Практическая работа № 2
студента группы ИТз-221
Дмитриев Дмитрий Анатольевич

Выполнение:_____ Защита:_____

Алгоритм Дейкстры.

Цель работы: научится реализовывать алгоритм Дейкстры, понять, как находить кратчайшие пути от одной из вершин графа до всех остальных.

Содержание работы

Ход работы

1) Реализовывал алгоритм Дейкстры, протестировал его, записал в отчет результаты.

```
/usr/bin/python3.10 /home/floy/www/university/dm/pw2/main.py
Введите конечную вершину (от 1 до 7): 4
Минимальный путь от вершины 1 до вершины 4: [1, 2, 6, 3, 4]
```

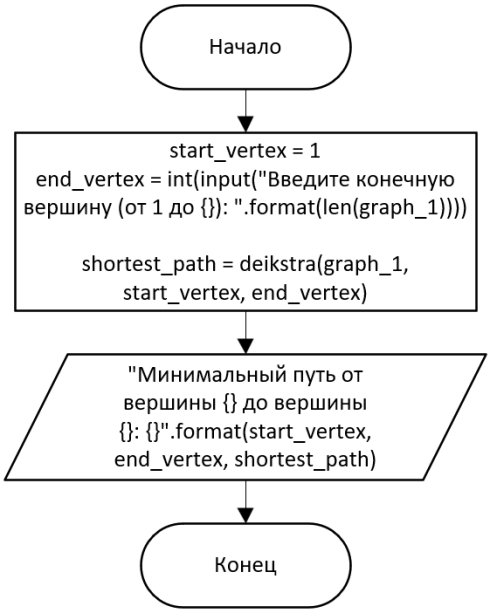
```
/usr/bin/python3.10 /home/floy/www/university/dm/pw2/main.py
Введите конечную вершину (от 1 до 7): 6
Минимальный путь от вершины 1 до вершины 6: [1, 2, 6]
```

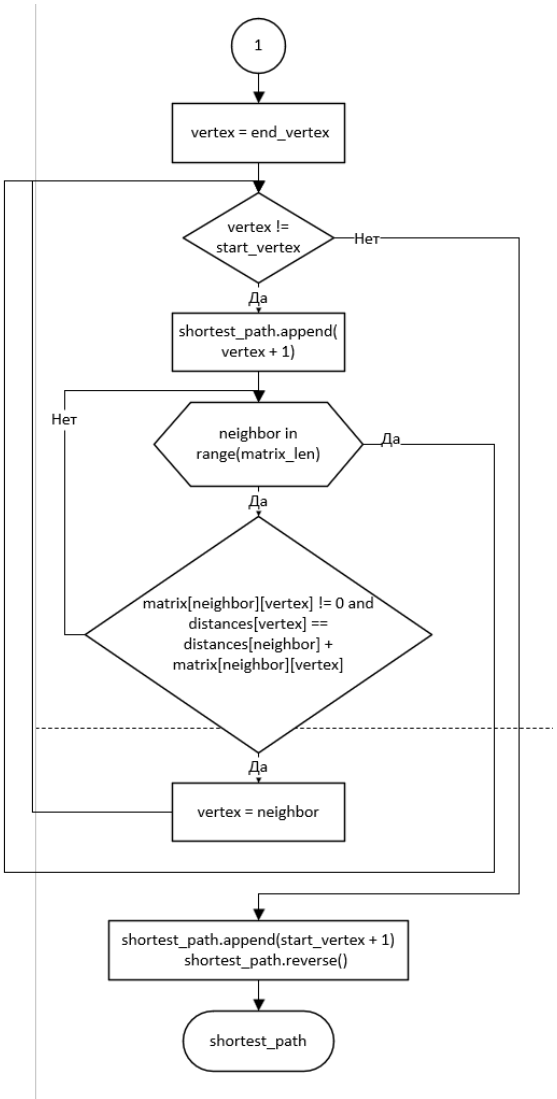
```
/usr/bin/python3.10 /home/floy/www/university/dm/pw2/main.py
Введите конечную вершину (от 1 до 7): 3
Минимальный путь от вершины 1 до вершины 3: [1, 2, 6, 3]
```

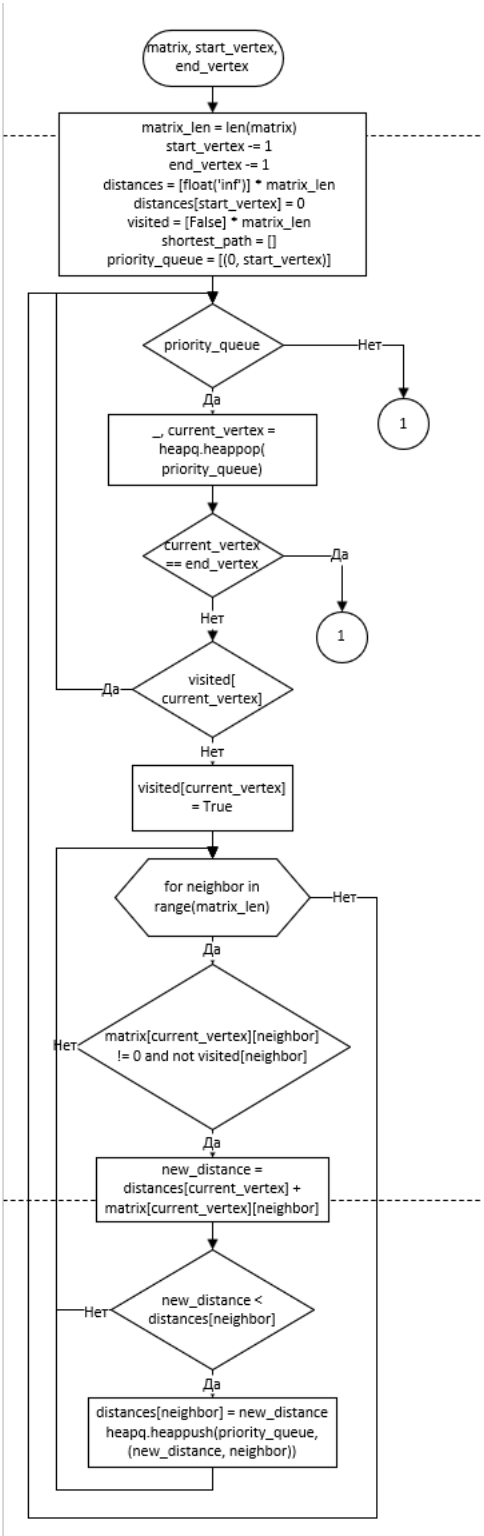
Код программы:

```
import heapq


def deikstra(matrix, start_vertex, end_vertex):
    matrix_len = len(matrix)

    start_vertex -= 1
    end_vertex -= 1

    distances = [float('inf')] * matrix_len
    distances[start_vertex] = 0

    visited = [False] * matrix_len
    shortest_path = []

    priority_queue = [(0, start_vertex)]

    while priority_queue:
        _, current_vertex = heapq.heappop(priority_queue)

        if current_vertex == end_vertex:
            break

        if visited[current_vertex]:
            continue

        visited[current_vertex] = True

        for neighbor in range(matrix_len):
            if matrix[current_vertex][neighbor] != 0 and not visited[neighbor]:
                new_distance = distances[current_vertex] + matrix[current_vertex][neighbor]
                if new_distance < distances[neighbor]:


    vertex = end_vertex
    while vertex != start_vertex:
        shortest_path.append(vertex + 1)
        for neighbor in range(matrix_len):
            if matrix[neighbor][vertex] != 0 and distances[vertex] == distances[neighbor] + matrix[neighbor][vertex]:
                vertex = neighbor
                break

    shortest_path.append(start_vertex + 1)
    shortest_path.reverse()

    return shortest_path


graph_1 = [
    [0, 1, 0, 0, 0, 0, 1],
    [1, 0, 0, 0, 0, 1, 1],
    [0, 0, 0, 1, 1, 1, 0],
    [0, 0, 1, 0, 1, 0, 0],
    [0, 0, 1, 1, 0, 1, 0],
    [0, 1, 1, 0, 1, 0, 1],
    [1, 1, 0, 0, 0, 1, 0]
]

start_vertex = 1
end_vertex = int(input("Введите конечную вершину (от 1 до {}): ".format(len(graph_1))))
shortest_path = deikstra(graph_1, start_vertex, end_vertex)
print("Минимальный путь от вершины {} до вершины {}: {}".format(start_vertex, end_vertex, shortest_path))
```

Блок схема:


Начало
start_vertex = 1
end_vertex = int(input("Введите конечную вершину (от 1 до {}): ".format(len(graph_1))))
shortest_path = deikstra(graph_1, start_vertex, end_vertex)
"Минимальный путь от вершины {} до вершины {}: {}".format(start_vertex, end_vertex, shortest_path)
Конец


1
vertex = end_vertex
vertex != start_vertex
Нет
Да
shortest_path.append(vertex + 1)
Нет
neighbor in range(matrix_len)
Да
Да
matrix[neighbor][vertex] != 0 and distances[vertex] == distances[neighbor] + matrix[neighbor][vertex]
Да
vertex = neighbor
shortest_path.append(start_vertex + 1)
shortest_path.reverse()
shortest_path



Вывод: научился реализовывать алгоритм Дейкстры, понял, как находить кратчайшие пути от одной из вершин графа до всех остальных.