Министерство образования, культуры и исследований Республики Молдова Технический Университет Молдовы Департамент Программной Инженерии и Автоматики

Лабораторная работа №1

Хранение графов в памяти ЭВМ.

По предмету

Дискретная математика

Подготовил: Оларь Артём,

Группа TI – 210

г. Кишинев 2021

**Цель работы:**  
1. Освоение и изучение способов задания графов:

-- Матрица инцидентности  
-- Матрица смежности  
-- Список смежности

2. Разработка процедур преобразования видов хранения

-- графов с выдачей результатов на дисплей

**Задание:**

1. Разработать процедуры ввода графа в виде:

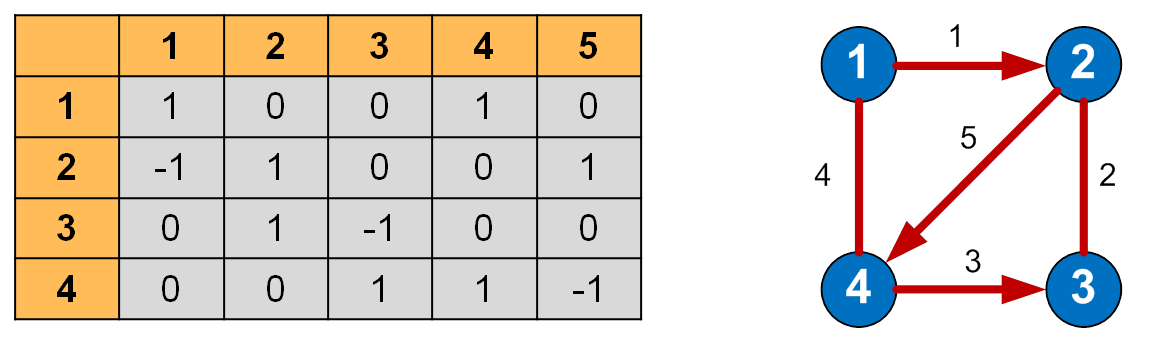
-- матрицы инцидентности  
-- матрицы смежности  
-- списка смежности

2. Разработать процедуры преобразования различных форм хранения графа:   
-- из матрицы смежности в список смежности и обратно  
-- из матрицы инцидентности в список смежности и обратно

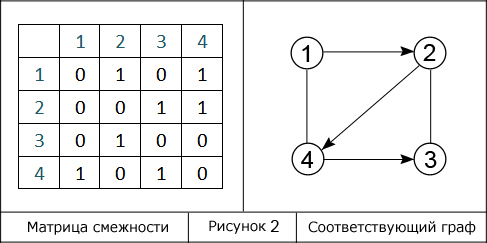
3. Используя указанные выше процедуры, получить программу, выполняющую следующие функции:   
-- ввод графа в любой из трех форм представления (по требованию пользователя)   
-- хранение введенного графа в памяти ЭВМ в виде списка смежности  
-- вывод информации о графе в любом из трех видов ( также по требованию пользователя ) на дисплей;

**Теория:**

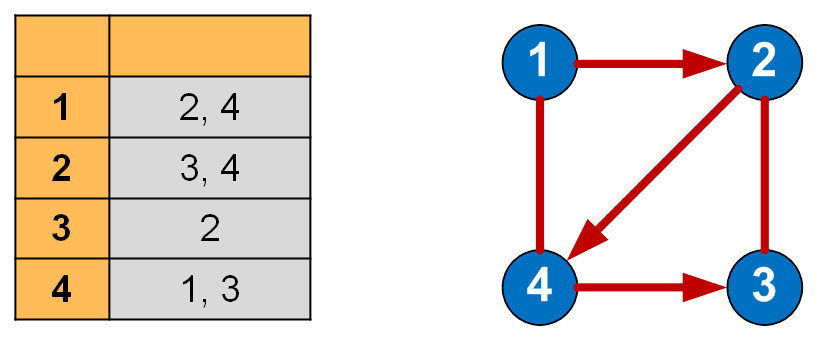
**Матрица инцидентности** — одна из форм представления графа, в которой указываются связи между инцидентными элементами графа (ребро(дуга) и вершина). Столбцы матрицы соответствуют ребрам, строки — вершинам.



**Матрица смежности** — один из способов представления графа в виде матрицы.  
Матрица смежности простого графа (не содержащего петель и кратных рёбер) является бинарной матрицей и содержит нули на главной диагонали.

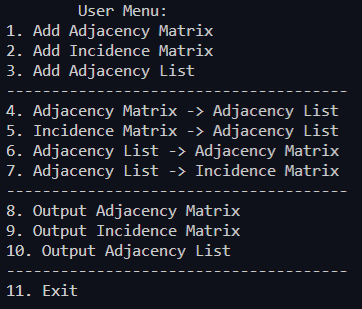


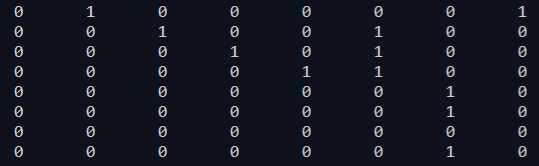
**Список смежности** — один из способов представления графа в виде коллекции списков вершин.   
Каждой вершине графа соответствует список, состоящий из «соседей» этой вершины.

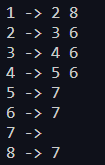


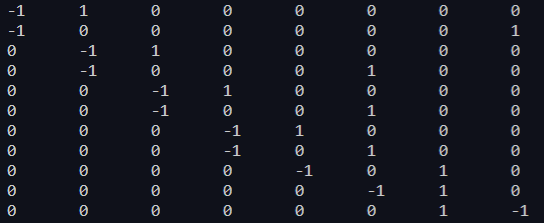
**Работа программы и скриншоты:**

1. Меню программы с вводом нужного значения



2. Ввод матрицы смежности + вывод  


3. Перевод матрицы смежности в список смежности  


4. Перевод списка смежности в матрицу инцидентности  


**Вывод:**

При помощи ЭВМ и структур данных есть возможность хранить и преобразовывать графы в нужный нам вид почти на любом языке программирования, в данном случае C++

**Код программы:**

*#include* <iostream>

*#include* <vector>

*using* *namespace* std;

void menu(int &choice);

void functionsMenu(

  int\*\* &adjMatrix, int\*\* &incMatrix,

  vector <vector <int>> &adjList,

  int &rows, int &cols,

  int &choice);

void addAdjacencyMatrix(int\*\* &array, int &rows);

void addIncidenceMatrix(int\*\* &array, int &rows, int &cols);

void addAdjacencyList(vector <vector <int>> &adjList, int &rows);

void adjacencyList\_adjacencyMatrix(vector <vector <int>> &adjList, int\*\* &array, int &rows);

int checkNumber(vector <int> partOfList, const int step);

void adjacencyMatrix\_adjacencyList(vector <vector <int>> &adjList, int\*\* &array, int &rows);

void incidenceMatrix\_adjacencyList(vector <vector <int>> &adjList, int\*\* &array, int &rows, int &cols);

void adjacencyList\_incidenceMatrix(vector <vector <int>> &adjList, int\*\* &array, int &rows, int &cols);

void showAdjacenctyMatrix(int\*\* &array, int &rows);

void showIncidenceMatrix(int\*\* &array, int &rows, int &cols);

void showAdjacencyList(vector <vector <int>> &adjList);

void deleteMemory(vector <vector <int>> &adjList, int\*\* &array);

int main() {

  int userChoice *=* 0;

  int *\*\**adjacencyMatrix *=* 0;

  int *\*\**incidenceMatrix *=* 0;

  vector *<*vector *<*int*>>* adjacencyList;

  int rows *=* 0;

  int cols *=* 0;

*while* (true) {

    menu(userChoice);

    functionsMenu(

      adjacencyMatrix,

      incidenceMatrix,

      adjacencyList,

      rows, cols,

      userChoice);

*if* (userChoice *==* 11) {

*break*;

    }

  }

*return* 0;

}

void menu(int &choice) {

  int tempUserChoice *=* 0;

*while* (true) {

    cout *<<* "\n\tUser Menu:" *<<* endl;

    cout *<<* "1. Add Adjacency Matrix" *<<* endl;

    cout *<<* "2. Add Incidence Matrix" *<<* endl;

    cout *<<* "3. Add Adjacency List" *<<* endl;

    cout *<<* "--------------------------------------" *<<* endl;

    cout *<<* "4. Adjacency Matrix -> Adjacency List" *<<* endl;

    cout *<<* "5. Incidence Matrix -> Adjacency List" *<<* endl;

    cout *<<* "6. Adjacency List -> Adjacency Matrix" *<<* endl;

    cout *<<* "7. Adjacency List -> Incidence Matrix" *<<* endl;

    cout *<<* "--------------------------------------" *<<* endl;

    cout *<<* "8. Output Adjacency Matrix" *<<* endl;

    cout *<<* "9. Output Incidence Matrix" *<<* endl;

    cout *<<* "10. Output Adjacency List" *<<* endl;

    cout *<<* "--------------------------------------" *<<* endl;

    cout *<<* "11. Exit\n" *<<* endl;

    cout *<<* "Your variant: ";

    cin *>>* tempUserChoice;

*if* (tempUserChoice *>* 0 *&&* tempUserChoice *<* 12) {

*break*;

    } *else* {

      cout *<<* "\nWrong number, again" *<<* endl;

    }

  };

  choice *=* tempUserChoice;

}

void functionsMenu(

  int\*\* &adjMatrix, int\*\* &incMatrix,

  vector <vector <int>> &adjList,

  int &rows, int &cols,

  int &choice) {

*switch* (choice) {

*case* 1:

      addAdjacencyMatrix(adjMatrix, rows);

*break*;

*case* 2:

      addIncidenceMatrix(incMatrix, rows, cols);

*break*;

*case* 3:

      addAdjacencyList(adjList, rows);

*break*;

*case* 4:

      adjacencyMatrix\_adjacencyList(adjList, adjMatrix, rows);

*break*;

*case* 5:

      incidenceMatrix\_adjacencyList(adjList, incMatrix, rows, cols);

*break*;

*case* 6:

      adjacencyList\_adjacencyMatrix(adjList, adjMatrix, rows);

*break*;

*case* 7:

      adjacencyList\_incidenceMatrix(adjList, incMatrix, rows, cols);

*break*;

*case* 8:

      showAdjacenctyMatrix(adjMatrix, rows);

*break*;

*case* 9:

      showIncidenceMatrix(incMatrix, rows, cols);

*break*;

*case* 10:

      showAdjacencyList(adjList);

*break*;

  }

}

void addAdjacencyMatrix(int\*\* &array, int &rows) {

  cout *<<* "Number of peaks? : ";

  cin *>>* rows;

  array *=* *new* int*\**[rows];

*for* (int i *=* 0; i *<* rows; i*++*) {

    array[i] *=* *new* int[rows];

  }

  int temp *=* 0;

*for* (int i *=* 0; i *<* rows; i*++*) {

*for* (int j *=* 0; j *<* rows; j*++*) {

      cout *<<* "Adj. matrix[" *<<* i *<<* "]" *<<* "[" *<<* j *<<* "]" *<<* " = ";

      cin *>>* temp;

      array[i][j] *=* temp;

    }

  }

}

void addIncidenceMatrix(int\*\* &array, int &rows, int &cols) {

  cout *<<* "Number of peaks? : ";

  cin *>>* cols;

  cout *<<* "Number of arcs? : ";

  cin *>>* rows;

  array *=* *new* int*\**[rows];

*for* (int i *=* 0; i *<* rows; i*++*) {

    array[i] *=* *new* int[cols];

  }

  int temp *=* 0;

*for* (int i *=* 0; i *<* rows; i*++*) {

*for* (int j *=* 0; j *<* cols; j*++*) {

      cout *<<* "Incid. matrix[" *<<* i *<<* "]" *<<* "[" *<<* j *<<* "]" *<<* " = ";

      cin *>>* temp;

      array[i][j] *=* temp;

    }

  }

}

void addAdjacencyList(vector <vector <int>> &adjList, int &rows) {

  cout *<<* "Number of peaks? : ";

  cin *>>* rows;

  int peaksInRow *=* 0;

  int tempNum *=* 0;

  vector *<*int*>* row;

*for* (int i *=* 0; i *<* rows; i*++*) {

    cout *<<* "Numbers in a line? : ";

    cin *>>* peaksInRow;

*for* (int j *=* 0; j *<* peaksInRow; j*++*) {

      cin *>>* tempNum;

      row.push\_back(tempNum);

    }

    adjList.push\_back(row);

    row.clear();

  }

}

void adjacencyList\_adjacencyMatrix(vector <vector <int>> &adjList, int\*\* &array, int &rows) {

  array *=* *new* int*\**[rows];

*for* (int i *=* 0; i *<* rows; i*++*) {

    array[i] *=* *new* int[rows];

  }

*for* (int i *=* 0; i *<* rows; i*++*) {

    vector *<*int*>* smallPart *=* adjList*[*i*]*;

*for* (int j *=* 0; j *<* rows; j*++*) {

      int temp *=* checkNumber(smallPart, j);

      array[i][j] *=* temp;

    }

  }

}

int checkNumber(vector <int> partOfList, const int step) {

*if* (partOfList.size() *==* 0) {

*return* 0;

  } *else* {

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

*if* (partOfList*[*i*]* *==* step *+* 1) {

*return* 1;

      }

    }

*return* 0;

  }

}

void adjacencyMatrix\_adjacencyList(vector <vector <int>> &adjList, int\*\* &array, int &rows) {

  vector *<*int*>* row;

*for* (int i *=* 0; i *<* rows; i*++*) {

*for* (int j *=* 0; j *<* rows; j*++*) {

*if* (array[i][j] *==* 1) {

        row.push\_back(j *+* 1);

      }

    }

    adjList.push\_back(row);

    row.clear();

  }

}

void incidenceMatrix\_adjacencyList(vector <vector <int>> &adjList, int\*\* &array, int &rows, int &cols) {

  vector *<*int*>* row;

*for* (int i *=* 0; i *<* cols; i*++*) {

*for* (int j *=* 0; j *<* rows; j*++*) {

*if* (array[j][i] *==* *-*1) {

*for* (int k *=* 0; k *<* cols; k*++*) {

*if* (array[j][k] *==* 1) {

            row.push\_back(k *+* 1);

          }

        }

      }

    }

    adjList.push\_back(row);

    row.clear();

  }

}

void adjacencyList\_incidenceMatrix(vector <vector <int>> &adjList, int\*\* &array, int &rows, int &cols) {

  cout *<<* "Number of arcs? : ";

  cin *>>* cols;

  int temp *=* rows;

  rows *=* cols;

  cols *=* temp;

  array *=* *new* int*\**[rows];

*for* (int i *=* 0; i *<* rows; i*++*) {

    array[i] *=* *new* int[cols];

  }

  int rowPosition *=* 0;

  int vectorNumber *=* 0;

  int i *=* 0;

*while* (rowPosition *<* rows) {

    vector *<*int*>* rowAdjList *=* adjList*[*vectorNumber*]*;

*while* (i *<* rowAdjList.size()) {

*for* (int j *=* 0; j *<* cols; j*++*) {

*if* (vectorNumber *==* j) {

          array[rowPosition][vectorNumber] *=* *-*1;

        } *else* *if* (rowAdjList*[*i*]* *==* j *+* 1) {

          array[rowPosition][j] *=* 1;

        } *else* {

          array[rowPosition][j] *=* 0;

        }

      }

      i*++*;

      rowPosition*++*;

    }

    i *=* 0;

    vectorNumber*++*;

  }

}

void showAdjacenctyMatrix(int\*\* &array, int &rows) {

*for* (int i *=* 0; i *<* rows; i*++*) {

*for* (int j *=* 0; j *<* rows; j*++*) {

      cout *<<* array[i][j] *<<* "\t";

    }

    cout *<<* endl;

  }

}

void showIncidenceMatrix(int\*\* &array, int &rows, int &cols) {

*for* (int i *=* 0; i *<* rows; i*++*) {

*for* (int j *=* 0; j *<* cols; j*++*) {

      cout *<<* array[i][j] *<<* "\t";

    }

    cout *<<* endl;

  }

}

void showAdjacencyList(vector <vector <int>> &adjList) {

  int i *=* 1;

*for* (vector *<*int*>* row : adjList) {

    cout *<<* i *<<* " -> ";

*for* (int elem : row) {

      cout *<<* elem *<<* " ";

    }

    cout *<<* endl;

    i*++*;

  }

}