算法设计与分析

第五、六章

姓名：张博康

学号：2014211383

班级：2014211309

一、 源代码

```
#include <iostream>
#include <sstream>
#include <fstream>
#include <cstring>
#include <cstdlib>
#include <vector>
#include <climits>
#include <cmath>
#include <queue>
#include <algorithm>
#include <ctime>

#define MAXSIZE 10000
#define NO_EDGE 99999
using namespace std;

const int SEQ_15[15] = {3, 5, 7, 8, 9, 10, 11, 12, 13, 16, 17, 19, 20, 21, 22};
const int SEQ_20[20] = {1, 2, 3, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22};

double map[42][42];
int mark[42];
int n, m;
long long l = 0;

bool Okay(int x)
{
	for (int j = 0; j < n; j++)
		if (map[x][j] > 0 && mark[j] == mark[x])
			return false;
	return true;
}

bool Color(int x)
{
	bool state = false;
	for (int i = 1; i <= m; i++) {
		mark[x] = i;
		if (Okay(x)) {
			l++;
			if (x + 1 == n) {
				for (int i = 0; i < n; i++)
```

```
                        cout << mark[i] << ' ';
                    cout << endl;
                    state = true;
                    return true;
                } else if (Color(x + 1))
                    return true;
            }
            mark[x] = 0;
        }
        return state;

}

double best = INT_MAX;
int origin;
int path[42];
int bestPath[42];
double maxn;

void Traveling(int node, int cost, int i)
{
    if (i == n) {
        if (map[node][origin] > 0) {
            if (best > cost + map[node][origin]) {
                best = cost + map[node][origin];
                path[i] = node;
                memcpy(bestPath, path, sizeof(path));
            }
        }
    } else {
        for (int j = 0; j < n; j++) {
            if (!mark[j] && map[node][j] > 0 && map[node][j] + cost < best) {
                l++;
                path[i] = j;
                mark[j] = 1;
                Traveling(j, cost + map[node][j], i + 1);
                mark[j] = 0;
            }
        }
    }
}

typedef struct node {
    double lcost; //  下界
```

```
    double cost;
    double rcost; //x[s:n-1]中顶点最小出边费用和
    int layer; //根结点到当前结点的路径为 x[0:s]
    int x[21]; //需要进一步搜索的顶点 x[s+1:n-1]
} Node, *pNode;


struct cmp {
    bool operator()(const pNode node1,const pNode node2) {
        return node1->lcost > node2->lcost;
    }
};

priority_queue<pNode, vector<pNode>, cmp > heap;
double MinOut[42];
double MinSum = 0;

void Traveling_BB(int origin)
{
    pNode cNode = new Node;
    for (int i = 0; i < n; i++)
        cNode->x[i] = i + 1;
    cNode->x[origin - 1] = 1;
    cNode->x[0] = origin;
    cNode->layer = 0;
    cNode->cost = 0;
    cNode->rcost = MinSum;

    while (cNode->layer < n - 1) {
        if (cNode->layer == n - 2) {
            //再加两条边构造回路
            if (map[cNode->x[n-2]][cNode->x[n-1]] > 0 && map[cNode->x[n-
1]][origin] > 0
                    && (cNode->cost + map[cNode->x[n-2]][cNode->x[n-1]] +
map[cNode->x[n-1]][origin] < best || best == INT_MAX)) {
                    best = cNode->cost + map[cNode->x[n-2]][cNode->x[n-1]] +
map[cNode->x[n-1]][origin];
                    cNode->layer += 1;
                    cNode->cost = best;
                    cNode->lcost = best;
                    heap.push(cNode);
            } else
                    delete cNode;
        } else { //产生当前
```

```
                for (int i = cNode->layer + 1; i < n; i++) {
                    if (map[cNode->x[cNode->layer]][cNode->x[i]] > 0) {
                        //可行儿子结点
                        double cc = cNode->cost +
map[cNode->x[cNode->layer]][cNode->x[i]];
                        double rcost = cNode->rcost -
MinOut[cNode->x[cNode->layer]];
                        double b = cc + rcost; //下界
                        if (b < best || best == INT_MAX) {
                            pNode tempNode = new Node;
                            for (int j = 0; j < n; j++)
                                tempNode->x[j] = cNode->x[j];
                            tempNode->x[cNode->layer + 1] = cNode->x[i];
                            tempNode->x[i] = cNode->x[cNode->layer + 1];
                            tempNode->cost = cc;
                            tempNode->layer = cNode->layer + 1;
                            tempNode->lcost = b;
                            tempNode->rcost = rcost;
                            heap.push(tempNode);
                        }
                    }
                }

                delete cNode;
            }

            if (!heap.empty())
                cNode = heap.top();
            else
                break;
            l++;
            heap.pop();
        }

        for (int i = 0; i < n; i++)
            bestPath[i] = cNode->x[i];
        while (!heap.empty())
        {
            delete heap.top();
            heap.pop();
        }
}

int main(int argc, char const *argv[])
```

```
{
    int choose = 0;
    while (choose != 5) {
        cout << "请选择以下操作：" << endl;
        cout << "1  图的 m 着色问题" << endl;
        cout << "2  旅行商问题（回溯法）" << endl;
        cout << "3  旅行商问题（分支界限法）" << endl;
        cout << "4  退出" << endl;

        while (cin >> choose, !(choose >= 1 && choose <= 5)) {
            cout << "输入不合法，请重新输入" << endl;
            cin.clear();
            cin.sync();
        }
        cout << "----------------------------------------------------------------" << endl;

        switch (choose) {
            case 1:
            {
                ifstream in1("附件 1-1.22 基站图的邻接矩阵.txt", ios_base::in);
                ifstream in2("附件 1-1.42 基站图的邻接矩阵.txt", ios_base::in);
                if (!in1.is_open() || !in2.is_open()) {
                    cout << "Error opening file..." << endl;
                    exit(1);
                }

                clock_t start, finish;
                double duration;

                double num;
                string line;
                getline(in1, line);
                istringstream iss(line);

                for (int i = 0; i < 22; i++) {
                    for (int j = 0; j <= 22; j++) {
                        in1 >> num;
                        if (j != 0)
                            map[i][j - 1] = map[j - 1][i] = ((num > NO_EDGE
- 1) ? -1 : num);
                    }
                }
                memset(mark, 0, sizeof(mark));
                n = 22;
```

```
                m = 1;

                cout << "22 个基站顶点组成的图: " << endl;
                start = clock();
                for(m = 1; m <= n; m++) {
                        if(Color(0))
                                break;
                }
                finish = clock();
                duration = (double)(finish - start) / CLOCKS_PER_SEC;
                cout << "花费时间：  " << duration << "s" << "\t 用到的颜色总数: " << m << "\t 搜索过的结点总数: " << l << endl;

                getline(in2, line);
                iss.clear();
                iss.str(line);

                for (int i = 0; i < 42; i++) {
                        for (int j = 0; j <= 42; j++) {
                                in2 >> num;
                                if (j != 0)
                                        map[i][j - 1] = map[j - 1][i] = ((num > NO_EDGE - 1) ? -1 : num);
                        }
                }
                memset(mark, 0, sizeof(mark));
                n = 42;
                m = 1;
                l = 0;

                cout << "42 个基站顶点组成的图: " << endl;
                start = clock();
                for(m = 1; m <= n; m++) {
                        if(Color(0))
                                break;
                }
                finish = clock();
                duration = (double)(finish - start) / CLOCKS_PER_SEC;

                cout << "花费时间：  " << duration << "s" << "\t 用到的颜色总数: " << m << "\t 搜索过的结点总数: " << l << endl;

                in1.close();
                in2.close();
```

```
                cout << "------------------------------------------------------------"
<< endl;
                break;
            }
            case 2:
            {
                ifstream in1("附件 1-1.15 基站图的邻接矩阵.txt", ios_base::in);
                ifstream in2("附件 1-1.20 基站图的邻接矩阵.txt", ios_base::in);
                ifstream in3("附件 1-1.22 基站图的邻接矩阵.txt", ios_base::in);
                if (!in1.is_open() || !in2.is_open() || !in3.is_open())
                {
                    cout << "Error opening file..." << endl;
                    exit(1);
                }

                clock_t start, finish;
                double num;

                string line;
                getline(in1, line);
                istringstream iss(line);

                n = 15;
                for (int i = 0; i < n; i++) {
                    for (int j = 0; j <= n; j++) {
                        in1 >> num;
                        if (j != 0) {
                            map[i][j - 1] = map[j - 1][i] = ((num > NO_EDGE
- 1) ? -1 : num);

                        }
                    }
                }

                l = 0;
                best = INT_MAX;
                memset(mark, 0, sizeof(mark));
                cout << "15 个基站顶点组成的图，起始城市为结点 20: " <<
endl;

                origin = 15 - 2 - 1;
                mark[origin] = 1;
                path[0] = origin;
```

```
                start = clock();
                Traveling(origin, 0, 1);
                finish = clock();

                if (best == INT_MAX)
                    cout << "无满足条件的路径" << endl;
                else {
                    cout <<   "最短路径长度为：  " << best << "\t 路径为：
";
                    for (int i = 0; i < n; i++)
                        cout << SEQ_15[bestPath[i]] << " -> ";
                    cout << SEQ_15[origin] << endl;
                    cout << endl;
                }
                cout <<"扫描过的节点总数：  " << l << "\t 运行时间：  " <<
(double)(finish - start) / CLOCKS_PER_SEC << endl;

                getline(in2, line);
                iss.clear();
                iss.str(line);

                n = 20;
                for (int i = 0; i < n; i++) {
                    for (int j = 0; j <= n; j++) {
                        in2 >> num;
                        if (j != 0)
                            map[i][j - 1] = map[j - 1][i] = ((num > NO_EDGE
- 1) ? -1 : num);
                    }
                }

                l = 0;
                best = INT_MAX;
                memset(mark, 0, sizeof(mark));
                cout << endl << "20 个基站顶点组成的图，起始城市为结点
20：  " << endl;

                origin = 20 - 2 - 1;
                mark[origin] = 1;
                path[0] = origin;

                start = clock();
                Traveling(origin, 0, 1);
                finish = clock();
```

```
                    if (best == INT_MAX)
                        cout << "无满足条件的路径" << endl;
                    else {
                        cout <<   "最短路径长度为：  " << best << "\t 路径为：
";
                        for (int i = 0; i < n; i++)
                            cout << SEQ_20[bestPath[i]] << " -> ";
                        cout << SEQ_20[origin] << endl;
                        cout << endl;
                    }
                    cout <<"扫描过的节点总数：  " << l << "\t 运行时间：  " <<
(double)(finish - start) / CLOCKS_PER_SEC << endl;

                    getline(in3, line);
                    iss.clear();
                    iss.str(line);

                    n = 22;
                    for (int i = 0; i < n; i++) {
                        for (int j = 0; j <= n; j++) {
                            in3 >> num;
                            if (j != 0) {
                                map[i][j - 1] = map[j - 1][i] = ((num > NO_EDGE
- 1) ? -1 : num);
                            }
                        }
                    }

                    l = 0;
                    best = INT_MAX;
                    memset(mark, 0, sizeof(mark));
                    cout << endl << "22 个基站顶点组成的图，起始城市为结点
20：  " << endl;

                    origin = 20 - 1;
                    mark[origin] = 1;
                    path[0] = origin;

                    start = clock();
                    Traveling(origin, 0, 1);
                    finish = clock();

                    if (best == INT_MAX)
```

```
                    cout << "无满足条件的路径" << endl;
                else {
                    cout <<   "最短路径长度为：  " << best << "\t 路径为：
";
                    for (int i = 0; i < n; i++)
                        cout << bestPath[i] + 1 << " -> ";
                    cout << origin + 1 << endl;
                }
                cout <<"扫描过的节点总数：  " << l << "\t 运行时间：  " <<
(double)(finish - start) / CLOCKS_PER_SEC << endl;

                in1.close();
                in2.close();
                in3.close();
                cout << "----------------------------------------------------------------"
<< endl;
                break;
            }
            case 3:
            {
                ifstream in1("附件 1-1.15 基站图的邻接矩阵.txt", ios_base::in);
                ifstream in2("附件 1-1.20 基站图的邻接矩阵.txt", ios_base::in);
                if (!in1.is_open() || !in2.is_open())
                {
                    cout << "Error opening file..." << endl;
                    exit(1);
                }

                clock_t start, finish;
                double num;

                string line;
                getline(in1, line);
                istringstream iss(line);

                n = 15;
                for (int i = 1; i <= n; i++) {
                    double minn = INT_MAX;
                    for (int j = 0; j <= n; j++) {
                        in1 >> num;
                        if (j != 0) {
                            map[i][j] = map[j][i] = ((num > NO_EDGE - 1) ? -
222 : num);
                            if (map[i][j] > 0)
```

```
                                minn = min(map[i][j], minn);
                        }
                    }
                    MinOut[i] = minn; //顶点 i 的最小出边费用
                    MinSum += minn;
                }

                l = 0;
                best = INT_MAX;
                memset(mark, 0, sizeof(mark));
                cout << "15 个基站顶点组成的图，起始城市为结点 20: " <<
endl;

                origin = 15 - 2;
                start = clock();
                Traveling_BB(origin);
                finish = clock();

                if (best == INT_MAX)
                    cout << "无满足条件的路径" << endl;
                else {
                    cout <<   "最短路径长度为：  " << best << "\t 路径为：
";
                    for (int i = 0; i < n; i++)
                        cout << SEQ_15[bestPath[i] - 1] << " -> ";
                    cout << SEQ_15[origin - 1] << endl;
                    cout << endl;
                }
                cout <<"扫描过的节点总数：  " << l << "\t 运行时间：  " <<
(double)(finish - start) / CLOCKS_PER_SEC << endl;

                getline(in2, line);
                iss.clear();
                iss.str(line);

                n = 20;
                MinSum = 0;
                for (int i = 1; i <= n; i++) {
                    double minn = INT_MAX;
                    for (int j = 0; j <= n; j++) {
                        in2 >> num;
                        if (j != 0) {
                            map[i][j] = map[j][i] = ((num > NO_EDGE - 1) ? -
1 : num);
```

```
                                if (map[i][j] > 0)
                                    minn = min(map[i][j], minn);
                            }
                        }
                        MinOut[i] = minn; //顶点 i 的最小出边费用
                        MinSum += minn;
                    }

                    l = 0;
                    best = INT_MAX;
                    memset(mark, 0, sizeof(mark));
                    cout << endl << "20 个基站顶点组成的图，起始城市为结点 20：  " << endl;

                    origin = 20 - 2;
                    start = clock();
                    Traveling_BB(origin);
                    finish = clock();

                    if (best == INT_MAX)
                        cout << "无满足条件的路径" << endl;
                    else {
                        cout <<   "最短路径长度为：  " << best << "\t 路径为：
";
                        for (int i = 0; i < n; i++)
                            cout << SEQ_20[bestPath[i] - 1] << " -> ";
                        cout << SEQ_20[origin - 1] << endl;
                        cout << endl;
                    }
                    cout <<"扫描过的节点总数：  " << l << "\t 运行时间：  " <<
(double)(finish - start) / CLOCKS_PER_SEC << endl;

                    in1.close();
                    in2.close();
                    cout << "------------------------------------------------------------"
<< endl;
                    break;
                }
                default:
                    break;
            }
        }
        return 0;
}
```

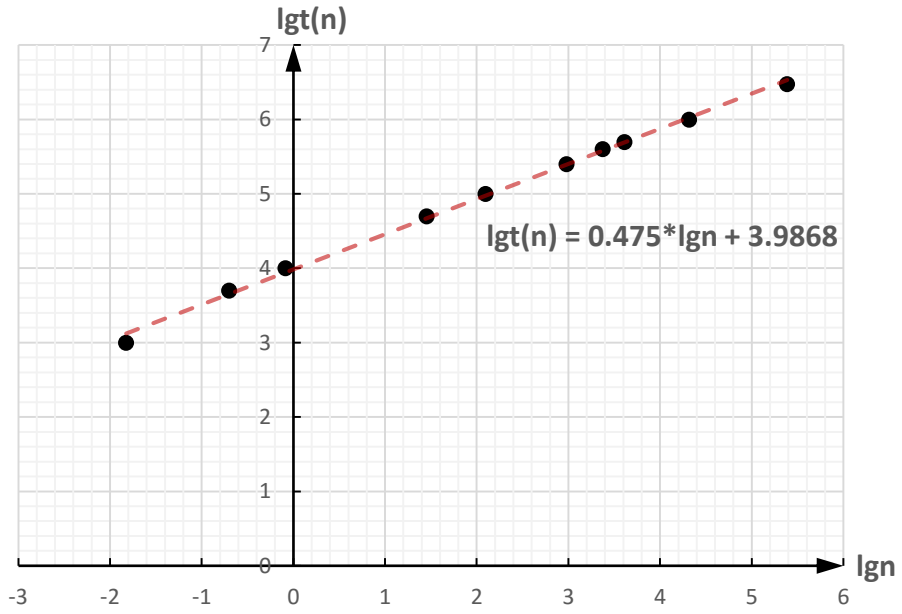
二、 运行结果

1. N 皇后问题

n	1000	5000	10000	50000	100000
m	1	1	1	1	1
t(n)	0.015	0.199	0.817	28.428	124.342
lg[t(n)]	-1.823	-0.701	-0.088	1.454	2.095
2lg(n)	6	7.398	8	9.398	10
3lg(n)	9	11.097	12	14.097	15
n	250000	400000	500000	1000000	3000000
m	1	1	1	1	1
t(n)	956.174	2363.955	4082.765	20716.622	243533.996
lg[t(n)]	2.981	3.374	3.611	4.316	5.387
2lg(n)	10.796	11.204	11.398	12	12.954
3lg(n)	16.194	16.806	17.097	18	19.431

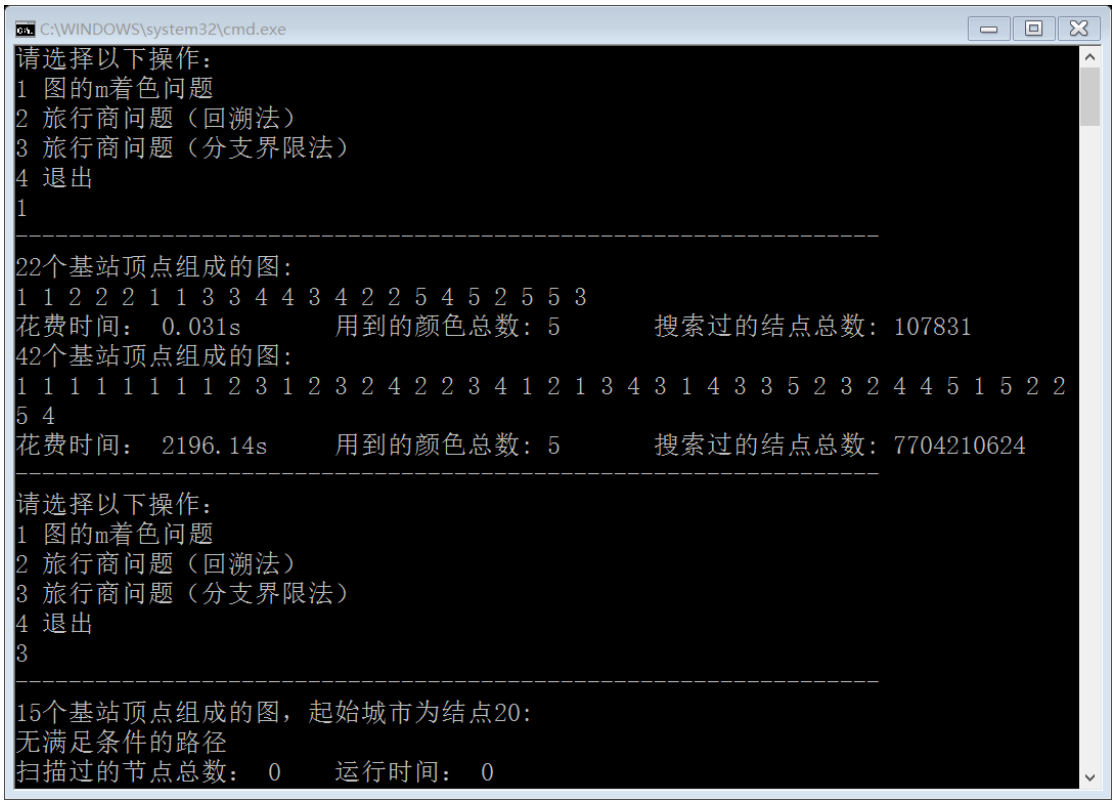
分别对 10 个不同的 n 取值，最终结果如上表所示：

从表中可知，初识随机解个数 m 不随问题规模 n 的变化而变化，而 t(n)随 n 的指数级增长



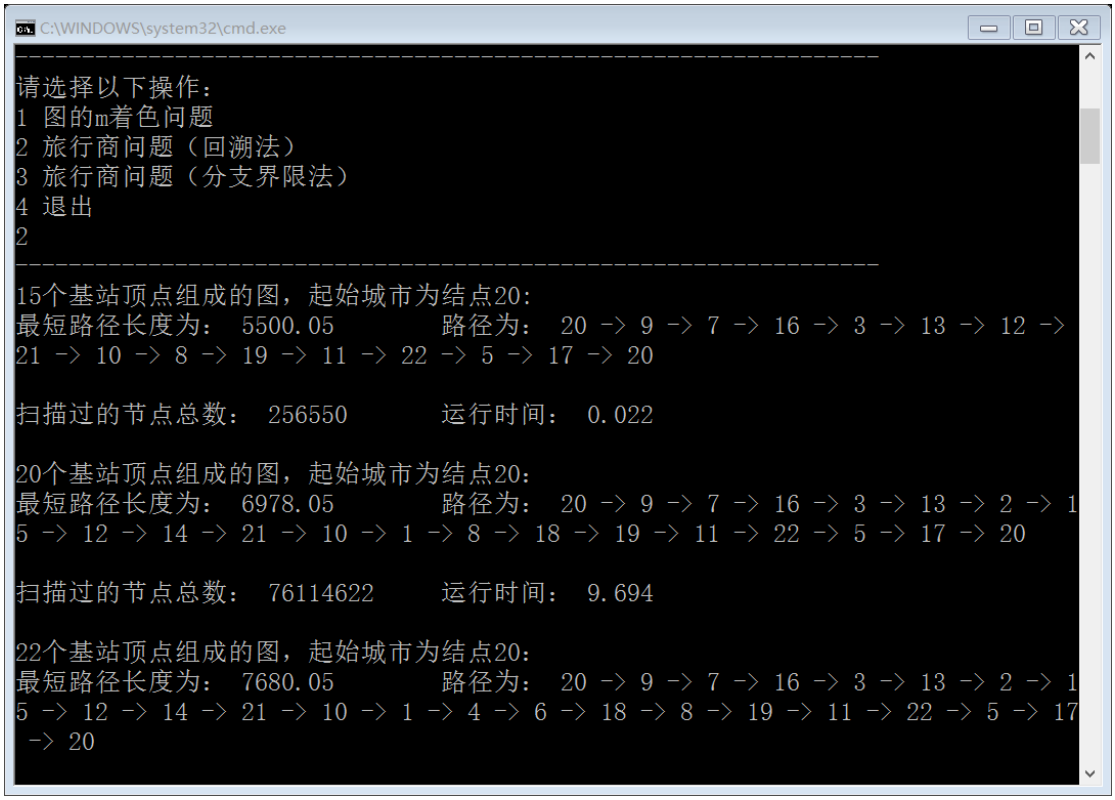
上图为<lgn, lgt(n)>散点图，可得线性回归方程为 lgt(n) = 0.475*lgn + 3.9868

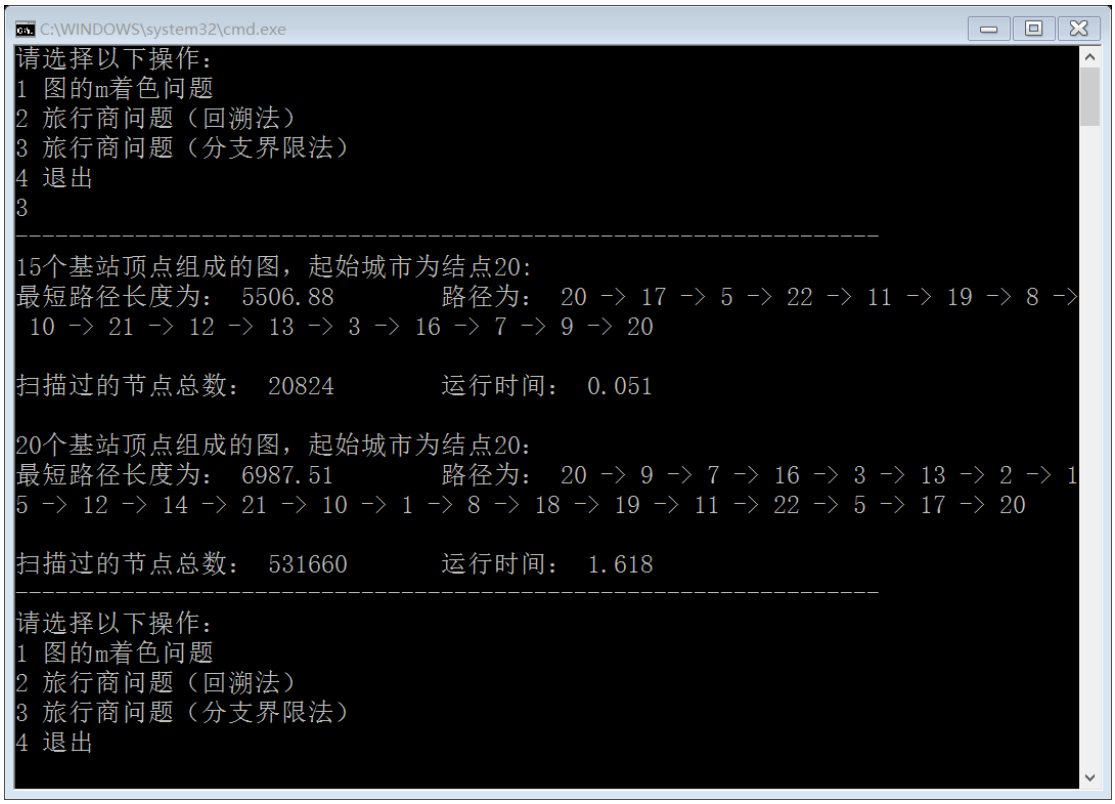
2. m 着色问题



问题	用到的颜色总数 m（色数）	搜索过的结点总数 L	程序运行时间 T（单位：s）
22 个基站	5	107831	0.031s
42 个基站	5	7704210624	2196.14

3. 旅行商问题


C:\WINDOWS\system32\cmd.exe
请选择以下操作：
1 图的m着色问题
2 旅行商问题（回溯法）
3 旅行商问题（分支界限法）
4 退出
2
15个基站顶点组成的图，起始城市为结点20：
最短路径长度为： 5500.05 路径为： 20 -> 9 -> 7 -> 16 -> 3 -> 13 -> 12 -> 21 -> 10 -> 8 -> 19 -> 11 -> 22 -> 5 -> 17 -> 20
扫描过的节点总数： 256550 运行时间： 0.022
20个基站顶点组成的图，起始城市为结点20：
最短路径长度为： 6978.05 路径为： 20 -> 9 -> 7 -> 16 -> 3 -> 13 -> 2 -> 15 -> 12 -> 14 -> 21 -> 10 -> 1 -> 8 -> 18 -> 19 -> 11 -> 22 -> 5 -> 17 -> 20
扫描过的节点总数： 76114622 运行时间： 9.694
22个基站顶点组成的图，起始城市为结点20：
最短路径长度为： 7680.05 路径为： 20 -> 9 -> 7 -> 16 -> 3 -> 13 -> 2 -> 15 -> 12 -> 14 -> 21 -> 10 -> 1 -> 4 -> 6 -> 18 -> 8 -> 19 -> 11 -> 22 -> 5 -> 17 -> 20


C:\WINDOWS\system32\cmd.exe
请选择以下操作：
1 图的m着色问题
2 旅行商问题（回溯法）
3 旅行商问题（分支界限法）
4 退出
3
15个基站顶点组成的图，起始城市为结点20：
最短路径长度为： 5506.88 路径为： 20 -> 17 -> 5 -> 22 -> 11 -> 19 -> 8 -> 10 -> 21 -> 12 -> 13 -> 3 -> 16 -> 7 -> 9 -> 20
扫描过的节点总数： 20824 运行时间： 0.051
20个基站顶点组成的图，起始城市为结点20：
最短路径长度为： 6987.51 路径为： 20 -> 9 -> 7 -> 16 -> 3 -> 13 -> 2 -> 15 -> 12 -> 14 -> 21 -> 10 -> 1 -> 8 -> 18 -> 19 -> 11 -> 22 -> 5 -> 17 -> 20
扫描过的节点总数： 531660 运行时间： 1.618
请选择以下操作：
1 图的m着色问题
2 旅行商问题（回溯法）
3 旅行商问题（分支界限法）
4 退出

问题	求解算法	最短回路	路径总长度（单位：m）	搜索过的结点总数	程序运行时间（单位：s）
15 个基站	回溯	20→9→7→16→3→13→12→21→10→8→19→11→22→5→17→20	5500.05	256550	0.022
	分支限界	20→9→7→16→3→13→12→21→10→8→19→11→22→5→17→20	5506.88	20824	0.051
20 个基站	回溯	20→9→7→16→3→13→2→15→12→14→21→10→1→8→18→19→11→22→5→17→20	6978.05	76114622	9.694
	分支限界	20→9→7→16→3→13→2→15→12→14→21→10→1→8→18→19→11→22→5→17→20	6987.51	531660	1.618
22 个基站	回溯法	20→9→7→16→3→13→2→15→12→14→21→10→1→4→6→18→8→8→19→11→22→5→17→20	7680.05	485744457	68.317