**Федеральное агентство по образованию РФ**

**Государственное образовательное учреждение высшего профессионального образования**

**Белгородский Государственный Технологический Университет им В. Г. Шухова**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем.

Лабораторная работа № 2 по дисциплине

«Дискретная математика»

**Транзитивное замыкание отношения**

Выполнил:

Студент группы ВТ-22

Богатырев В.В,

Белгород 2018

Цель работы: изучить и выполнить сравнительный анализ алгоритмов вычисления транзитивного замыкания отношения.

**Задания**

1. Изучить и программно реализовать алгоритмы объединения степеней и Уоршалла для вычисления транзитивного замыкания отношения.

2. Разработать и программно реализовать генератор отношений на множестве мощности N и содержащих заданное число пар.

3. Разработать и написать программу, которая генерирует 1000 отношений на множестве мощности N с заданным числом пар, для каждого отношения вычисляет транзитивное замыкание двумя алгоритмами и определяет время выполнения каждого алгоритма. Время вычисления транзитивного замыкания различных отношений на множестве мощности N с заданным числом пар может быть разным, поэтому программа так же должна определять минимальное и максимальное время вычисления транзитивного замыкания сгенерированных отношений. Выполнить программу при N = 50, 100 и 150. Результат для каждого N представить в виде таблицы (табл. 3).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N = 50 | | | | | | | | | | | | | | |
| Число пар в отношении | 1 | | |  | | |  | | |  | | |  | |
| min | max | min | | max | min | | max | min | | max | min | | max |
| Алгоритм объединения степеней | 0,0001 | 0,016 | 0,0001 | | 0,032 | 0,0001 | | 0,047 | 0,0001 | | 0,047 | 0,0001 | | 0,016 |
| Алгоритм Уоршала | 0,0001 | 0,032 | 0,0001 | | 0,016 | 0,0001 | | 0,032 | 0,0001 | | 0,016 | 0,0001 | | 0,016 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N = 100 | | | | | | | | | | | | | | |
| Число пар в отношении | 1 | | |  | | |  | | |  | | |  | |
| min | max | min | | max | min | | max | min | | max | min | | max |
| Алгоритм объединения степеней | 0,0001 | 0,046 | 0,031 | | 0,063 | 0,015 | | 0,047 | 0,015 | | 0,125 | 0,0001 | | 0,014 |
| Алгоритм Уоршала | 0,0001 | 0,032 | 0,0001 | | 0,032 | 0,0001 | | 0,032 | 0,0001 | | 0,032 | 0,0001 | | 0,032 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N = 150 | | | | | | | | | | |
| Число пар в отношении | 1 | |  | |  | |  | |  | |
| min | max | min | max | min | max | min | max | min | max |
| Алгоритм объединения степеней | 0,031 | 0,063 | 0,078 | 0,391 | 0,078 | 0,14 | 0,078 | 0,125 | 0,031 | 0,097 |
| Алгоритм Уоршала | 0,031 | 0,063 | 0,015 | 0,047 | 0,015 | 0,062 | 0,015 | 0,047 | 0,015 | 0,078 |

mod.c

#include "mod1.h"

#include "operations.h"

#include <stdlib.h>

#include <time.h>

void getmem(int \*\*\*a, int n)

{

int i = 0;

\*a = calloc(n,sizeof(int\*));

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

(\*a)[i] = calloc(n,sizeof(int));

}

void transitive\_closure1(int \*\*a, int \*\*\*c, int n)

{

int \*\*c2;

getmem(&c2,n);

\*c = a;

Compos(\*c,\*c,c2,n);

while (!Includ(c2,\*c,n))

{

Union(\*c,c2,\*c,n);

Compos(\*c,\*c,c2,n);

}

// free\_mem(c2,n);

}

void transitive\_closure2(int \*\*a, int \*\*\*c, int n)

{

int x,y,z;

\*c = a;

for (x = 0; x < n; x++)

for (y = 0; y < n; y++)

for (z = 0; z < n; z++)

(\*c)[x][y] = (\*c)[x][y] || ((\*c)[x][z] && (\*c)[z][y]);

}

void generator(int \*\*a, int k, int n)

{

int x = 0,i,j;

(time(NULL));

while (x < k)

{

i = rand() % n;

j = rand() % n;

if (a[i][j] != 1)

{

a[i][j] = 1;

x++;

}

}

}

Main.c

#include <stdio.h>

#include <stdlib.h>

#include "operations.h"

#include "mod1.h"

#include "time.h"

void nulling(int \*\*\*a, int n)

{

int i,j;

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

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

(\*a)[i][j] = 0;

}

int main()

{

int i,j,n = 150,k = 1, \*\*a, \*\*c,p;

double min = 10, max = 0,t;

clock\_t time;

getmem(&a,n);

getmem(&c,n);

for (p = 0; p < 1000; p++)

{

nulling(&a,n);

nulling(&c,n);

generator(a,k,n);

time = clock();

transitive\_closure2(a, &c, n);

time = clock() - time;

t = (double) time /CLOCKS\_PER\_SEC;

if (t < min)

min = t;

if (t > max)

max = t;

}

}