Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

# по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «обход графа в ширину»

**Выполнил:**

студент группы 20ВВ3

Духнов Олег

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2021

# Название

Обход графа в ширину.

**Цель работы** – изучение алгоритма обхода матрицы смежности в ширину с помощью библиотеки queue.

# Лабораторное задание

**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.

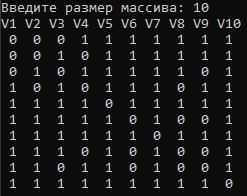
**Задание 2\***

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.

# Результат работы программы

**Задание 1.1.**

Результаты работы программы показаны на рисунке 1.



**Рисунок 1 – Результат работы программы**

**Задание 1.2. Результат обхода в ширину.**



**Задание 2.1.**

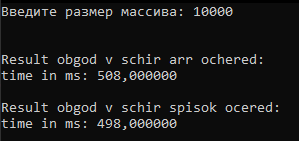
Результаты работы программы показаны на рисунке 2.



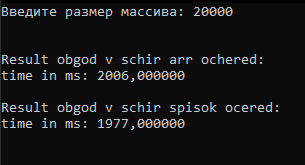
**Рисунок 2 – Результат работы программы**

**Задание 2.2.**

Результаты работы программы показаны на рисунке 3.



**Рисунок 3 – Результат работы программы**



**Рисунок 4 – Результат работы программы**

**Вывод по второй части:**

При измерении времени работы алгоритмов выяснилось, что написанные функции для реализации обхода в ширину работают немного быстрее, чем встроенные функции библиотеки queue.

**Вывод:** яизучил алгоритм обхода матрицы смежности в ширину с помощью библиотеки queue.

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <stdbool.h>

#include <conio.h>

#include <time.h>

#include <stack>

#include <queue>

using namespace std;

typedef struct List

{

int data;

List\* next;

}List;

typedef struct Queue

{

int size;

List\* Front;

List\* Back;

}Queue;

void init(Queue\* Que)

{

Que->Front = new List;

Que->Front->next = NULL;

Que->Back = Que->Front;

Que->size = 0;

}

bool Empty(Queue\* Que)

{

if (Que->Back == Que->Front)

{

return false;

}

else

{

return true;

}

}

void Push(Queue\* Que, int value)

{

Que->Back->next = new List;

Que->Back = Que->Back->next;

Que->Back->data = value;

Que->Back->next = NULL;

Que->size++;

}

void Pop(Queue\* Que)

{

List\* p;

p = Que->Front;

Que->Front = Que->Front->next;

Que->size--;

free(p);

}

int \_Front(Queue\* Que)

{

return Que->Front->next->data;

}

void BFS\_QUEUE(int n, bool\* Arr2, int i, int\*\* Arr)

{

Queue Que;

init(&Que);

Push(&Que, 0);

Arr2[i] = true;

while (Empty(&Que))

{

i = \_Front(&Que);

Pop(&Que);

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

for (int j = 1; j < n; j++)

{

if (Arr[i][j] == 1 && Arr2[j] == false)

{

Push(&Que, j);

Arr2[j] = true;

}

}

}

}

int\*\* create\_arr(int n)

{

int s = 1;

int\*\* Arr;

Arr = (int\*\*)malloc(n \* sizeof(int\*));

srand(time(NULL));

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

{

Arr[i] = (int\*)malloc(n \* sizeof(int));

Arr[i][i] = 0;

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

{

if (rand() % 100 > 70)

{

Arr[i][j] = 0;

}

else {

Arr[i][j] = 1;

}

}

s++;

}

s = 1;

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

{

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

Arr[j][i] = Arr[i][j];

s++;

}

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

printf("V%d ", i + 1);

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

{

printf("\n");

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

{

printf("%2d ", Arr[i][j]);

}

}

return Arr;

}

void BFS(int n, bool\* Arr2, int i, int\*\* Arr)

{

queue<int> que;

que.push(i);

Arr2[i] = true;

while (!que.empty())

{

i = que.front();

que.pop();

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

for (int j = 1; j < n; j++)

{

if (Arr[i][j] == 1 && Arr2[j] == false)

{

que.push(j);

Arr2[j] = true;

}

}

}

}

void Free(int\*\* Arr, bool\* Arr2, int n)

{

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

{

free(Arr[i]);

}

free(Arr);

free(Arr2);

}

//void POSCH(int n, int\*\* Arr)

//{

// bool\* Arr2;

// Arr2 = (bool\*)malloc(n \* sizeof(bool));

//

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

// Arr2[i] = false;

//

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

// {

// if (Arr2[i] == false)

// BFS(n, Arr2, i, Arr);

// }

//}

int main()

{

setlocale(LC\_ALL, "Ru");

clock\_t start, stop, st, end;

int n, \*\* t;

bool\* Arr2;

printf("Введите размер массива: ");

scanf("%d", &n);

Arr2 = (bool\*)malloc(n \* sizeof(bool));

t = create\_arr(n);

printf("\n\n");

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

Arr2[i] = false;

printf("Result obgod v schir arr ochered: ");

start = clock();

BFS(n, Arr2, 0, t);

stop = clock();

float res = stop - start;

printf("\ntime in ms: %f", res);

printf("\n\n");

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

Arr2[i] = false;

printf("Result obgod v schir spisok ocered: ");

st = clock();

BFS\_QUEUE(n, Arr2, 0, t);

end = clock();

float result = end - st;

printf("\ntime in ms: %f", result);

\_getch();

Free(t, Arr2, n);

}