一、题目说明

题目64. Minimum Path Sum，给一个m*n矩阵，每个元素的值非负，计算从左上角到右下角的最小路径和。难度是Medium！

二、我的解答

乍一看，这个是计算最短路径的，迪杰斯特拉或者弗洛伊德算法都可以。不用这么复杂，同上一个题目一样：

刷题62. Unique Paths()

不多啰嗦，直接代码，注释中有原理：

```
#include<iostream>
#include<vector>
using namespace std;
class Solution{
    public:
        int minPathSum(vector<vector<int>>& grid){
            int m = grid.size();
            if(m<1){
                return 0;
            }

            int n = grid[0].size();
            vector<vector<int>> dp(m,vector<int>(n,0));

            dp[0][0] = grid[0][0];
            //初始化第1行
            for(int i=1;i<n;i++){
                dp[0][i] = dp[0][i-1]+grid[0][i];
            }
            //初始化第1列
            for(int i=1;i<m;i++){
                dp[i][0] = dp[i-1][0]+grid[i][0];
            }

            for(int i=1;i<n;i++){//计算第i列
                for(int j=1;j<m;j++){//计算第j行
                    if(dp[j-1][i]>dp[j][i-1]){
                        dp[j][i] = dp[j][i-1]+grid[j][i];
                    }else{
                        dp[j][i] = dp[j-1][i]+grid[j][i];
                    }

                }
            }

            return dp[m-1][n-1];
        }
};
int main(){
    Solution s;
    vector<vector<int>> grid;
    grid = {{1,3,1},{1,5,1},{4,2,1}};
    cout<<"7=="<<s.minPathSum(grid)<<"\n";

    grid = {{0,1},{1,0}};
    cout<<"1=="<<s.minPathSum(grid)<<"\n";

    grid = {{1,2,5},{3,2,1}};
    cout<<"6=="<<s.minPathSum(grid)<<"\n";

    return 0;
}
```

性能，第一次提交16ms，一行代码没修改再次提交12ms：

```
Runtime: 12 ms, faster than 49.38% of C++ online submissions for Minimum Path
Sum.
Memory Usage: 10.9 MB, less than 50.00% of C++ online submissions for Minimum
Path Sum.
```

三、优化措施

本来想用迪杰斯特拉算法写的，也不废这个劲了。