**第2周上机实验报告**

# 一、题目：

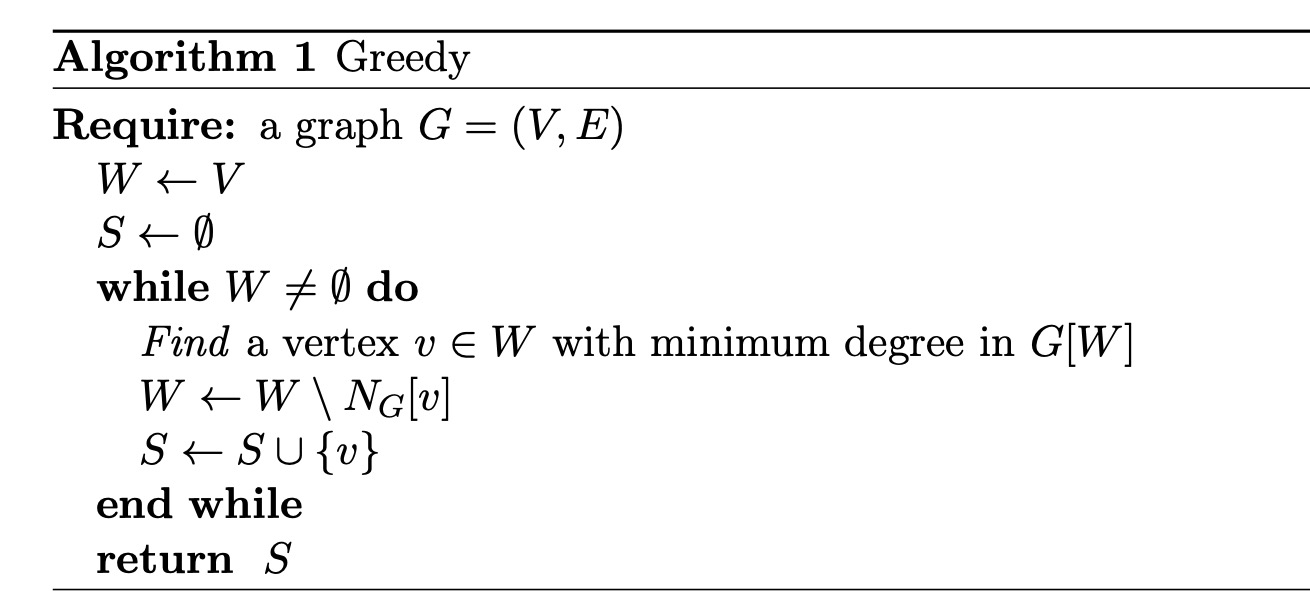
设计一个独立集算法，在真实图上找到尽可能大的独立集。

使用近似算法。

# 二、算法思路：

具体描述算法的设计思想

## 伪代码

****

**贪心策略：每次都挑选具有最小度的点，移除该点和其邻域中的所有点**

## 时间复杂度分析

* 使用l\_min寻找到最小度的点是O(1)
  + 由于独立集大小最大为n，while最多循环n次，故所有该操作的时间复杂度为O(n)
* 删除顶点v以及其临域，需要遍历
  + 对于每一个顶点，删除它是O(1)
  + 由于每个顶点最多删除一次，故所有删除操作加起来是O(n)
* 生成结果数组，每一次操作都是O(1)
  + 独立集最大为n，所以总的该操作的时间复杂度为O(n)
* 综上所述，总的时间复杂度是O(n)

## 近似比分析

* 将v不断添加进S，最终S的大小就是独立集的大小
* 每一次临域时，删除的结点数最多为（）
* 总共删除｜S｜次（S为近似解产生的独立集）
* 因此最终至多有个顶点被删除
* 图G至多有个点，即
* 化简得
* 所以近似比为

# 三、程序设计框架：

## 核心函数的名称和功能：

贪心策略的主体：

* l\_min记录了当前degree最小的结点
* lengths按照大小储存了所有结点的degree
* while循环中
  + 选取最小度的点
  + 删除该点以及其邻域
  + 独立集中增加该点

1. **void** find\_MIS()
2. {
4. **int** ans = 0;
5. **int** w = n;
6. **while** (w > 0 && !lengths[l\_min].empty())
7. {
8. **int** v = lengths[l\_min].front();
9. w = w - delete\_nodes(v);
11. ans++;
12. }
13. cout << ans << endl;
14. }

删除结点：

* 在lengths中将当前结点删除
* 通过遍历临接链表将所有的邻域中的点删除
* 更新l\_min

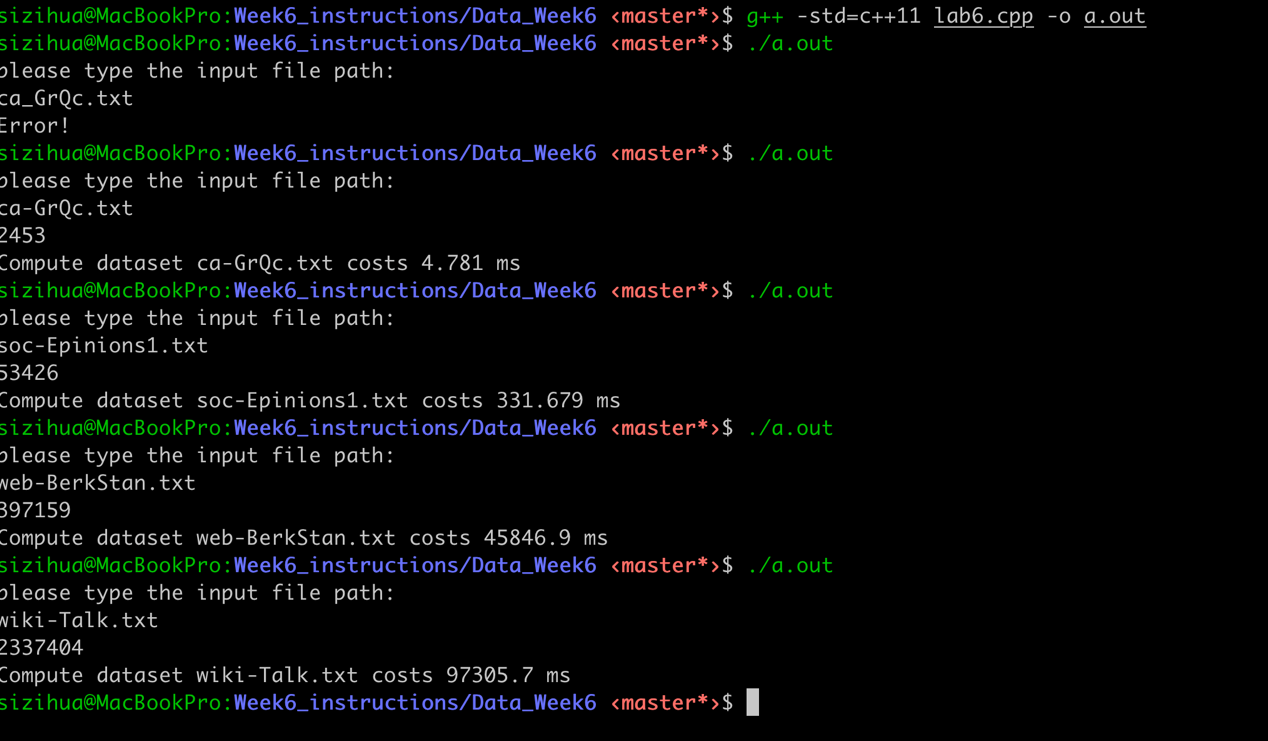
1. **int** delete\_nodes(**int** v)
2. {
3. **int** sum = 1;
4. judge[v] = **false**;
6. lengths[l\_min].pop\_front(); //在长度链表中删除v
7. **for** (auto i = adjacent\_table[v].begin(); i != adjacent\_table[v].end(); i++)
8. {
9. **if** (judge[\*i])
10. {
11. **int** pre\_size = remaining\_len[\*i];
12. judge[\*i] = **false**;
13. sum++;
14. lengths[pre\_size].remove(\*i);
15. }
16. }
17. **while** (lengths[l\_min].empty())
18. {
19. l\_min++;
20. }
21. **return** sum;
22. }

## 输入输出的格式等：

使用fstream中的open函数打开文件，填写相对路径即可。如“./ca-GrQc.txt”

# 四、实验结果说明：

## 实验结果截图：



上图中第一次输入文件名错误，所以显示Error。

## 运行结果比较

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Grpahs** | **Referring approximate Independent set number** | **Referring processing time(ms)** | **My independent set number** | **My processing time(ms)** |
| ca-GrQc | 2,458 | 0.811 | 2,435 | 4.781 |
| soc-Epinions1 | 53,598 | 10.165 | 53,426 | 331.679 |
| web-BerkStan | 407,174 | 733.648 | 397,159 | 45,846.9 |
| wiki-Talk | 2,335,644 | 147.647 | 2,337,404 | 97,305.7 |

## 结果分析

* 从运行结果来看，前3个图中，本实验采用的算法略差于最优算法，第4个图中，本实验采用算法结果要优于最优算法
* 从运行时间看，不知道最优算法在什么机器上运行的，所以个人认为时间不具有比较的价值。若想比较时间，应该在同一台机器上运行。而且本次实验实现的算法可能在细节上不够精致，所以导致O(n)中的常数较大。