```cpp
#include "../bits/stdc++.h"
// verified: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3385218
// 複数 hash の方がいいかも
template <typename T>
class RollingHash2D
{
    int h, w;
    static const uint64_t b1 = 9973, b2 = 100000007; // (1000000007, 1000000009)
    std::vector<uint64_t> rh, rw;
    std::vector<std::vector<uint64_t>> hash;

  public:
    RollingHash2D(const std::vector<std::vector<T>> &mat) : h(mat.size()), w(mat[0].size()), rh(h + 1), rw(w + 1), hash(h + 1, std::vector<uint64_t>(w + 1))
    {
        for (int i = 0; i < h; i++)
        {
            for (int j = 0; j < w; j++)
            {
                hash[i + 1][j + 1] = hash[i + 1][j] * b2 + mat[i][j];
            }
            for (int j = 0; j < w; j++)
            {
                hash[i + 1][j + 1] += hash[i][j + 1] * b1;
            }
        }
        rh[0] = rw[0] = 1;
        for (int i = 0; i < h; i++)
            rh[i + 1] = rh[i] * b1;
        for (int i = 0; i < w; i++)
            rw[i + 1] = rw[i] * b2;
    }
    // [sx, gx) * [sy, gy) のハッシュ値を取得
    uint64_t query(int sx, int sy, int gx, int gy)
    {
        assert(0 <= sx && sx <= gx && gx <= h);
        assert(0 <= sy && sy <= gy && gy <= w);
        uint64_t h1 = hash[gx][gy] - hash[gx][sy] * rw[gy - sy];
        uint64_t h2 = hash[sx][gy] - hash[sx][sy] * rw[gy - sy];
        return h1 - h2 * rh[gx - sx];
    }
};

```