Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе № 5

по дисциплине: "Логика и основы алгоритмизации в инженерных задачах"

на тему: "Обход графа в ширину"

Выполнили:

студенты группы 20ВВ2

Макарова А. Ю.

Тельнова А. Д.

Принял:

д.т.н., профессор

Митрохин М. А.

к.т.н., доцент

Юрова О. В.

Пенза, 2021

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа G. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру обхода в ширину, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди используйте класс queue из стандартной библиотеки С++.
3. \* Реализуйте процедуру обхода в ширину для графа, представленного списками смежности.

**Описание работы программы:**

**1-2.** Объявляются переменные int N – количество вершин графа, visited – массив для хранения информации о посещенных вершинах, G – матрица смежности графа G.

Количество вершин графа вводится с клавиатуры. Затем все N элементов массива visit инициализируются нулями. С помощью генератора случайных чисел заполняется значениями массив G[N][N]. Он представляет собой симметричную матрицу, элементы главной диагонали которой – нули.

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

for (int j = i; j < N; j++)

if (i == j)

G[i][j] = 0;

else

{

if (rand() % 10 < 3)

G[i][j] = 0;

else

G[i][j] = 1;

G[j][i] = G[i][j];

}

Полученная матрица выводится на экран.

Вводится номер вершины num, с которой следует начать обход графа в ширину. Функция BFS получает значение num-1, так как индексация элементов массива начинается с 0.

void BFS(int num)

{

queue <int> q;

q.push(num);

visited[num] = 1;

while (!q.empty())

{

num = q.front();

q.pop();

printf("%d ", num+1);

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

if (visited[i] == 0 && G[num][i] == 1)

{

q.push(i);

visited[i] = 1;

}

}

}

Функция получает на вход номер вершины, с которой начнется обход в ширину. Создается очередь q. Номер вершины, полученный функцией на вход, помещается с очередь. Элемент массива visited с соответствующим индексом помечается как посещенный.

Пока очередь не опустеет будет выполняться следующее: переменная num получает значение первого элемента очереди, после чего элемент будет удален. Значение num+1 выводится на экран. Затем осуществляется проход по всем вершинам матрицы смежности. Если находится не посещенная вершина, смежная с текущей, ее номер помещается в очередь и элемент массива visited с ее индексом получает значение 1.

В итоге на экран будет выведен граф и порядок его обхода в ширину.

**3\*.** Перед началом обхода в глубину графа, представленного списками смежности, необходимо обнулить массив visited и создать списки смежности.

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

visited[i] = 0;

Для преобразования представления графа из матрицы смежности в списки смежности используется функция CreateList.

void CreateList(int\*\* matrix, Node\*\* head, int N)

{

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

{

AddFirst(&head[i], i);

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

if (matrix[i][j] == 1)

AddLast(head[i], j);

}

}

С помощью функции AddFirst добавляются элементы в начало списка. После добавления очередного элемента в список, проверяется условие на наличие связи между вершинами графа с индексами i и j. Если эта связь есть, то в конец списка смежности i-ой вершины, добавляется вершина j с помощью функции AddLast.

Вводится номер вершины num, с которой следует начать обход. Функция BFSlist реализует обход графа в ширину.

void BFSlist(Node\*\* head, int num)

{

queue <int> q;

Node\*\* tmp\_node = head;

q.push(tmp\_node[num]->vertex);

visited[tmp\_node[num]->vertex] = 1;

while (!q.empty())

{

num = q.front();

q.pop();

printf("%d ", num + 1);

while (tmp\_node[num])

{

if (visited[tmp\_node[num]->vertex] == 0)

{

q.push(tmp\_node[num]->vertex);

visited[tmp\_node[num]->vertex] = 1;

}

tmp\_node[num] = tmp\_node[num]->next;

}

}

}

На вход функция получает указатель на массив голов списков смежности и номер вершины, с которой начнется обход.

Создается очередь q, куда помещается номер вершины старта. Элемент массива visited с индексом, равным этому номеру, получает значение 1. Затем пока в очереди будут элементы будут выполняться следующие действия: переменная num получит значение первого элемента в очереди, который затем будет удален, а значение num+1 выведется на экран. Далее будут просмотрены все элементы списка. Если в списке смежности найдется не посещенная вершина, ее номер занесется в очередь и в массиве visited она отметится посещенной.

**Задание 2\***

1. Для матричной формы представления графов реализуйте алгоритм обхода в ширину с использованием очереди, построенной на основе структуры данных «список», самостоятельно созданной в лабораторной работе № 3.
2. Оцените время работы двух реализаций алгоритмов обхода в ширину (использующего стандартный класс **queue** и использующего очередь, реализованную самостоятельно) для графов разных порядков.

**Описание работы программы**

1. Обход графа в ширину с помощью самостоятельно созданной очереди осуществляется с помощью функции BFSqueue.

void BFSqueue(int num)

{

Node\* QueueHead = NULL;

AddFirst(&QueueHead, num);

visited[num] = 1;

while (QueueHead)

{

num = QueueHead->vertex;

QueueHead = QueueHead->next;

printf("%d ", num + 1);

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

if (visited[i] == 0 && G[num][i] == 1)

{

if (QueueHead == NULL)

AddFirst(&QueueHead, i);

else

AddLast(QueueHead, i);

visited[i] = 1;

}

}

}

Перед вызовом этой функции все элементы массива посещенных вершин получают значения 0. Номер вершины, с которой начнется обход, вводится с клавиатуры.

Созданная очередь представляет собой односвязный список. Указатель на первый его элемент – QueueHead. Первый элемент очереди – элемент с номером стартовой вершины, которая помечается как посещённая.

Пока в очереди есть хотя бы один элемент, выполняется следующее: значение поля vertex первого элемента присваивается переменной num, а сам элемент удаляется из очереди, значение num+1 выводится на экран. Затем просматривается столбец с индексом num в исходной матрице смежности. Если для текущей вершины находится смежная не посещённая, в конце списка добавляется новый элемент, содержащий её номер. В массиве посещенных вершин она так же отмечается посещенной.

1. Для сравнения времени работы реализаций алгоритмов необходимо было замерить время выполнения каждого из них. Для этого были созданы переменные типа clock\_t start и end. Время обхода выводилось на экран в секундах.

start = clock();

BFS(num - 1);

end = clock();

printf("\n\nВремя обхода: %f сек", ((float)end - (float)start) / CLOCKS\_PER\_SEC);

|  |  |  |
| --- | --- | --- |
| **Порядок графа** | **Время выполнения алгоритма, использующего стандартный класс queue, сек** | **Время выполнения алгоритма, использующего очередь, реализованную самостоятельно, сек** |
| 10 | 0,001 | 0,001 |
| 100 | 0,01 | 0,007 |
| 200 | 0,016 | 0,021 |
| 400 | 0,05 | 0,044 |
| 800 | 0,054 | 0,054 |
| 1000 | 0,071 | 0,066 |

Обход графа в ширину алгоритмом, использующем реализованную самостоятельно очередь, происходит быстрее.

**Результаты работы программы**

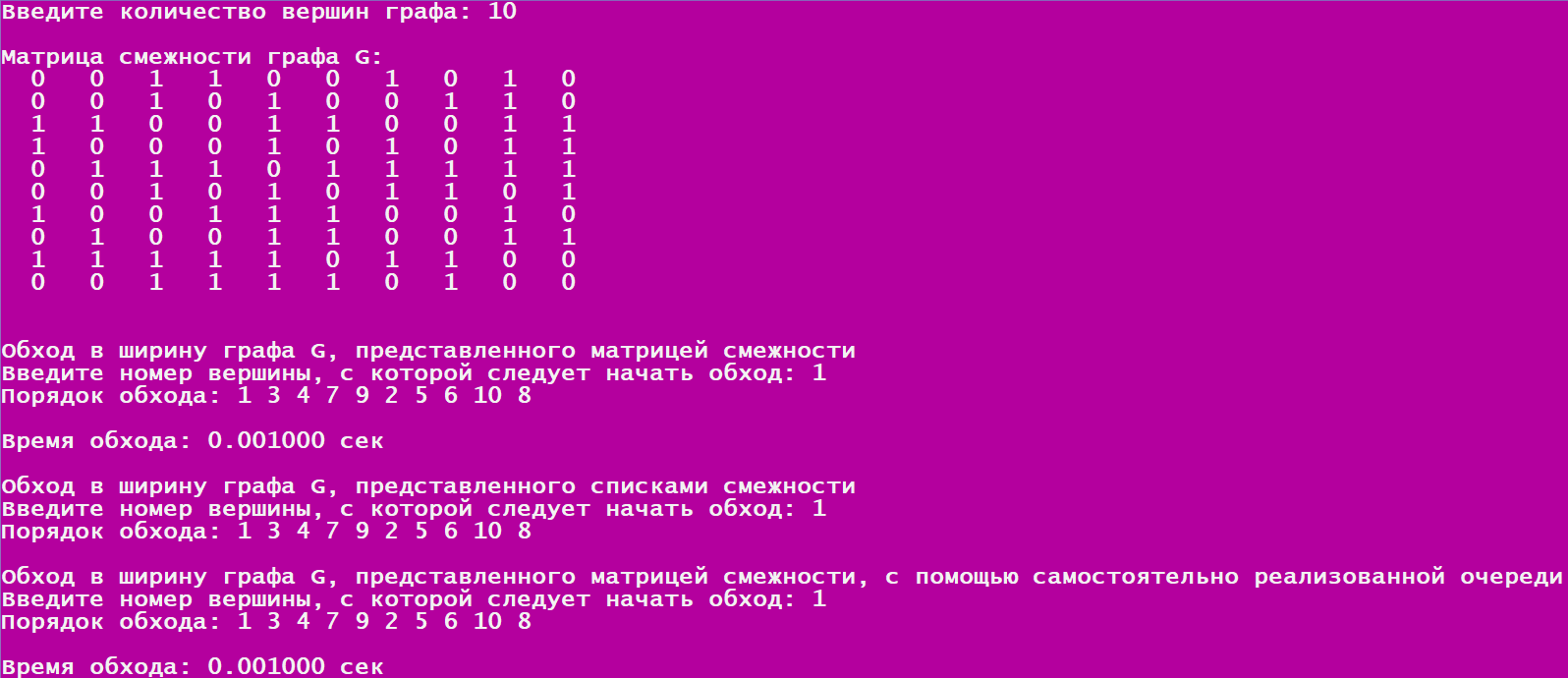


Рисунок 1 - Результаты работы программы

**Вывод**: написали программу, реализующую алгоритм обхода в ширину графов, представленных матрицей смежности и списками смежности. Матрица смежности графа, результаты обхода выводятся на экран, а количество вершин вводится пользователем. Для реализации использована очередь – класс queue из стандартной библиотеки С++. Реализовали алгоритм обхода в ширину графа, представленного матрицей смежности, с использованием очереди, созданной самостоятельно в виде односвязного списка. Сравнили время работы этой реализации и временем работы алгоритма, использующего стандартный класс queue.