МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЕТ

по курсовой работе

по дисциплине «АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ»
Тема: Графы

Студент гр. 3312 _____ Нукусси М.Ф.

Студент гр. 3312 _____ Петров М.

Преподаватель _____ Колинько П.Г.

Санкт-Петербург

2024

1. **Текст индивидуального задания:**
Поиск изоморфного ориентированного подграфа

2. **Математическая формулировка задачи в терминах теории множеств:**

Входом являются два графа G и H и нужно определить, не содержит ли G подграф, изоморфный графу H

Пусть $G = (V, E)$, $H = (V', E')$ — два графа. Существует ли подграф $G_0 = (V_0, E_0) : V_0 \subseteq V, E_0 \subseteq E \cap (V_0 \times V_0)$, такой, что $G_0 \cong H$? Т.е. существует ли отображение $f: V_0 \to V'$, такое, что $(v_1, v_2) \in E_0 \Leftrightarrow (f(v_1), f(v_2)) \in E'$?

3. **Выбор и обоснование способа представления данных.**

Если вы выбрали список смежности для представления графа, это действительно может быть более эффективным способом обхода соседей каждой вершины, особенно если граф разреженный.

4. **Описание алгоритма и оценка его временной сложности.**

Для решения задачи было принято решение использовать алгоритм полного перебора с оптимизацией, основанной на методе Ульмана. Этот метод заключается в создании матрицы соответствия между вершинами двух графов и её последующем уточнении.

Сначала вычисляются степени вершин, и на основе этой информации формируется первоначальный список соответствий. Вершина шаблона может быть изоморфной вершине графа только в том случае, если её степень меньше или равна степени соответствующей вершины графа. Далее происходит процесс уточнения этого списка: пока для каждой вершины шаблона не будет найдено соответствие в графе или не будет доказано, что изоморфный подграф не существует.

Уточнение выполняется следующим образом: выбираем вершину шаблона A, рассматриваем её список соответствий и выбираем вершину A' из этого списка. Затем проверяем, может ли A' быть изоморфной A. Для этого анализируем соседние вершины A, и для каждой из них хотя бы одна вершина из их списка соответствий должна находиться в списке смежности A'. Если это условие не выполняется, удаляем A' из списка соответствий для A. Процесс уточнения продолжается до тех пор, пока это возможно.

После этого устанавливаем вершину A в какое-то значение A'' из её списка соответствий и снова запускаем алгоритм уточнения. Если в списке соответствий образуется пустой элемент, то A'' удаляется из списка соответствий для A. Если же нет, то мы устанавливаем значение для следующей вершины и повторяем алгоритм. Если удаётся установить значения для всех вершин, выводим получившийся список соответствий.

5. **Набор тестов и результаты проверки алгоритма на ЭВМ.**

```
99                  return list;
100             }
101             else{
102                 auto checkingList = list;
103                 for (int g : checkingList[i]){
104                     auto testList = setElement(checkingList, i, g);
105                     auto newList = subgraphSearch(sample, testList, set + 1);
106                     vector<vector<int>> empty = {{}};
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
modervnt@modervnt-Vivobook-Go-E1504FA-E1504FA:~/C++/courseWork/final$ ./a.out
0 --> 1 3
1 --> 2
2 --> 0
3 --> 3 4
4 --> 5
5 --> 3

6 --> 6 7
7 --> 8
8 --> 6

result
6 : 3
7 : 4
8 : 5

modervnt@modervnt-Vivobook-Go-E1504FA-E1504FA:~/C++/courseWork/final$
```

```
modervnt@modervnt-Vivobook-Go-E1504FA-E1504FA:~/C++/courseWork/final$ g++ main.cpp Graph.cpp
modervnt@modervnt-Vivobook-Go-E1504FA-E1504FA:~/C++/courseWork/final$ ./a.out
0 --> 1 2 5
1 --> 2 4 5
2 --> 3 5
3 --> 0 1 2 3 5
4 --> 1 4 5
5 --> 0 1 2 3 4

6 --> 7 10
7 --> 6 7 8 9
8 --> 8
9 --> 8 9 10
10 --> 6

There is no isomorphic subgraphmodervnt@modervnt-Vivobook-Go-E1504FA-E1504FA:~/C++/courseWork/final$
```

6. Выводы.

Алгоритм поиска изоморфных подграфов может иметь экспоненциальную сложность в худшем случае, что делает его неэффективным для больших графов. Это стоит учитывать при использовании данного подхода.

Список использованных источников:

1.Ullmann Julian R.: An algorithm for Subgraph Isomorphism. Journal of the Association for Computing Machinery. 1976

2. coderoad.ru/17480142/Есть-ли-простой-пример-для-объяснения-алгоритма-Ульмана

7. Приложение: исходный текст программы для ЭВМ (на машинном носителе), файлы с тестами.

main.cpp

```
#include <iostream>
#include <vector>
#include "Graph.h"
using namespace std;

int Graph::cnt = 0;
int main()
{
        //test1
   vector<vector<int>> matrix1 = {{1, 2, 5}, {2, 4, 5}, {3, 5}, {0, 1, 2, 3, 5}, {1, 4, 5}, {0, 1, 2, 3, 4}};
   vector<vector<int>> matrix2 = {{7, 10}, {6, 7, 8, 9}, {8}, {8, 9, 10}, {6}};
        //test2
   vector<vector<int>> matrix3 = {{1, 2, 4}, {0}, {0, 1, 3}, {0, 1, 2}, {1, 2}};
   vector<vector<int>> matrix4 = {{6}, {5, 7}, {5}, {5, 6, 7}};
        //test3
   vector<vector<int>> matrix5 = {{1}, {2}, {3}, {4}, {5}, {}};
   vector<vector<int>> matrix6 = {{7}, {8}, {9}, {10}, {11}, {}};
        //test4
   vector<vector<int>> matrix7 = {{0, 1, 2, 3}, {1, 2 ,3}, {0, 1, 3}, {0, 1 , 2}};
   vector<vector<int>> matrix8 = {{4, 5, 6, 7}, {6, 7}, {5, 7}, {5, 6, 7}};
        //test5
   vector<vector<int>> matrix9 = {{1, 3}, {2}, {0}, {3, 4}, {5}, {3}};
   vector<vector<int>> matrix10 = {{6, 7}, {8}, {6}};


   Graph graph(matrix9);
```

```
    graph.printGraph();
    Graph sample(matrix10);
    sample.printGraph();

    clock_t  begin = clock( );
    auto result = graph.subgraphSearch(sample,
graph.makeEquivalenceList(sample), 0);
    clock_t  end = clock( );
    double time = ((double) (end - begin)) / CLOCKS_PER_SEC;
    cout << "Time is " << time << " seconds";


    vector<vector<int>> empty = {{}};
    if (result != empty){
        cout << "There is no isomorphic subgraph";
    }
 }
```

Graph.cpp

```
#include "Graph.h"
#include <algorithm>

Graph :: Graph(const vector<vector<int>>& matrix) {
    amount = matrix.size();
    adjList.resize(amount);
    adjList = (matrix);
    cnt += amount;
}

Graph :: Graph(int n) {
    amount = n;
    adjList.resize(amount);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++){
            if (rand() %  2 == 0 ) {
                adjList[i].push_back(cnt + j);
            }
        }
    }
    cnt += amount;
}

void Graph :: printGraph()
{
```

```
    for (int i = 0; i < amount; i++)
    {
        cout << cnt - amount + i << " --> ";
        for (int v: adjList[i]) {
            cout << v << " ";
        }
        cout <<  endl;
    }
    cout<< endl;
}

vector<vector<int>> Graph::makeEquivalenceList(Graph sample) {
    vector<vector<int>> list(sample.amount, vector<int>());
    for (int i = 0; i < sample.amount; ++i){
        for (int j = 0; j < amount; ++j){
            if (sample.adjList[i].size() <= adjList[j].size()){
                list[i].push_back(j);
            }
        }
    }
    return list;
}

vector<vector<int>> Graph::updateEquivalenceList(Graph
sample,vector<vector<int>> list) {
    vector<vector<int>> newList(sample.amount, vector<int>());
    for (int i = 0; i < sample.amount; ++i) {
        if (!sample.adjList[i].empty()){
            for (int j = 0; j < list[i].size(); j++) {
                bool conclusion = true;
                int testedElememt = list[i][j];
                for (auto el: sample.adjList[i]) {
                    bool flag = false;
                    int num = el - amount;
                    for (auto k: list[num]) {
                        if (elementInVector(adjList[testedElememt], k)) {flag = true;}
                    }
                    if (!flag){
                        conclusion = false;
                    }
                }
                if (conclusion) newList[i].push_back(list[i][j]);
            }
        }
        else {
```

```
            newList[i] = list[i];
        }
    }
    return newList;
}


vector<vector<int>> Graph:: subgraphSearch(const Graph& sample,
vector<vector<int>> list, int set){
    auto checkList = updateEquivalenceList(sample, list);
    while (checkList != list){
        list = checkList;
        checkList = updateEquivalenceList(sample, list);
    }
    list = checkList;
    if (haveEmptyRows(list)) {
        return list;
    }
    else {
//        printMatrix(list, amount);
        if (set == sample.amount){
            vector<vector<int>> empty = {{}};
            cout << "result" << endl;
            printMatrix(list, amount);
            return empty;
        }
        else{
            for (int i = set; i < sample.amount; i++){
                if (haveEmptyRows(list)){
                    return list;
                }
                else{
                    auto checkingList = list;
                    for (int g : checkingList[i]){
                        auto testList = setElement(checkingList, i, g);
                        auto newList = subgraphSearch(sample, testList, set + 1);
                        vector<vector<int>> empty = {{}};
                        if (newList == empty){
                            return empty;
                        }
                        if (haveEmptyRows(newList)){
                            list[i] = deleteElement(list[i], g);
                        }
                    }
                }
```

```
            }
            return list;
        }
    }
}

bool haveEmptyRows(const vector<vector<int>>& matrix){
//    any_of(matrix.cbegin(), matrix.cend(), [](vector<int> vector){return
vector.empty();});
    for (auto i : matrix){
        if (i.empty()){
            return true;
        }
    }
    return false;
}

vector<vector<int>> setElement(const vector<vector<int>> &matrix, int pos, int
elem) {
    vector<vector<int>> newMatrix(matrix.size());
    for (int i =0; i < matrix.size(); i++){
        if (i == pos) newMatrix[i].push_back(elem);
        else {
            for (int j = 0; j < matrix[i].size(); j++){
                if (matrix[i][j] != elem) newMatrix[i].push_back(matrix[i][j]);
            }
        }
    }
    return newMatrix;
}


vector<int> deleteElement(const vector<int>& vec, int el){
    vector<int> newVec;
    for (int i : vec){
        if (i != el)   newVec.push_back(i);
    }
    return newVec;
}

void printMatrix(const vector<vector<int>>& matrix, int n){
    int printingElement = n;
    for (const auto& i : matrix){
        cout << printingElement << " : ";
        for (auto j : i){
```

```
            cout << j << " ";
        }
        cout << endl;
        printingElement += 1;
    }
    cout << endl;
}

bool elementInVector(const vector<int>& vector, int element){
    //    any_of(vector.cbegin(), vector.cend(), [](int el){return int el == element;});
    for (auto el : vector){
        if (el == element){
            return true;
        }
    }
    return false;
}

vector<vector<int>> inputAdjList(){
    vector<vector<int>> matrix;
    vector<int> row;
    int n, k, element;
    cin >> n;
    cout << "Enter number of rows in adjList";
    for (int i = 0; i < n; i++){
        cout << "Enter number of neighbours of " << i;
        cin >> k;
        for (int j = 0; j < k; j++){
            cout << "Enter an element";
            cin >> element;
            row.push_back(element);
        }

    }
    return matrix;
}
```

Graph.h

```
#ifndef AICD_LAB4
#define AICD_LAB4


#include <iostream>
#include <vector>
```

```
using namespace std;


class Graph
{
private:
    int amount = 0;
    vector<vector<int>> adjList;
public:
    static int cnt;
    Graph(const vector<vector<int>>& matrix);
    explicit Graph(int n);
    void printGraph();
    vector<vector<int>> makeEquivalenceList(Graph sample);
    vector<vector<int>> updateEquivalenceList(Graph sample,vector<vector<int>>
list);
    vector<vector<int>> subgraphSearch(const Graph& sample,
vector<vector<int>> list, int set);
};

bool haveEmptyRows(const vector<vector<int>>& matrix);
vector<vector<int>> setElement(const vector<vector<int>> &matrix, int pos, int
elem);
vector<int> deleteElement(const vector<int>& vec, int el);
void printMatrix(const vector<vector<int>>& matrix, int n);
bool elementInVector(const vector<int>& vector, int element);
vector<vector<int>> inputAdjList();


#endif
```