稀疏矩阵

【问题描述】

大部分元素是 0 的矩阵称为稀疏矩阵。假设有k个非 0 元素，就可以把稀疏矩阵用$k \times 3$的矩阵进行记录。其中第一列是行号，第二列是列号，第三列是该行、该列下的非零元素的值。如：

$$\begin{matrix} 0 & 0 & 0 & 5 \\ 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

简记成：

$$\begin{matrix} 1 & 4 & 5 & \text{//第 1 行第 4 列有个数是 5} \\ 2 & 2 & 2 & \text{//第 2 行第 2 列有个数是 2} \\ 3 & 2 & 1 & \text{//第 3 行第 2 列有个数是 1} \end{matrix}$$

读入一个 M 行 N 列的稀疏矩阵（最大 5 行 5 列，最多 10 个非零元素），输出简记形式。

【分析】

查找非零元素并记忆位置。将原始矩阵存于数组 a。转换后的矩阵存于数组 b。

【输入样例】

```
3 5
0 0 0 0 5
0 0 4 0 0
1 0 0 0 1
```

【输出样例】

```
1 5 5
2 3 4
3 1 1
3 5 1
```

【参考程序】

```
#include<iostream>
using namespace std;
const int M = 5;
const int N = 5;
```

```
const int K = 10;
int main()
{
    //输入
    int a[M][N];
    int m, n; //m 行 n 列
    cin >> m >> n;
    for (int i=0; i<m; i++)
    {
        for (int j=0; j<n; j++)
        {
            cin >> a[i][j];
        }
    }
    //生成稀疏矩阵表示法
    int b[K][3];
    int k = 0;
    for (int i=0; i<m; i++)
    {
        for (int j=0; j<n; j++)
        {
            //找到非零值，存储
            if (a[i][j]!=0)
            {
                b[k][0] = i+1;
                b[k][1] = j+1;
                b[k][2] = a[i][j];
                k++;
            }
        }
    }
    //输出
    for (int i=0; i<k; i++)
    {
        for (int j=0; j<3; j++)
        {
            cout << b[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

杨辉三角形

【问题描述】

打印杨辉三角形的前 n 行（最大 8 行）。

数字的输出宽度为 4。

两个数字之间有 4 个空格。

【输入样例】

```
5
```

【输出样例】

```
                1
            1       1
        1       2       1
    1       3       3       1
1       4       6       4       1
```

【分析】

杨辉三角形如下图：

```
                1
            1       1
        1       2       1
    1       3       3       1
1       4       6       4       1
```

把上图转化为下图，可以发现杨辉三角形其实就是一个二维表的下三角形部分。

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
```

使用二维数组 a 存储：

- 每行首尾元素为 1，
- 任意非首尾元素$a[i][j] = a[i-1][j-1] + a[i-1][j]$，
- 每一行的元素个数刚好等于行数。

有了数组元素的值，只需要控制好输出起始位置，就可以打印杨辉三角形了。

【参考程序】

```
#include<iostream>
#include<iomanip>
using namespace std;
const int N = 8;
int a[N][N];
int main()
{
    int n;
    cin >> n;
    //设定第一行的值
    a[0][0] = 1;
    //从第二行开始推
    for (int i = 1; i < n; i++)
    {
        //设定每一行的首尾值为 1
        a[i][0] = 1;
        a[i][i] = 1;
        //当前行非首尾的数
        for (int j = 1; j <= i - 1; j++)
        {
            //每个数等于上一行的两个数之和
            a[i][j] = a[i - 1][j - 1] + a[i - 1][j];
        }
    }
    for (int i = 0; i < n; i++)
    {
        //控制每行的起始位置
        for (int j = 0; j < n - i - 1; j++)
        {
            cout << "    ";
        }
        for (int j = 0; j <= i; j++)
        {
            cout << setw(4) << a[i][j] << "    ";
        }
        cout << endl;
    }
    return 0;
}
```