动态规划

W.J.Z

1 动态规划的特点

1. 如果问题的解是最优解，那么该问题的子问题的解也是最优解.
2. 解问题的过程可以分为若干个阶段，第 $i+1$ 个阶段的的行为仅依赖于第 i 个阶段状态，与第 i 个阶段之前的行为过程都无关。

2 动态规划解题的一般思路

1. 将原问题分解为若干个子问题，每个子问题和原问题形式相同或类似，每个子问题的解只计算一次且被保存，所有子问题的解可构成原问题的解。
2. 确定状态，整个问题的时间复杂度是状态数目乘以计算每个状态所需的时间。
3. 确定初始状态的值
4. 确定状态转移方程

从大脑神经角度来看，整个问题相当于一个神经系统，每个神经元相当于一个状态对应的子问题，状态对神经元进行刺激，神经元根据状态转移方程产生结果，在某种程度上和人工神经网络相似。

3 动态规划距离

3.1 问题描述

从下面的半三角矩阵中从顶部至底部寻找一条路径使得该路径上的和最大，每一步只能向下或向右下部行走。

3
2 4
2 3 4
1 4 5 6
2 3 4 1 3

3.2 分解为子问题

可以将问题分为五个阶段，每个阶段对应矩阵的一行，令 $Date(i,j)$ 代表矩阵第 i 行第 j 列的数据，$maxSum(i,j)$ 代表 $D(i,j)$ 到第一层的最长路径的数值和 (也可以到最后一层，称呼和算法不同为前向算法和后向算法)，只要保证每个阶段取得最优解，则可保证最后问题的解也是最优的。

3.3 确定初始状态的值

$Date(1,1)=3$，。

3.4 确定状态转移方程

$$maxSum(i,j)=\begin{cases}Date(i,j) & i=1\\ MAX\{maxSum(i-1,j),maxSum(i-1,j-1)\}+Date(i,j) & \text{其他}\end{cases}$$

3.5 C++ 代码

为防止数组越界下标都是从一开始的，maxSum 数组初始值全部为 0.

```
#include<iostream>
#include<algorithm>
using namespace std;
#define MAX 100
int Date[MAX][MAX];
int row;
int maxSum[MAX][MAX]

int main()
{
    int i,j;
    cin >> row;
    for(i=1;i<=row;i++)
        for(j=1;j<=i;j++)
        cin >> Date[i][j];
    maxSum[1][1]=Date[1][1];
    for(i=2;i<=row;i++)
        for(j=1;j<=row;j++)
        maxSum[i][j]=max(maxSum[i-1][j],maxSum[i-1][j-1])+Date[i][j];
    int result = maxSum[row][1];
    for(i=2;i<=row;i++)
        result = max(result,maxSum[row][i]);
    cout << result;
}
```