|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

по дисциплине: «Общесистемное программное обеспечение параллельных вычислительных систем»

|  |  |  |
| --- | --- | --- |
| Студент |  | Гольцов Илья Сергеевич |
| Группа |  | РК6-32М |
| Тип задания |  | лабораторная работа №2 |
| Тема лабораторной работы |  | Волновой алгоритм Финна |

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Гольцов И. С.\_\_**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Грошев С. В.\_**

*подпись, дата фамилия, и.о.*

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

*Москва, 2022 г.*

**Оглавление**

[Задание на лабораторную работу 3](#_Toc119288266)

[Выполнение лабораторной работы 3](#_Toc119288267)

[Заключение 7](#_Toc119288268)

[Код программы 8](#_Toc119288269)

# Задание на лабораторную работу

Написать программу иллюстрирующую работу волнового алгоритма Финна для распределенной сети, представленной в виде графа.

# Выполнение лабораторной работы

Алгоритм Финна – волновой алгоритм, который можно использовать в ориентированных сетях произвольной топологии. Он не требует того, чтобы диаметр сети был известен заранее, но подразумевает наличие уникальных идентификаторов процессов. В сообщениях передаются множества идентификаторов процессов, что приводит к высокой битовой сложности алгоритма.

Процесс содержит два множества идентификаторов процессов:

* – это множество процессов таких, что событие в предшествует последнему произошедшему событию в ;
* – множество процессов таких, что для всех соседей процесса событие в предшествует последнему произошедшему событию в . Эта зависимость поддерживается следующим образом.

1. Изначально , а .
2. Каждый раз, когда одно из множеств пополняется, процесс посылает сообщение, включая в него и .
3. Когда получает сообщение, включающее множества и , полученные идентификаторы включаются в версии этих множеств в процессе .
4. Когда получит сообщения от всех соседей по входу, включается в .
5. Когда два множества становятся равны, выполняет процедуру .

Из неформального смысла двух множеств следует, что для каждого процесса такого, что событие в предшествует некоторому событию , выполняется следующее: для каждого соседа процесса событие в также предшествует событию .

Ниже представлен псевдокод волнового алгоритма Финна:

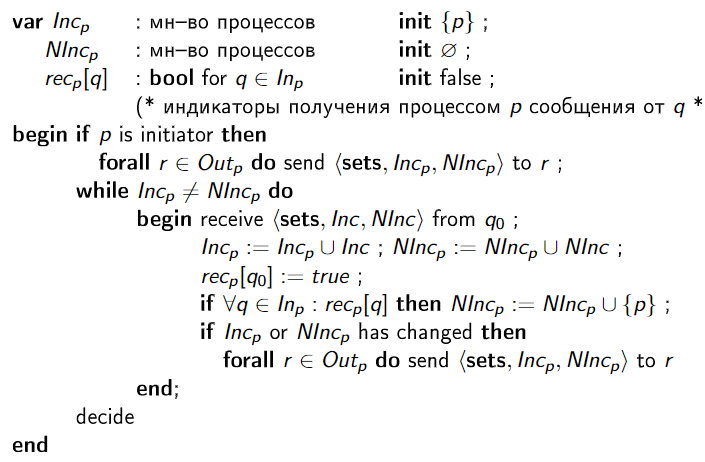


Рисунок 1 - Псевдокод волнового алгоритма Финна

В рамках выполнения лабораторной работы была разработана программа, иллюстрирующая работу волнового алгоритма Финна. Решение производилось на графе, представленном на рисунке 2.

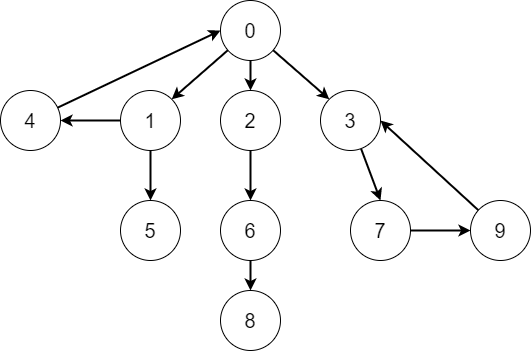


Рисунок 2 – Граф, используемый для решения

Граф задается в программе парами чисел, представляющих его ребра. В начале работы программы происходит считывание структуры графа. На основе заданной структуры строится матрица смежности.

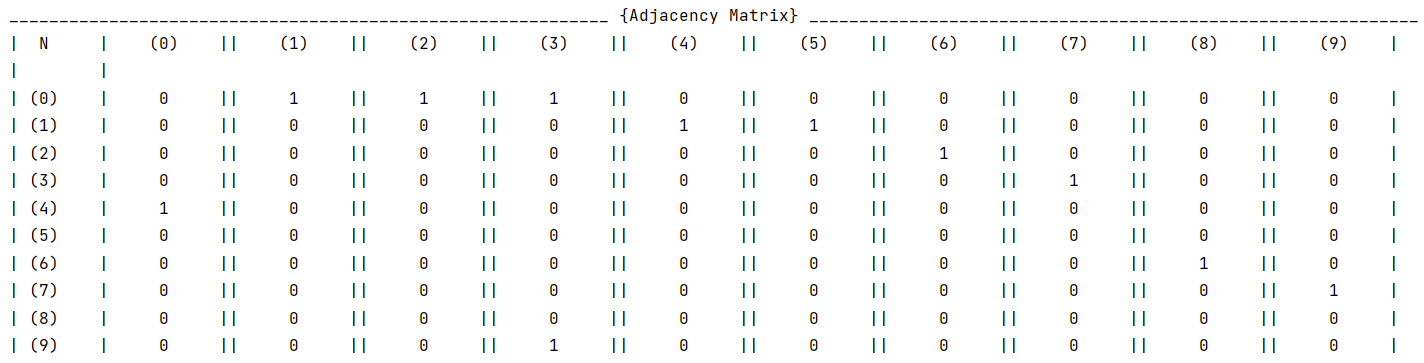
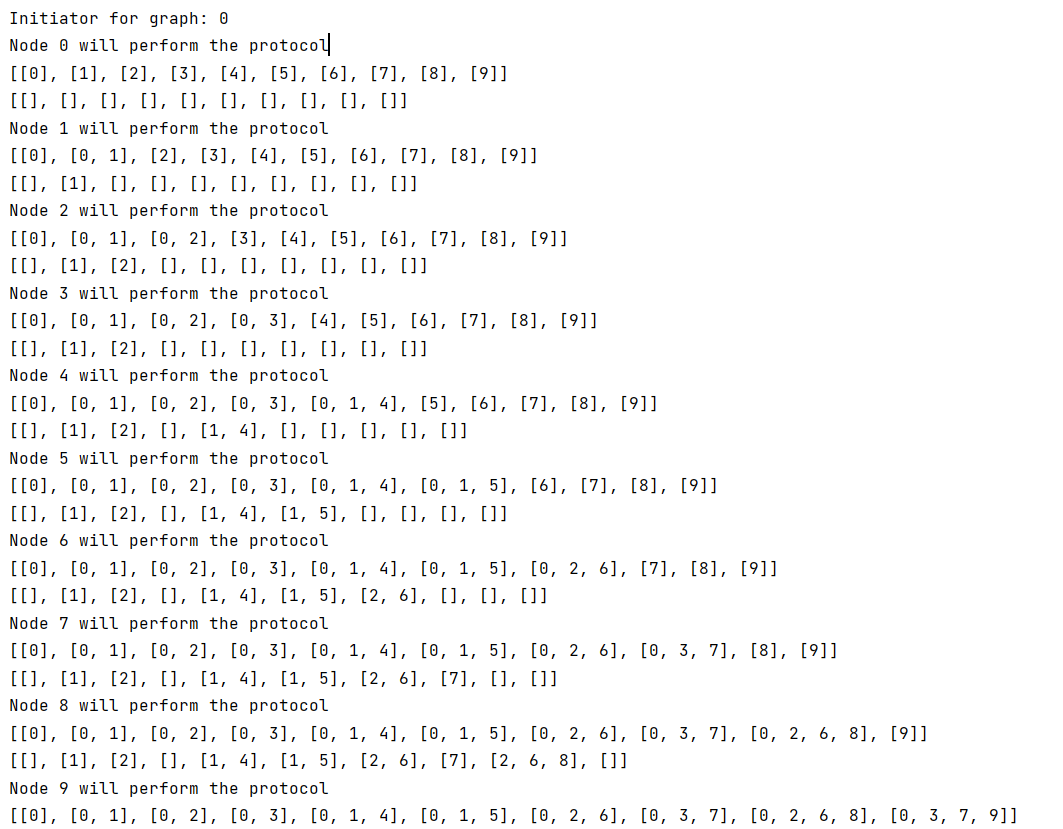


Рисунок 3 – Полученная программой матрица смежности

Вершиной-инициатором выбирается корневая. Затем происходит рассылка сообщений всем потомкам инициатора. Потомки, получая сообщения, обновляют свои множества и , далее рассылая сообщения другим вершинам. При этом происходит последовательный перебор всех вершин, что имитирует общий канал передачи данных между вершинами распределенной сети. Каждая вершина в один момент времени производит некоторую операцию, в зависимости от своего состояния. То есть либо ожидает, либо выполняет рассылку сообщений, либо получает сообщения. После того, как для некоторой вершины множества и становятся одинаковыми, происходит принятие решения и алгоритм завершает работу.



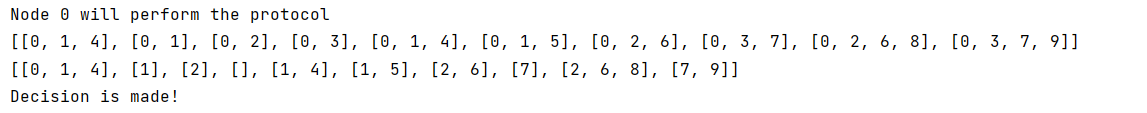
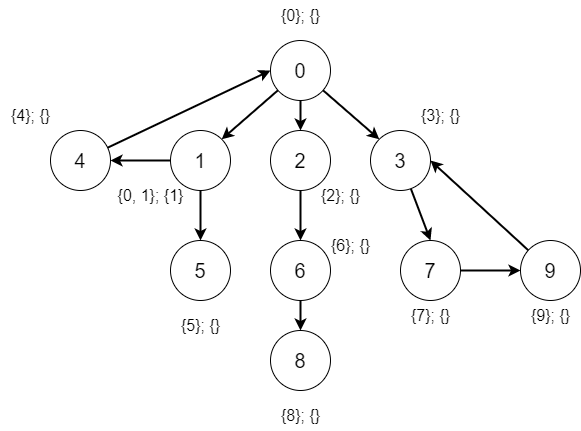


Рисунок 4 – Результат работы разработанного алгоритма

На рисунках 5-7 представлена иллюстрация работы алгоритма в виде последовательных состояний сети. Рядом с каждой вершиной обозначены соответственно множества и .



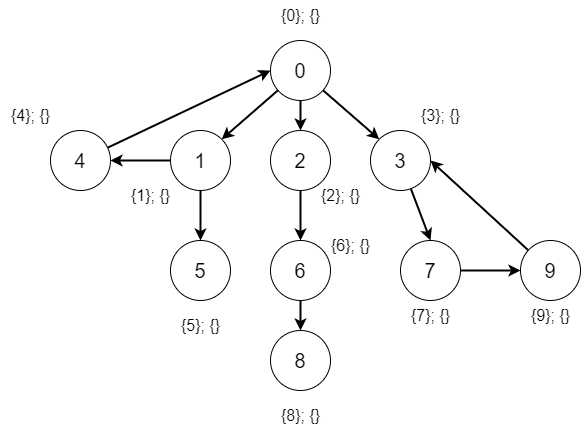


Рисунок 5 – Ход работы алгоритма (шаги 1 и 2)

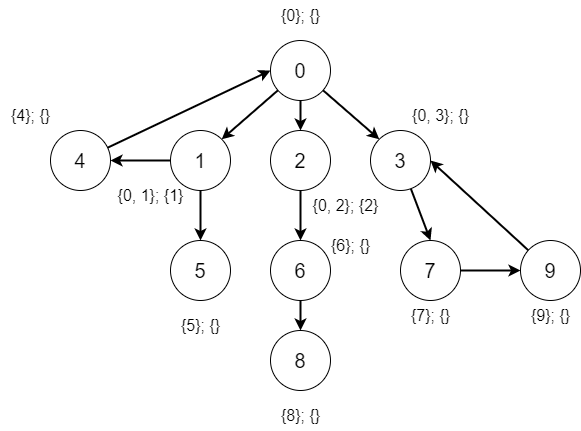
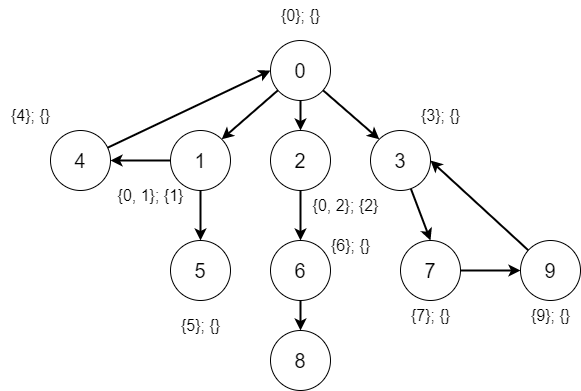


Рисунок 6 – Ход работы алгоритма (шаги 3 и 4)

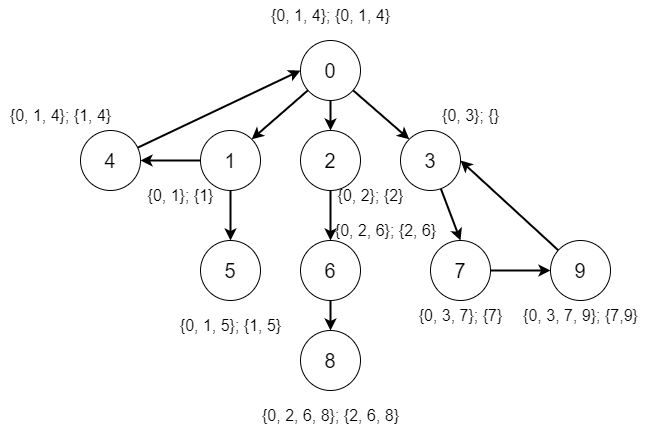
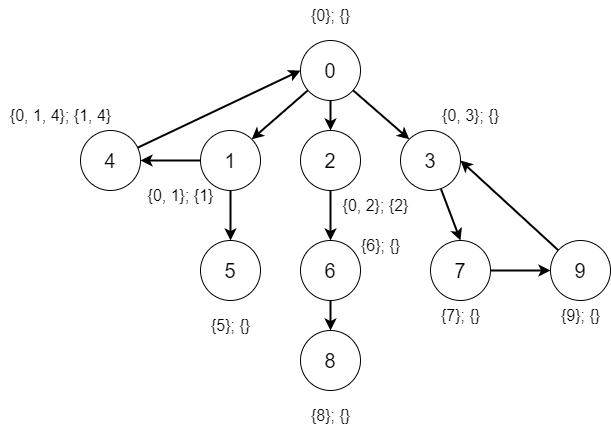


Рисунок 7 – Ход работы алгоритма (шаг 5 и финальное состояние)

# Заключение

В ходе выполнения лабораторной работы был изучен волновой алгоритм Финна и реализован программный код на языке Java, имитирующий работу распределенной сети на заданном графе, в ходе которой при помощи исследуемого алгоритма происходит рассылка сообщений всем вершинам графа. Рассмотренный алгоритм может использоваться для загрузки и верификации узлов в алгоритмах более высокого уровня.

# Код программы

Ниже представлен программный код, отвечающий за выполнение алгоритма:

Листинг 1 – Функция, отвечающая за выполнение алгоритма

**public void** executeEchoFinn(Graph g, Integer start) {  
 List<Set<Integer>> Inc = **new** ArrayList<>();  
 List<Set<Integer>> NInc = **new** ArrayList<>();  
 List<List<Map.Entry<Integer, Boolean>>> rec = **new** ArrayList<>();  
 **for** (**var** i = 0; i < g.getGraphSize(); i++) {  
 Inc.add(Set.*of*(g.getGraphNodes()[i].getValue()));  
 NInc.add(Set.*of*());  
  
 List<Map.Entry<Integer, Boolean>> r = **new** ArrayList<>();  
 **for** (**var** e : g.getPredecessors(i)) {  
 r.add(**new** AbstractMap.SimpleEntry<>(e, Boolean.***FALSE***));  
 }  
 rec.add(r);  
 }  
  
 **int** graphSize = g.getGraphSize();  
 GraphNode[] nodes = g.getGraphNodes();  
  
 **int** initiator = start;  
 System.***out***.println(**"Initiator for graph: "** + initiator);  
  
 **boolean** isEnd = **false**;  
 **while** (!isEnd) {  
 **for** (**int** j = 0; j < graphSize; j++) {  
 FinnGraphNode finnNode = (FinnGraphNode) nodes[j];  
  
 System.***out***.println(**"Node "** + j + **" will perform the protocol"**);  
  
 *// Ветка для инициатора, должна отработать один раз* **if** (j == initiator && !finnNode.isVisited()) {  
 ((FinnGraph) g).sentMessagesToNeighbors(j);  
 }  
  
 **for** (**var** node : finnNode.getNodesFrom()) {  
 Set<Integer> IncNew = Stream.*concat*(Inc.get(j).stream(), Inc.get(node).stream())  
 .collect(Collectors.*toSet*());  
 Set<Integer> NIncNew = Stream.*concat*(NInc.get(j).stream(), NInc.get(node).stream())  
 .collect(Collectors.*toSet*());  
  
 rec.get(j).stream()  
 .filter(el -> node.equals(el.getKey()))  
 .findFirst()  
 .map(el -> el.setValue(Boolean.***TRUE***)); *// received* **if** (rec.get(j).stream()  
 .allMatch(el -> Boolean.***TRUE***.equals(el.getValue()))) {  
 NIncNew.add(j);  
 }  
  
 **if** (!IncNew.equals(Inc.get(j)) || !NIncNew.equals(NInc.get(j))) {  
 ((FinnGraph) g).sentMessagesToNeighbors(j);  
 }  
  
 Inc.set(j, IncNew);  
 NInc.set(j, NIncNew);  
 }  
 finnNode.getNodesFrom().clear();  
  
 **if** (Inc.get(j).equals(NInc.get(j))) {  
 isEnd = **true**;  
 **break**;  
 }  
  
 System.***out***.println(Arrays.*toString*(Inc.toArray()));  
 System.***out***.println(Arrays.*toString*(NInc.toArray()));  
 }  
 }  
  
 System.***out***.println(Arrays.*toString*(Inc.toArray()));  
 System.***out***.println(Arrays.*toString*(NInc.toArray()));  
}