Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
"Белгородский государственный технологический университет им. В. Г. Шухова"
(БГТУ им. В.Г. Шухова)

Институт энергетики, информационных технологий и управляющих систем

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 3.2
по дисциплине дискретная математика
тема: Транзитивное замыкание отношения

Выполнил: студент группы ПВ-223
Игнатьев Артур Олегович
Проверил: доцент
Рязанов Юрий Дмитриевич

Белгород 2023

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.

```
void tr_zam_st(int **a, int **c, int N) {
    int i, j;
    int **b, **b1;
    b = (int **) calloc(N, sizeof(int *));
    for (i = 0; i < N; i++)
        b[i] = (int *) calloc(N, sizeof(int));
    b1 = (int **) calloc(N, sizeof(int *));
    for (i = 0; i < N; i++)
        b1[i] = (int *) calloc(N, sizeof(int));
    for (i = 1; i < N; i++)
        for (j = 1; j < N; j++)
            b[i][j] = b1[i][j] = a[i][j];
    for (i = 2; i < N; i++) {
        compose(b1, a, b, N);
        unit(c, b, c, N);
        for (i = 1; i < N; i++)
            for (j = 1; j < N; j++)
                b1[i][j] = b[i][j];
    }
}

void tr_zam_W(int **a, int **c, int n) {
    int x, y, z;
    for (x = 1; x < n; x++)
        for (y = 1; y < n; y++)
            for (z = 1; z < n; z++)
                c[x][y] = c[x][y] || c[x][z] && c[z][y];
}
```

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

```
void form(int **a, int N, int k) {
    int x, y, i = 0;
    srand(124);
    while (i < k) {
        x = rand() % N;
        y = rand() % N;
        if (!a[x][y]) {
            a[x][y] = 1;
            i++;
        }
    }
}
```

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.

```
void main()
{
    setlocale(LC_ALL, "Rus");
    int i, N=200;
    int **a;
    a=(int **)calloc(N,sizeof(int*));
    for (i=0;i<N;i++)
        a[i]=(int *)calloc(N,sizeof(int));
    int **c;
    c=(int **)calloc(N,sizeof(int*));
    for (i=0;i<N;i++)
        c[i]=(int *)calloc(N,sizeof(int));
    time_count(a,c,N);
}
void time_count (int **a, int **c, int N)
{
    int k;
    k=1;
    sravn (a,c,N,k);
    k=N*N/4;
    sravn (a,c,N,k);
    k=N*N/2;
    sravn (a,c,N,k);
    k=N*N*2/3;
    sravn (a,c,N,k);
    k=N*N;
    sravn (a,c,N,k);
}
void sravn (int **a, int **c, int N, int k)
{

    double l,min1,max1,s,min2,max2;
    clock_t start, end,d1,d2;
    int i,x,y;
    printf ("k=%d\n",k);
    max1=0;
    max2=0;
    min1=min2=clock();
    for (i=1;i<=1000;i++)
    {
        start=clock();
        form(a,N,k);
        for (x=1;x<N;x++)
            for (y=1;y<N;y++)
                c[x][y]=a[x][y];

        tr_zam_st(a,c,N);
        end=clock();
        s=((double) end - start) / ((double) CLOCKS_PER_SEC);
        if (s>max1)
```

```
            max1=s;
        if (s<min1)
            min1=s;
        clear (c,N);

        start=clock();
        tr_zam_W(a,c,N);
        end=clock();
        s=((double) end - start) / ((double) CLOCKS_PER_SEC);
        if (s>max2)
            max2=s;
        if (s<min2)
            min2=s;
        clear (c,N);
        clear (a,N);
    }
    printf ("\nmin1 = %f\nmax1 = %f\nmin2 = %f\nmax2 = %f\n", min1,max1,
min2,max2);
}

void tr_zam_st (int **a, int **c, int N)
{
    int i,j;
    int **b,**b1;
    b=(int **)calloc(N,sizeof(int*));
    for (i=0;i<N;i++)
        b[i]=(int *)calloc(N,sizeof(int));
    b1=(int **)calloc(N,sizeof(int*));
    for (i=0;i<N;i++)
        b1[i]=(int *)calloc(N,sizeof(int));
    for (i=1;i<N;i++)
        for (j=1;j<N;j++)
            b[i][j]=b1[i][j]=a[i][j];
    for (i=2;i<N;i++)
    {
        compose (b1,a,b,N);
        unit (c,b,c,N);
        for (i=1;i<N;i++)
            for (j=1;j<N;j++)
                b1[i][j]=b[i][j];
    }
}

void clear (int **a, int n)
{
    int x,y;
    for (x=0;x<n;x++)
        for (y=0;y<n;y++)
            a[x][y]=0;
}
//объединение
void unit (int **a, int **b, int **res, int N)
{
    int x, y;
    for (x=1;x<N;x++)
        for (y=1;y<N;y++)
            res[x][y]=a[x][y] || b[x][y];
}
//композиция
void compose (int **a,int **b, int **res, int N)
{
    int x, y, z;
    for (x=1;x<N;x++)
        for (y=1;y<N;y++)
```

```
        {
            res[x][y]=0;
            for(z=1;z<N;z++)
                res[x][y]=res[x][y] || a[x][z] && b[z][y];
        }
}
```

4. Время выполнения алгоритмов

N=50

Невозможно определить время при выполнении алгоритма Уоршалла.

N=100

Число пар в отношении	1		$N^2/4$		$N^2/2$		$N^2*2/3$		N^2	
	min	max	min	max	min	max	min	max	min	max
Алгоритм объединения степеней	0,008	0,031	0,007	0,019	0,006	0,018	0,007	0,018	0,012	0,03
Алгоритм Уоршалла	0,007	0,028	0,007	0,017	0,007	0,019	0,007	0,022	0,008	0,03

N=150

Число пар в отношении	1		$N^2/4$		$N^2/2$		$N^2*2/3$		N^2	
	min	max	min	max	min	max	min	max	min	max
Алгоритм объединения степеней	0,027	0,074	0,022	0,052	0,022	0,061	0,021	0,061	0,036	0,095
Алгоритм Уоршалла	0,026	0,058	0,025	0,063	0,025	0,084	0,025	0,084	0,025	0,068

N=200

Число пар в отношении	1		$N^2/4$		$N^2/2$		$N^2*2/3$		N^2	
	min	max	min	max	min	max	min	max	min	max

Алгоритм объединения степеней	0,067	0,144	0,053	0,107	0,052	0,214	0,053	0,248	0,09	0,68
Алгоритм Уоршалла	0,064	0,162	0,065	0,161	0,063	0,367	0,063	0,321	0,063	0,382

В случае невозможности определения времени выполнения алгоритмов, рекомендуется изменить количество генерируемых отношений и их мощности.

Вывод: на этой лабораторной работе я изучил и выполнил сравнительный анализ алгоритмов вычисления транзитивного замыкания отношения.