Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РтФ

Школа профессионального и академического образования

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель: Мирвода С.Г.\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СИСТЕМЫ ГОЛОСОВАНИЙ. МЕТОД ШУЛЬЦЕ.**

Проект по модулю

по дисциплине "Алгоритмы обработки данных во внешней памяти"

Студент: Слобцов В.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО) (Подпись)

Группа: РИМ-171226

Екатеринбург

2018

Оглавление

[ВВЕДЕНИЕ 3](#_Toc516005746)

[ПРИНЦИП КОНДОРСЕ 4](#_Toc516005747)

[Парадокс Кондорсе 5](#_Toc516005748)

[МЕТОД ШУЛЬЦЕ 6](#_Toc516005749)

[ОПИСАНИЕ АЛГОРИТМА 7](#_Toc516005750)

[РЕАЛИЗАЦИЯ алгоритма 12](#_Toc516005751)

[Результат работы алгоритма 13](#_Toc516005752)

[ЗАКЛЮЧЕНИЕ 14](#_Toc516005753)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 15](#_Toc516005754)

[КОД ПРОГРАММЫ 16](#_Toc516005755)

# ВВЕДЕНИЕ

Применяющиеся в настоящее время системы, базирующиеся на абсолютном или относительном большинстве голосов, не могут обеспечить адекватного отражения желания даже участвовавших в выборах людей. Это было показано с помощью элементарных рассуждений французским математиком Николем де Кондорсе в 1785 году.

# ПРИНЦИП КОНДОРСЕ

Согласно принципу Кондорсе, для определения истинной воли большинства необходимо, чтобы каждый голосующий провел ранжировку всех кандидатов в порядке их предпочтения. После этого для каждой пары кандидатов определяется, сколько голосующих предпочитает одного кандидата другому, исходя из чего формируется полная матрица попарных предпочтений голосующих. Победителем становится кандидат, побеждающий всех при парном сравнении.

Рассмотрим принцип Кондорсе на примере ранжированных альтернатив a1,a2,a3,a4,a5. Сначала избирателя осуществляют ранжировку альтернатив, как на таблице 1. После чего находятся оценки mik, характеризующий предпочтения альтернатив в парных сравнениях. Оценки представлены в таблице 2 По принципу Кондорсе наилучшей является альтернатива а, если mik>=mki для всех k не равных I, чему удовлетворяет только вариант а1.

Таблица 1 – Ранжировка альтернатив избирателями.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **И1** | **И2** | **И3** | **И4** | **И5** |
| а1 | а1 | а1 | а2 | а2 |
| а3 | а2 | а2 | а3 | а4 |
| а2 | а4 | а5 | а1 | а3 |
| а5 | а3 | а3 | а5 | а1 |
| а4 | а5 | а4 | а4 | а5 |

Таблица 2 – Оценка предпочтения альтернатив.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| mik | a1 | a2 | a3 | a4 | a5 |
| a1 |  | 3 | 3 | 4 | 5 |
| a2 | 2 |  | 4 | 5 | 5 |
| a3 | 2 | 1 |  | 3 | 4 |
| a4 | 1 | 0 | 2 |  | 2 |
| a5 | 0 | 0 | 1 | 3 |  |

## Парадокс Кондорсе

При анализе реальных профилей предпочтения избирателей возникает цикл, так называемый парадокс Кондорсе, и тогда победитель отсутствует.

Парадокс Кондорсе заключается в возможной противоречивости коллективного выбора избирателей при транзитивности выбора каждого избирателя. Таким образом ранжировки разных избирателей могут вступать в парадоксальные противоречия друг с другом. Коллективная оценка вариантов становится цикличной. Подобный парадокс возникает в случаях выбора из трех и более вариантов.

Для примера парадокса рассмотрим результат ранжировки стратегий руководителями высшего звена Digital Equipment Corporation из книги Ричарда Румельта, представленный в таблице 3.

Таблица 3 – Результат ранжировки стратегий.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Алек** | **Беверли** | **Крейг** |
| «Железо» | 1 | 2 | 3 |
| «Чипы» | 2 | 3 | 1 |
| «Решения» | 3 | 1 | 2 |

На примере из таблицы можно вывести три цепочки.

* Железо > Чипы > Решения
* Решения > Железо > Чипы
* Чипы > Решения > Железо

Из этих трех цепочек получается три утверждения:

Чипы > Решения, Решения > Железо и Железо > Чипы,

которые вместе парадоксально противоречивы и цикличны. Не смотря на то, что каждый вариант обошел одного соперника, всех конкурентов никто победить не смогу. Поэтому победителя нет.

На практике метод Кондорсе реализован в системе голосования по методу Шульце в 1997 году.

# МЕТОД ШУЛЬЦЕ

При объективном подсчете она позволяет определить победителя с использованием бюллетеней для голосования с полным списком кандидатов, при помощи которых голосующие ранжируют представленные кандидатуры, указывая свои предпочтения. Формально этот метод способен определять победителя даже, когда согласно критерию Кондорсе его нет, а именно при наличии парадокса Кондорсе.

Существуют различные эвристики, позволяющие определить победителя голосования по таким исходным данным. Метод Шульце использует концепцию учета косвенных побед, которая наследует подход Кондорсе в том, что учитывает победы кандидата над всеми другими при непосредственных парных сопоставлениях, однако помимо прямого сравнения используются косвенные победы, составленные по цепочке прямых с учетом количественного определения силы пути по графу бинарного отношения парных побед по Кондорсе. Учет этого параметра позволяет выявить победителя из любой пары кандидатов.

Достоинством метода является более полный и тонкий учет реальных предпочтений избирателей в большинстве случаев достаточный для преодоления парадокса Кондорсе. Недостатком является сложность анализа вследствие полноты матрицы прямых побед и обилия вариантов косвенных побед.

# ОПИСАНИЕ АЛГОРИТМА

Первый этап в методе Щульце – нахождение попарных предпочтений для кандидатов. Предположим, что у нас имеется n бюллетеней B с m кандидатами C. Каждый избиратель в своей бюллетени ранжирует кандидатов, создавая список предпочитаемых кандидатов в порядке убывания предпочтений. В результате мы получаем следующий массив ранжировок.

6 x [A,C,D,B],

4 x [B,A,D,C],

3 x [C,D,B,A],

4 x [D,B,A,C],

4 x [D,C,B,A].

На основе этого массива списков проводится подсчет попарных сравнений. Получается матрица оценок как при применении метода Кондорсе, который мы рассматривали ранее.

При выборе из четырех кандидатов с участием 21 избирателя получаем следующее распределение списков предпочтений. Чтобы подсчитать массив предпочтений, мы находим, что кандидат А предпочтительнее кандидата В только в первой подборке, поэтому элемент в массиве для попарных предпочтений между А и В будет равен числу этих списков, т.е. 6. Продолжая мы получаем следующую матрицу предпочтений. Процесс составления матрицы оценок представлен на рисунке 1, а сама матрица на рисунке 2.

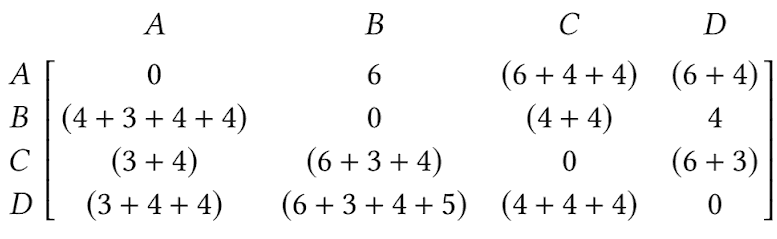


Рисунок 1 – Принцип подсчета матрицы оценок.

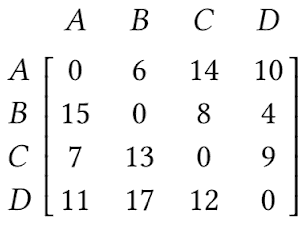


Рисунок 2 – матрица оценок.

Второй этап метода Шульце – построение графа, в котором узлами являются кандидаты, а весами ребер – интервалы предпочтений одного кандидата другому. Если для двух кандидатов А и В число избирателей, которые предпочитают А вместо В больше числа избирателей, предпочитающих В вместо А, мы добавляем ребро А – В и называем разницу между числом избирателей весом этого ребра. Остальные ребра считаем несуществующими. В конечном счете мы получаем представленные на рисунке 3 матрицу весов и соответствующей ей граф выборов. Сам процесс расчета матрицы представлен на рисунке 4.

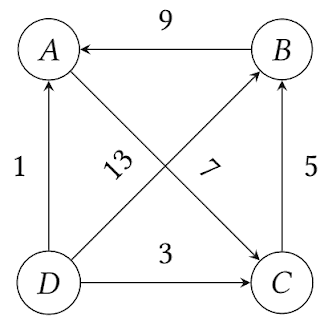
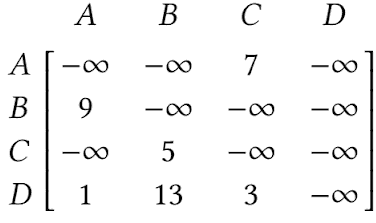


Рисунок 3 – матрица весов и соответствующий ей граф выборов.

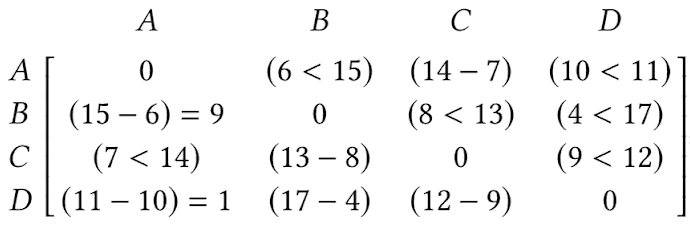


Рисунок 4 – Принцип подсчета матрицы весов.

Построив граф, мы вычисляем сильнейшие пути между всеми узлами графа. Мы определяем силу пути, как минимальый вес в ребрах, составляющий этот путь. Для нахождения прочнейших путей между всеми парами кандидатов в графе мы берем по очереди каждый узел, после чего сначала находим сильнейшие пути между всеми парами узлов, игнорируя промежуточные узлы, т.е. только между теми парами узлов, между которыми есть прямо соединяющее их ребро, как на рисунке 5.

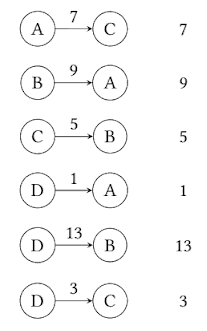


Рисунок 5 – Первый этап нахождения кратчайших путей.

После этого мы продолжаем поиски путей, используя первый выбранный узел в графе в качестве промежуточного. Сравниваем силу путей мы выбираем самый сильный. Так продолжается до тех пор, пока не будут перебраны все узлы графа. Процесс показан на рисунке 6. В итоге мы получаем следующие сильнейшие пути в графе выборов (См. Рисунок 7) и соответствующую им матрицу (См. Рисунок 8).

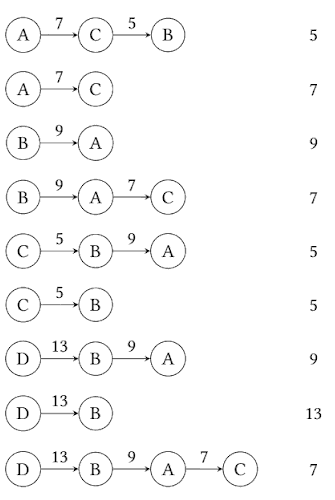
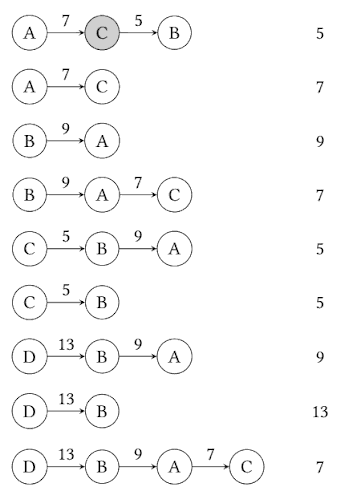
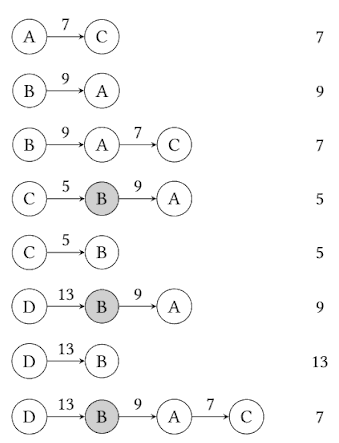
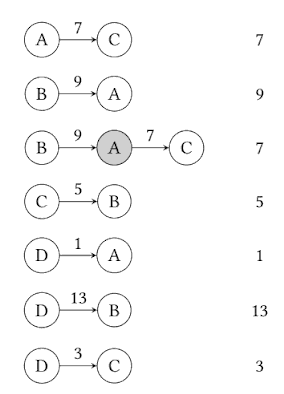


Рисунок 6 – Этапы нахождения сильнейших путей.

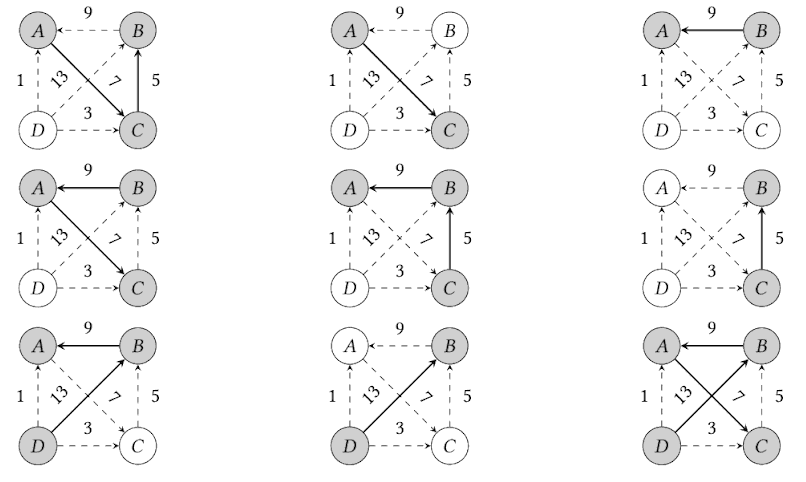


Рисунок 7 – Карта сильнейших путей в графе выборов.

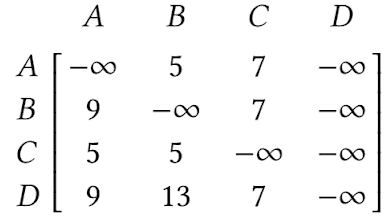


Рисунок 8 – Матрица карты сильнейших путей в графе выборов.

С этими данными можно перейти к третьему этапу метода Шульце – подсчету результатов. Поддержка кандидатов избирателями выражена в виде прочнейших путей. Если прочность пути от А к В больше, нежели от В к А – то можно считать, что кандидат А одержал победу над кандидатом В. После чего нужно узнать, сколько других кандидатов обошел кандидат А. Это можно легко узнать, пройдясь по массиву с прочностями путей и сосчитать, сколько раз кандидат А был победителем:

Winners = [[2], [2, 0], [ ], [2, 1, 0]]

A = 1, B = 2, C = 0, D = 3.

**Победитель кандидат D.**

# РЕАЛИЗАЦИЯ алгоритма

Алгоритм был реализован посредством языка программирования C#. Программа состоит из главной функции, в которой инициализирован массив с тестовыми данными, а так же осуществляется их обработка при помощи вспомогательных функций, реализующих этапы работы метода Шульце. Дополнительно используется функция для перевода результатов, возвращаемых функцией подсчета результатов (3 шаг метода Шульце) в строку для вывода на экран консоли. Используется StringBuilder.

Время выполнения первой функции, представляющей собой первый шаг метода Шульце в худшем случае составляет О(m2+n2) времени, где m – число кандидатов, а n – число бюллетеней. Это обусловлено тем, что алгоритму необходимо перебрать всех кандидатов во всех бюллетенях. А так же необходимо обнулить созданную матрицу предпочтений, что занимает О(n2) времени.

Время выполнения второй функции, представляющей собой второй шаг метода Шульце в худшем случае составляет О(m3) времени.

Время выполнения третей функции, представляющей собой последний шаг метода Шульце в худшем случае составляет О(m2)

# Результат работы алгоритма

Программная реализация алгоритма обработала входной поток данных, аналогичный указанному в примере для возможности оценки корректности работы. Результаты совпали, что показано на рисунке 9.

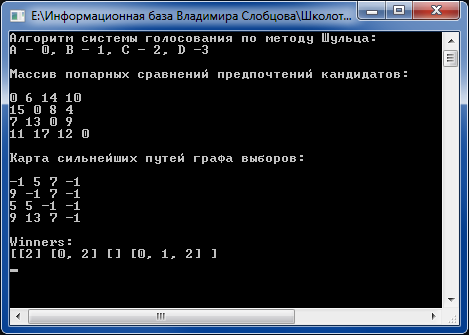


Рисунок 9 – Результат работы программы.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта был изучен и реализован алгоритм системы голосования по методу Шульце на основе принципа Кондорсе по ранжированным кандидатам. Был произведен ручной и программный подсчет победителя при голосовании на тестовых данных. Результаты совпадают, исходя из чего можно судить о корректной работе программной реализации алгоритма.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. С.А.Смирнов, К.А.Ильчук, Алгоритмы определения победителя при коллективном выборе на основе подхода Кондорсе, 2011. Режим доступа: <http://ena.lp.edu.ua/bitstream/ntb/19544/1/39-Smirnov-86.pdf>
2. Ричард Румельт, Хорошая стратегия, плохая стратегия. В чем отличие и почему это важно, 2013. <https://www.mann-ivanov-ferber.ru/books/paperbook/good-strategy-bad-strategy/>
3. Искуственный интеллект: стратегии и методы решения сложных проблем, 4-е издания. Режим доступа: <https://books.google.ru/books?id=jaoDX9-e_McC&pg=PA167&lpg=PA167&dq=%D1%8D%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0+%D0%BF%D1%83%D1%82%D0%B8&source=bl&ots=lIdSE0oBei&sig=gNrQ6MEONz_AZO_dIpINICHIfwU&hl=ru&sa=X&ved=0ahUKEwjZxMfvh-zaAhXDIpoKHeCVAFwQ6AEIczAF#v=onepage&q=%D1%8D%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0%20%D0%BF%D1%83%D1%82%D0%B8&f=false>

# КОД ПРОГРАММЫ

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Schulz

{

class Program

{

public static int[][] Pairs(int[][] B, int n)

{

int[][] arr = new int[n][];

for (int i =0; i<n;i++)

{

arr[i] = new int[n];

for (int j = 0; j<n; j++)

{

arr[i][j] = 0;

}

}

for (int i = 0; i<B.Length; i++)

{

int[] elem = B[i];

for (int j = 0; j<elem.Length;j++)

{

int a1 = elem[j];

for (int k = j+1; k<elem.Length;k++)

{

int a2 = elem[k];

arr[a1][a2]++;

}

}

}

return arr;

}

public static Tuple<int[][], int[][]> PowerWays(int[][] arr, int n)

{

int[][] arr2 = new int[n][];

int[][] parr = new int[n][];

for (int i =0; i<n;i++)

{

arr2[i] = new int[n];

parr[i] = new int[n];

for (int j =0;j<n;j++)

{

if (arr[i][j] > arr[j][i])

{

arr2[i][j] = arr[i][j] - arr[j][i];

parr[i][j] = i;

}

else

{

arr2[i][j] = -1;

parr[i][j] = -1;

}

}

}

for (int k = 0; k < n; k++)

{

for (int i = 0; i < n; i++)

{

if (i != k)

{

for (int j = 0; j < n; j++)

{

if (j != i)

{

if (arr2[i][j] < Math.Min(arr2[i][k], arr2[k][j]))

{

arr2[i][j] = Math.Min(arr2[i][k], arr2[k][j]);

parr[i][j] = parr[k][j];

}

}

}

}

}

}

return new Tuple<int[][], int[][]>(arr2,parr);

}

public static string ListToString(List<List<int>> thing)

{

StringBuilder sb = new StringBuilder();

sb.Append("[");

foreach (var li in thing)

{

sb.Append("[" + string.Join<int>(", ", li) + "] ");

}

sb.Append("] ");

return sb.ToString();

}

public static List<List<int>> Results(int[][] arr, int n)

{

List<List<int>> wins = new List<List<int>>();

for (int i =0;i<n;i++)

{

List<int> list = new List<int>();

wins.Add(list);

for (int j =0; j<n;j++)

{

if (i != j)

{

if (arr[i][j]>arr[j][i])

{

list.Add(j);

}

}

}

}

return wins;

}

static void Main(string[] args)

{

int[][] B = new int[21][] { new int[]{ 0, 2, 3, 1}, new int[]{ 0, 2, 3, 1},new int[]{ 0, 2, 3, 1},new int[]{ 0, 2, 3, 1},new int[]{ 0, 2, 3, 1},new int[]{ 0, 2, 3, 1},

new int[]{ 1, 0, 3, 2},new int[]{ 1, 0, 3, 2},new int[]{ 1, 0, 3, 2},new int[]{ 1, 0, 3, 2},

new int[]{ 2, 3, 1, 0},new int[]{ 2, 3, 1, 0},new int[]{ 2, 3, 1, 0},

new int[]{ 3, 1, 0, 2},new int[]{ 3, 1, 0, 2},new int[]{ 3, 1, 0, 2},new int[]{ 3, 1, 0, 2},

new int[]{ 3, 2, 1, 0},new int[]{ 3, 2, 1, 0},new int[]{ 3, 2, 1, 0},new int[]{ 3, 2, 1, 0}};

int[][] arr = Pairs(B, 4);

Tuple<int[][], int[][]> Pair = PowerWays(arr, 4);

int[][] arr2 = Pair.Item1;

List<List<int>> win = Results(arr2, 4);

Console.WriteLine("Алгоритм системы голосования по методу Шульца:");

Console.WriteLine("A - 0, B - 1, C - 2, D -3\n");

Console.WriteLine("Массив попарных сравнений предпочтений кандидатов:");

for (int i =0;i<4;i++)

{

Console.WriteLine(" ");

for (int j=0;j<4;j++)

{

Console.Write(arr[i][j] + " ");

}

}

Console.WriteLine("\n");

Console.WriteLine("Карта сильнейших путей графа выборов:");

for (int i = 0; i < 4; i++)

{

Console.WriteLine(" ");

for (int j = 0; j < 4; j++)

{

Console.Write(arr2[i][j] + " ");

}

}

Console.WriteLine("\n");

Console.WriteLine("Winners: ");

Console.WriteLine(ListToString(win));

Console.ReadKey();

}

}

}