HDOJ 2254 – 奥运

——by A Code Rabbit

Description

求 v1 到 v2 两个点之间，长度为 t1～t2 的路径总数。

Types

Maths :: Matrix

Analysis

这题可以用传递闭包的思想。
即这张图邻接矩阵的 n 次幂，即为两点之间长度为 n 的路径总数。
然后矩阵乘法+快速幂求解。
但是这道题有几个要注意的：

- 题目中城市的总数小于 30，这保证了我们构造的矩阵不会太大。
  但是输入中的城市 p1、p2 的取值范围是 int 的范围。
  因此我们要用一个 map，把城市映射到 0～30 的点上。
- 这题的数据规模较大，在计算 t1～t2 路径总数时，不能对[t1, t2]中的每一点都做矩阵乘法+快速幂，否则会 TLE。
  正确方法应该是，计算出指数为 t1 时的邻接矩阵，然后指数递增到 t2，其中对所求的路径求和。
- 注意 t1 = 0 的情况，这时候路径数应该为 0。

Solution

```
// HDOJ 2254
// 奥运
// by A Code Rabbit

#include <cstdio>
#include <cstring>
#include <map>

using namespace std;

const int LIMITS = 32;
const int DIVISOR = 2008;
```

```
struct Matrix {
    int element[LIMITS][LIMITS];
};

int n;
int p_1, p_2;
int k;
int v_1, v_2, t_1, t_2;

map<int, int> city;
int num_city;

Matrix mat_unit;
Matrix mat_one;
Matrix mat_ans;

int GetPos(int x);

Matrix QuickPower(Matrix mat_result, Matrix mat_one, int index);
Matrix Multiply(Matrix mat_a, Matrix mat_b);

int main() {
    while (scanf("%d", &n) != EOF) {
        // Initialize.
        memset(mat_one.element, 0, sizeof(mat_one.element));
        city.clear();
        num_city = 0;
        // Inputs and count.
        for (int i = 0; i < n; ++i) {
            scanf("%d%d", &p_1, &p_2);
            int pos_1 = GetPos(p_1);
            int pos_2 = GetPos(p_2);
            ++mat_one.element[pos_1][pos_2];
        }
        // Make mat_unit.
        for (int i = 0; i < num_city; ++i) {
            for (int j = 0; j < num_city; ++j) {
                mat_unit.element[i][j] = i == j ? 1 : 0;
            }
        }
        // Inputs, run and outputs.
```

```
        scanf("%d", &k);
        for (int i = 0; i < k; ++i) {
            scanf("%d%d%d%d", &v_1, &v_2, &t_1, &t_2);
            if (city.find(v_1) == city.end() ||
                city.find(v_2) == city.end())
            {
                printf("0\n");
                continue;
            }
            mat_ans = QuickPower(mat_unit, mat_one, t_1);
            int pos_1 = city[v_1];
            int pos_2 = city[v_2];
            int sum = !t_1 ? 0 : mat_ans.element[pos_1][pos_2];
            for (int j = t_1 + 1; j <= t_2; ++j) {
                mat_ans = Multiply(mat_ans, mat_one);
                sum += mat_ans.element[pos_1][pos_2];
            }
            printf("%d\n", sum % DIVISOR);
        }
    }

    return 0;
}

int GetPos(int x) {
    if (city.find(x) == city.end()) {
        city[x] = num_city++;
    }
    return city[x];
}

Matrix QuickPower(Matrix mat_result, Matrix mat_one, int index) {
    while (index) {
        if (index & 1) {
            mat_result = Multiply(mat_result, mat_one);
        }
        mat_one = Multiply(mat_one, mat_one);
        index >>= 1;
    }
    return mat_result;
}
```

```
Matrix Multiply(Matrix mat_a, Matrix mat_b) {
    Matrix mat_result;
    for (int i = 0; i < num_city; ++i) {
        for (int j = 0; j < num_city; ++j) {
            mat_result.element[i][j] = 0;
            for (int k = 0; k < num_city; ++k) {
                mat_result.element[i][j] += mat_a.element[i][k] * mat_b.element[k][j];
            }
            mat_result.element[i][j] %= DIVISOR;
        }
    }
    return mat_result;
}
```