输入一个存储着整数的二维数组grid，如果grid[i][j] > 0，说明这个格子装着血瓶，经过它可以增加对应的生命值；如果grid[i][j] == 0，则这是一个空格子，经过它不会发生任何事情；如果grid[i][j] < 0，说明这个格子有怪物，经过它会损失对应的生命值。
现在你是一名骑士，将会出现在最上角，公主被困在最右下角，你只能向右和向下移动，请问骑士的初始生命值至少为多少，才能成功救出公主?

分析：

1. 该题求解的为初始能量为多少？且状态转移方程已给定只能向下或向右；
2. **对初始值的推定，一般采用反向遍历的方式。**
3. 对本题需要从row-1,col-1向上倒推；
4. base case的确定可以看出，对<row-1, col-1>其值为dungeon[row-1][col-1]的判定，如果dungeon[row-1][col-1] > 0 时，到达该点其值为1即可，如果dungeon[row-1][col-1] <= 0 到达该点时最小值应该为-dungeon[row-1][col-1]+1
5. 所以base的确定应该为[row-1][..]与[..][col-1], 对尾行和尾列而言，其值可以化为int t = dp[row-1][i+1] - dungeon[row-1][i],如果t <= 0 那么到达该点最小值为1,t > 0时到达该点的值应该为t.
6. 对dp数组的计算：int t = std::min(dp[i + 1][j], dp[i][j + 1]) - dungeon[i][j], 但其值不能直接等于t, 需要判断：t 是否大于 0；如果> 0 表示到达该点至少需要t的能量，如果<=0 表示到达该点至少需要1的能量。
7. 结果存储在dp[0][0]中。

```
class Solution {
public:
  int calculateMinimumHP(std::vector<std::vector<int>>& dungeon) {
    int row = dungeon.size(), col = 0;
    if (row != 0) {
      col = dungeon[0].size();
    } else {
      return 1;
    }

    std::vector<std::vector<int>> dp =
        std::vector<std::vector<int>>(row + 1, std::vector<int>(col + 1,
0));

    dp[row - 1][col - 1] =
        dungeon[row - 1][col - 1] <= 0 ? -dungeon[row - 1][col - 1] + 1 :
1;

    // base case
    for (int i = row - 2; i >= 0; i--) {
      int tmp        = dp[i + 1][col - 1] - dungeon[i][col - 1];
      dp[i][col - 1] = tmp <= 0 ? 1 : tmp;
    }

    for (int i = col - 2; i >= 0; i--) {
      int tmp        = dp[row - 1][i + 1] - dungeon[row - 1][i];
```

```
        dp[row - 1][i] = tmp <= 0 ? 1 : tmp;
      }

      // dp 数据计算
      for (int i = row - 2; i >= 0; i--) {
        for (int j = col - 2; j >= 0; j--) {
          int tmp  = std::min(dp[i + 1][j], dp[i][j + 1]) - dungeon[i][j];
          dp[i][j] = tmp <= 0 ? 1 : tmp;
        }
      }

      return dp[0][0];
    }
};
```