ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

«Структуры и алгоритмы обработки данных»

на тему

Алгоритмы ранжирования. PageRank

Выполнил студент Д.А. Мильтов
Ф.И.О.

Группы ИС-341

Работу принял _____ ассистент Кафедры ВС А. О. Насонова
подпись

Защищена _____ Оценка _____

Новосибирск – 2024

ВВЕДЕНИЕ

Всемирная паутина (WWW) быстро развивается во всех аспектах и представляет собой массивное, разнообразное, динамичное и в основном неструктурированное хранилище данных. На сегодняшний день WWW является огромным информационным хранилищем справочных данных [1]. Когда пользователь вводит запрос с использованием ключевых слов в интерфейсе поисковой системы, компонент обработки запросов сопоставляет ключевые слова запроса с индексом и возвращает пользователю URL-адреса страниц. Но прежде, чем показывать страницы пользователю, поисковые системы запускают механизм ранжирования, чтобы наиболее релевантные страницы отображались вверху, а менее релевантные - внизу [1].

Одним из известных алгоритмов является PageRank, разработанный Ларри Пейджем и Сергеем Брином, который стал основой для Google.

В данной работе рассмотрены основные принципы работы этого алгоритма, его вычислительная сложность, а также проведение экспериментальных исследований.

1. Алгоритмы ранжирования

1.1 Основы алгоритмов ранжирования

Алгоритмы ранжирования являются основополагающими для построения современных поисковых систем. Ключевая идея ранжирования состоит в определении порядка, в котором результаты должны быть показаны пользователю,

основываясь на их релевантности (насколько результат соответствует запросу) и значимости (общей важности элемента в системе) [3].

Они могут быть основаны на различных принципах, включая:

- Контент-анализ (анализ текстов, метаданных и других атрибутов объекта).
- Probabilistic Ranking Models (Основаны на вероятностной модели релевантности: каждому документу d приписывается вероятность P(d|q) быть релевантным запросу q, как в BM25).
- Ссылочные связи (учет количества и качества ссылок на объект, как в PageRank).
- Машинное обучение (учет множества факторов для оптимального упорядочивания).

Методы, связанные с ранжированием:

- Жадные алгоритмы и их применение в задачах оптимизации, включая построение кратчайших путей, которые могут быть основой для выбора релевантных документов [4].
- Графовые алгоритмы, особенно важные в системах ранжирования, таких как PageRank, где страницы веб-графа моделируются вершинами, а ссылки — рёбрами [5].

1.2 Алгоритм PageRank

Алгоритм PageRank основывается на концепции, согласно которой, если страница содержит важные входящие ссылки, то и ссылки этой страницы на другие ресурсы также считаются важными [3]. Алгоритм учитывает обратные ссылки при определении значения ранга страницы [6]. Если сумма рангов всех обратных ссылок

страницы велика, то и её ранг будет высоким [6]. Таким образом, PageRank предлагает более усовершенствованный способ вычисления важности или релевантности веб-страницы, чем просто подсчёт числа страниц, ссылающихся на неё [3].

Если обратная ссылка поступает с важной страницы, то она получает больший вес, чем ссылки, поступающие с менее значимых страниц [3]. В упрощённой форме можно считать, что ссылка с одной страницы на другую является своего рода "голосом" [3]. Однако важны не только количество "голосов", которые получает страница, но также значение или релевантность тех, кто эти голоса подает [3].

Предположим, что страница A имеет страницы T1,T2,…,Tn, которые на неё ссылаются (т.е. это входящие ссылки). Переменная d называется фактором затухания, значение которого находится в диапазоне от 0 до 1. Обычно устанавливается значение d=0,85. PR(T1) — это значение ранга страницы, ссылающейся на страницу A, а C(T1) — количество исходящих ссылок со страницы T1 (например, PR(T1)) [1].

Формула для вычисления PageRank страницы A выглядит следующим образом:

$$PR(A) = (1 - d) + d(PR(T1)/C(T1) + \cdots + PR(Tn/C(Tn)) \quad (1)$$

Упрощённая интерпретация PageRank:

$$PR(N) = \sum_{m \in Bn} R(M)/L(M) \quad (2)$$

Где значение PageRank для веб-страницы N зависит от значений PageRank всех страниц M из множества Bn (этот набор включает все страницы, ссылающиеся на веб-страницу N), делённое на число L(M) ссылок со страницы M [2].

Пример обратной ссылки показан на рисунке 1: страница N является обратной ссылкой для M и Q, а M и Q являются обратными ссылками для страницы O.

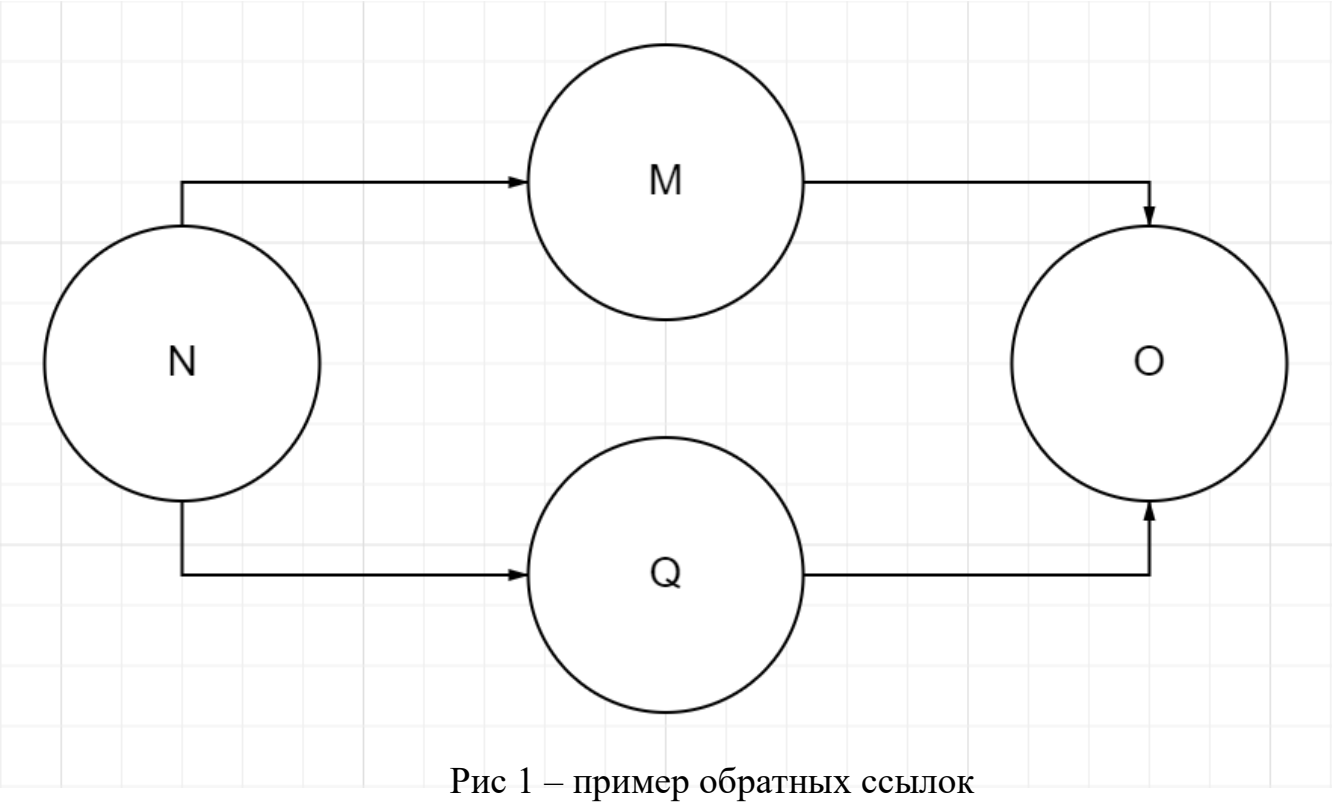

Рис 1 – пример обратных ссылок

Рассмотрим пример структуры гиперссылок на четырех страницах A, B, C и D, как показано на рисунке 2.

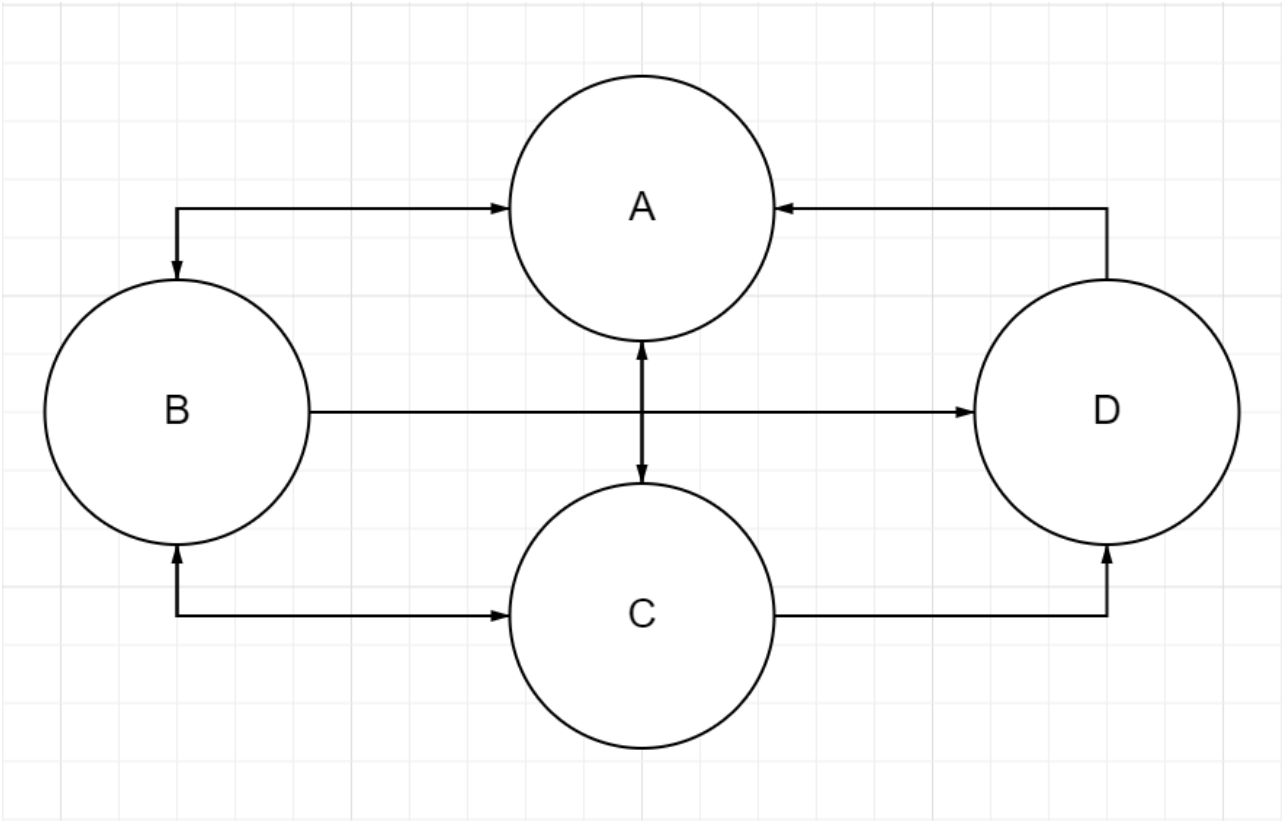

Рис 2 – Структура гиперссылок для четырёх страниц

PageRank для страниц A, B, C и D может быть рассчитан с помощью (1). Предположим, что начальный PageRank равен 1, и произведем расчет. Значение коэффициента демпфирования d равно 0,85.

$$PR(A) = (1 - d) + d\big(PR(B)/C(B) + PR(C)/C(C) + PR(D)/C(D)\big) =$$

$$= (1 - 0.85) + 0.85(1/3 + 1/3 + 1/1) = 1.5666667$$

$$(3) PR(B) = (1 - d) + d((PR(A)/C(A) + (PR(C)/C(C)) =$$

$$= (1 - 0.85) + 0.85(1.5666667/2 + 1/3) = 1.0991667$$

$$(4) \quad PR(C) = (1 - d) + d((PR(A)/C(A) + (PR(B)/C(B)) =$$

$= (1 - 0.85) + 0.85(1.5666667/2 + 1.0991667/3) = 1.127264$ (5) $PR(D) = (1 - d) + d((PR(B)/C(B) + (PR(C)/C(C)) =$

$= (1 - .085) + 0.85(1.0991666/3 + 1.127264/3) = 0.7808221$
(6)

Для второй итерации, используя приведенные выше значения PageRank из (3), (4), (5) и (6). Значения PageRank для второй итерации следующие:

$PR(A) = 0.15 + 0.85((1.0991667/3) + (1.127264/3) + (0.7808221/1) =$
1.4445208 (7)

$PR(B) = 0.15 + 0.85\big((1.4445208/2) + (1.127264/3)\big) = 1.0833128$
(8)

$PR(C) = 0.15 + 0.85\big((1.4445208/2) + (1.0833128/3)\big) = 1.07086$
(9)

$PR(D) = 0.15 + 0.85\big((1.0833128/3) + (1.07086/3)\big) = 0.760349$
(10)

Во время вычисления на 34-й итерации среднее значение для всех веб-страниц равно 1. Некоторые значения PageRank приведены в таблице 1.

Iteration	A	B	C	D
1	1	1	1	1

2	1.5666667	1.0991667	1.127264	0.7808221
3	1.4445208	1.0833128	1.07086	0.760349
..	..	..	..	..
..	..	..	..	..
17	1.3141432	0.9886763	0.9886358	0.7102384
18	1.313941	0.9885384	0.98851085	0.71016395
19	1.3138034	0.98844457	0.98842573	0.7101132

Таблица 1 - Вычисления PageRank

Как видно из таблицы 1, PageRank страницы A выше, чем у страниц B, C и D. Это связано с тем, что страница A имеет три входящие ссылки, тогда как страницы B, C и D имеют по две, как показано на рисунке 2. При этом страница B имеет две входящие и три исходящие ссылки, страница C — две входящие и три исходящие ссылки, а страница D — одну входящую и две исходящие ссылки. Из данных таблицы 1 следует, что после 34-й итерации значения PageRank для всех страниц нормализуются [1].

Следует отметить, что рейтинг страницы распределяется равномерно между всеми её внешними ссылками, что позволяет повышать значимость связанных страниц. Исходное уравнение PageRank имеет рекурсивный характер: процесс вычисления начинается с установленного по умолчанию значения PageRank (например, 1) и продолжается через итерации до тех пор, пока значения PageRank для всех страниц не стабилизируются, то есть не станут повторяться, и в итоге их среднее значение будет равно 1 [3].

Алгоритм PageRank вычисляется с помощью итерационного метода и связан с основным собственным вектором нормализованной матрицы ссылок в веб-графе. Для обработки миллионов страниц алгоритму требуется несколько часов, но он обеспечивает эффективное ранжирование большого количества веб-страниц. Хотя для небольших наборов страниц вычисление PageRank не представляет сложности, для крупных веб-сайтов или систем с миллиардом страниц реализация этого процесса становится значительно сложнее [3].

1.3 Область применения

Основная область применения — использование PageRank в поисковых системах [3]. Упор сделан на том, как алгоритм ранжирует веб-страницы на основе ссылочной структуры, оценивая их значимость для предоставления пользователям релевантных результатов поиска [3]. Научное цитирование, PageRank рассматривается как пример алгоритма на графах, который может быть адаптирован для анализа цитирования научных работ [4]. Это связано с оценкой значимости научных публикаций на основе структуры ссылок (аналогично тому, как анализируются ссылки в веб-графе) [4]. Транспортные и инфраструктурные сети,

адаптация PageRank для оценки значимости узлов в сетях, таких как транспортные или коммуникационные сети [5].

1.4 Выводы

Алгоритм PageRank вычисляется с помощью итерационного метода и связан с основным собственным вектором нормализованной матрицы ссылок в веб-графе [3]. Для обработки миллионов страниц алгоритму требуется несколько часов, но он обеспечивает эффективное ранжирование большого количества веб-страниц. Хотя для небольших наборов страниц вычисление PageRank не представляет сложности, для крупных веб-сайтов или систем с миллиардом страниц реализация этого процесса становится значительно сложнее [3]. Вычислительная сложность $O(n^2)$, где n — количество узлов.

2. Экспериментальное исследование эфективности алгоритма

2.1 Организация экспериментального исследования

Была поставлена задача выполнить асимптотический анализ заявленной вычислительной сложности. Как время выполнения изменяется при увеличении размера графа. Для этого генерировались графы с возрастающим размером, которые исполняли алгоритм PageRank. Далее время, полученное после выполнения алгоритма, сравнивалось с размером графа.

2.2 Результаты экспериментального исследования

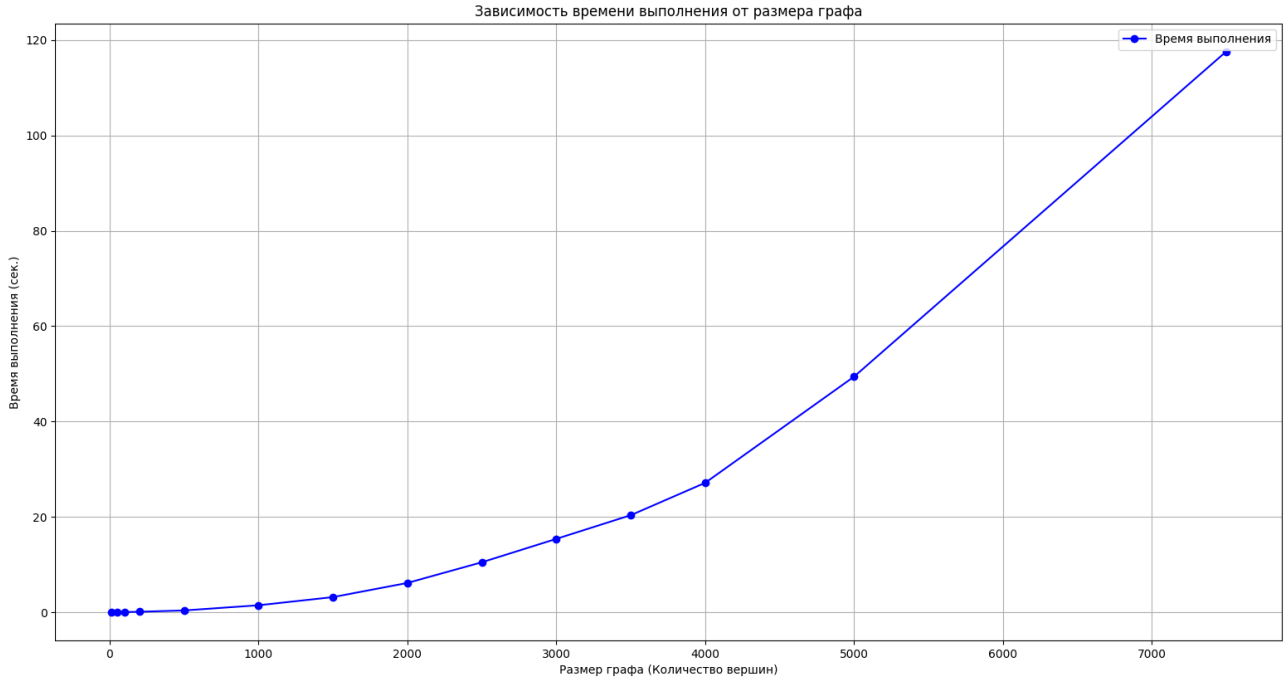

Рис 3.1 – График зависимости времени выполнения от размера графа

На рисунке 3.1 изображена кривая, являющаяся заявленной вычислительной сложностью алгоритма, а именно O(n^2).

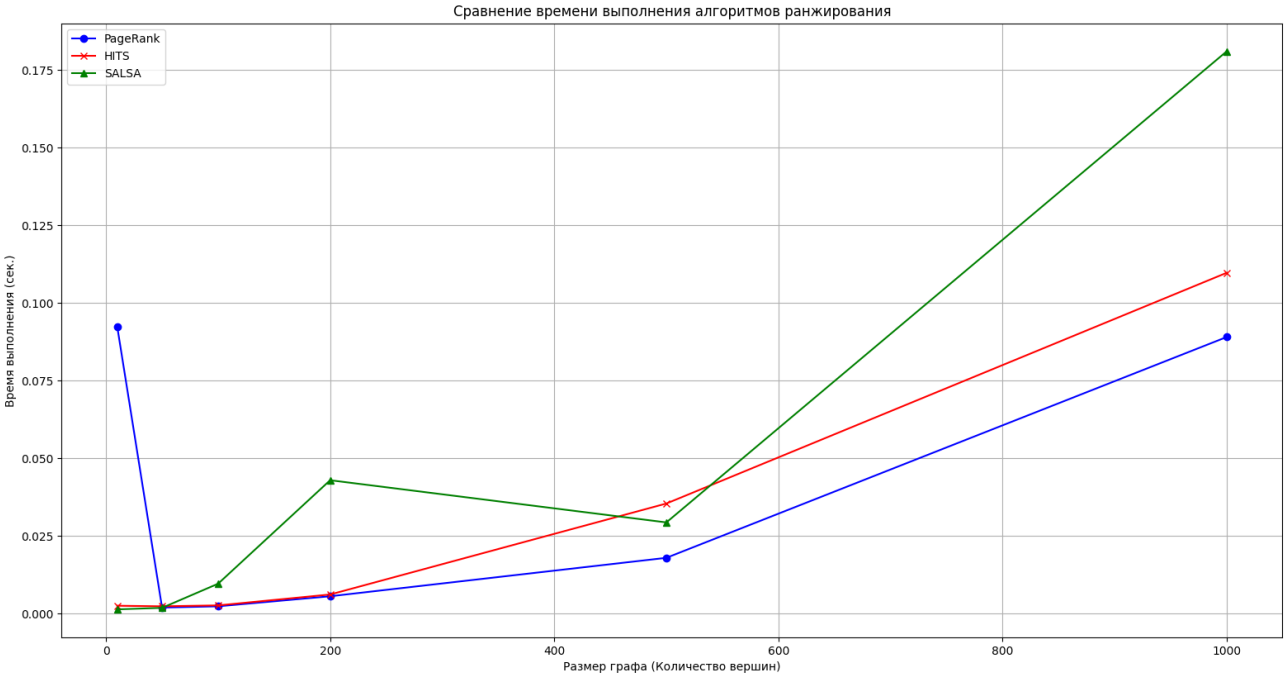

Рис 3.2 – График сравнения времени выполнения разных алгоритмов

На рисунке 3.2 изображены другие алгоритмы ранжирования, которые позволяют сравнить скорость выполнения, это позволяет сказать, что PageRank хорошо справляется с небольшими графами, но при увеличении размеров падает скорость выполнения.

ЗАКЛЮЧЕНИЕ

В результате работы реализован и исследован алгоритм ранжирования PageRank. Было рассмотрено уравнение алгоритма, разработанного Ларри Пейджем

и Сергеем Брином. Также в результате работы были проведены экспериментальные исследования, которые доказали, поставленную вычислительную сложность O(n^2). Но также был сделан вывод, что использование, выбранного алгоритма будет недостаточно эффективным при большом количестве страниц.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Laxmi Choudhary, Bhawani Shankar Burdak . // " Role of Ranking Algorithms for Information Retrieval" International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.4, July 2012*
2. *GeeksforGeeks. // "Page Rank Algorithm and Implementation"[Электронный ресурс] URL: https://www.geeksforgeeks.org/page-rank-algorithm-implementation/*
3. *Маннинг К., Рагхаван П., Шютце Х. // "Введение в информационный поиск." Перевод с английского языка-М.: Издательский дом «Вильямс». – 2011.*

4. *Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К. // "Алгоритмы: построение и анализ." – 3-е изд. Перевод с английского языка-М.: Издательский дом «Вильямс. – 2013.*
5. *Левитин А. В. // "Алгоритмы: введение в разработку и анализ." Перевод с английского языка-М.: Издательский дом «Вильямс». – 2006.*
6. *Скиена С. С. // "Алгоритмы. Руководство по разработке." – 2-е изд. – Перевод с английского языка-М.: СПб: БХВ-Петербург. – 2011.*

ПРИЛОЖЕНИЕ

Main.cpp

```cpp
#include <iostream>
#include <vector>
#include <unordered_map>
#include <fstream>
#include <cmath>
#include <omp.h>

const double DAMPING_FACTOR = 0.85;
const double EPSILON = 1e-6;
const int MAX_ITERATIONS = 100;

// Функция для расчета PageRank
std::unordered_map<int, double> calculatePageRank(
    const std::vector<std::vector<int>>& graph,
    const std::unordered_map<int, double>& personalizedVector = {},
    const std::unordered_map<int, std::unordered_map<int, double>>& linkWeights = {})
{
```

```
    int numPages = graph.size();
    std::unordered_map<int, double> rank;
    std::unordered_map<int, double> newRank;

    for (int i = 0; i < numPages; i++) {
        rank[i] = 1.0 / numPages;
    }

    for (int iter = 0; iter < MAX_ITERATIONS; iter++) {
        for (int i = 0; i < numPages; i++) {
            newRank[i] = (1.0 - DAMPING_FACTOR) / numPages;
        }

        double danglingRank = 0.0;
        #pragma omp parallel for reduction(+:danglingRank)
        for (int i = 0; i < numPages; i++) {
            if (graph[i].empty()) {
                danglingRank += rank[i];
            }
        }

        #pragma omp parallel for
        for (int j = 0; j < numPages; j++) {
            newRank[j] += DAMPING_FACTOR * danglingRank / numPages;
        }

        #pragma omp parallel for
        for (int i = 0; i < numPages; i++) {
            int outDegree = graph[i].size();
            if (outDegree == 0) continue;

            for (int j : graph[i]) {
                double weight = linkWeights.count(i) && linkWeights.at(i).count(j) ?
linkWeights.at(i).at(j) : 1.0;
                newRank[j] += DAMPING_FACTOR * rank[i] * weight / outDegree;
            }
        }

        if (!personalizedVector.empty()) {
            for (const auto& [page, weight] : personalizedVector) {
                newRank[page] += weight * (1.0 - DAMPING_FACTOR);
```

```
            }
        }

        double diff = 0.0;
        #pragma omp parallel for reduction(+:diff)
        for (int i = 0; i < numPages; i++) {
            diff += std::fabs(newRank[i] - rank[i]);
            rank[i] = newRank[i];
        }

        if (diff < EPSILON) {
            std::cout << "Converged after " << iter + 1 << " iterations." <<
std::endl;
            break;
        }
    }

    return rank;
}

void exportGraphToDOT(const std::vector<std::vector<int>>& graph, const std::string&
filename) {
    std::ofstream file(filename);
    file << "digraph G {\n";
    for (size_t i = 0; i < graph.size(); i++) {
        for (int j : graph[i]) {
            file << "  " << i << " -> " << j << ";\n";
        }
    }
    file << "}\n";
    file.close();
}

int main() {
    std::vector<std::vector<int>> graph = {
        {1, 2},   // Страница 0 ссылается на 1 и 2
        {2},      // Страница 1 ссылается на 2
        {0},      // Страница 2 ссылается на 0
        {0, 2}    // Страница 3 ссылается на 0 и 2
    };

    std::unordered_map<int, std::unordered_map<int, double>> linkWeights = {
        {0, {{1, 1.5}, {2, 1.0}}},
```

```
        {3, {{0, 2.0}, {2, 0.5}}}
    };

    std::unordered_map<int, double> personalizedVector = {
        {0, 0.5},
        {2, 0.3}
    };

    auto rank = calculatePageRank(graph, personalizedVector, linkWeights);

    std::cout << "PageRank values:" << std::endl;
    for (const auto& [page, value] : rank) {
        std::cout << "Page " << page << ": " << value << std::endl;
    }

    exportGraphToDOT(graph, "graph.dot");
    std::cout << "Граф сохранён в 'graph.dot'." << std::endl;

    return 0;
}
```