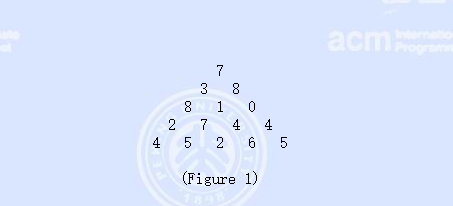
# 实验二：动态规划

**一、实验目的**

理解动态规划的基本思想，理解动态规划算法的两个基本要素最优子结构性质和子问题的重叠性质。熟练掌握典型的动态规划问题。掌握动态规划思想分析问题的一般方法，对较简单的问题能正确分析，设计出动态规划算法，并能快速编程实现。

**二、实验内容**



在上面的数字三角形中寻找一条从顶部到底边的路径，使得路径上所经过的数字之和最大。路径上的每一步都只能往左下或 右下走。只需要求出这个最大和即可，不必给出具体路径。 三角形的行数大于1小于等于100，数字为0~99

输入格式：

5 //表示三角形的行数 接下来输入三角形

7

3 8

8 1 0

2 7 4 4

4 5 2 6 5

要求输出最大和

1. **动态规划基本思想**

**自底向上dp**

**设dp[i][j]表示从第i行第j列出发能得到的最大数字和**

**初始条件:最后一行dp[i][j]=a[i][j]**

**转移方程:** **dp[i][j] = max(dp[i+1][j], dp[i+1][j+1])+a[i][j]**

1. **实验过程**

#include <bits/stdc++.h>

using namespace std;

signed main() {

  int n;

  cin >> n;

  vector<vector<int>> a(n), dp(n);

  for (int i = 0; i < n; ++i) {

    a[i].resize(i+1);

    dp[i].resize(i+1);

    for (int j = 0; j <= i; ++j) {

      cin >> a[i][j];

      dp[i][j] = a[i][j];

    }

  }

  for (int i = n-2; i >= 0; --i) {

    for (int j = 0; j <= i; ++j) {

      dp[i][j] = max(dp[i+1][j], dp[i+1][j+1])+a[i][j];

    }

  }

  cout << dp[0][0] << '\n';

  return 0;

}

1. **实验结果**

