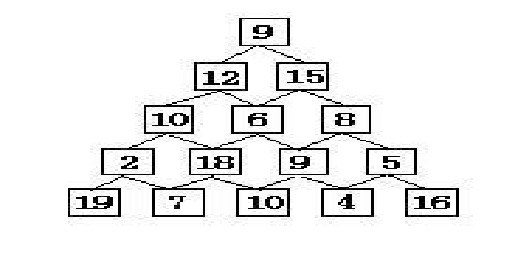
**[动态规划“数塔”类型题目总结](http://www.cnblogs.com/DiaoCow/archive/2010/04/18/1714859.html)**

前几天做了好几个DP题目，感觉都是一个类型的，因此有必要总结一下。



数塔问题 ：要求从顶层走到底层，若每一步只能走到相邻的结点，则经过的结点的数字之和最大是多少？

分析：站在位置9，我们可以选择沿12方向移动，也可以选择沿着15方向移动，现在我们假设“已经求的”沿12方向的最大值x和沿15方向的最大值y，那么站在9的最大值必然是：Max(x,y) + 9。

因此不难得出，对于任意节点i,其状态转移方程为：m[i] = Max(a[i的左孩子] , a[i的右孩子]) + a[i]

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include <stdio.h>  
  
#define N 10000  
#define Max(a,b) ((a) > (b) ? (a) : (b))  
  
int a[N];   
   
int main(void)  
{  
 int n , m , i , k , j;  
  
 scanf("%d",&m);  
 while(m-- > 0)  
 {  
 scanf("%d",&n);  
 k = (1 + n) \* n / 2;  
 for(i = 1 ; i <= k; i++)   
 {  
 scanf("%d",a+i);  
 }  
  
 k = k - n;  
 for(i = k , j = 0 ; i >= 1 ; i--)  
 {  
 a[i] = a[i] + Max(a[i+n],a[i+n-1]);  
 if(++j == n -1)  
 {  
 n--;  
 j = 0;  
 }  
 }  
 printf("%d\n",a[1]);  
   
 }  
   
 return 0;  
}

首先什么是“数塔类型”？从某一点转向另一点或者说是从某一状态转向另一状态，有多种选择方式（比如这里的9->12 , 9->15），从中选取一条能产生最优值的路径。

这类问题的思考方法：假设后续步骤的结果已知，比如这里假设已经知道沿12方向的最大值x和沿15方向的最大值y。

接下来看几个题，加深印象吧

1.免费馅饼问题

                     5 (起始位置）

     4       |      5       |       6

3   4   5  |  4   5   6  |  5   6   7

..................

和“数塔”一样，它也是从某一点出发，有多个选择的问题（往前走一步，呆在原地，往后走一步）从中选择一条最优值路径（获得馅饼最多）。还是按照“数塔”的思考方式，我们可以假设“已经求得”下一个站在位置4获得的最大值x和呆在原地获得的最大值y以及站在位置6获得的最大值z,那么对于起始位置5获得最大值就是Max(x,y,z) ，因此可以得到状态转移方程为：m[t][x] = Max(m[t+1][x-1] , m[t+1][x] , m[t+1][x+1])

并且我们可以通过“列表格”的方式，自底向上求解：

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include <stdio.h>  
#include <string.h>  
  
#define N 100000  
int a[N][11];

int Max(int a , int b , int c)  
{  
 int n;  
  
 n = a > b ? a : b;  
  
 return n > c ? n : c;  
}  
  
  
int main(void)  
{  
 int n , x , t , max , i;  
  
   
 while(scanf("%d",&n))  
 {  
 if(!n) break;  
  
 max = 0;  
 memset(a , 0 , sizeof(a));  
 for(i = 0 ; i < n ; i++)   
 {  
 scanf("%d%d",&x,&t);  
 a[t][x] += 1;  
 if(t > max) max = t;  
 }  
   
  
 //DP  
 for(t = max - 1 ; t >= 0 ; t--)  
 {  
 a[t][0] += Max(0 , a[t + 1][0] , a[t + 1][1]) ;  
   
 for(x = 1 ; x < 10 ; x++)  
 {  
 a[t][x] += Max(a[t + 1][x - 1] , a[t + 1][x] , a[t + 1][x + 1]) ;  
 }  
 a[t][10] += Max(a[t + 1][9] , a[t + 1][10] , 0) ;  
 }  
 printf("%d\n",a[0][5]);   
 }  
   
 return 0;  
}

2.滑雪问题

      上

左   A    右

　　下

依然和“数塔”一样，从某一点出发，面临多个选择（往上，往左，往下，往右）从中选择一条最优值路径（滑雪距离最长）

若对A点求，很显然它的最大值就为： Max(上，右，下，左) + 1

因此对于任意位置[i,j], 其状态转移方程为：m[i][j] = Max(m[i-1][j] , m[i][j+1] , m[i+1][j] , m[i][j-1]) + 1

由于这道题很难画出它的路径图（起点和终点都不知道）因此很难用“列表格”的方式自底向上求解，因此我们采用备忘录法：

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include <stdio.h>  
#include <string.h>  
#define N 101  
  
int a[N][N] , m[N][N] , r , c;

int OK(int i ,int j)  
{  
 return (i >= 1 && i <= r && j >= 1 && j <= c);  
}  
  
int search(int i , int j)  
{  
 int k;  
  
 if(m[i][j] > 0) return m[i][j];  
  
 if(OK(i - 1, j) && a[i][j] > a[i - 1][j])  
 {  
 k = search(i - 1, j) + 1;  
 m[i][j] = m[i][j] < k ? k : m[i][j];  
 }  
 if(OK(i, j + 1) && a[i][j] > a[i][j + 1])  
 {  
 k = search(i, j + 1) + 1;  
 m[i][j] = m[i][j] < k ? k : m[i][j];  
 }  
 if(OK(i + 1, j) && a[i][j] > a[i + 1][j])  
 {  
 k = search(i + 1, j) + 1;  
 m[i][j] = m[i][j] < k ? k : m[i][j];  
 }  
 if(OK(i, j - 1) && a[i][j] > a[i][j - 1])  
 {  
 k = search(i, j - 1) + 1;  
 m[i][j] = m[i][j] < k ? k : m[i][j];  
 }  
  
 return m[i][j] ;  
}  
  
int main(void)  
{  
 int i , j , k , t;  
  
 while(scanf("%d%d", &r, &c) != EOF)  
 {  
 for(i = 1 ; i <= r ; i++)  
 for(j = 1 ; j <= c ; j++)  
 scanf("%d", &a[i][j]);  
  
 memset(m, 0, sizeof(m)); k = 0;  
 for(i = 1 ; i <= r ; i++)  
 for(j = 1 ; j <= c ; j++)  
 {  
 t = search(i, j);  
 k = k < t ? t : k;  
 }  
  
 printf("%d\n", k + 1);  
  
 }  
 return 0;  
}

3.Worm问题，这题和免费馅饼几乎是一样的，我们同样可以使用“列表格”的方式自底向上求解：

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttp://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

#include <stdio.h>  
#include <string.h>  
#define N 100  
  
int a[N][N];  
  
int main(void)  
{  
 int t , x , n , p , m , T;  
  
 while(scanf("%d%d%d%d", &n, &p, &m, &T) != EOF) //苹果树n,毛毛虫其实位置p,m分钟，终点位置T  
 {   
 memset(a, 0, sizeof(a));  
 a[m][T] = 1;  
 for(t = m - 1; t >= 0 ; t--)  
 {  
 a[t][1] += a[t + 1][2];  
 for(x = 2 ; x < n ; x++)  
 a[t][x] += a[t + 1][x - 1] + a[t + 1][x + 1];  
 a[t][n] += a[t + 1][n - 1];  
 }  
 printf("%d\n", a[0][p]);  
 }  
 return 0;  
}