PageRank Algorithm Implementation

Gerardo Gomez
CS470 - Data Mining
HW4

November 14, 2025

1 Test Cases

Five directed graphs were created to test the PageRank implementation under different conditions. Each graph was designed to test specific features of the algorithm, particularly how it handles dead ends and spider traps.

1.1 Graph 1: Small Symmetric Graph

A simple 5-vertex graph with vertices A through E was created to verify the basic algorithm implementation. The graph is perfectly symmetric with 10 edges, where every vertex has both incoming and outgoing edges. No dead ends or spider traps are present.

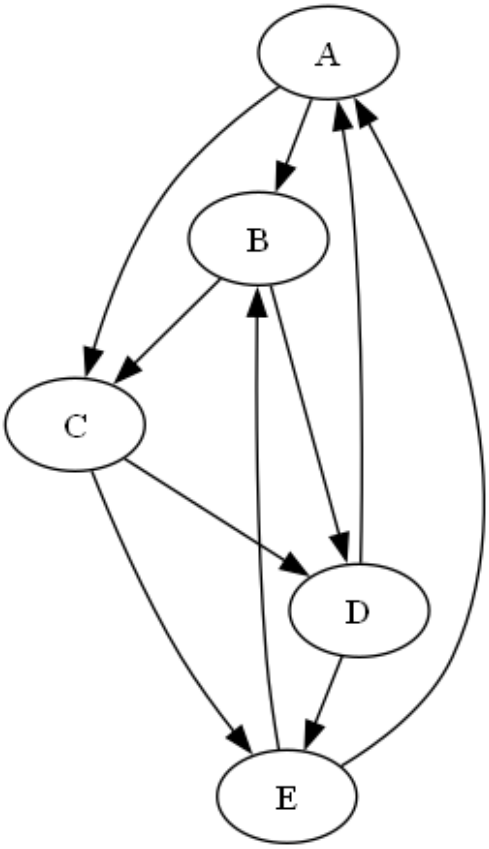

Figure 1: Visualization of Graph 1

The symmetric structure means that each vertex should be equally important, with all five vertices expected to have a PageRank of approximately 0.20.

1.2 Graph 2: Medium Connected Graph

A larger graph with 10 vertices (V1 through V10) and 20 edges was created. The graph is intentionally asymmetric with multiple cycles and paths to test how the algorithm distributes importance unevenly. No dead ends or spider traps are present.

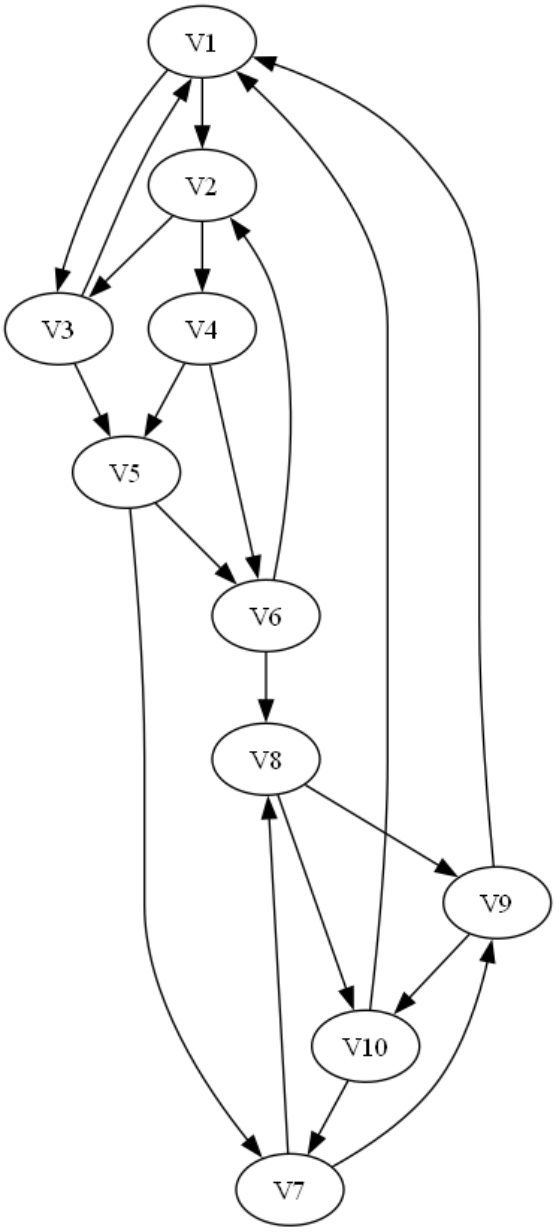

Figure 2: Visualization of Graph 2

V1 was set up to receive links from several important vertices, so it was predicted to have the highest PageRank.

1.3 Graph 3: Testing Dead Ends

A graph similar to Graph 2 was created, but all outgoing edges from V10 were removed, making it a dead end. The graph has 10 vertices and 17 edges. This tests whether the algorithm can properly handle a vertex that receives PageRank but cannot pass it along.

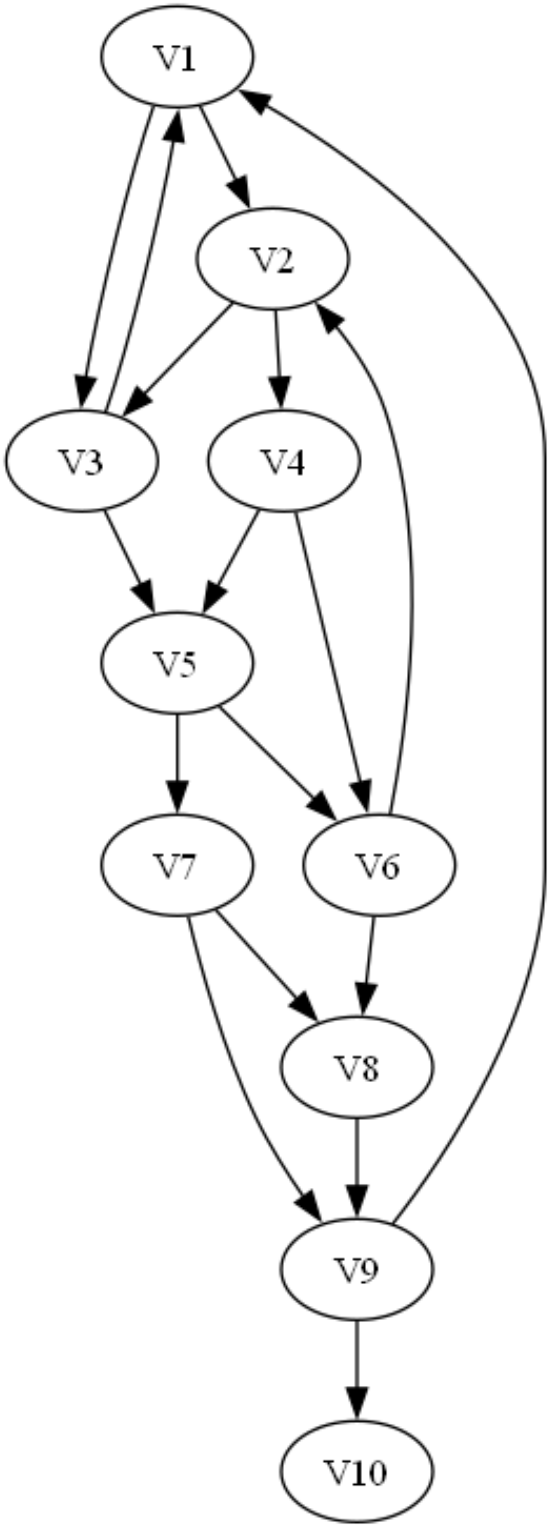

Figure 3: Visualization of Graph 3 - V10 has no outgoing edges

Without proper handling, V10 would act as a sink that absorbs PageRank without re-distributing it.

1.4 Graph 4: Testing Spider Traps

A spider trap was created using vertices V7, V8, and V9. These three vertices form a cycle where they only link to each other, with no edges leading out. Edges from V1 and V10 lead into this trap. The graph has 10 vertices total and 16 edges.

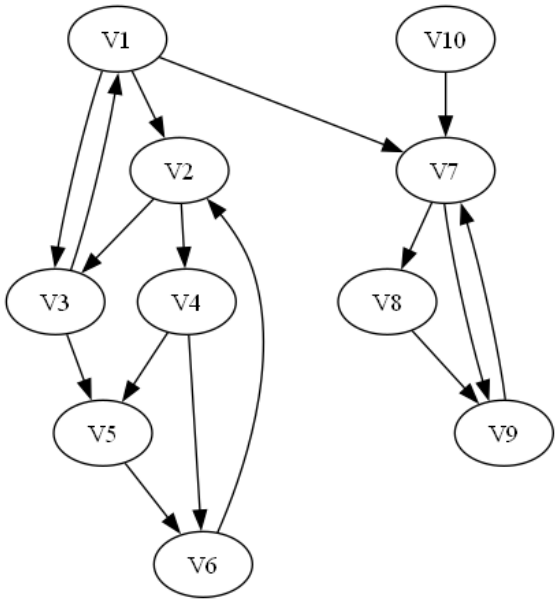

Figure 4: Visualization of Graph 4 - Spider trap formed by V7, V8, V9

The spider trap vertices were expected to accumulate high PageRank values because once PageRank flows in, it cannot escape through normal link following.

1.5 Graph 5: Large Complex Graph

A larger graph with 50 vertices and 63 edges was created to test scalability. This graph has a complex structure with tree-like branches, multiple cycles, a small spider trap (V37-V38-V39), and a long chain from V40 to V49. Both dead ends and spider traps are included.

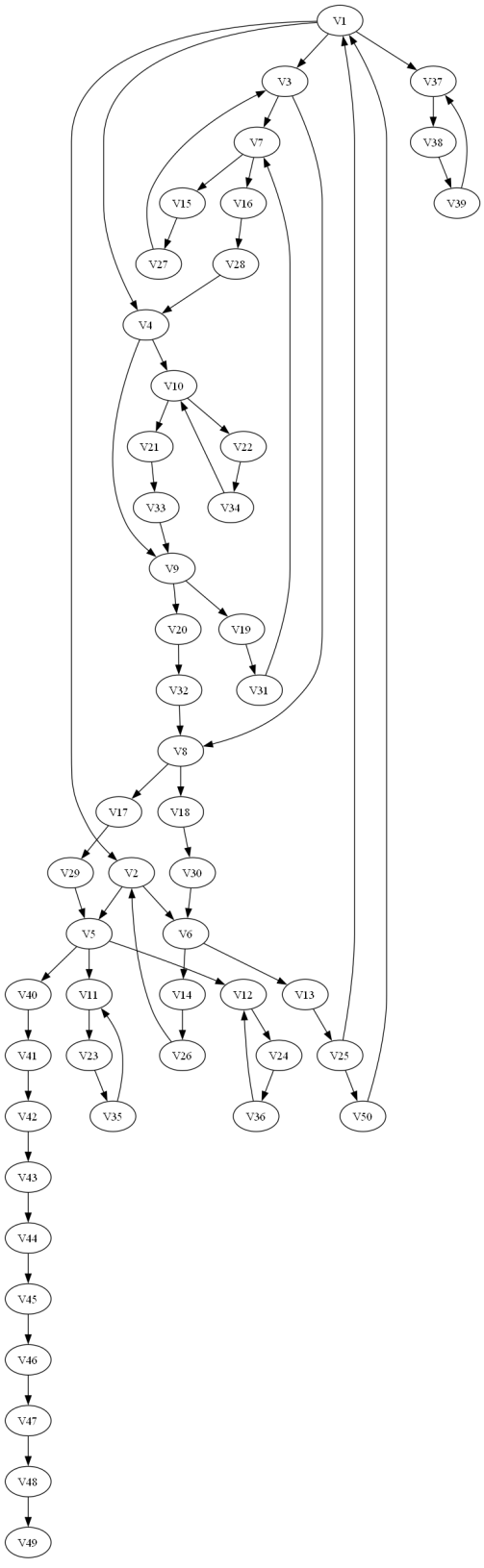

Figure 5: Visualization of Graph 5 - Complex 50-vertex graph

2 Implementation

The PageRank algorithm was implemented in Python using only standard libraries (sys, re, and typing). The core algorithm uses the iterative power method with the formula:

$$PR(v) = \frac{1-d}{N} + d \sum_{u \in In(v)} \frac{PR(u)}{Out(u)}$$

where d is the damping factor (0.85 was used), N is the total number of vertices, $In(v)$ represents vertices linking to v, and $Out(u)$ is the number of outgoing edges from u.

2.1 Handling Dead Ends

Dead ends create "sinks" where PageRank gets stuck. All dead end vertices are identified at the start, then at each iteration, their PageRank is distributed equally to all vertices in the graph. This simulates a random surfer who, when stuck at a dead end, randomly jumps to any page with equal probability.

A teleportation contribution from all dead ends is calculated:

$$teleport_{contrib} = \sum_{v \in DeadEnds} \frac{PR(v)}{N}$$

This is added to every vertex's PageRank during each iteration, maintaining the probabilistic interpretation of the random surfer model.

2.2 Handling Spider Traps

Spider traps are strongly connected components that have no outgoing edges to the rest of the graph. The damping factor handles these naturally. With a damping factor of 0.85, there is a 15% chance at each step that the random surfer teleports to a random page instead of following a link.

This allows PageRank to escape from spider traps through random jumps, and every vertex gets a base PageRank of $\frac{1-d}{N}$ from the teleportation term. Without the damping factor, spider traps would eventually absorb all the PageRank.

The convergence threshold was set to 10^{-8} and maximum iterations to 100.

3 Results

Tables 1 through 5 show the PageRank values computed for each graph, sorted by PageRank (descending) and then by vertex name for ties.

3.1 Graph 1 Results

Vertex	PageRank
A	0.2000
B	0.2000
C	0.2000
D	0.2000
E	0.2000

Table 1: PageRank values for Graph 1

All vertices have exactly equal PageRank (0.20 each), as predicted. This confirms that the basic algorithm implementation is correct. The symmetric structure resulted in every vertex being equally important.

3.2 Graph 2 Results

Vertex	PageRank
V1	0.1480
V3	0.1259
V2	0.1129
V5	0.0953
V7	0.0951
V9	0.0831
V8	0.0773
V10	0.0765
V6	0.0725
V4	0.0635

Table 2: PageRank values for Graph 2

V1 has the highest PageRank of 0.148, which is expected because it receives links from multiple important vertices like V3, V9, and V10. The asymmetric structure created a varied distribution of PageRank values.

3.3 Graph 3 Results (Dead End)

Vertex	PageRank
V1	0.1283
V9	0.1261
V3	0.1255
V2	0.1168
V5	0.1049
V7	0.0866
V8	0.0847
V6	0.0783
V10	0.0752
V4	0.0736

Table 3: PageRank values for Graph 3 (with dead end V10)

The dead end vertex V10 has a relatively low PageRank (0.0752) because it has no outgoing edges. The dead end handling worked correctly—instead of becoming a sink, V10 redistributed its PageRank to all vertices equally. The overall distribution is similar to Graph 2 but adjusted for the missing edges.

3.4 Graph 4 Results (Spider Trap)

Vertex	PageRank
V7	0.1970
V9	0.1826
V2	0.1221
V6	0.1096
V8	0.0987
V3	0.0880
V1	0.0752
V5	0.0616
V4	0.0426
V10	0.0227

Table 4: PageRank values for Graph 4 (with spider trap V7-V8-V9)

The spider trap vertices V7, V9, and V8 dominate with values of 0.197, 0.183, and 0.099. This demonstrates how spider traps absorb PageRank. However, due to the damping factor of 0.85, the other vertices maintained reasonable PageRank values. Without the damping factor, these three vertices would hold nearly all the PageRank.

3.5 Graph 5 Results

Vertex	PageRank
V11	0.0403
V12	0.0403
V23	0.0376
V24	0.0376
V35	0.0353
...(top 5 of 50 shown)	

Table 5: PageRank values for Graph 5 (top 5 vertices shown)

With 50 vertices, the PageRank is much more evenly distributed—the highest value is only 0.040 compared to 0.20 in the base case. V11 and V12 tied for the top spot because they are positioned in the middle of feedback cycles with multiple incoming links. The algorithm handled the complex structure well, including the embedded spider trap and various dead ends.

4 Discussion and Conclusions

The damping factor was found to be crucial for two purposes: it prevents spider traps from absorbing all PageRank, and it ensures every vertex gets some minimum PageRank value. Without it, the algorithm would fail on graphs with spider traps.

The dead end handling worked correctly. By redistributing dead end PageRank equally to all vertices, the random surfer interpretation was maintained while preventing PageRank from getting stuck.

The algorithm converged quickly in all test cases. Even the complex 50-vertex graph only took about 15-20 iterations to reach convergence with a threshold of 10^{-8}. Simple graphs like Graph 1 converged in just 10 iterations.

Perfectly symmetric graphs (like Graph 1) produced equal PageRank values, which serves as a verification of implementation correctness. The asymmetric graphs showed how link structure creates natural hierarchies of importance.

The spider trap test case (Graph 4) demonstrated the effectiveness of the damping factor. The trap vertices accumulated the most PageRank, but did not absorb everything. Without the damping factor, these vertices would incorrectly appear as the most important pages.

PageRank was successfully implemented with proper handling of dead ends and spider traps. The algorithm correctly identified important vertices in all test cases, from simple 5-vertex graphs to complex 50-vertex structures. The implementation is deterministic and initializing with equal PageRank values (1/N) was effective.

The current $O(N^2)$ implementation would not scale to web-sized graphs with millions of vertices. A sparse matrix version would be needed for better efficiency on very large graphs.