БУ ВО Ханты-Мансийского автономного округа – Югры

«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

ОТЧЁТ

по лабораторной работе №8

по дисциплине: «Алгоритмы и структуры данных»

Выполнил: студент(ка) группы №609-31,

Гаврилов Е.Е.

Принял: старший преподаватель кафедры АиКС

Назаров Е.В.

Сургут

2025г.

**Цель работы:** изучить основные алгоритмы работы с графами – обход, построение транзитивного замыкания, поиск кратчайших путей, закрепить навыки структурного и объектно-ориентированного программирования.

**Задание:**

1. Используя исходный код, приведенный в приложении, разработать функции (методы) реализующие задания, представленные в таблице 1 в соответствии с вариантом задания. При этом необходимо выбрать тип графа и методы, которые позволят эффективно решить поставленную задачу. Решение аргументировать.

**Вариант 3:**

Три графа представляют маршруты трех автобусов (множество вершин считается одинаковым). Определить можно ли добраться от остановки A до остановки B.

**Ход работы:**

Для реализации функций был выбран неорентированный невзвешанный граф, поскольку речь в задаче идет только о наличии маршрутов, но не об их длине.

Для решения задачи необходимо реализовать два метода:

Слияние нескольких графов в один (что бы свести задачу к задаче о поиске наличия пути в одном графе).

Обход графа в ширину для определения наличия пути.

Листинг функции слияния графов:

SGraph SGraph::Merge(Graph \* graphs, int size, int n){

SGraph result(size);

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

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

for (int j = i; j< size; j++){

if(graphs[c](i,j) < MAXDOUBLE)

result.AddEdge(i,j);

}

}

}

return result;

}

Листинг функции определения наличия пути в графе:

bool SGraph::PathExist(int a, int b){

if(a<0 || b< 0 || a>= Size() || b>= Size()){

return false;

}

if(a==b) return true;

std::queue<int> q;

std::vector<bool>v(Size(),false);

q.push(a);

v[a]=true;

while(!q.empty() && q.front()!=b){

int p = q.front();

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

if(operator()(p,i) < MAXDOUBLE && v[i]==false){

v[i] = true;

q.push(i);

}

}

q.pop();

}

return (!q.empty() && q.front()==b);

}

Листинг основной функции:

int main(int argc, char\* argv[])

{

try{

std::ifstream inStream("input.txt");

int a =0;

int b = 0;

int size, len;

inStream >> size >> len;

SGraph graphs[3]{{size},{size},{size}};

for(int i = 0; i < 3; i++){

inStream>>a>>b;

while(a!=-1 && b!=-1){

graphs[i].AddEdge(a,b);

inStream>>a>>b;

}

}

SGraph graph = SGraph::Merge(graphs,size,len);

inStream>>a>>b;

graph.Print();

std::cout<<"Points: "<<a<<" "<<b<<std::endl;

if(graph.PathExist(a,b)){

std::cout<<"Path is exists"<<std::endl;

}

else{

std::cout<<"Path is not exists"<<std::endl;

}

}

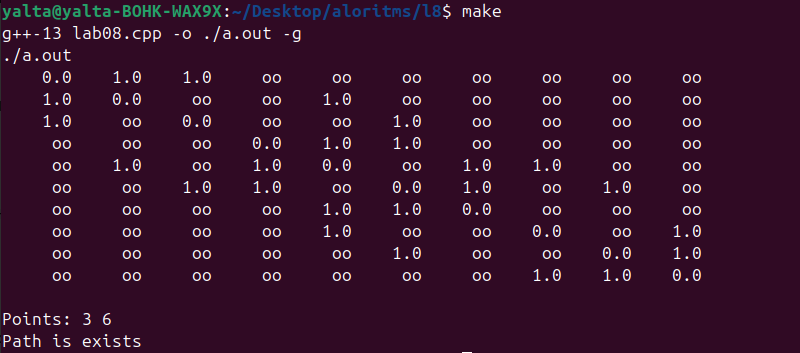
catch(...){

std::cerr<<"ERROR!";

}

return 0;

}

Рис 1. Результат выполнения программы.

**Вывод:** изучены основные алгоритмы работы с графами — обход, построение, поиск кратчайших путей.