：
```
47
```
【代码】
```
//
//                练习一 数字游戏                          //
//
#include <stdio.h>
#include <stdlib.h>

#define MAXNUM 100

int n;//数字个数
int m;//回合数
int Num[MAXNUM];//原始数据数组
int DescNum[MAXNUM];//递减数据数组
int F[MAXNUM][MAXNUM];//过程数组

void swap(int ,int );
//交换两个整数
int max(int ,int );
//取两个中的最大值

int main()
{
    int i,j;
    printf("输入数字的个数n(1-100)：\n");
    scanf("%d",&n);
    printf("输入回合数m(1-100)<=n：\n");
    scanf("%d",&m);
    printf("输入n个原始序列：\n");
    for(i=1; i<=n; i++)
    {
        scanf("%d",&Num[i]);
    }
    printf("输入n个递减的数字：\n");
    for(i=1; i<=n; i++)
    {
        scanf("%d",&DescNum[i]);
    }


    for(i=1; i<=n; i++)
    {
        for(j=i+1; j<=n; j++)
        {
            if(DescNum[i]<DescNum[j])
            {
                swap(Num[i],Num[j]);
                swap(DescNum[i],DescNum[j]);
            }
        }
    }
    for(i=1; i<=n; i++)
```

```
    {
        F[i-1][0]=0;
        for(j=1; j<=m; j++)
            F[i][j]=max(F[i-1][j],F[i-1][j-1]+Num[i]-DescNum[i]*(j-1));
    }


    printf("得分：%d\n",F[n][m]+DescNum[1]);
    return 0;
}

void swap(int x,int y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}

int max(int x,int y)
{
    if(x<y)
        return y;
    else
        return x;
}
```

7、挖地雷

在一个地图上有N个地窖（N<=200）,每个地窖中埋有一定数量的地雷。同时，给出地窖之间的连接路径，并规定路径都是单向的。某人可以从任一处开始挖地雷，然后沿着指出的连接往下挖（仅能选择一条路径），当无连接时挖地雷工作结束。设计一个挖地雷的方案，使他能挖到最多的地雷。

【代码】

```
//
//                    挖雷问题                              //
//

#include <stdio.h>
#include <stdlib.h>

#define MAXNUM 200

int n;//地窖的个数
int w[MAXNUM];//每个地窖中的地雷数
int Sum[MAXNUM];//挖到的地雷总数
int G[MAXNUM][MAXNUM];//形成的图
int next[MAXNUM];//记录路径
int max;//最大值
int start;

//void init(int G[MAXNUM][MAXNUM]);
//void init2(int n[MAXNUM]);

int main()
{
    int i,j,x,y;
    printf("输入地窖的个数n(1-200)：\n");
    scanf("%d",&n);
    printf("输入各个地窖中的地雷数：\n");
    for(i=1;i<=n;i++)
    {
        scanf("%d",&w[i]);
    }
    //init(G);//初始化有向图
    printf("输入各个地窖之间的链接(x,y):\n");
```

```
    scanf("%d,%d",&x,&y);
    while(x!=0&&y!=0)
    {
        G[x][y]=1;
        scanf("%d,%d",&x,&y);
    }
    //init2(Sum);

    Sum[n]=w[n];
    for(i=n-1;i>=1;i--)
    {
        for(j=i+1;j<=n;j++)
        {
            if(G[i][j]&&Sum[j]>Sum[i])
            {
                Sum[i]=Sum[j];
                next[i]=j;
            }
        }
        Sum[i] += w[i];
    }
    max = 0;
    for(i=1;i<=n;i++)
    {
        if(Sum[i]>max)
        {
            max=Sum[i];
            start = i;
        }
    }


    printf("最大路径：");
    printf("%d-",start);
    while(next[start]!=0)
    {
        printf("%d-",next[start]);
        start = next[start];
    }
    printf("最大挖雷数：%d\n",max);
    return 0;
}
```

8、最小代价子母树

设有n堆沙子排成一排，其编号为1，2，3，…，n（n≤100）。每堆沙子有一定的数量，如下表

13 7 8 16 21 4 18

现在要将n堆沙子归并成一堆。归并的过程为每次只能将相邻的两堆沙子堆成一堆，这样经过n-1次归并之后最后成为一堆，如上面7堆沙子，可以有多种方法归并成一堆，归并的代价是这样定义的，将两堆沙子归并为一堆时，两堆沙子数量的和称为归并2堆沙子的代价。由此可见，不同归并过程得到的总的归并代价是不一样的。

问题：n堆沙子的数量给出之后，找出一种合理的归并方法，使总的归并代价为最小。

[输入格式]

n {表示沙子的堆数, 2<=n<=100}

a1 a2 … an {表示每堆沙子的数量,1<=Ai<=100}

[输出格式]

x {表示最小的归并总代价 }

输入样例：

7

13 7 8 16 21 4 18

输出样例：

239

【代码】

```
//
//                  练习3 最小代价子母树                    //
//                   沙堆归并问题                           //
```

```
//
#include <STDIO.H>
#include <STDLIB.H>

#define INF 1000000
#define MAXNUM 100

int n;//沙子的堆数
int Num[MAXNUM];//每堆沙子的数量
int F[MAXNUM][MAXNUM];//过程函数
int G[MAXNUM][MAXNUM];

int fnMin(int ,int );
//返回两个数的最小值

int main()
{
    int i,j,k,m,L,t;
    printf("输入沙堆的堆数n(1-100)：\n");
    scanf("%d",&n);
    printf("输入每堆中沙子的个数：\n");
    for(i=1; i<=n; i++)
    {
        scanf("%d",&Num[i]);
        G[i][i]=Num[i];
        F[i][i]=0;
    }
    for (m=n-1;m>=1;m--)
    {
        for (i=1;i<=m;i++)
        {
            L=n-m+1;
            j=i+L-1;
            for (k=i;k<=j;k++)
            {
                G[i][j]=G[i][j]+G[k][k];
            }
            F[i][j]=INF;
            for (k=i;k<=j-1;k++)
            {
                t=F[i][k]+F[k+1][j]+G[i][j];
                if (t<F[i][j])
                {
                    F[i][j]=t;
                }
            }
        }
    }
    printf("最小代价：%d\n",F[1][n]);
    return 0;
}
```

9、数字移动问题

给出一个1到n的排列，每次可以移动一个数到一个任意位置。问要达到状态1,2,3……n至少移动多少次？

Sample Input

5

2 1 4 5 3

Sample Output

2

【代码】

```
//
//                    练习4 数字移动                                //
//
#include <stdio.h>
```

```
#include <stdlib.h>

#define MAXNUM 100//最大数字个数
#define INF 100000000

int n;//数字个数
int Num[MAXNUM];//未排序数字
int Last[MAXNUM];

void update(int x,int y,int L[],int N[],int i);
//递归函数

int main()
{
    int i,j;
    int lis;
    printf("输入数字的个数n(1-100)：\n");
    scanf("%d",&n);
    printf("输入数字：\n");
    for(i=1;i<=n;i++)
    {
        scanf("%d",&Num[i]);
    }

    lis = 0;
    Last[lis]=-INF;
    for(i=1;i<=n;i++)
    {
        if(Num[i]>Last[lis])
        {
            lis++;
            Last[lis]=Num[i];
        }
        else

        update(0,lis,Last,Num,i);
    }


    printf("最少移动次数：%d\n",n-lis);
    return 0;
}

void update(int x,int y,int L[],int N[],int i)
{
    if(x==y)
    {
        L[x]=N[i];
    }
    else
    {
        if(L[(x+y)/2]>N[i])
        {
            update(x,(x+y)/2,L,N,i);
        }
        else
        {
            update((x+y)/2+1,y,L,N,i);
        }
    }
}
```

10、叠放箱子问题

某港口有一批箱子，将其编号，分别为1至N。每一个箱子的尺寸规格是一样的，现在要将其中某些箱子叠放起来，箱子叠放的规则如下：

一、每个箱子上最多只能直接叠放一个箱子；

二、编号较小的箱子不能放在编号较大的箱子之上；

三、每个箱子都给出了自身重量与可承受重量，每个箱子之上的所有箱子重量之和不得超过该箱的可承受重量。

为了节约堆放场地，希望你编程从中选出最多个箱子，使之能够在满足条件的情况下叠放起来。

【输入】

第一行是一个整数N（1≤N≤1000）。

以下共有N行，每行两个整数，中间以空格分隔，分别表示每个箱子的自身重量与可承受重量，两个数值均为小于等于3000的正整数。

【输出】

第一行应当输出最多可叠放的箱子总数M。

【样例】有五个箱子，如下表：

1 19 15
2 7 13
3 5 7
4 6 8
5 1 2

则最多可以叠放4个箱子，方案之一如：1、2、3、5

【代码】

```
//
//                  叠放箱子问题                        //
//
#include <stdio.h>
#include <stdlib.h>

#define MAXNUM 1000//最大的箱子个数
#define MAXWIGHT 3000//
int n;//箱子数目
int F[MAXNUM+1][6000];//第i个箱子到第n个箱子中总重量为j的最大箱子数
int Weight[MAXNUM];//箱子重量
int Capacity[MAXNUM];//箱子所能承受的压力

int max(int ,int );
//返回两个数的最大值

int main()
{
    int i,j,ans;
    printf("输入箱子的个数n(1-1000):\n");;
    scanf("%d",&n);
    printf("分别输入箱子的重量和能承受的重量(x y):\n");
    for(i=1; i<=n; i++)
        scanf("%d %d",&Weight[i],&Capacity[i]);

    F[n+1][0] = 0;
    for(i=n; i>=1; i--)
    {
        for(j=0; j<=2*MAXWIGHT; j++)
        {
            F[i][j]=F[i+1][j];
            if(j>=Weight[i]&&Capacity[i]>=j-Weight[i])
            {
                F[i][j]=max(F[i][j],F[i+1][j-Weight[i]]+1);
            }
        }
    }
    ans=0;
    for(i=0; i<=6000; i++)
        ans = max(ans,F[1][i]);
    printf("最大叠放：%d",ans);

    return 0;
}
```

```
int max(int x,int y)
{
    if(x>y)
        return x;
    else
        return y;
}
```

8 0

喜欢 赠金笔

分享：

前一篇：Cholesky分解
后一篇：关于孙鑫VC++深入详解中遇到的两个问题

〈 前一篇
Cholesky分解

后一篇 〉
关于孙鑫VC++深入详解中遇到的两个问题