Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рязанский государственный радиотехнический университет имени В. Ф. Уткина»

Кафедра «САПР ВС»

Отчет по лабораторной работе №3

по дисциплине «Дискретная математика»

на тему

**«Раскраска графа»**

Выполнил: студенты гр.1413  
Бригада №4  
Аверкин М.Е.  
Стаховский А.В.

Проверили:   
доц. Орешков В.И.  
доц. Бакулева М.А.

**Цель работы**: изучение способа правильной раскраски графа на основе эвристического алгоритма.

**Граф:**

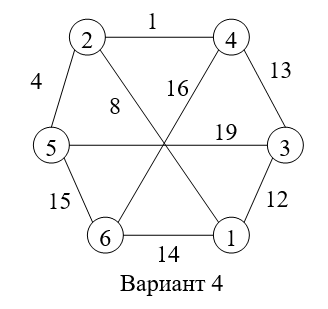
****

Рисунок 1 ‑ вариант задания

**Порядок выполнения работы:**

1 Получить задание у преподавателя в виде исходного неориентированного графа:

2 Составить блок-схему программы, определяющей раскраску графа с помощью эвристического алгоритма, указанного в п.2.2.

3 Создать программу, реализующую эвристический алгоритм раскраски графа. Исходный граф задается в виде матрицы смежности, вводимой построчно с помощью консоли. Программа должна вывести список полученных цветов для всех вершин графа.

**Получившийся раскрашенный граф:**

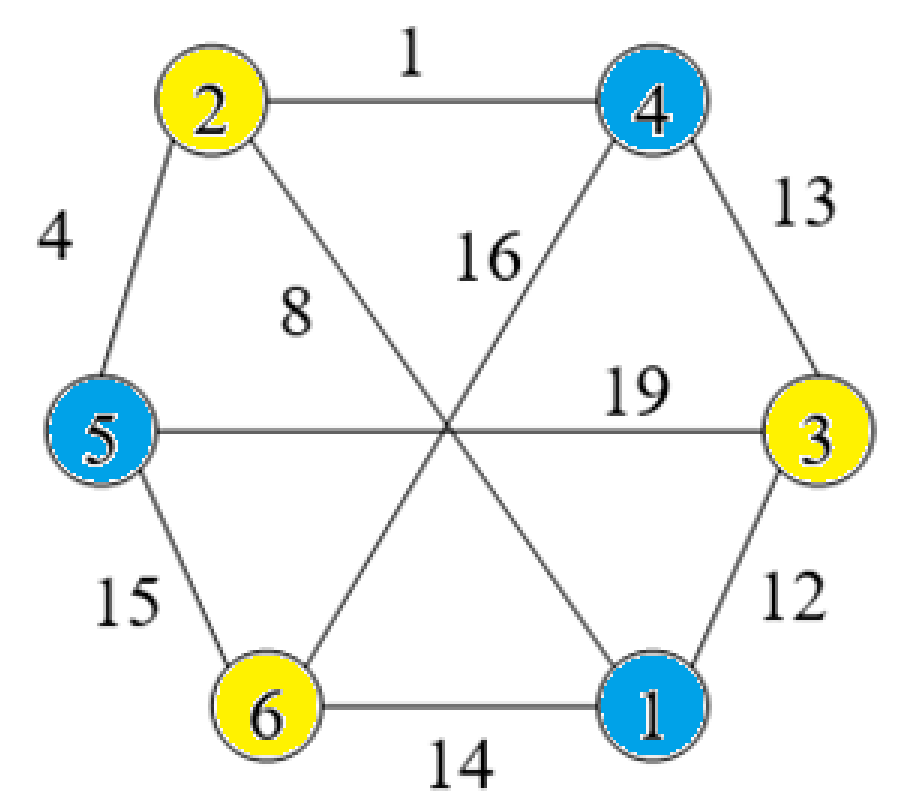


Рисунок 2 - раскрашенный граф

**Типы переменных:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

graph: Тип: словарь

Назначение: хранит граф в виде словаря, где ключи - вершины, а значения - словари соседей с соответствующими весами ребер.

p:

Тип: целое число

Назначение: используется для присваивания цветов вершинам графа. Значение переменной p соответствует текущему цвету, который будет присвоен вершине, если ей еще не был назначен цвет.

n:

Тип: целое число

Назначение: хранит количество вершин в графе, вычисляется как длина словаря graph.

index:

Тип: список целых чисел

Назначение: хранит порядковые номера вершин в отсортированном графе. Индекс в списке соответствует номеру вершины, а значение - порядковому номеру вершины после сортировки.

col:

Тип: список целых чисел

Назначение: хранит цвета вершин графа. Индекс в списке соответствует номеру вершины, а значение - цвету вершины.

sorted\_graph:

Тип: словарь

Назначение: хранит граф, отсортированный по убыванию степени вершин с использованием функции sortGraph.

sorted\_nodes:

Тип: список строк

Назначение: хранит список вершин графа, отсортированных по убыванию степени. Это ключи словаря sorted\_graph в виде списка.

i (в цикле for):

Тип: целое число

Назначение: служит индексом для итерации по списку отсортированных вершин sorted\_nodes.

node (в цикле for):

Тип: строка

Назначение: хранит текущую вершину графа в виде строки вида 'x\_i' при итерации по списку sorted\_nodes.

node\_index (в цикле for):

Тип: целое число

Назначение: хранит индекс текущей вершины в виде целого числа, полученного из строки node.

j (во вложенном цикле for):

Тип: целое число

Назначение: служит индексом для итерации по списку отсортированных вершин sorted\_nodes во вложенном цикле.

other\_node (во вложенном цикле for):

Тип: строка

Назначение: хранит вершину графа, с которой сравнивается текущая вершина node при итерации по списку sorted\_nodes во вложенном цикле. Используется для определения, можно ли присвоить текущий цвет другой вершине, если она не соседствует с вершиной node.

**Блок схема:**

Рисунок 3 ‑ Алгоритм нахождения кратчайшего пути shortest\_path

Рисунок 4‑ Реализация метода динамического программирования и алгоритма топологической сортировки вершин

**Листинг программы:**

def sortGraph(gr):

sorted\_gr = sorted(gr.items(), key=lambda x: len(x[1]), reverse=True)

return dict(sorted\_gr)

graph = {

'x\_1': {'x\_2': 8, 'x\_3': 12, 'x\_6': 14},

'x\_2': {'x\_1': 8, 'x\_4': 1, 'x\_5': 4},

'x\_3': {'x\_1': 12, 'x\_4': 13, 'x\_5': 19},

'x\_4': {'x\_2': 1, 'x\_6': 16, 'x\_3': 13},

'x\_5': {'x\_2': 4, 'x\_3': 19, 'x\_6': 15},

'x\_6': {'x\_5': 15, 'x\_1': 14, 'x\_4': 16}

}

p = 1

n = len(graph)

index = [0 for i in range(n)]

col = [0 for i in range(n)]

sorted\_graph = sortGraph(graph)

sorted\_nodes = list(sorted\_graph.keys())

for i in range(n):

node = sorted\_nodes[i]

node\_index = int(node.split('\_')[1]) - 1

index[node\_index] = i + 1

if col[node\_index] == 0:

col[node\_index] = p

for j in range(n):

other\_node = sorted\_nodes[j]

if other\_node not in graph[node] and col[int(other\_node.split('\_')[1]) - 1] == 0:

col[int(other\_node.split('\_')[1]) - 1] = p

p += 1

print("Номера вершин X\* " + str(index))

print(" Цвета " + str(col))

**ЗАКЛЮЧЕНИЕ**

Таким образом, в этой лабораторной работе мы разработали программу, способную раскрасить граф в минимальное количество цветов.