Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №3**

По дисциплине «Теоретические и множественные основы интеллектуальных систем»

Тема: «Графы»

**Выполнил:**

Студент 1 курса

Группы ИИ-21

Литвинюк Т. В.

**Проверил:**

Глущенко Т. А.

Брест 2022

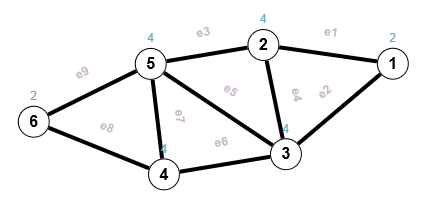
**Цель работы:** нахождение компонент связности неориентированного графа.

**Ход работы:**

**Вариант 8**

**Задание 1**

Построить матрицу смежности и инцидентности для заданного графа. Изобразить граф.

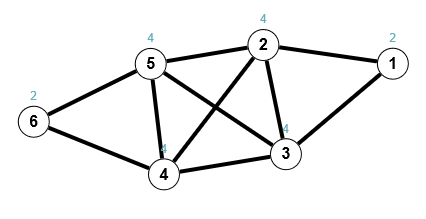
****Матрица смежности:

**** 1 2 3 4 5 6

Матрица инцидентности:

**Задание 2**

По матрице смежности (инцидентности) для каждой из вершин вычислить ее степень.

****

**Задание 3**

Используя поиск в глубину и поиск в ширину написать программу, определяющую число компонент связности графа. Методы представляются в виде отдельных функций (или классов).

import collections

def bfs(graph, root):

visited, queue = set(), collections.deque([root])

visited.add(root)

while queue:

vertex = queue.popleft()

for neighbor in graph[vertex]:

if neighbor not in visited:

visited.add(neighbor)

queue.append(neighbor)

return list(visited)

def find\_num(graph):

num = 1

vertexes = list(graph.keys()) # Все вершины

while True:

bfs\_list = bfs(graph, vertexes[0]) # Все посещенные вершины, стартуя с vertexes[0]

for x in bfs\_list:

vertexes.remove(x)

if len(vertexes) == 0:

return num

num += 1

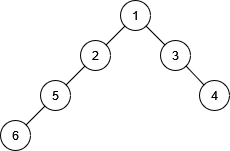


graph = {1: [2,3], 2: [1,3,4,5], 3: [1,2,5], 4: [2,3,5,6], 5:[2,3,4,6], 6:[4,5]}

print(find\_num(graph))

**Задание 4**

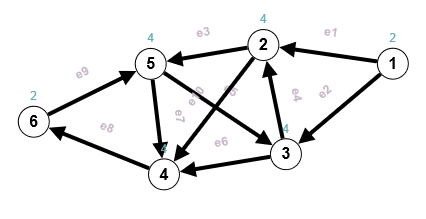
Построить деревья поиска в ширину и глубину.





**Задание 5**

Из заданного неориентированного графа построить произвольным образом ориентированный граф (добавить к каждому ребру стрелку). Для полученного таким образом ориентированного графа построить матрицу смежности и инцидентности.

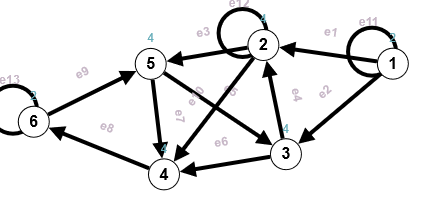
Матрица смежности:

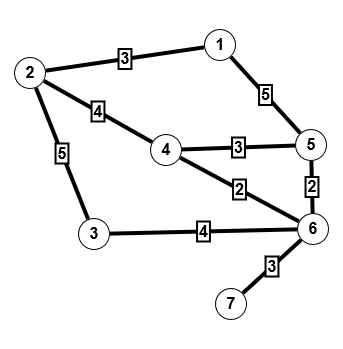
****  1 2 3 4 5 6

Матрица инцидентности:

**Задание 6**

Из заданного неориентированного графа построить произвольным образом псевдограф.



**Задание 7**

Нарисовать граф.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 |  | 3 |  |  | 5 |  |  |
| 2 | 3 |  | 5 | 4 |  |  |  |
| 3 |  | 5 |  |  |  | 4 |  |
| 4 |  | 4 |  |  | 3 | 2 |  |
| 5 | 5 |  |  | 3 |  | 2 |  |
| 6 |  |  | 4 | 2 | 2 |  | 3 |
| 7 |  |  |  |  |  | 3 |  |

**Задание 8**

Программно алгоритмом Дейкстры вычислить кратчайшие пути от вершины ко всем вершинам графа.

def dijkstra(graph, src):

src -= 1

visited = []

distance = {src+1: 0}

node = list(range(len(graph[0])))

if src in node:

node.remove(src)

visited.append(src)

else:

return None

for i in node:

distance[i+1] = graph[src][i]

prefer = src

while node:

\_distance = float('inf')

for i in visited:

for j in node:

if graph[i][j] > 0:

if \_distance > distance[i+1] + graph[i][j]:

\_distance = distance[j+1] = distance[i+1] + graph[i][j]

prefer = j

visited.append(prefer)

node.remove(prefer)

return distance

graph = [

[0, 3, 0, 0, 5, 0, 0],

[3, 0, 5, 4, 0, 0, 0],

[0, 5, 0, 0, 0, 4, 0],

[0, 4, 0, 0, 3, 2, 0],

[5, 0, 0, 3, 0, 2, 0],

 [0, 0, 4, 2, 2, 0, 3],

[0, 0, 0, 0, 0, 3, 0]

]

print(dijkstra(graph, 1))

**Задание 9**

Нарисовать таблицу пошагового выполнения алгоритма Дейкстры.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № итерации | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Кратчайшее расстояние  от 1 до i |
| 1 | 0\* | 3\* | ∞ | ∞ | 5 | ∞ | ∞ | D2=3 |
| 2 |  |  | 8 | 7\* | 5 | ∞ | ∞ | D4=7 |
| 3 |  |  | 8 |  | 5\* | 9 | ∞ | D5=5 |
| 4 |  |  | 8 |  |  | 7\* | ∞ | D6=7 |
| 5 |  |  | 8\* |  |  |  | 10 | D3=8 |
| 6 |  |  |  |  |  |  | 10\* | D7=10 |

**Задание 10**

Программно алгоритмом Флойда-Уоршолла вычислить кратчайшие пути от вершины ко всем вершинам графа.

from math import inf

def get\_path(P, u, v):

path = [u]

while u != v:

u = P[u][v]

path.append(u)

return path

V = [

[inf, 3, inf, inf, 5, inf, 0],

[3, inf, 5, 4, inf, inf, 0],

[inf, 5, inf, inf, inf, 4, 0],

[inf, 4, inf, inf, 3, 2, 0],

[5, inf, inf, 3, inf, 2, 0],

[inf, inf, 4, 2, 2, inf, 3],

[inf, inf, inf, inf, inf, 3, 0]]

N = len(V)

P = [[v for v in range(N)] for u in range(N)]

for k in range(N):

for i in range(N):

 for j in range(N):

d = V[i][k] + V[k][j]

if V[i][j] > d:

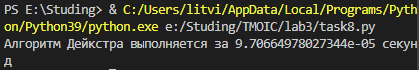
V[i][j] = d

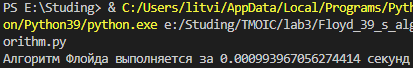
P[i][j] = k

print(get\_path(P, 2, 0))

**Задание 11**

Сравнить скорость выполнения обоих алгоритмов для заданного графа.





**Вывод:** в ходе лабораторной работы я научился находить компонент связности неориентированного графа, находить кратчайший путь в графе.