图与网络：第一次编程作业 报告

小组成员：
黎锦灏 518021910771
沈刘 518030910114
程欣雯 518021910031

GitHub repo 链接：
https://github.com/ljh2000/MATH6010_Graph-and-Network

一、问题描述

图 G 的支配集是顶点集合 V 的一个子集 $S \subset V$，集合 R 由不属于支配集的顶点组成，R 中的任意一个顶点至少与 S 中的一个顶点相邻。给定一个图 $G=(V,E)$ 有 n 个顶点，每个点的最小度 $\delta > 1$，则图 G 的最小支配集最多有 $\frac{n[1+ln(\delta+1)]}{\delta+1}$ 个顶点。

实验内容：

随机生成满足课堂定理条件的图，编写一个贪心算法求解最小支配集问题，比较所求结果与定理给出的上界。

二、算法流程

Step 1:
由当前支配集 S 出发，覆盖 S 中的点及其 $neighbors$，从而得到当前未被支配的点集 R（R 中点满足：自己及其 $neighbors$ 均未被选进支配集）。

Step 2:
对 R 中每一点，遍历生成自己及其 $neighbors$ 的集合，即得 $|R|$ 个未被支配的点集。

Step 3:
统计 $|R|$ 个集合中各点的出现次数，根据贪心算法策略，选取出现次数最多的点作为新选入支配集的点，得到新支配集 S'。

Step 4:
若当前支配集已覆盖图的所有点，则算法结束，得到由贪心算法求解的支配集；
否则重复执行上述操作，扩充支配集。

三、程序框架

贪心算法求解支配集

比较未被支配的顶点的度数，选度数最大的点加入支配集 V，直至所有不属于支配集中的顶点都与支配集里至少一个顶点相邻。

本程序分为三个文件：

- 文件 `Graph_generator.cpp` 用于生成满足条件的图，并将结果存储至中间文件。程序输入两个参数 m 和 n 分别表示这个图的边数和节点数目，程序随机生成不重复的 m 条边：
    - 函数 `find(int)` 用于实现并查集，确保生成图为连通图；

```cpp
int find(int x){
    if(fa[x] != x) fa[x] = find(fa[x]);
    return fa[x];
}
```

    - 函数 `check()` 使用随机函数，生成 m 条符合条件的边，排除重复边和自环的情况，并通过并查集检查图的连通性；

```cpp
bool check(){
    memset(graph,0,sizeof(graph));
    int mcnt = 0;
    while(mcnt < m) {
        int x,y;
        x = rand() % n + 1;
        y = rand() % n + 1;
        if(x > y) swap(x,y);

        if(x == y || graph[x][y]) continue;

        graph[x][y] = 1;
        mcnt ++;
        //cout << x << " " << y << endl;
        e[mcnt].x = x; e[mcnt].y = y;
    }

    for(int i = 1;i <= n;i++) fa[i] = i;
    for(int i = 1;i <= m;i++){
        int fx,fy;
        fx = find(e[i].x);
        fy = find(e[i].y);
        if(fx != fy)
            fa[fx] = fy;
    }

    for(int i = 2;i <= n;i++)
        if(find(i) != find(1))
            return false;
    return true;
}
```

    - 函数 `main()` 是程序的入口，设置参数 m 和 n 用于确立图的边数和点数；
- 文件 `Minimun_Dominating_Set.cpp` 读取上一步函数的中间结果， 计算支配集的大小和理论上界。此处图的结构使用邻接表进行存储：
    - 函数 `add_edge(int ,int)` 将图的边添加到邻接表中，并增加对应节点的度，本函数接收两个参数，表示某一条边的两个端点；

```cpp
void add_edge(int x,int y){
    e[++ecnt].nxt = first[x];
    e[ecnt].to = y;
    first[x] = ecnt;

    e[++ecnt].nxt = first[y];
    e[ecnt].to = x;
    first[y] = ecnt;
}
```

    - 函数 `solve()` 使用贪心算法求解最小支配集；

```cpp
    // initialize
    set<int>::iterator it = doms.begin();
    while (it != doms.end()){
        //cout << *it << " ";
        int x = (*it);

        cont[x] = 1;
        for(int i = first[x] ; i ; i = e[i].nxt)
            cont[ e[i].to ] = 1;

        it++;
    }

    // check
    int dom_sz = 0;
    for(int x = 1;x <= n;x++)
        if(cont[x])
            dom_sz++;
    if(dom_sz == n) return true;

    // caculate ranks
    for(int x = 1;x <= n;x++){
        if(cont[x]) continue;
        for(int i = first[x] ; i ; i = e[i].nxt) {
            if(in_set[ e[i].to ]) continue;
            rk[ e[i].to ] ++;
        }
    }
```

```cpp

    // greedy algorithm
    int maxrk = -1, flag = 1;
    for(int x = 1;x <= n;x++) {
        if(rk[x] > maxrk) {
            maxrk = rk[x];
            flag = x;
        }
    }

    doms.insert(flag);
    in_set[flag] = 1;
    sz ++;

    //cout << sz << " : " << flag << endl;
    return false;
}
```

- 函数 `calc_upper_bound()` 会依次比较每个点的度，得出此图中的最小度δ，然后根据公式计算最小支配集的理论上界；

```cpp
void calc_upper_bound(){
    int mind = n;
    for(int x = 1; x <= n;x++)
        if(deg[x] < mind)
            mind = deg[x];

    double limit;
    limit = n * (1 + log(1 + mind) );
    limit /= mind + 1;

    //upper_bd = (int)limit;
    upper_bd = limit;
}
```

- 函数 `main()` 是程序的入口，读取中间结果，并调用上面三个函数计算结果；

* 文件 `Test.cpp` 是测试程序，重复运行 `Graph_generator.cpp` 和 `Minimun_Dominating_Set.cpp` 100次，展示程序运行结果。

四、结果分析

程序运行的结果的截图如下所示（每次实验截取前 8 组测试结果），由实验结果分析可知：**使用贪心法计算得到的结果小于理论上界**：

实验一：
n = 10, m = 19
（10个点，19条边的连通图）

```
Test 1 :
The theory of upper bound : 8.46574
Dominating set size : 3
1 4 9

Test 2 :
The theory of upper bound : 6.99537
Dominating set size : 2
7 10

Test 3 :
The theory of upper bound : 6.99537
Dominating set size : 2
2 9

Test 4 :
The theory of upper bound : 6.99537
Dominating set size : 2
2 9

Test 5 :
The theory of upper bound : 6.99537
Dominating set size : 2
3 8

Test 6 :
The theory of upper bound : 6.99537
Dominating set size : 2
3 8

Test 7 :
The theory of upper bound : 8.46574
Dominating set size : 4
1 2 4 10

Test 8 :
The theory of upper bound : 6.99537
Dominating set size : 3
1 3 8
```

实验二：
n = 15, m = 22
（15个点，22条边的连通图）

```
Test 1 :
The theory of upper bound : 12.6986
Dominating set size : 6
2 3 4 6 7 8

Test 2 :
The theory of upper bound : 12.6986
Dominating set size : 5
1 6 7 11 13

Test 3 :
The theory of upper bound : 12.6986
Dominating set size : 5
1 6 7 11 13

Test 4 :
The theory of upper bound : 12.6986
Dominating set size : 5
1 6 7 8 11

Test 5 :
The theory of upper bound : 12.6986
Dominating set size : 6
1 2 3 4 7 12

Test 6 :
The theory of upper bound : 12.6986
Dominating set size : 6
1 2 3 4 7 12

Test 7 :
The theory of upper bound : 12.6986
Dominating set size : 6
1 2 5 6 9 13

Test 8 :
The theory of upper bound : 10.4931
Dominating set size : 5
1 3 5 6 8
```

实验三：
n = 30, m = 50
（30个点，50条边的连通图）

```
Test 1 :
The theory of upper bound : 25.3972
Dominating set size : 8
2 4 6 8 18 19 21 24

Test 2 :
The theory of upper bound : 25.3972
Dominating set size : 8
2 4 6 8 18 19 21 24

Test 3 :
The theory of upper bound : 25.3972
Dominating set size : 10
1 3 4 6 8 9 12 14 19 27

Test 4 :
The theory of upper bound : 25.3972
Dominating set size : 10
1 3 4 6 8 9 12 14 19 27

Test 5 :
The theory of upper bound : 25.3972
Dominating set size : 9
10 12 13 14 16 18 25 26 27

Test 6 :
The theory of upper bound : 25.3972
Dominating set size : 9
2 4 8 10 12 14 20 21 25

Test 7 :
The theory of upper bound : 25.3972
Dominating set size : 9
2 4 8 10 12 14 20 21 25

Test 8 :
The theory of upper bound : 25.3972
Dominating set size : 9
2 4 5 6 15 18 21 24 26
```

实验四:

n = 50, m = 120

（50个点，120条边的连通图）

```
Test 1 :
The theory of upper bound : 34.9769
Dominating set size : 12
3 5 8 9 13 15 18 19 31 32 35 38

Test 2 :
The theory of upper bound : 34.9769
Dominating set size : 12
3 5 8 9 13 15 18 19 31 32 35 38

Test 3 :
The theory of upper bound : 42.3287
Dominating set size : 14
2 3 5 6 7 8 10 19 21 22 24 27 37 41

Test 4 :
The theory of upper bound : 42.3287
Dominating set size : 11
2 6 14 16 17 22 28 31 32 39 44

Test 5 :
The theory of upper bound : 42.3287
Dominating set size : 11
2 6 14 16 17 22 28 31 32 39 44

Test 6 :
The theory of upper bound : 42.3287
Dominating set size : 12
1 6 8 18 20 21 24 30 32 34 47 49

Test 7 :
The theory of upper bound : 34.9769
Dominating set size : 13
2 3 5 13 14 15 19 25 30 35 38 39 41

Test 8 :
The theory of upper bound : 34.9769
Dominating set size : 13
2 3 5 13 14 15 19 25 30 35 38 39 41
```