Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работ №3 по курсу

«Операционные системы»

Группа: М80 – 207Б-18

Студент: Цапков Александр Максимович

Преподаватель: Миронов Евгений Сергеевич

Оценка: _____

Дата: _____

Москва, 2019.

Содержание

Постановка задачи.

Произвести поиск кратчайшего пути в графе поиском в ширину. Граф задается матрицей смежности, где элементы этой матрицы указывают расстояние между вершинами. Ключом программе может задаваться максимальное количество потоков.

Общие сведения о программе

Программа состоит из одного исполняемого файла. В лабораторной работе были использованы следующие системные вызовы.

1. **thread()** – для создания потока
2. **join** – блокировка до завершения потока
3. **lock** – для блокировка мьютекса.
4. **unlock** – для разблокировки мьютекса

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Реализовать ввод матрицы смежности и узлов по которым считаем путь.
2. Реализовать функцию преобразования взвешенного графа в невзвешенный.
3. Реализовать в функцию поиска пути. Остальные пункты описывают ее реализацию.
4. Создаем 2 очереди: слоя с которым работаем сейчас и следующего слоя
5. Для каждого узла из очереди настоящего слоя делаем поток, который добавляет дочерние узлы данного узла в очередь следующего уровня, если они еще не обрабатывались. Тут используется мьютекс для работы с очередью следующего уровня.
6. После создания всех потоков мы ждем их завершения с помощью join.
7. После меняем местами очереди следующего уровня и настоящего и повторяем этот процесс до тех пор пока не найдется искомый узел или пока не кончится граф (ничего не будет в очереди следующего уровня).

Основные файлы программы.

Файл main.cpp

```
#include <iostream>
#include <thread>
```

```
#include <mutex>
#include <list>
#include <vector>
#include <queue>

int THREAD_COUNT = 1;
std::mutex NL_m;

template<typename L>
void start_thread(std::vector<std::thread>& threads, L&& task)
{
   for(auto&& thread: threads)
   {
      if(thread.joinable())
         continue;
      thread = std::thread(task);
      puts("New thread created");
      return;
   }

   for(auto&& thread: threads)
   {
      if(!thread.joinable())
         continue;
      puts("Waiting for other thread to finish");
      thread.join();
      puts("Done. New thread created");
      thread = std::thread(task);
      return;
   }
}

template <typename T>
int minPath(T M, int from, int to) {
   if (from == to)
      return 0;
   std::vector<std::thread> threads(THREAD_COUNT);
```

```
std::queue<int> CL;
std::queue<int> NL;
int visited[M.size()]{0};
visited[from] = 1;
int path = 0;
int end = 0, closed = 1;
CL.push(from);
while (!end) {
    if (CL.empty())
        return -1;
    path++;
    puts("New layer");
    while (!CL.empty()) {
        int CN = CL.front();
        CL.pop();
        start_thread(threads, [&, CN]{
            for (int i: M[CN]) {
                if (visited[i] == 0) {
                    visited[i] = 1;
                    if (i == to) {
                        end = 1;
                    } else {
                        NL_m.lock();
                        NL.push(i);
                        NL_m.unlock();
                    }
                }
            }
        });
    }
    for(auto&& thread: threads)
    {
        if(!thread.joinable())
            continue;
        thread.join();
    }
    swap(CL, NL);
```

```
    }
    return path;
}

template <typename T>
auto toNoW(T& M, int N) {
    std::vector<std::list<int>> out(N);
    int add = N;
    for (int i = 0; i < N; i++) {
        for (int k = 0; k < N; k++) {
            if (M[i][k] > 0) {
                if (M[i][k] == 1){
                    out[i].push_back(k);
                } else {
                    int rest = M[i][k] - 1;
                    out[i].push_back(add);
                    rest--;
                    while(rest--) {
                        out.push_back(std::list<int>{++add});
                    }
                    out.push_back(std::list<int>{k});
                    add++;
                }

            }
        }
    }
    std::cout << add - N << " nodes added\n";
    return out;
}

int main(int argc, char *argv[]) {
    if (argc == 2)
        THREAD_COUNT = std::atoi(argv[1]);
    int N;
    std::cin >> N;
    int** M = new int*[N];
```

```
    for(int i = 0; i < N; i++)
        M[i] = new int[N];
    for (int i = 0; i < N; i++) {
        for (int k = 0; k < N; k++) {
            std::cin >> M[i][k];
        }
    }
    int from, to;
    std::cin >> from >> to;
    auto ML = toNoW(M, N);
    for (int i = 0; i < ML.size(); i++) {
        std::cout << i << ": ";
        for (int k : ML[i]) {
            std::cout << k << ' ';
        }
        std::cout << '\n';
    }
    int min = minPath(ML, from, to);
    std::cout << "Total path: " << min << std::endl;
    delete[] M;
    return 0;
}
```

Пример работы программы

```
iPad-Alex:~/Documents/Study/Labs/OS/sem3os/lab3/src# g++
-std=c++14 main.cpp
iPad-Alex:~/Documents/Study/Labs/OS/sem3os/lab3/src# cat
test.txt
10
0 1 0 0 0 0 0 8 1 1
0 0 2 0 0 0 0 0 1 0
0 0 0 1 0 0 0 1 0 0
```

```
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 7

iPad-Alex:~/Documents/Study/Labs/OS/sem3os/lab3/src# ./
a.out 2 < test.txt
8 nodes added
0: 1 10 8 9
1: 17 8
2: 3 7
3: 4
4: 5
5: 6
6:
7: 6
8:
9: 2
10: 11
11: 12
12: 13
13: 14
14: 15
15: 16
16: 7
17: 2
New layer
New thread created
New layer
New thread created
New thread created
Waiting for other thread to finish
Done. New thread created
Waiting for other thread to finish
Done. New thread created
New layer
New thread created
New thread created
Waiting for other thread to finish
Done. New thread created
Total path: 3
```

```
iPad-Alex:~/Documents/Study/Labs/OS/sem3os/lab3/src#
```

Вывод

Для начала из данной лабораторной работы я вынес что некоторые алгоритмы больше подходят для распараллеливания и некоторые меньше (что, в принципе, очевидно). В моем случае мне пришлось подумать над тем как распараллелить обход графа в ширину. Еще я понял, что лабораторные работы не нужно начинать за 10 часов до начала сдачи. Также я еще раз убедился что англоязычные ресурсы обладают большими объемами данных по темам проходимым в нашем курсе.