算法作业7-4

2154312 郑博远

- 该问题在一维上即最大子段和问题，对于$b[j]=\max\limits_{1\le i\le j}\sum\limits_{k=i}^{j}a[k]$，有动态规划递归式$b[j]=\max b[j-1]+a[j],a[j]$。可以根据该式求出一维上的最大子段和问题；
- 二维上，需要遍历横向上线段的每一种子线段，通过求和的方式将其塌缩为一维上的一个值。这些值在纵向上便形成了一维的最大子段和问题，可以调用maxSum解决；
- 三维上类似的调用二维的maxSum2即可。

```
#include <iostream>
#include <fstream>
using namespace std;

// 求最长字段和
int maxSum(int n, int* a)
{
    int sum = 0, b = 0;
    for (int i = 1; i ≤ n; i++) {
        if (b > 0)
            b += a[i];
        else
            b = a[i];
        if (b > sum)
            sum = b;
    }
    return sum;
}

// 求平面最大子矩形
int maxSum2(int m, int n, int** a)
{
    int sum = 0;
    int* b = new int[n + 1];
    for (int i = 1; i ≤ m; i++) {
        for (int k = 1; k ≤ n; k++)
            b[k] = 0;
        // i与j之间的部分被累加到b[]上
        for (int j = i; j ≤ m; j++) {
            // 这里的k相当于一维上的i
            for (int k = 1; k ≤ n; k++)
                b[k] += a[j][k];
            int max = maxSum(n, b);
            if (max > sum)
```

```
                sum = max;
        }
    }
    delete[] b;
    return sum;
}

// 求长方形最大子矩形
int maxSum3(int p, int m, int n, int*** a)
{
    int sum = 0;

    int** b = new int*[m + 1];
    for (int i = 1; i ≤ m; i++)
        b[i] = new int[n + 1];

    for (int i = 1; i ≤ p; i++) {
        for(int k = 1; k ≤ m; k++)
            for (int l = 1; l ≤ n; l++)
                b[k][l] = 0;
        for (int j = i; j ≤ p; j++) {
            for (int k = 1; k ≤ m; k++)
                for (int l = 1; l ≤ n; l++)
                    b[k][l] += a[j][k][l];
            int max = maxSum2(m, n, b);
            if (max > sum)
                sum = max;
        }
    }

    for (int i = 1; i ≤ m; i++)
        delete[] b[i];
    delete[] b;

    return sum;
}

int main()
{
    ifstream infile;
    infile.open("./input.txt", ios::in);
    ofstream outfile;
    outfile.open("./out.txt", ios::out);

    if (!infile.is_open() || !outfile.is_open())
        return 0;

    int p, m, n;
    infile >> p >> m >> n;
    int*** a = new int** [p + 1];
    for (int i = 1; i ≤ p; i++) {
        a[i] = new int* [m + 1];
        for (int j = 1; j ≤ m; j++)
            a[i][j] = new int[n + 1];
```

```
    }

    for (int i = 1; i ≤ p; i++)
        for (int j = 1; j ≤ m; j++)
            for (int k = 1; k ≤ m; k++)
                infile >> a[i][j][k];

    outfile << maxSum3(p, m, n, a);


    for (int i = 1; i ≤ p; i++) {
        for (int j = 1; j ≤ m; j++)
            delete[] a[i][j];
        delete[] a[i];
    }
    delete[] a;

    infile.close();
    outfile.close();

    return 0;
}
```