



南開大學
Nankai University

计算机学院
大数据计算及应用作业报告

PageRank

姓名：王志铭
学号：2212285
专业：计算机科学与技术

2025 年 4 月 29 日

目录

1	作业要求	2
2	数据集描述	2
3	关键代码细节	2
4	内存优化方法	3
5	结果分析	5

1 作业要求

读取 Data.txt，计算 PageRank，并将 Top 100 节点及其分数按以下格式写入 Res.txt。要求程序运行期间内存占用不得超过 80MB，并且程序需在 60 秒内完成运行，不能以牺牲过多时间性能为代价优化内存。

2 数据集描述

数据集中包含若干行数据，每行数据格式为 FromNodeID ToNodeID，为两个整数，共同描绘了网络图上的一条边。FromNodeID 表示该边的起始点，ToNodeID 表示该边的终点。

3 关键代码细节

核心的 PageRank 算法函数如下：

```
1
2 import numpy as np
3 def pagerank(M, total, rounds, teleport, target):
4     # 初始化权重
5     pr = np.full(total, 1.0 / total)
6     # 筛出黑洞节点
7     sum_ = np.array(M.sum(axis=0)).flatten()
8     blackholes = np.where(sum_ == 0)[0]
9
10    for i in range(rounds):
11        last = pr.copy()
12
13        pr = M.dot(last) * teleport
14        # 处理黑洞节点，将权重平均分配到所有节点
15        if len(blackholes) > 0:
16            pr += np.sum(last[blackholes]) * teleport / total
17        pr += (1 - teleport) / total
18
19        # 计算收敛情况
20        ans = 0.0
21        for i in range(total):
22            ans += abs(pr[i] - last[i])
23        if ans < target:
24            break
25
26    return pr
```

该函数主要有三个关键点：

PageRank 的核心计算（第 12, 16 行）遵从如下公式：

$$PR = d \cdot M \times PR + \frac{1-d}{N} \quad (1)$$

其中：

- PR 是页面的 PageRank 值
- d 是阻尼因子及转移参数 (teleport)
- M 是转移矩阵
- N 是网页总数
- $\frac{1-d}{N}$ 表示随机跳转到任意页面的概率，用于处理蜘蛛网陷阱和黑洞节点。

第 15 行主要是针对黑洞节点 (dead end) 进行了进一步处理，在每次迭代计算中，将所有黑洞节点的权重值分散给其他所有节点，即额外与其他节点建立一条边。

第 19-23 行判断每轮迭代后 PageRank 值的更新情况，若低于阈值即可停止迭代。

4 内存优化方法

本次实验中主要使用了稀疏矩阵来进行内存方面的优化，调用了 python 中的 scipy 库。此项工作主要在数据的预处理阶段完成；

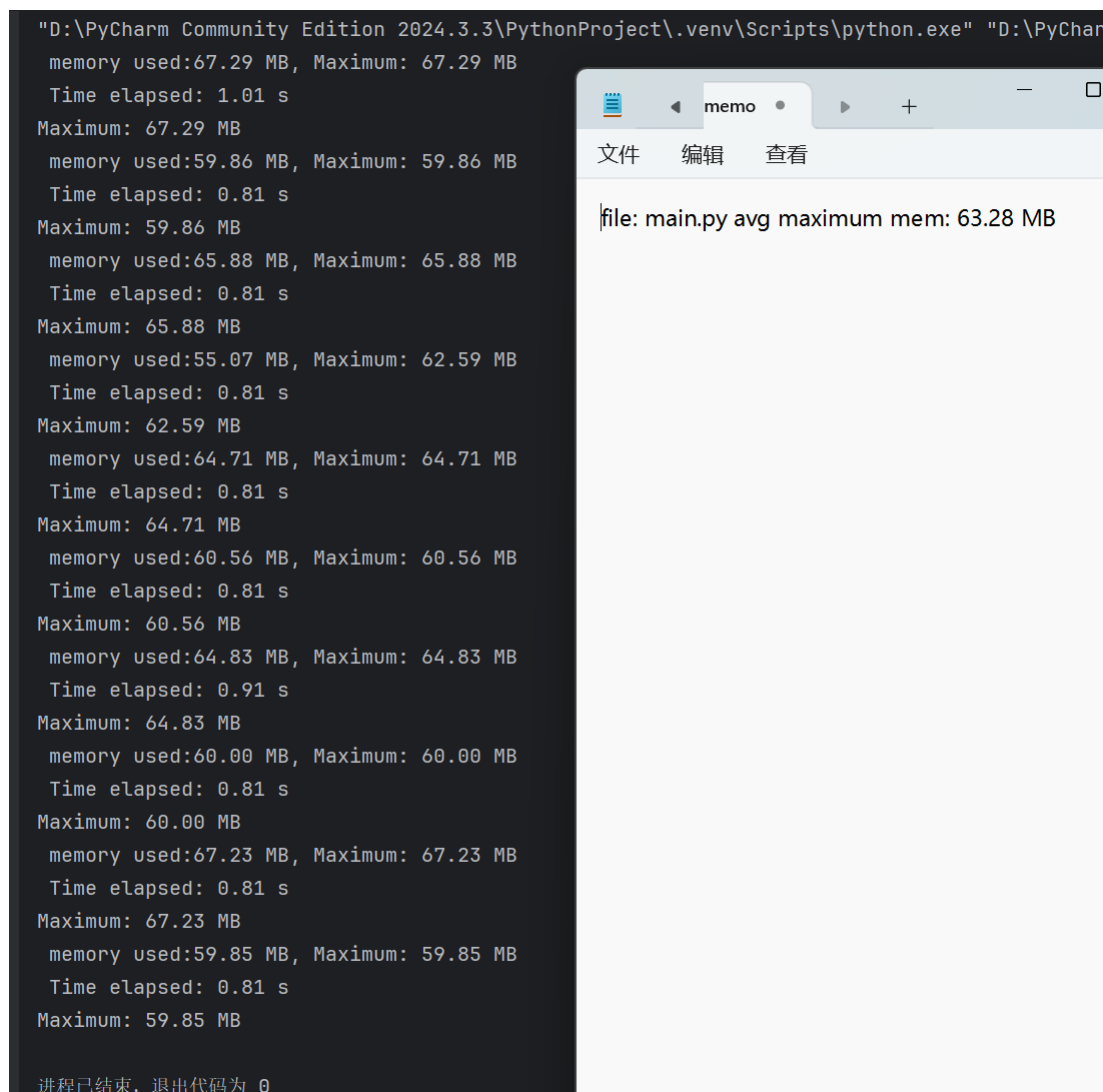
```
1
2 from scipy.sparse import csr_matrix
3 def inital_data():
4     # 遍历，记录节点信息
5     all_nodes = set()
6     degrees = {} # 记录每个节点的出度
7
8     # 读取文件获取所有节点
9     with open(file_, 'r') as f:
10         for line in f:
11             # 滤去空行
12             if line.strip() == '':
13                 continue
14             # 筛出出入点对，记录总共的节点信息
15             parts = line.split()
16             FromNode = int(parts[0])
17             ToNode = int(parts[1])
18             all_nodes.add(FromNode)
19             all_nodes.add(ToNode)
20             # 统计 出节点 的个数
```

```
21         if FromNode in degrees:
22             degrees[FromNode] += 1
23         else:
24             degrees[FromNode] = 1
25
26     # 排序节点, 建立索引
27     nodes_sort = sorted(all_nodes)
28     total_node = len(nodes_sort)
29     # 创建节点序号映射信息, 防止有空缺的序号占用空间
30     node_id = {node: i for i, node in enumerate(nodes_sort)}
31
32     # 再遍历, 构建矩阵
33     row = []
34     col = []
35     values = []
36
37     with open(file_, 'r') as f:
38         for line in f:
39             if line.strip() == '':
40                 continue
41             parts = line.split()
42             FromNode = int(parts[0])
43             ToNode = int(parts[1])
44             # 滤去黑洞节点
45             if FromNode in degrees:
46                 FromId = node_id[FromNode]
47                 col.append(FromId)
48                 ToId = node_id[ToNode]
49                 row.append(ToId)
50                 # 初始化权重
51                 weight = 1.0 / degrees[FromNode]
52                 values.append(weight)
53
54     # 创建稀疏矩阵
55     ans = csr_matrix((values, (row, col)), shape=(total_node, total_node))
56     return ans, nodes_sort, total_node, node_id
```

其中, 我分两次对数据集进行读取, 第一轮主要记录节点的总体信息, 并筛出黑洞节点。第二轮则是关注每条边的源节点、目标节点和权重值, 利用这三元素来构建最终存储网络信息的稀疏矩阵。

5 结果分析

teleport 参数设定为 0.85 时，利用压缩包中给出的运行内存监测文件运行代码，得到结果如下：



```
"D:\PyCharm Community Edition 2024.3.3\PythonProject\.venv\Scripts\python.exe" "D:\PyCharm
memory used:67.29 MB, Maximum: 67.29 MB
Time elapsed: 1.01 s
Maximum: 67.29 MB
memory used:59.86 MB, Maximum: 59.86 MB
Time elapsed: 0.81 s
Maximum: 59.86 MB
memory used:65.88 MB, Maximum: 65.88 MB
Time elapsed: 0.81 s
Maximum: 65.88 MB
memory used:55.07 MB, Maximum: 62.59 MB
Time elapsed: 0.81 s
Maximum: 62.59 MB
memory used:64.71 MB, Maximum: 64.71 MB
Time elapsed: 0.81 s
Maximum: 64.71 MB
memory used:60.56 MB, Maximum: 60.56 MB
Time elapsed: 0.81 s
Maximum: 60.56 MB
memory used:64.83 MB, Maximum: 64.83 MB
Time elapsed: 0.91 s
Maximum: 64.83 MB
memory used:60.00 MB, Maximum: 60.00 MB
Time elapsed: 0.81 s
Maximum: 60.00 MB
memory used:67.23 MB, Maximum: 67.23 MB
Time elapsed: 0.81 s
Maximum: 67.23 MB
memory used:59.85 MB, Maximum: 59.85 MB
Time elapsed: 0.81 s
Maximum: 59.85 MB

进程已结束，退出代码为 0
```

file: main.py avg maximum mem: 63.28 MB

图 5.1: 运行内存与时间结果

平均的运行内存使用为 63.28 MB，且时间均在 1s 内，满足作业要求。

接下来更改 teleport 参数，进行了多次实验并对结果进行可视化分析表示。（由于迭代次数均在 10 次左右波动，**不具可比较性**，故不作对比，只有得到结论为本次实验数据集上 PageRank 计算的收敛迭代次数与 teleport 参数关系不大，可以很快收敛）

以 teleport 参数 0.85 为基准，测试 teleport 参数为 0.95、0.9、0.85、0.8、0.75、0.7 共六种情况。

计算 Jaccard 相似度，衡量前 100 节点的重叠程度，并针对共同节点的排名顺序计算 Spearman 相关系数，针对二者的结果绘制热力图如下图5.2：

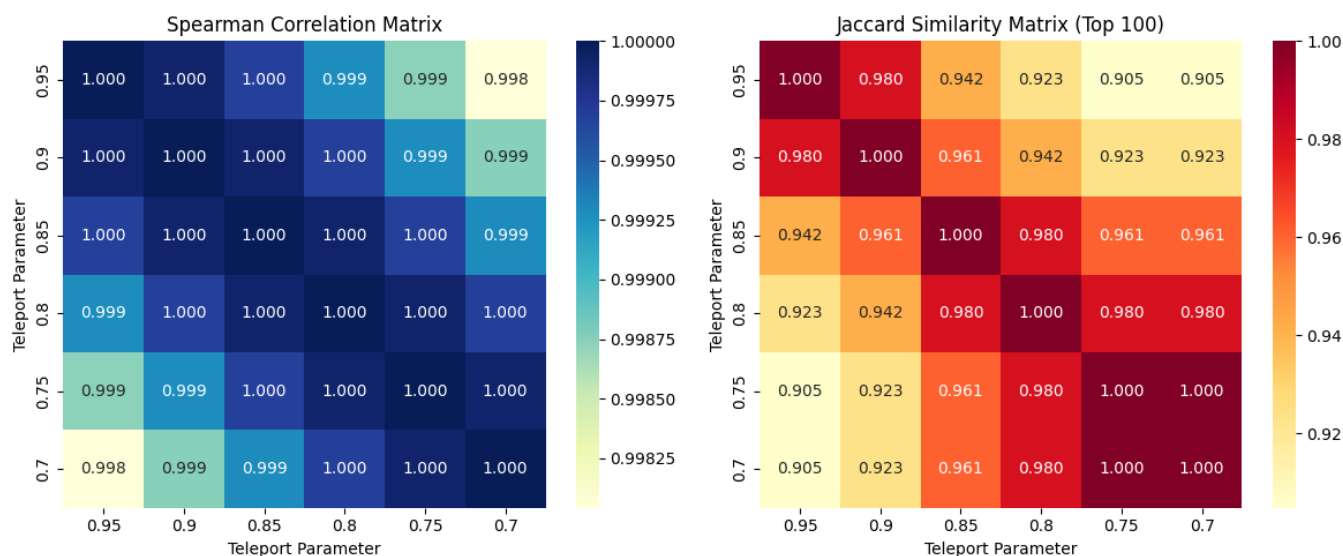


图 5.2: Spearman 相关系数 (左) 与 Jaccard 相似度 (右) 热力图

对于左侧蓝色的 Spearman 相关系数矩阵, 所有参数间的相关系数都 >0.998 , 可知系统对 teleport 参数并不敏感。

右侧红色的 Jaccard 相似矩阵, Jaccard 值均 >0.9 , 可知排名在前 100 的节点组成非常稳定, 受 teleport 参数的影响不大。

接下来利用各 teleport 参数下的节点分数绘制分布直方图, 观察分析不同 teleport 参数计算 PageRank 结果的分布情况:

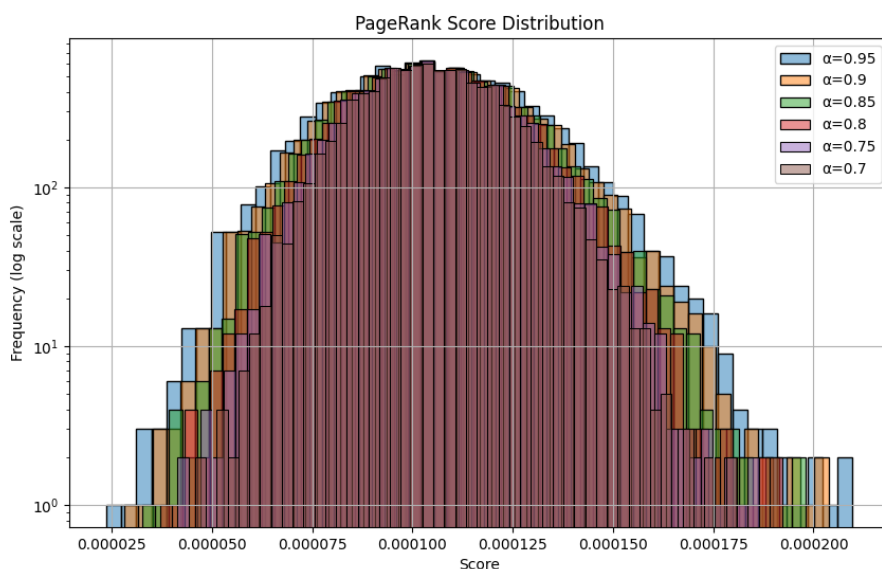


图 5.3: PageRank 分数分布直方图

整体上分布的差别不大, 但 teleport 参数越小分数分布的要更集中一些, 且较大的 teleport 参数对应的结果会存在少量极高值节点, 体现为“尾部延长”, 这表示这某些重要的节点 (也有可能是 dead end) 获得了更高的权重得分。