МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н. Э. Баумана

КАФЕДРА ПРОЕКТИРОВАНИЕ И ТЕХНОЛОГИЯ ПРОИЗВОДСТВА ЭЛЕКТРОННОЙ АППАРАТУРЫ

Отчет о выполнении

практического задания №5

«Теоретико-множественные операции»

по курсу «Функциональная логика и теория алгоритмов»

Выполнили: студенты гр. ИУ4-21Б

Горохов И.А.

Проверил: д.т.н., профессор, профессор каф. ИУ-4

Л.А. Зинченко

Москва 2024

Цель работы:

Разработка программы, выполняющей ограниченный набор действий над множествами.

1. Исходные данные:

Написать на языке C программу, реализующую следующие функции:

* Чтение списка ребер, представляющего собой граф;
* Поиск вершин графа с наибольшей степенью;

Для программы выполнить:

* Тестирование на скорость
* Оценку o(n) теоретическая
* Оценку o(n) практическая
* График скорости работы

2. Выполнение.

Программа вызывается с набором аргументов командной строки, первый из которых является именем файла, содержащего список ребер, а второй – количество ребер. В случае отсутствия второго аргумента количество ребер будет прочитано из первой строки файла.

Содержимое основного кода программы находится в файле 3.c:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define DATA\_TYPE long

typedef unsigned long vertex;

typedef struct {

vertex start;

vertex end;

} edge;

void sort(unsigned long long len, DATA\_TYPE arr[]);

edge\* read\_file(FILE\* fp, unsigned long long len){

edge\* arr = malloc(len \* sizeof(edge));

for(unsigned long long i = 0; i<len; i++)

fscanf(fp, "%Ld %Ld", &((arr+i)->start), &((arr+i)->end));

return arr;

}

int main(int argc, char \*argv[])

{

if (argc < 2) {

printf("No filename provided\n");

return(0);

}

FILE\* fp;

if (!(fp=fopen(argv[1],"r"))){

printf("Can't open file\n");

return(0);

}

unsigned long long len;

if (argc < 3) {

fscanf(fp, "%ULLd", &len);

} else {

len = atoi(argv[2]);

}

edge\* arr = read\_file(fp, len);

fclose(fp);

time\_t begin = clock();

search\_max(len, arr);

time\_t end = clock();

printf("Time taken \t%f\n", ((double)(end - begin))/CLOCKS\_PER\_SEC);

}

vertex count(unsigned long long len, edge arr[]){

vertex vertex\_len = 0;

for (unsigned long long i = 0; i<len; i++){

if (arr[i].end > vertex\_len) vertex\_len = arr[i].end;

if (arr[i].start > vertex\_len) vertex\_len = arr[i].start;

}

return vertex\_len + 1;

}

void search\_max(unsigned long long len\_e, edge arr\_e[]){

printf("\nHighest degree vertexes:\t");

vertex len\_v = count(len\_e, arr\_e);

unsigned long\* arr\_v = malloc(len\_v \* sizeof(unsigned long));

for (vertex i = 0; i<len\_v; i++) arr\_v[i] = 0;

for (unsigned long long i = 0; i<len\_e; i++){

arr\_v[arr\_e[i].end]++;

arr\_v[arr\_e[i].start]++;

}

unsigned long max\_v = 0;

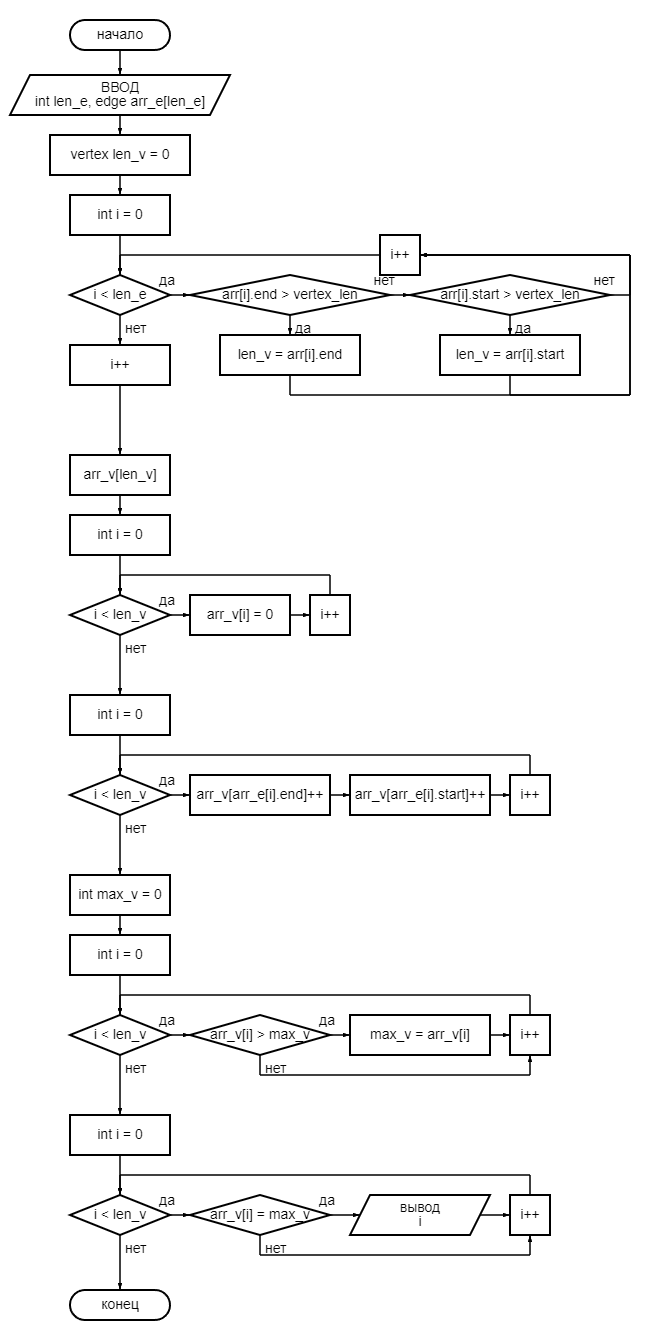
for (vertex i = 0; i<len\_v; i++) if (arr\_v[i] > max\_v) max\_v = arr\_v[i];

for (vertex i = 0; i<len\_v; i++) if (arr\_v[i] == max\_v) printf("%d ", i);

printf("\nwith a degree of %d\n", max\_v);

}

Схема используемого алгоритма представлена на рисунке 1.

  
Рисунок 1.

3. Результаты работы.

Пример вызовов программы находится в файле 1.bat:

C:\Users\goroh\Documents\GitHub\FLiTA\_M2> ./3.exe "tmp\_graph.txt" 10

highest degree vertexes: 5

with a degree of 5

Time taken 0.010000

Содержимое файла tmp\_graph.txt (первые строки):

6 9

4 9

7 3

4 9

9 0

5 9

5 6

5 6

9 7

5 5

...

Оценка скорости алгоритма:

Программа содержит циклы по ребрам и по вершинам, сложность которых O(n) и O(m), где n и m количество соответственно. Вложенность циклов отсутствует. Таким образом сложность алгоритма сводится к O(n+m).  
Зная, что количество вершин не превышает удвоенного количества ребер, можно утверждать, что O(n+m) ~ O(3n) ~ O(n).

Теоретическая сложность алгоритма O(n).

Практическая сложность измерена на графах различной длины (от 0 до 108 с шагом 5\*106, 21 измерение). Результат измерений приведен на рисунке 2.

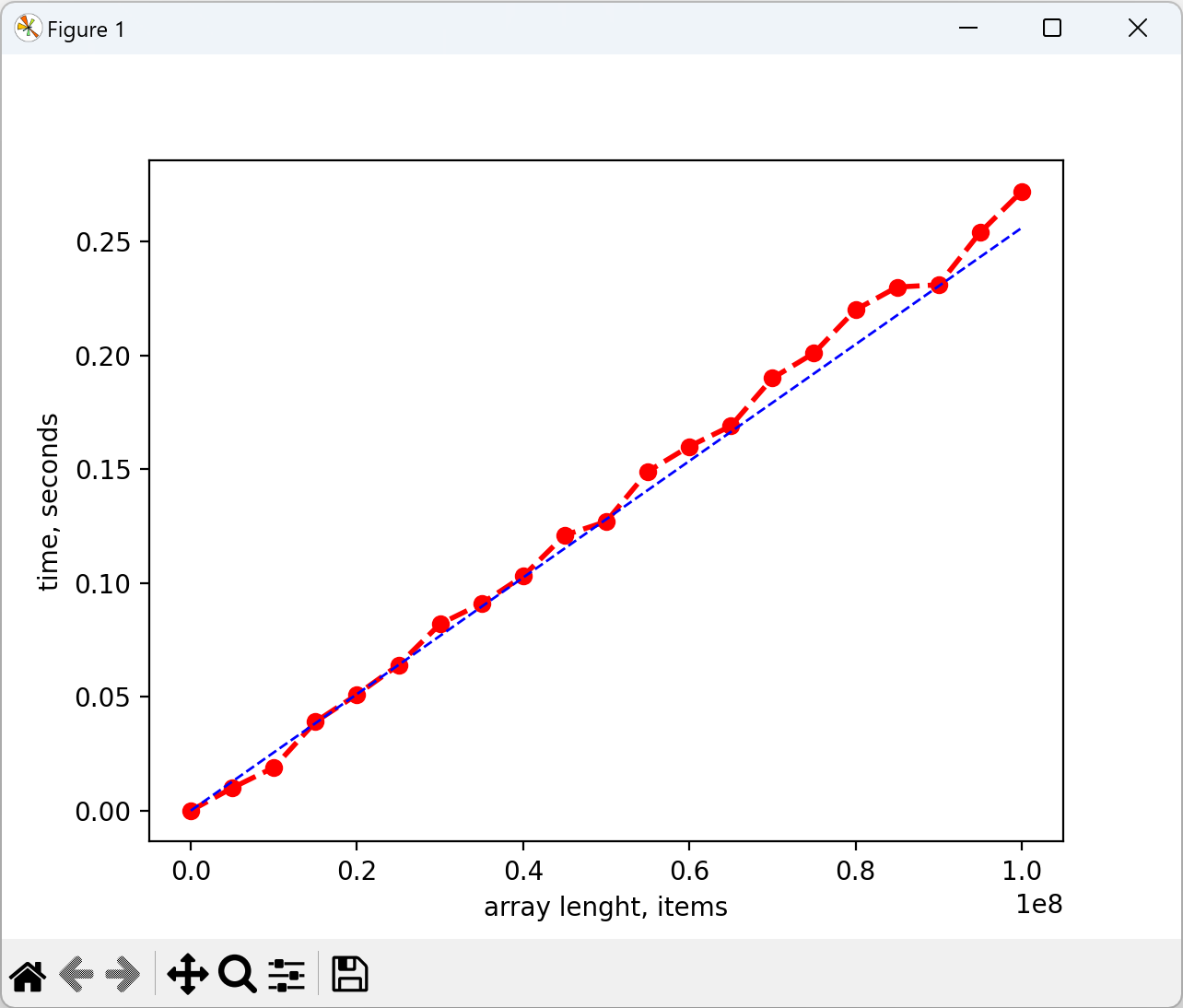


Рисунок 2.

Данные измерения практического времени выполнения выделены красным (толстым), теоретическая сложность – синим.

4. Вывод:

Программа корректно выполняет поиск вершин графа с наибольшей степенью;

Для программы выполнены:

* Тестирование на скорость
* Оценка o(n) теоретическая
* Оценка o(n) практическая
* График скорости работы