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 №3

«Работа с графами в программировании»

Вариант №11.

по курсу «Функциональная логика и теория алгоритмов»

Выполнил: студент гр. ИУ4-23Б

Якимов Павел Николаевич

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

Л.А. Зинченко

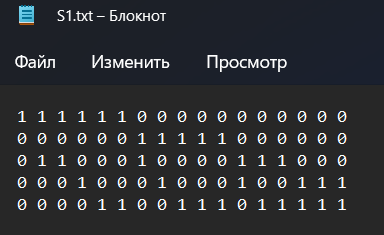
Москва 2022

Цель работы: Создать программу, определяющую по матрице инцидентности вершину графа с минимальной степенью.

1. Исходные данные:

Файл с матрицей инцидентности для графа.

#Пример вводимых данных:



2. Выполнение.

Реализуем считывание из файла матрицы инцидентности для определения вершины графа с минимальной степенью.

#Код программы:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include<iostream> //С++

#include<chrono> //С++

#include<iomanip> //C++

using namespace std;

int comp(const void\*, const void\*);

void fail(long int n) {

int ch;

FILE\* S1;

S1 = fopen("S1.txt", "w");

for (long int i = 1; i <= n; i++)

{

for (long int j = 1; j <= ((n+1)\*n/2); j++)

{

ch = rand() % 2;

//printf("\*");

fprintf(S1, "%d ", ch);

}

if (i != n)

{

fprintf(S1, "\n");

}

//printf("\*");

}

fclose(S1);

}

int main() {

FILE\* S2;

S2 = fopen("S2.txt", "w");

for (long int n = 1; n <= 100000000; n++)

{

FILE\* S1;

char s = '~', rez2;

long int k = 1;

fail(n);

auto begin = std::chrono::steady\_clock::now(); //начало подсчета времени выполнения проги (С++)

S1 = fopen("S1.txt", "r");

int\* rez = NULL;

rez = (int\*)malloc(k \* sizeof(long int));

rez[k - 1] = 0;

fscanf(S1, "%c", &s);

while (!feof(S1))

{

//printf("\*");

//cout << s;

if (s == '1')

{

rez[k-1]++;

//printf("+!%d!+", rez[k - 1]);

}

if (s == '\n')

{

//printf("\n+%d+\n", rez[k - 1]);

k++;

rez = (int\*)realloc(rez, k \* sizeof(long int));

rez[k - 1] = 0;

}

fscanf(S1, "%c", &s);

}

//printf("\n+%d+\n", rez[k - 1]);

fclose(S1);

/\*for (long int i = 0; i <= k - 1; i++) {

printf("%d ", rez[i]);

}

printf("\n");

qsort(rez, k, sizeof(long int), comp);

for (long int i = 0; i <= k - 1; i++) {

printf("%d ", rez[i]);

}

printf("\n%d\n", rez[0]);\*/

auto end = std::chrono::steady\_clock::now(); //С++

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin); //С++

printf("%d ", n);

fprintf(S2, "%d ", n);

long double time = elapsed\_ms.count();

time = time / 1000;

cout << fixed << setprecision(3) << time << "\n"; //конец подсчета

fprintf(S2, "%.\*f\n", 3, time);

//printf("\n\n");

}

fclose(S2);

return 0;

}

/\* сравнение целых \*/

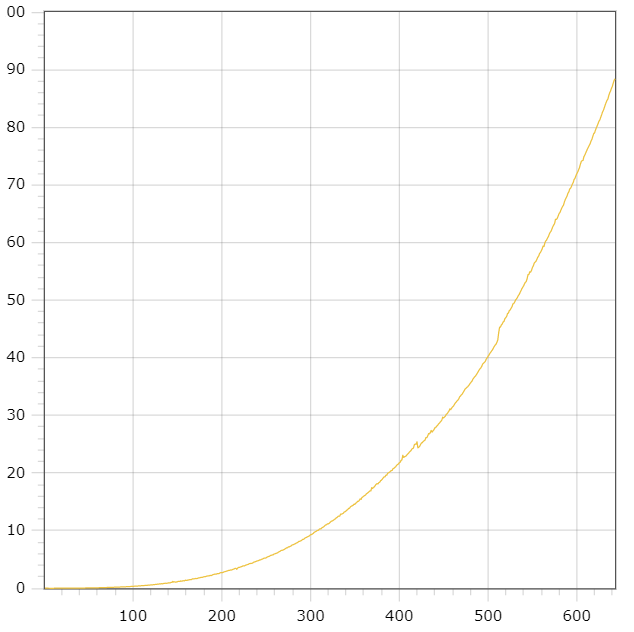
int comp(const void\* i, const void\* j) {

return \*(int\*)i - \*(int\*)j;

}

#Блок-схема:

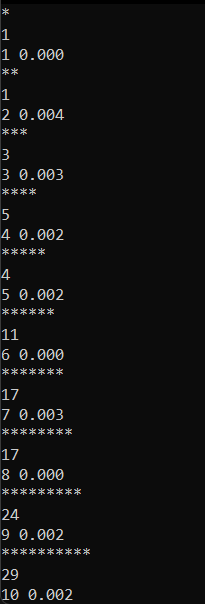
#График зависимости времени работы алгоритма от количества вводимых данных:



3. Результаты работы.

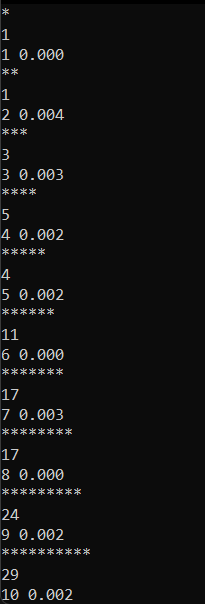
Получили вполне юзабильную программу, которая отвечает всем условиям поставленной программы.

#Пример выводимых данных:

**

4. Вывод:

Юзеру будет удобно пользоваться данной программой, т. к. имеет фид-бек интерфейс.

**