Хмелёв И.С. Разработка методики расчёта тепловой сети по алгоритму Дейкстры на PYTHON // Международный журнал информационных технологий и энергоэффективности. – 2023. –
Т. 8 № 5(31) ч.2 с. 145–150

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 697.343

РАЗРАБОТКА МЕТОДИКИ РАСЧЁТА ТЕПЛОВОЙ СЕТИ ПО АЛГОРИТМУ ДЕЙКСТРЫ НА PYTHON

Хмелёв И.С.
ФГБОУ ВО "Самарский Государственный Технический Университет", Самара, Россия (443100, г. Самара, Молодогвардейская ул., д.244), e-mail: *igori111@mail.ru*

Рассмотрен алгоритм расчёта потерь давления на вводах потребителей, подключенных к тепловой сети. Методика расчёта разработана на основе пошагового алгоритма Дейкстры и написана на языке программирования Python.

Ключевые слова: Тепловая сеть, потери давления, алгоритм Дейкстры, Python

DEVELOPMENT OF A METHOD FOR CALCULATING THE HEAT NETWORK USING DIJKSTRA'S ALGORITHM IN PYTHON

Khmelev I.S.
Samara State Technical University, Samara, Russia (443100, Samara, Molodogvardeyskaya St., 244), e-mail: *igori111@mail.ru*

An algorithm for calculating pressure losses at the inputs of consumers connected to the heating network is considered. The calculation method is based on Dijkstra's step-by-step algorithm and is written in the Python programming language.

Keywords: Heat network, pressure loss, Dijkstra algorithm, Python.

В настоящее время существуют различные методы и способы расчёта тепловых сетей от «классической» реализации алгоритма в таблицах Excel до использования специализированного программного обеспечения, например, системы ZuluThermo.

Рассмотрим методику, разработанную на основе алгоритма Дейкстры и реализованную с помощью языка программирования Python. Данная методика предназначена для расчёта потерь давления в тепловой сети на вводах подключенных к ней потребителей. Для начала вкратце разберём, что из себя представляет сам алгоритм Дейкстры.

Алгоритм Дейкстры – это пошаговый метод, который находит кратчайший путь от одной вершины графа до другой [1]. Граф – это некоторая структура, состоящая из узлов, соединенных гранями каждая из которых имеет свой весовой коэффициент – в данном случае потери давления.

На этапе инициализации задается начальный узел графа, и расстояние до всех узлов приравнивается к бесконечности. Далее на первом шаге выбирается узел ближайший к

начальному, расстояние до него приравнивается к весовому коэффициенту пройденной грани. На всех последующих шагах сравниваются узлы-соседи текущего узла графа. К расстоянию, пройденному от начального узла до текущего прибавляется расстояние от текущего узла до каждого его «соседа». После чего выбирается узел с наименьшим расстоянием и аналогичные действия проделываются с его «соседями». Расчёт завершается после того как все узлы были посещены.

В качестве исходных данных используются две таблицы, реализованные в виде Pandas DataFrame. В первой содержится информация об узлах тепловой сети: наименование узла, порядковый номер узла и тип (источник или потребитель). Во второй таблице хранится информация об участках тепловой сети: наименование начального и конечного узла, диаметр трубопровода и длина участка.

Наименование	Номер	Тип
ТЭЦ-1	0	Источник
ТК6-19	1	NaN
ПРК-1	2	NaN
ПРК-2	3	NaN
СВК-1	4	NaN
...	...	...
ул. Труда,15	249	Потребитель
ул. Труда,39	250	Потребитель
ул. Труда,37а	251	Потребитель
ул. Чехова,2	252	Потребитель
ул. Чехова,8	253	Потребитель

254 rows × 2 columns

	Начало линии	Конец линии	Диаметр	Длина
0	ТК6-19	ПРК-1	0.225	61.875
1	ПРК-1	ПРК-2	0.225	38.750
2	ПРК-2	СВК-1	0.225	56.087
3	СВК-1	СВК-2	0.225	14.375
4	СВК-2	СВК-2а	0.225	37.500
...	...	...	...	...
258	ТРК-24	ул. Труда,15	0.050	27.900
259	ТРК-16	ул. Труда,39	0.050	67.067
260	ТРК-16	ул. Труда,37а	0.050	54.374
261	СВК-3б	ул. Чехова,2	0.050	17.288
262	ТК-1 Чехова,8	ул. Чехова,8	0.050	18.728

263 rows × 5 columns

Рисунок 1 – Таблицы исходных данных

Приступим к реализации алгоритма Дейкстры на языке Python. Для начала создадим класс для нашего графа и методы необходимые для заполнения его данными (Рисунок 1).

```
class Graph:
    def __init__(self):
        self.v = num_of_nodes
        self.edges = [[-1 for i in range(num_of_nodes)] for j in range(num_of_nodes)]
        self.visited = []

    def add_line(self, begin_line, end_line, diametr, lenght):
        self.diametr = diametr
        self.lenght = lenght
        self.hydro = hydro_by_d[diametr]*lenght
        self.num_beginline = NameNodes_num.loc[begin_line, 'Номер']
        self.num_endline = NameNodes_num.loc[end_line, 'Номер']
        self.edges[self.num_beginline][self.num_endline] = self.hydro
```

Рисунок 2 – *Класс Graph*

Внутри метода init находится три переменные:

- v – определяет количество узлов в созданном графе;
- edges – представляет собой матрицу, в которую будут заноситься значения потерь давления;
- visited – это список посещенных узлов.

Далее разберём метод add_line, он используется для заполнения графа исходными данными. В этом методе прописаны следующие переменные:

- diametr – диаметр участка трубопровода;
- lenght – длина участка;
- hydro – абсолютные потери давления, по значению диаметра из словаря подтягиваются удельные потери давления и умножаются на длину участка;
- num_beginline – номер начального узла, определяется из Pandas DataFrame по наименованию узла и столбцу, где NameNodes_num – название DataFrame;
- num_endline – номер конечного узла, определяется аналогично начальному узлу;
- edges – заполнение матрицы граней значениями абсолютных потерь давления, каждый элемент матрицы соответствует комбинации номеров начального и конечного узлов.

Теперь импортируем класс PriorityQueue и создадим новую функцию, которая будет описывать непосредственно алгоритм расчёта (Рисунок 2).

```
from queue import PriorityQueue

def dijkstra(graph, start_vertex):
    D = {v:float('inf') for v in range(graph.v)}
    D[start_vertex] = 0

    pq = PriorityQueue()
    pq.put((0, start_vertex))

    while not pq.empty():
        (dist, current_vertex) = pq.get()
        graph.visited.append(current_vertex)

        for neighbor in range(graph.v):
            if graph.edges[current_vertex][neighbor] != -1:
                distance = graph.edges[current_vertex][neighbor]
                if neighbor not in graph.visited:
                    old_cost = D[neighbor]
                    new_cost = D[current_vertex] + distance
                    if new_cost < old_cost:
                        pq.put((new_cost, neighbor))
                        D[neighbor] = new_cost
    return D
```

Рисунок 3 – Функция Дейкстра

В функции Дейкстра вводится несколько новых переменных:

- graph – переменная присвоенная созданному графу;
- start_vertex – номер начального узла;
- D – словарь с результатами расчёта;
- dist – значение весового коэффициента (потери давления);
- current_vertex – номер текущего узла;
- neighbor – номер узла-соседа;
- distance - значение потерь давления, берётся из матрицы граней;
- old_cost – старое значение потерь давления узла-соседа, берётся из словаря D;
- new_cost – новое значение потерь давления узла-соседа, определяется суммой значения текущего узла из словаря D и переменной distance.

Следующим этапом является создание цикла, который передаст все исходные данные в метод add_line и функцию Дейкстра, а также настроим алгоритм добавления результатов расчёта вТаблицу.

```
# Расчёт
  g = Graph()
  for i in range(0, num_of_line):
    g.add_line(line_info.iloc[i, 0], line_info.iloc[i, 1],
               line_info.iloc[i, 2], line_info.iloc[i, 3])
  D = dijkstra(g, 0)
#Создание таблицы с результатами
for vertex in range(len(D)):
  new_row = { 'Наименование': num_NameNodes.iloc[vertex, 0],
              'Тип': num_NameNodes.iloc[vertex, 2],
              'Потери давления, Па': D[vertex],}
  All_results = All_results.append(new_row, ignore_index=True)

mask_user = (All_results['Тип'] == 'Потребитель')
User_results = All_results[mask_user]
```

Рисунок 4 – Запуск расчёта и вывод результатов

Как результат выполнения описанного алгоритма получаем таблицу в которой сведена информация по всем потребителям подключенным к тепловой сети: их наименования и суммарные потери давления на абонентских вводах каждого из них.

	Наименование	Тип	Потери давления, Па
110	пер. Дерендяева,19	Потребитель	68002.35
122	ул. Володарского,46	Потребитель	47792.95
126	ул. К.Маркса,35	Потребитель	32076.89
133	ул. Р.Люксембург,33	Потребитель	53861.00
136	ул. Р.Люксембург,68	Потребитель	36963.67
...	...	...	...
249	ул. Труда,15	Потребитель	74363.65
250	ул. Труда,39	Потребитель	63764.19
251	ул. Труда,37а	Потребитель	61077.84
252	ул. Чехова,2	Потребитель	49644.61
253	ул. Чехова,8	Потребитель	49255.30

118 rows × 3 columns

Рисунок 5 – Таблица результатов расчёта

Список литературы

1. Skillfactory media [Электронный ресурс] – Электрон. Текстовые дан. – Москва – Режим доступа: https://blog.skillfactory.ru/glossary/algoritm-dejkstry/

2. Помощник Python [Электронный ресурс] – Электрон. Текстовые дан. – Режим доступа: https://pythonpip.ru/examples/ochered-python
3. Академия Яндекса [Электронный ресурс] – Электрон. Текстовые дан. – Москва – Режим доступа: https://academy.yandex.ru/handbook/python/article/obuektnaya-model-python-klassy-polya-i-metody

References

1. Skillfactory media [Electronic resource] – Electron. Text data. – Moscow – Access mode: https://blog.skillfactory.ru/glossary/algoritm-dejkstry/
2. Python Assistant [Electronic resource] – Electron. Text data. – Access mode: https://pythonpip.ru/examples/ochered-python
3. Yandex Academy [Electronic resource] – Electron. Text data. – Moscow – Access mode: https://academy .yandex.ru/handbook/python/article/obuektnaya-model-python-klassy-polya-i-metody