|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №5**

**«Остовы графов»**

**ДИСЦИПЛИНА: «Дискретная математика»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-32Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_Зудин Д.В.\_\_\_\_\_)  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_Никитенко У.В.\_\_\_)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2022 г.  **Цель**: изучение алгоритмов обхода графов и алгоритмов поиска остова.  **Задачи**:  1. Построение остова;  2. Поиск в ширину и поиск в глубину.  **Вариант №4**  **Формулировка общего задания**  Для графа *G4*, полученного в лабораторной работе №4, построить остов. Четные варианты – методом обхода в ширину, нечетные варианты – методом обхода в глубину.  **Результат выполнения программы для общего задания**  [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]  [1, 0, 1, 0, 0, 0, 0, 0, 0, 0]  [0, 1, 0, 1, 0, 0, 0, 0, 0, 0]  [0, 0, 1, 0, 1, 0, 0, 0, 0, 0]  [0, 0, 0, 1, 0, 1, 0, 0, 0, 0]  [0, 0, 0, 0, 1, 0, 1, 0, 0, 0]  [0, 0, 0, 0, 0, 1, 0, 1, 0, 0]  [0, 0, 0, 0, 0, 0, 1, 0, 1, 0]  [0, 0, 0, 0, 0, 0, 0, 1, 0, 1]  [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]  **Листинг программы для общего задания**  *class* Graph:   *def \_\_init\_\_*(*self*, adj):  *self*.adj = adj  *self*.v = len(adj)  *self*.vlist = []   *def* add\_edge(*self*, a, b):  *self*.adj[a][b] = 1  *self*.adj[b][a] = 1  *if* a == b:  *self*.adj[a][b] = 2   *def* BFS(*self*, start):  visited = [*False*] \* *self*.v  q = [start]  visited[start] = *True   while* q:  vis = q[0]  *self*.vlist.append(vis)  q.pop(0)  *for* i *in* range(*self*.v):  *if* (*self*.adj[vis][i] != 0 *and* (*not* visited[i])):  q.append(i)  visited[i] = *True* G\_1 = [[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, 1, 1, 0, 0],  [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],  [0, 1, 0, 0, 0, 1, 0, 0, 0, 0],  [0, 0, 0, 0, 1, 0, 0, 1, 1, 0],  [0, 0, 1, 0, 0, 0, 0, 1, 1, 0],  [0, 0, 1, 0, 0, 1, 1, 0, 0, 0],  [0, 0, 0, 0, 0, 1, 1, 0, 0, 0],  [0, 0, 0, 1, 0, 0, 0, 0, 0, 0], ]  dop\_G\_1 = G\_1  *for* i *in* range(len(G\_1)):  *for* j *in* range(len(G\_1)):  e = 1  *if* i == j:  e = 2  *if* G\_1[i][j] == e:  dop\_G\_1[i][j] = 0  *if* G\_1[i][j] == 0:  dop\_G\_1[i][j] = e  G\_0 = [[0 *for* \_ *in* range(len(G\_1))] *for* \_ *in* range(len(G\_1))]  a = Graph(dop\_G\_1) b = Graph(G\_0)  a.BFS(0)  *for* i *in* range(len(a.vlist) - 1):  b.add\_edge(a.vlist[i], a.vlist[i + 1])  print(\*b.adj, sep='\n')  **Выводы:**  В ходе работы были изучены основы теории графов, базовые понятия и определения; освоены компьютерные способы представления графов и операции над ними. | | |