```
import numpy as np

def build_transition_matrix(filename):
    with open(filename, 'r') as file:
        links = file.readlines()

    pages = set()
    for link in links:
        parts = link.strip().split()
        if not parts:
            continue
        if len(parts) < 2:
            print("error formating!")
            continue
        pages.add(parts[0])
        pages.update(parts[1:])

    pages = list(pages)
    page_index = {page: i for i, page in enumerate(pages)}
    num_pages = len(pages)

    M = np.zeros((num_pages, num_pages))

    for link in links:
        parts = link.strip().split()
        if not parts or len(parts) < 2:
            continue
        source_index = page_index[parts[0]]
        num_links = len(parts) - 1
        if num_links > 0:
            for target in parts[1:]:
                target_index = page_index[target]
                M[target_index][source_index] = 1.0 / num_links

    print("Transition Matrix:")
    print(M)

    return M, page_index, pages

def matrix_vector_multiply(matrix, vector):
    result = np.dot(matrix, vector)
    print('\n')
    print("Matrix-Vector Multiplication Result:")
    print(result)
    return result

def pagerank(M, num_iterations=10, damping_factor=0.85):
    num_pages = M.shape[0]
    rank_vector = np.ones(num_pages) / num_pages

    for iteration in range(num_iterations):
```

```
        new_rank_vector = (1 - damping_factor) / num_pages + damping_factor *
matrix_vector_multiply(M, rank_vector)
        print('\n')
        print(f"Iteration {iteration + 1}:\n")
        print(new_rank_vector)
        rank_vector = new_rank_vector

    return rank_vector

def print_pagerank_scores(rank_vector, page_index):
    index_page = {v: k for k, v in page_index.items()}
    sorted_pages = sorted(((index_page[i], rank) for i, rank in enumerate(rank_vector)), key=lambda
x: -x[1])

    print("PageRank Scores:")
    for page, score in sorted_pages:
        print(f"{page}: {score}")

if __name__ == "__main__":
    M, page_index, pages = build_transition_matrix('links.txt')
    rank_vector = pagerank(M)
    print_pagerank_scores(rank_vector, page_index)
```