Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Кубанский государственный университет»

Кафедра вычислительных технологий

**ОТЧЕТ**

о выполнении лабораторной работы № 6

по дисциплине Теория параллельных алгоритмов

Тема: Алгоритм Флойда поиска кратчайших путей между всеми вершинами графа

Выполнила: ст. гр. 36/1

Фролова А.А.

Проверила: Патыковская М.В.

Краснодар

2024

**Цель работы:**

Построить последовательную и параллельную программы для поиска кратчайших путей между всеми вершинами взвешенного графа с помощью алгоритма Флойда. Проследить зависимость времени работы от количества потоков и ядер.

**Ход работы.**

Данные, полученные в результате экспериментов:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 потока | 3 потока | 4 потока | 5 потоков | 6 потоков | 7 потоков | 8 потоков | последовательно |
| 100 | 0,013 | 0,014 | 0,013 | 0,011 | 0,011 | 0,013 | 0,013 | 0,023 |
| 200 | 0,115 | 0,055 | 0,083 | 0,033 | 0,03 | 0,03 | 0,036 | 0,184 |
| 300 | 0,198 | 0,131 | 0,092 | 0,089 | 0,095 | 0,078 | 0,091 | 0,618 |
| 400 | 0,397 | 0,269 | 0,24 | 0,191 | 0,182 | 0,165 | 0,172 | 1,626 |
| 500 | 0,83 | 0,549 | 0,463 | 0,411 | 0,397 | 0,436 | 0,328 | 3,255 |

Полученные диаграммы:

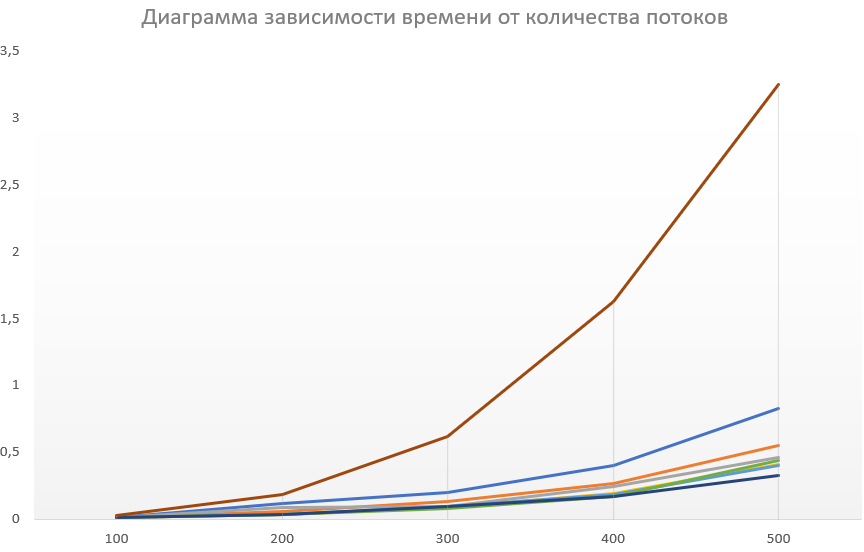
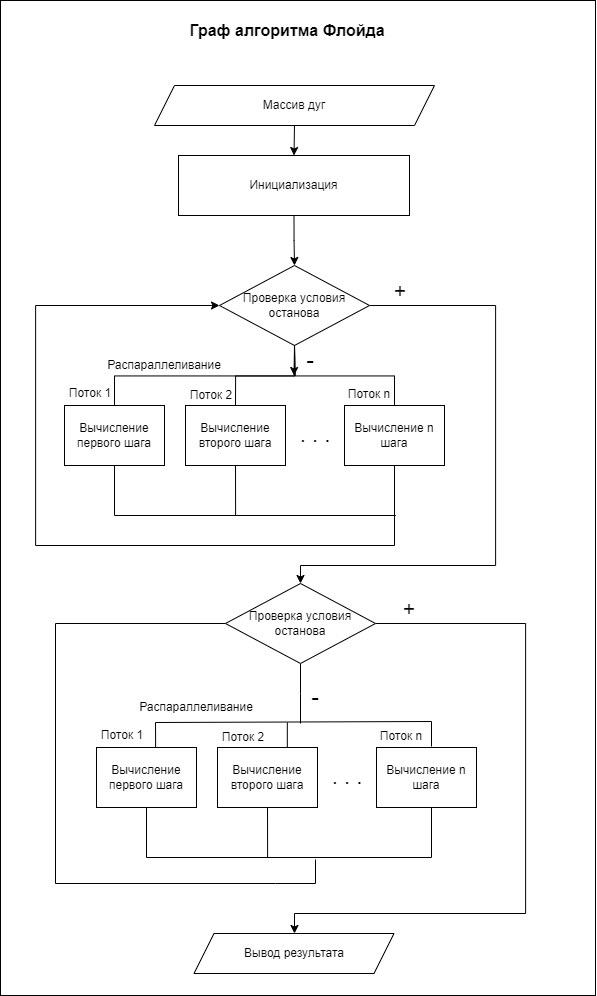


Рисунок 3 – Диаграмма зависимости времени работы от количества потоков

**Граф алгоритма**

****

**Листинг программ**

Сортировка Шелла:

- Последовательно:

#include <cstdlib>

#include <iostream>

#include <omp.h>

#include <ctime>

using namespace std;

int main() {

setlocale(LC\_ALL, "rus");

clock\_t start = clock();

int n = 200;

int a[200][200];

long long maxr = 0;

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

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

a[i][j] = rand() % 10 + 2;

}

}

for (int k = 0; k < n; k++) {

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

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

if (i != j && a[i][k] != -1 && a[k][j] != -1) {

if (a[i][j] == -1) {

a[i][j] = a[i][k] + a[k][j];

}

else

{

a[i][j] = min(a[i][j], a[i][k] + a[k][j]);

}

}

}

}

}

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

{

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

{

if (a[i][j] > maxr) maxr = a[i][j];

}

}

cout << maxr << endl;

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("\nВремя выполнения: %f секунд\n", seconds);

system("pause");

return 0;

}

- Параллельно:

#include <cstdlib>

#include <iostream>

#include <omp.h>

#include <ctime>

using namespace std;

int main() {

setlocale(LC\_ALL, "rus");

clock\_t start = clock();

int n = 500;

int a[500][500];

long long maxr = 0;

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

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

a[i][j] = rand() % 10 + 2;

}

}

#pragma omp parallel for num\_threads(8)

for (int k = 0; k < n; k++) {

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

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

if (i != j && a[i][k] != -1 && a[k][j] != -1) {

if (a[i][j] == -1) {

a[i][j] = a[i][k] + a[k][j];

}

else

{

a[i][j] = min(a[i][j], a[i][k] + a[k][j]);

}

}

}

}

}

#pragma omp parallel for reduction(max: maxr)

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

{

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

{

if (a[i][j] > maxr) maxr = a[i][j];

}

}

cout << maxr << endl;

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("\nВремя выполнения: %f секунд\n", seconds);

system("pause");

return 0;

}

Вывод:

В результате выполнения лабораторной работы №6 были созданы программы для поиска кратчайших путей между всеми вершинами взвешенного графа с помощью алгоритма Флойда. В ходе работы было установлено, что оптимальное количество потоков было примерно равно 8 ядер. Также можно отметить, что обе реализации довольно быстро работают при малой размерности систему, но последовательный значительно замедляется с ростом этого показателя. В дальнейшем полученные знания будут использованы для написания более сложных программ.