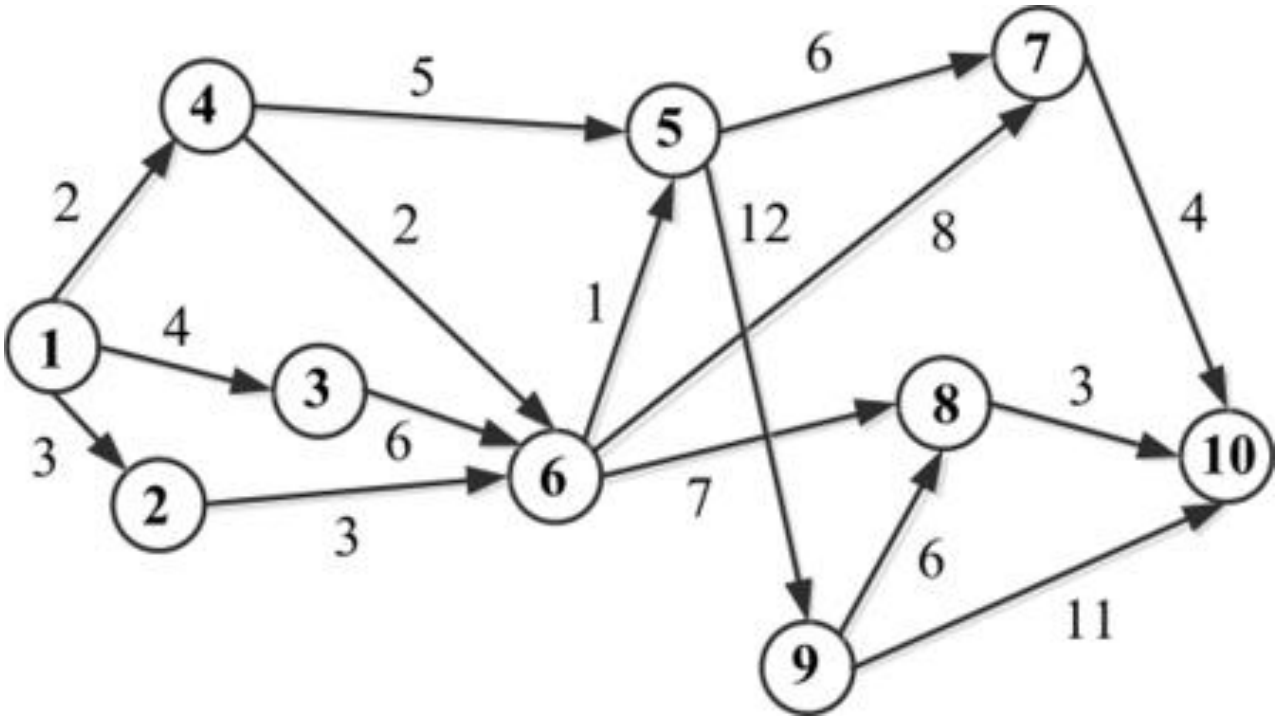
Задания для студентов III курса в рамках Midterm N2

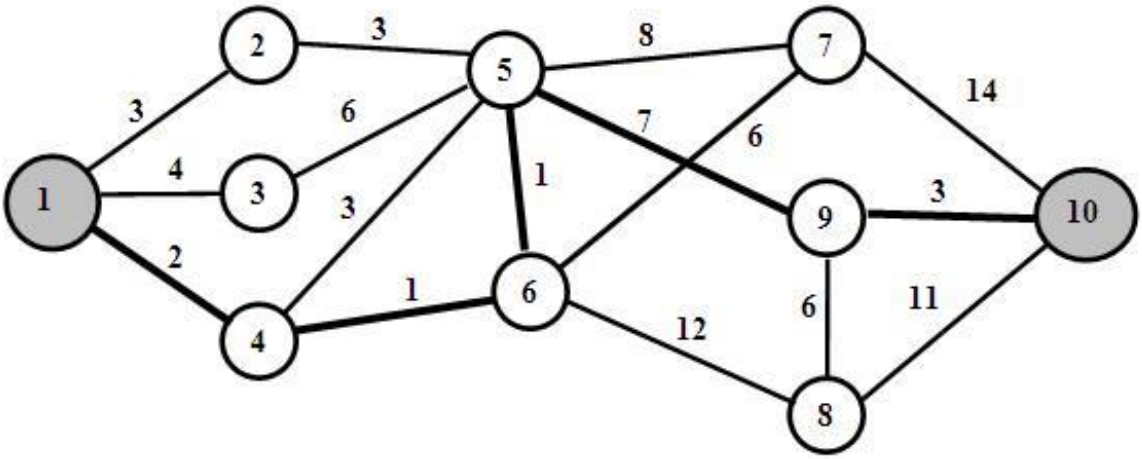
в TSI AUCA по предмету

«Разработка программного обеспечения».

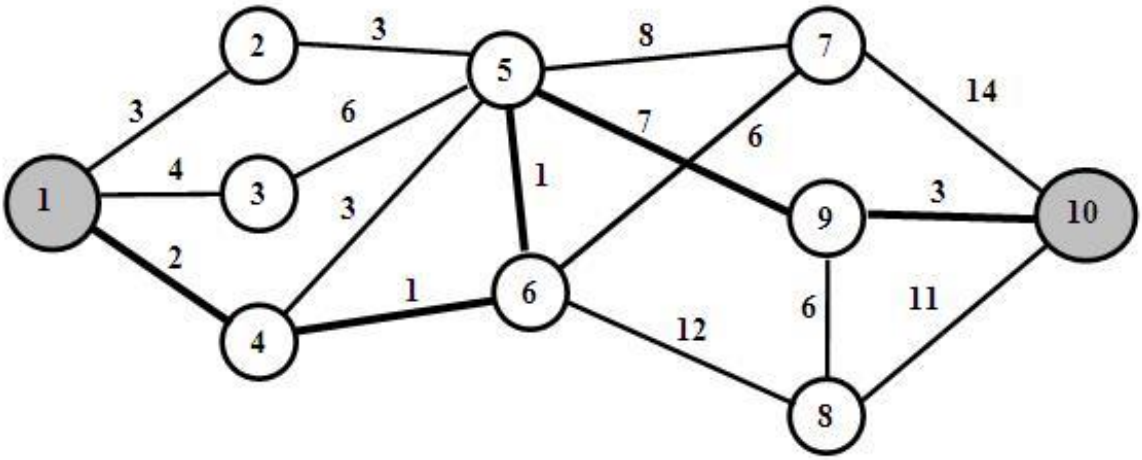
Задача №1 (2,5 балла). Дан взвешенный ориентированный граф, состоящий из 10 вершин. Используя алгоритм Дейкстры, найти кратчайшие расстояния от вершины №1 до каждой из остальных девяти вершин. Работу алгоритма Дейкстры подробно расписать в таблице. В итоге, выписать значения кратчайших расстояний от вершины №1 до каждой из остальных девяти вершин.



Задача №2 (2,0 балла). Дан взвешенный неориентированный граф, состоящий из 10 вершин. Используя алгоритм Крускала, построить минимальное остовное дерево. Подробно описать работу алгоритма Крускала. Отобразить в итоге графически построенное минимальное остовное дерево.



Задача №3 (2,0 балла). Дан взвешенный неориентированный граф, состоящий из 10 вершин. Используя алгоритм Прима, построить минимальное остовное дерево. Подробно описать работу алгоритма Прима. Отобразить в итоге графически построенное минимальное остовное дерево.



Дополнительный вопрос к задачам №2 и №3 (1,0 балл): может ли случиться так, что при использовании алгоритмов Прима и Крускала по отношению к одному и тому же графу, могут быть получены разные решения? Если да, то в каких случаях?

Задача №4 (2,5 балла). Дан невзвешенный неориентированный граф, состоящий из 10 вершин. Ниже приводится рабочая программа, реализующая два алгоритма - алгоритм поиска в глубину (DFS) и алгоритм поиска в ширину (BFS).

Используя данные программы, написанные на языке программирования Python, применить алгоритмы DFS и BFS по отношению к заданному выше графу. Найти все вершины и порядок их посещения вершин графа при использовании алгоритмов DFS и BFS, считая, что путь начинается из вершины №1. Т.е. найти все вершины, которые можно посетить, стартуя из вершины №1.

Программа на языке Python для DFS – алгоритма.

```
# Смежность вершин
inc = {
    'Россия': ['Казахстан','Украина','Беларусь','Эстония','Латвия','Литва'],
    'Украина': ['Россия','Беларусь','Молдова'],
    'Армения': ['Грузия','Азербайджан'],
    'Грузия': ['Армения','Азербайджан'],
    'Азербайджан': ['Армения','Грузия'],
    'Молдова': ['Украина'],
    'Кыргызстан': ['Казахстан','Узбекистан','Таджикистан'],
    'Казахстан': ['Россия','Кыргызстан','Узбекистан','Туркменистан'],
    'Узбекистан': ['Туркменистан','Кыргызстан','Казахстан','Таджикистан'],
    'Таджикистан': ['Кыргызстан','Узбекистан'],
    'Туркменистан': ['Казахстан','Азербайджан','Узбекистан'],
    'Эстония': ['Россия','Латвия'],
    'Латвия': ['Литва','Россия', 'Беларусь'],
    'Литва': ['Латвия','Беларусь','Россия'],
    'Беларусь': ['Украина','Россия','Латвия','Литва'],

}

visited = set()  # Посещена ли вершина?


# Поиск в глубину - ПВГ (Depth First Search - DFS)
def dfs(v):
    if v in visited:  # Если вершина уже посещена, выходим
        return
    visited.add(v)  # Посетили вершину v
    for i in inc[v]:  # Все смежные с v вершины
        if not i in visited:
            dfs(i)

start = 'Эстония'
dfs(start)  # start - начальная вершина обхода
print(visited)  #
```

Программа на языке Python для BFS – алгоритма.

```
# Смежность вершин
inc = {
    'Россия': ['Казахстан','Украина','Беларусь','Эстония','Латвия','Литва'],
    'Украина': ['Россия','Беларусь','Молдова'],
    'Армения': ['Грузия','Азербайджан'],
    'Грузия': ['Армения','Азербайджан'],
    'Азербайджан': ['Армения','Грузия'],
    'Молдова': ['Украина'],
    'Кыргызстан': ['Казахстан','Узбекистан','Таджикистан'],
    'Казахстан': ['Россия','Кыргызстан','Узбекистан','Туркменистан'],
```

```
    'Узбекистан': ['Туркменистан','Кыргызстан','Казахстан','Таджикистан'],
    'Таджикистан': ['Кыргызстан','Узбекистан'],
    'Туркменистан': ['Казахстан','Азербайджан','Узбекистан'],
    'Эстония': ['Россия','Латвия'],
    'Латвия': ['Литва','Россия', 'Беларусь'],
    'Литва': ['Латвия','Беларусь','Россия'],
    'Беларусь': ['Украина','Россия','Латвия','Литва'],

}

visited = set()  # Посещена ли вершина?
Q = []  # Очередь
BFS = []

# Поиск в ширину - ПВШ (Breadth First Search - BFS)
def bfs(v):
    if v in visited:  # Если вершина уже посещена, выходим
        return
    visited.add(v)  # Посетили вершину v
    BFS.append(v)  # Запоминаем порядок обхода
    # print("v = %d" % v)
    for i in inc[v]:  # Все смежные с v вершины
        if not i in visited:
            Q.append(i)
    while Q:
        bfs(Q.pop(0))

start = 'Эстония'
bfs(start)  # start - начальная вершина обхода
print(BFS)  #
```