417. Pacific Atlantic Water Flow

题目描述：https://leetcode.com/problems/pacific-atlantic-water-flow/

> 太平洋和大西洋中间有洋流，太平洋用~代表，大西洋用*代表，水流可以从数字小的流向数字大的，求问能同时流到大西洋和太平洋的水流坐标集合。
> 例如：

```
Given the following 5x5 matrix:
Pacific ~   ~   ~   ~   ~
   ~  1   2   2   3  (5) *
   ~  3   2   3  (4) (4) *
   ~  2   4  (5)  3   1  *
   ~ (6) (7)  1   4   5  *
   ~ (5)  1   1   2   4  *
      *   *   *   *   * Atlantic
   Return:
   [[0, 4], [1, 3], [1, 4], [2, 2], [3, 0], [3, 1], [4, 0]]
   (positions with parentheses in above matrix).
```

解题思路：

BFS，用一个数组存着大西洋可以留到的位置，一个数组存储太平洋可以留到的位置，求两个交集。

代码：

```
class Solution {
public:
    void bfs(queue<pair<int, int> > &q, vector<vector<bool> > &visited, vector<vec
tor<int> > matrix)
    {
        while(!q.empty()) {
            pair<int, int> p = q.front();
            int i = p.first, j = p.second;
            q.pop();
            visited[p.first][p.second] = true;
            if(i+1 < matrix.size() && matrix[i+1][j] >= matrix[i][j] && !visited[i
+1][j]){
                q.push(pair<int, int>(i+1, j));
            }
            if(i-1 >= 0 && matrix[i-1][j] >= matrix[i][j] && !visited[i-1][j]){
                q.push(pair<int, int>(i-1, j));
            }
```

```
            if(j+1 < matrix[0].size() && matrix[i][j+1] >= matrix[i][j] && !visite
d[i][j+1]){
                q.push(pair<int, int>(i, j+1));
            }
            if(j-1 >= 0 && matrix[i][j-1] >= matrix[i][j] && !visited[i][j-1]){
                q.push(pair<int, int>(i, j-1));
            }
        }
        return;
    }
    vector<pair<int, int>> pacificAtlantic(vector<vector<int>>& matrix) {
        int m = matrix.size();
        queue<pair<int, int> > q;
        vector<pair<int, int> > res;
        if(m == 0) return {};
        int n = matrix[0].size();
        vector<vector<bool> > visited1(m, vector<bool>(n, false));
        vector<vector<bool> > visited2(m, vector<bool>(n, false));
        for(int i = 0; i < n; i++) {
            q.push(pair<int, int>(0, i));
        }
        for(int i = 0; i < m; i++) {
            q.push(pair<int, int>(i, 0));
        }
        bfs(q, visited1, matrix);
        for(int i = 0; i < n; i++) {
            q.push(pair<int, int>(m-1, i));
        }
        for(int i = 0; i < m; i++) {
            q.push(pair<int, int>(i, n-1));
        }
        bfs(q, visited2, matrix);
        for(int i = 0; i < m; i++) {
            for(int j = 0; j < n; j++) {
                if(visited1[i][j]&&visited2[i][j]) {
                    res.push_back(pair<int, int>(i, j));
                }
            }
        }
        return res;
    }

};
```