目录

pagerank 算法实验报告

王启翔 524072910012

April 2025

1 任务 1

1.1 任务 1 的实现

在用改进的 PageRank 算法讨论图 7.2 所示的小型网络时，我们取 p = 0.85，请依次改取 p = 0.75，p = 0.8 以及 p = 0.9，然后观察网页排名结果的变化情况。

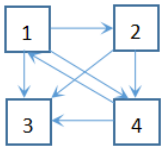

图 7.2

得到根据改进的 PageRank 算法公式

$$x_k = p \sum_{j \in L_k} \frac{x_j}{n_j} + (1-p)\frac{1}{n}$$

以邻接矩阵

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

作为输入用代码实现改进的 PageRank（代码见附录 1），输出结果为表 1。

1.2 p 值对排序的影响分析

通过观察不同 p 值下的 PageRank 结果，可以总结出以下规律：

1. 网页重要性分布变化：随着 p 值增大，网页 3 的 PageRank 值显著增加（从 0.375 增至 0.410），而其他网页的重要性逐渐降低。这表明更高的 p 值增强了“权威页面”（如网页 3）的影响力。
2. 排序稳定性：在所有 p 值下，网页的排名顺序保持不变（3→4→1→2）。这说明当前网络结构中存在明显的权威层级，即使调整 p 值也不会改变相对顺序。

表 1: 不同阻尼系数 p 下的 PageRank 结果对比

$p = 0.75$		$p = 0.80$		$p = 0.85$		$p = 0.90$	
网页	PageRank 值	网页	PageRank 值	网页	PageRank 值	网页	PageRank 值
3	0.37546	3	0.38633	3	0.39790	3	0.41031
4	0.24669	4	0.24468	4	0.24229	4	0.23946
1	0.21577	1	0.21402	1	0.21227	1	0.21048
2	0.16208	2	0.15497	2	0.14754	2	0.13975

3. 收敛速度：较大的 p 值（如 0.90）会使 PageRank 分布更加集中，导致头部网页与尾部网页的差距拉大；较小的 p 值（如 0.75）则使分布更均匀。

1.3 p 值的合理设置建议

阻尼系数 p 的选择取决于具体应用场景：

1. 理论标准值：Google 创始人最初提出的 PageRank 算法中，建议使用 $p = 0.85$。这一数值在理论上平衡了“用户继续浏览”和“随机跳转”的概率。
2. 网络特性适配：
   - 对于链接结构稀疏的小网络，可适当降低 p 值（如 0.7 0.8）以增加随机性。
   - 对于链接密集的大型网络，$p = 0.85$ 通常是合理选择。
3. 应用目标调整：
   - 若需突出核心权威页面，可增大 p 值。
   - 若希望更多边缘页面获得曝光，可减小 p 值。

在本案例中，由于所有 p 值下排名稳定且网页 3 优势明显，建议保持 $p = 0.85$ 作为默认选择。若需进一步分析，可以尝试在 0.8 0.9 范围内进行敏感性测试，观察排名变化的临界点。

2 任务 2

2.1 任务 2 的实现

计算图 7.4 所示小型网络的排名, 分析其排名与图 7.2 所示小型网络排名发生变化的原因.

同样地，将邻接矩阵设置为

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

运行代码，得到表二的结果。

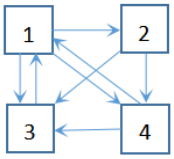

图 7. 4

$p = 0.75$		$p = 0.80$		$p = 0.85$		$p = 0.90$	
网页	PageRank 值	网页	PageRank 值	网页	PageRank 值	网页	PageRank 值
1	0.35491	1	0.36159	1	0.36815	1	0.37459
3	0.28592	3	0.28699	3	0.28796	3	0.28884
4	0.20794	4	0.20499	4	0.20208	4	0.19920
2	0.15123	2	0.14642	2	0.14181	2	0.13738

表 2: 增加一条网页 3 指向网页 1 的链接后不同 p 值下的 PageRank 结果

2.2 排名变化原因分析

当增加一条从网页 3 指向网页 1 的线后，排名发生了变化。原因在于 PageRank 算法是基于网页之间的链接结构来计算网页的重要性。网页 3 原本就具有一定的影响力，增加指向网页 1 的链接后，网页 1 获得了额外的“投票”（从网页 3 传递过来的权重）。

随着阻尼系数 p 的增大，网页之间链接的权重对 PageRank 值的影响更为显著。因为较大的 p 值意味着用户更倾向于沿着现有链接进行浏览，所以新增加的链接所传递的权重对目标网页（网页 1）的 PageRank 值提升效果更加明显。

在不同的 p 值下，虽然网页 1 的排名始终保持第一，但它的 PageRank 值随着 p 的增大而不断增加，同时其他网页（如网页 2、4）的 PageRank 值有所下降，这是由于权重在网页之间的重新分配导致的。而网页 3 的 PageRank 值变化相对较小，可能是因为它自身的链接结构较为复杂，新增加的一条链接对其整体权重影响不大。

3 任务 3

3.1 利用幂法计算网页排名的思路

利用幂法计算网页排名主要基于 PageRank 算法的原理。首先，根据网页之间的链接关系构建邻接矩阵。在邻接矩阵中，若网页 i 指向网页 j ，则对应矩阵元素设为 1 ，否则为 0 。构建好邻接矩阵后，将其每一列元素进行标准化处理，得到转移矩阵。转移矩阵的每一列元素之和为 1 ，它表示从一个网页跳转到其他网页的概率。

幂法通过迭代计算来逼近 PageRank 向量。迭代公式为 $\mathbf{r}^{(k+1)} = (1-d)\frac{\mathbf{e}}{n} + dP\mathbf{r}^{(k)}$ ，其中 $\mathbf{r}^{(k)}$ 是第 k 次迭代的 PageRank 向量，$\mathbf{e}$ 是全 1 向量，n 是网页总数，d 是阻尼系数（通常取 $0.8-0.9$ 之间的

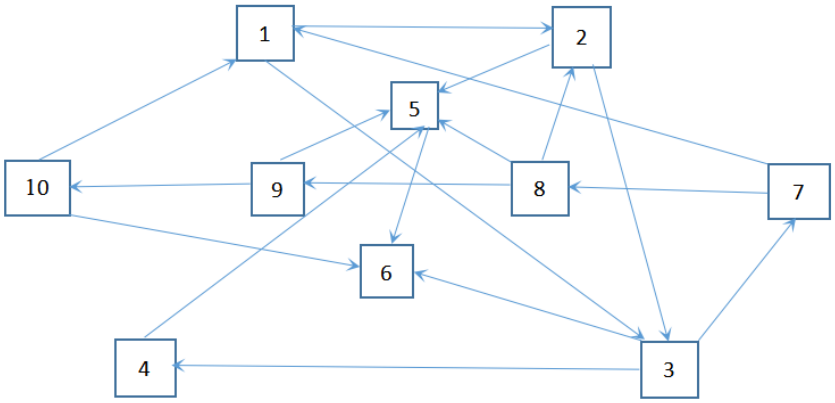

图 1: 图 7.5

值，如 0.85 ），P 是转移矩阵。从一个初始的 PageRank 向量（通常是每个元素都相等的向量）开始，不断迭代，直到相邻两次迭代的 PageRank 向量差异小于预先设定的收敛阈值（如 10^{-6} ），此时得到的向量就是各个网页的 PageRank 值，根据这些值可以对网页进行排名。

3.2 任务 3 的实现

利用幂法计算图 7.5 所示网络的排名.

按照上面的思路，书写代码（见附录 2）并运算，运算结果为表 3 所示。

排名	1	2	3	4	5	6	7	8	9	10
网页	6	5	3	2	1	4	7	8	10	9
PageRank 值	0.1009	0.0729	0.0474	0.0386	0.0376	0.0284	0.0284	0.0271	0.0246	0.0227

表 3: 网页排名结果

4 任务 4

4.1 三种改变顺序的思路

4.1.1 方法一：增加目标网页的入链接数量

让网页 5 和网页 6(PageRank 值最高的两个) 增加指向网页 9(最差的) 的链接，构建邻接矩阵 A' ($A'_{4,1}=1$ ，$A'_{4,2}=1$)，再构建转移矩阵 P' ，通过幂法 ($d=0.85$ ，$n=10$ ，$\mathbf{r}_0=\frac{\mathbf{e}}{n}$ ，$tol=1e-6$ ，$max\_iter=1000$) 迭代计算 $\mathbf{r}^{(k+1)}=(1-d)\frac{\mathbf{e}}{n}+dP'\mathbf{r}^{(k)}$ ，得 PageRank 值向量 $\mathbf{r}'$ 。

4.1.2 方法二：调整出链接分布

减少网页 3 指向网页 6 和网页 4 指向网页 5 的链接，增加指向网页 4 的链接（$A''_{6,3}=0$，$A''_{8,3}=0$，$A''_{4,3}=2$），构建邻接矩阵 A'' 及转移矩阵 P''，用幂法同参数迭代 $\mathbf{r}^{(k+1)}=(1-d)\frac{\mathbf{e}}{n}+dP''\mathbf{r}^{(k)}$，得 $\mathbf{r}''$。

4.1.3 方法三：构建链接环路

在网页 6 和 9 间添加双向链接（$A'''_{5,4}=1$，$A'''_{4,5}=1$），构建邻接矩阵 A''' 及转移矩阵 P'''，幂法同参数迭代 $\mathbf{r}^{(k+1)}=(1-d)\frac{\mathbf{e}}{n}+dP'''\mathbf{r}^{(k)}$，得 $\mathbf{r}'''$。

4.1.4 方法比较

比较三种方法所得 PageRank 值向量中网页 4 对应值 r'_4、r''_4、r'''_4，若 $r'_4>r''_4$ 且 $r'_4>r'''_4$，则增加入链接数量法更优；若 r''_4 最大，调整出链接分布法更优；若 r'''_4 最大，构建链接环路法更优。也可观察对整个网络排名影响综合评估。

4.2 任务 4 的实现

按照上面的思路，书写代码（见附录 3）并运算，运算结果如下：

原始排名: 网页 9 排名第 10, PageRank 值 =0.022673
方法 1: 网页 9 排名第 1, PageRank 值 =0.235616 (提升了 9 位)
方法 2: 网页 9 排名第 10, PageRank 值 =0.023609 (提升了 0 位)
方法 3: 网页 9 排名第 2, PageRank 值 =0.242295 (提升了 8 位)

最优方法是方法 1，网页 9 的排名提升了 9 位
各方法的 PageRank 值提升百分比:
方法 1: 939.21%
方法 2: 4.13%
方法 3: 968.66%

4.3 结果分析结论

通过对上述结果的分析，可以得出以下结论：

1. 不同方法效果差异明显：三种方法对网页 9 的排名提升效果呈现出显著差异。方法 1 大幅提升了网页 9 的排名和 PageRank 值，使其从第 10 位跃升至第 1 位，PageRank 值提升幅度高达 939.21%；方法 2 几乎未改变网页 9 的排名，PageRank 值仅提升了 4.13%；方法 3 使网页 9 排名提升至第 2 位，PageRank 值提升幅度为 968.66%。这表明不同的链接添加策略在影响网页排名方面具有不同的效力。

2. 增加入链接数量效果最佳：方法 1 采用增加目标网页入链接数量的策略，成为提升网页 9 排名的最优方法。这体现了在当前网络结构中，依据 PageRank 算法“更多高质量页面指向的页面排名更高”的原则，增加指向目标网页的链接能有效提升其重要性和排名。

3. 调整出链接分布效果不佳：方法 2 通过调整出链接分布的方式，对网页 9 的排名提升作用甚微。可能是因为当前网络结构中，这种调整未能有效改变目标网页的权重分配，或者该调整相对于其他因素对目标网页的影响较小。

4. 构建链接环路有一定效果：方法 3 利用构建链接环路的策略，使网页 9 的排名有了较大提升，虽不及方法 1，但也将其提升至第 2 位。说明在该网络中，合适的链接环路可增强相关网页间的相互影响，从而提升目标网页排名。

5. 综合评估的必要性：在评估不同方法的优劣时，不能仅依据目标网页排名的提升幅度，还需综合考虑对整个网络排名的影响。尽管方法 1 对网页 9 的提升效果显著，但需进一步分析其是否对其他网页排名产生不利影响，如是否导致重要网页排名大幅下降等。只有全面考量这些因素，才能准确评估不同策略在实际应用中的价值。

5 任务 5

使用幂迭代法，得出计算结果以图标展示 (表 4 和图 2，代码在附录 4)：

第一组			第二组			第三组		
排名	PageRank 值	网站索引	排名	PageRank 值	网站索引	排名	PageRank 值	网站索引
1	0.103640	7	11	0.011635	76	21	0.009086	85
2	0.048393	54	12	0.010597	51	22	0.009086	109
3	0.038737	53	13	0.010234	223	23	0.007897	102
4	0.030473	18	14	0.010044	101	24	0.006217	32
5	0.024795	9	15	0.009989	342	25	0.005412	41
6	0.024160	15	16	0.009940	382	26	0.004790	187
7	0.020895	1	17	0.009884	19	27	0.004782	124
8	0.020707	10	18	0.009758	3	28	0.004640	83
9	0.018037	222	19	0.009698	56	29	0.004640	84
10	0.011996	55	20	0.009317	421	30	0.004640	88

表 4: 哈佛大学相关网站排名（前 30 名）

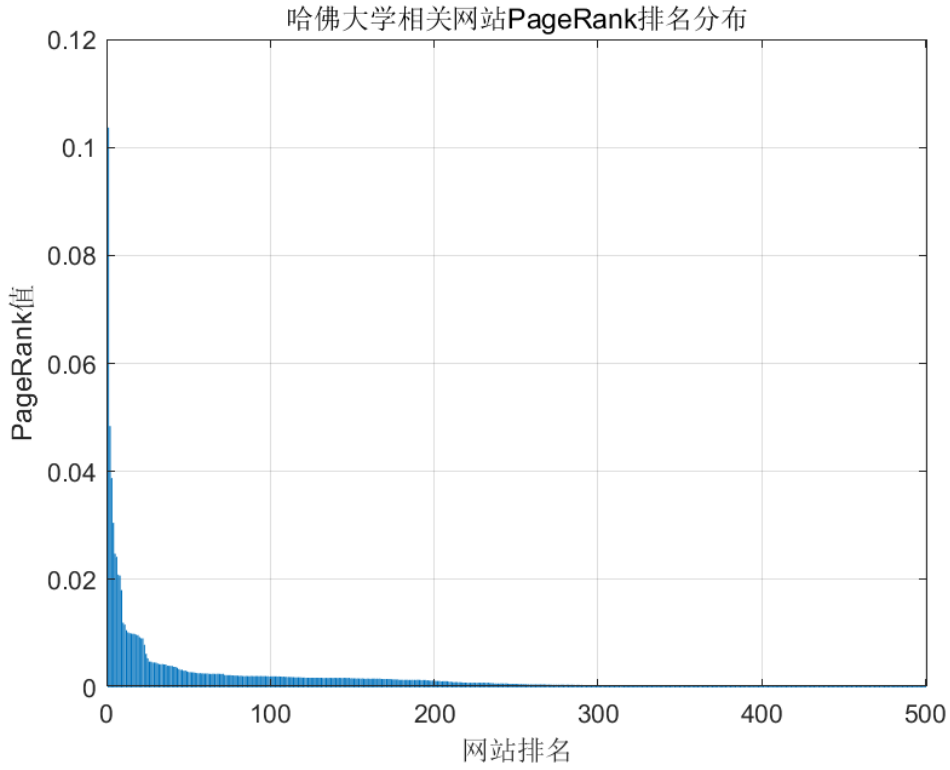

图 2: 哈佛 pagerank

6 附加任务

6.1 解题思路

6 支篮球队进行单循环比赛，每场比赛只计胜负，没有平局，比赛结果如下，请给出排名。

	A	B	C	D	E	F
A		1	0	1	1	1
B	0		0	1	1	1
C	1	1		1	0	0
D	0	0	0		1	1
E	0	0	1	0		1
F	0	0	1	0	0	

构建比赛结果矩阵

根据题目所给比赛结果构建矩阵 A，矩阵元素 $A(i,j)$ 表示球队 j 战胜球队 i 的情况（1 表示胜，0 表

示负)。

归一化处理

对矩阵 A 按列进行归一化，若某列元素和大于 0，则该列元素除以其列元素之和，使每列元素和为 1。数学表达式为：若 $\sum_{i=1}^{n} A(i,j) > 0$，则 $A(i,j) = \frac{A(i,j)}{\sum_{i=1}^{n} A(i,j)}$，其中 n 为矩阵 A 的行数（即球队数量）。

构建矩阵 M

设置阻尼系数 $p = 0.85$，结合归一化后的 A 和初始得分矩阵 S 构建矩阵 M，模拟比赛结果对球队得分的影响。具体构建方式为 $M = pA + (1-p)S$。

幂法迭代

设置收敛阈值 tol 和最大迭代次数 max_iter，从初始得分向量 r 开始迭代，每次迭代更新得分向量 r。迭代公式为 $r^{(k+1)} = Mr^{(k)}$，其中 k 表示迭代次数。当相邻两次迭代的得分向量差值的范数小于阈值时停止迭代，即当 $\|r^{(k+1)} - r^{(k)}\| < tol$ 时停止。

计算排名

对最终得分向量 r 归一化后，按得分从高到低排序得到排名 $ranking$。并输出各球队排名、名称和得分，从而确定各球队在比赛中的名次情况。

6.2 附加任务的实现

使用代码实现上述过程，结果如表 5 所示。

排名	球队	得分
1	C	0.26947
2	A	0.18760
3	E	0.16918
4	F	0.13953
5	B	0.13165
6	D	0.10258

表 5: 6 支篮球队比赛排名结果

7 附录

7.1 附录一：任务 1,2 核心代码

```
A = [0 1 1 0;
     0 0 0 1;
     1 0 0 0;
     0 1 1 0];
n = size(A, 1);
```

```
p_values = [0.75, 0.8, 0.9]; % 不同的p值

for i = 1:length(p_values)
    p = p_values(i);
    S = ones(n)/n;
    M = p*A + (1 - p)*S;
   [V,D]=eig(M);
diag(D);

% 计算排名
[~, ranking] = sort(x, 'descend');
```

7.2 附录 2：任务 3 核心代码

```
% 构建邻接矩阵
A = zeros(10);
A([2 3 5 6 9 10], 1) = 1;
A([1 3 5 6 8], 2) = 1;
A([4 6 7 8 10], 3) = 1;
A(10, 4) = 1;
A([5 6 9], 7) = 1;
A([5 6 9], 8) = 1;
A([5 6 10], 9) = 1;

% 构建转移矩阵
col_sums = sum(A, 1);
col_sums(col_sums == 0) = 1;
P = bsxfun(@rdivide, A, col_sums);

% 幂法参数设置
d = 0.85; % 阻尼系数
n = 10;    % 节点个数
r = ones(n, 1) / n; % 初始 PageRank 向量
tol = 1e-6; % 收敛阈值
max_iter = 1000; % 最大迭代次数

% 幂法迭代
for k = 1:max_iter
```

```
    r_prev = r;
    r = (1 - d) * ones(n, 1) / n + d * P * r;
    if norm(r - r_prev) < tol
        break;
    end
end

% 排序处理
[ranked_values, indices] = sort(r, 'descend');
ranking = (1:n)';
```

7.3 附录 3：任务 4 核心代码

```
% 构建邻接矩阵
A = zeros(10);
A([2 3], 1) = 1;
A([3 5], 2) = 1;
A([4 6 7], 3) = 1;
A(5, 4) = 1;
A(6, 5) = 1;
A([1 8], 7) = 1;
A([2 5 9], 8) = 1;
A([5 10], 9) = 1;
A([1 6], 10) = 1;

% 目标网页
target_page = 9;

% 方法1: 增加入链接数量
A1 = A;
A1(target_page, 6) = 1;
A1(target_page, 5) = 1;

% 方法2: 调整出链接分布
A2 = A;
A2(6, 3) = 0;
A2(5, 4) = 0;
```

```
% 方法3: 构建链接环路
A3 = A;
A3(6, target_page) = 1;
A3(target_page, 6) = 1;

% 计算PageRank的函数
function r = compute_pagerank(A, d, tol, max_iter)
    % 构建转移矩阵
    col_sums = sum(A, 1);
    col_sums(col_sums == 0) = 1;
    P = bsxfun(@rdivide, A, col_sums);

    % 初始化PageRank向量
    n = size(A, 1);
    r = ones(n, 1) / n;

    % 幂法迭代
    for k = 1:max_iter
        r_prev = r;
        r = (1 - d) * ones(n, 1) / n + d * P * r;
        if norm(r - r_prev) < tol
            break;
        end
    end
end

% 参数设置
d = 0.85;          % 阻尼系数
tol = 1e-6;        % 收敛阈值
max_iter = 1000; % 最大迭代次数

% 计算原始网络的PageRank
r_original = compute_pagerank(A, d, tol, max_iter);

% 计算三种方法的PageRank
r_method1 = compute_pagerank(A1, d, tol, max_iter);
r_method2 = compute_pagerank(A2, d, tol, max_iter);
```

```
r_method3 = compute_pagerank(A3, d, tol, max_iter);

% 比较目标网页的排名提升
[~, original_rank] = sort(r_original, 'descend');
[~, rank1] = sort(r_method1, 'descend');
[~, rank2] = sort(r_method2, 'descend');
[~, rank3] = sort(r_method3, 'descend');

% 查找目标网页在各排名中的位置
original_pos = find(original_rank == target_page);
pos1 = find(rank1 == target_page);
pos2 = find(rank2 == target_page);
pos3 = find(rank3 == target_page);

% 比较目标网页的PageRank值
value_original = r_original(target_page);
value1 = r_method1(target_page);
value2 = r_method2(target_page);
value3 = r_method3(target_page);
```

7.4 附录 4: 任务 5 核心代码

```
load Harvard500.mat
whos
A = Problem.A;
n = size(A, 1);
d = 0.85;
v = ones(n, 1) / n;

% 处理没有出链的页面 (即行和为0的情况)
rowsum = sum(A, 2);
no_outlinks = (rowsum == 0);
A(no_outlinks, :) = 1/n;

P = A ./ sum(A, 2);

% 计算PageRank (使用幂迭代法)
max_iter = 100;
```

```
tolerance = 1e-8;
x = v;
for iter = 1:max_iter
    x_old = x;
    x = d * P' * x + (1-d) * v;
    if norm(x - x_old, 1) < tolerance
        fprintf('PageRank算法在第␣%d␣次迭代后收敛\n', iter);
        break;
    end
end

% 按PageRank值排序
[rank_values, rank_indices] = sort(x, 'descend');

% 显示排名前十的网站
fprintf('排名\tPageRank值\t网站索引\n');
for i = 1:50
    fprintf('%d\t%.6f\t\t%d\n', i, rank_values(i), rank_indices(i));
end

% 可视化排名分布
figure;
bar(1:n, x(rank_indices));
title('哈佛大学相关网站PageRank排名分布');
xlabel('网站排名');
ylabel('PageRank值');
grid on;
```

7.5 附录 5：附加任务核心代码

```
% 构建比赛结果矩阵
A = [0 1 0 1 1 1;
     0 0 0 1 1 1;
     1 1 0 1 0 0;
     0 0 0 0 1 1;
     0 0 1 0 0 1;
     0 0 1 0 0 0];
n = size(A, 1); % 球队数量
```

```
% 归一化处理
for i = 1:n
    if sum(A(:,i)) > 0
        A(:,i) = A(:,i)/sum(A(:,i));
    end
end
S = ones(n)/n; % 初始得分矩阵

% 设置阻尼系数
p = 0.85;

% 构建矩阵M
M = p*A + (1 - p)*S;

% 幂法迭代参数设置
tol = 1e-6; % 收敛阈值
max_iter = 1000; % 最大迭代次数
r = ones(n, 1) / n; % 初始得分向量

% 幂法迭代
for k = 1:max_iter
    r_prev = r;
    r = M * r;
    if norm(r - r_prev) < tol
        break;
    end
end

% 归一化最终得分向量
r = r / sum(r);

% 计算排名
[~, ranking] = sort(r, 'descend');
```