Сетевое программирование

Лекция 7

Поиск путей в графах. Алгоритмы BFS и DFS. Кратчайшие пути (Dijkstra, Bellman–Ford)

Усіпов Н.М., PhD · КазНУ имени аль-Фараби

Цель и план лекции

Цель

Изучение алгоритмов обхода графа и поиска кратчайших путей, анализ их вычислительной сложности и границ применимости, а также рассмотрение их связи с протоколами маршрутизации.

1. Введение
2. Алгоритмы обхода графа
3. Задача о кратчайшем пути
4. Алгоритм Дейкстры
5. Алгоритм Беллмана–Форда
6. Сравнение алгоритмов и другие методы
7. Практические особенности маршрутизации
8. Варианты задачи о кратчайших путях

1 Введение

- Основная функция сети передачи данных – доставка информации от источника к получателю.
- Алгоритмы данной лекции являются основой протоколов маршрутизации. Понимание их работы и условий применимости необходимо для того, чтобы правильно настроить сеть и корректно выбрать алгоритм при решении прикладной задачи.

2 Алгоритмы обхода графа

Поиск в ширину (BFS)

Обходит вершины по уровням удалённости от источника, используя очередь. Сначала посещаются все соседи источника, затем соседи соседей и т. д. Поэтому в невзвешенном графе BFS находит пути с наименьшим числом рёбер.

Поиск в глубину (DFS)

Продвигается как можно дальше вдоль одной ветви и возвращается назад, только если продолжать движение невозможно.

Различие алгоритмов определяется применяемой структурой данных: очередь обеспечивает обход по уровням, а стек – продвижение в глубину.

2 Обход графа в ширину и в глубину

Листинг 7.1.

```
import networkx as nx

G = nx.Graph()
G.add_weighted_edges_from([
    ("A", "B", 4), ("A", "C", 2), ("B", "C", 1), ("B", "D", 5),
    ("C", "D", 8), ("C", "E", 10), ("D", "E", 2), ("D", "F", 6),
    ("E", "F", 3)])

bfs = [v for _, v in nx.bfs_edges(G, "D")]
print("BFS из D:", ["D"] + bfs)
print("DFS из D:", list(nx.dfs_preorder_nodes(G, "D")))
print("Переходы A->F:", nx.shortest_path(G, "A", "F"))
print("Уровни:", nx.single_source_shortest_path_length(G, "A"))
```

```
BFS из D: ['D', 'B', 'C', 'E', 'F', 'A']
DFS из D: ['D', 'B', 'A', 'C', 'E', 'F']
Переходы A->F: ['A', 'B', 'D', 'F']
Уровни: {'A': 0, 'B': 1, 'C': 1, 'D': 2, 'E': 2, 'F': 3}
```

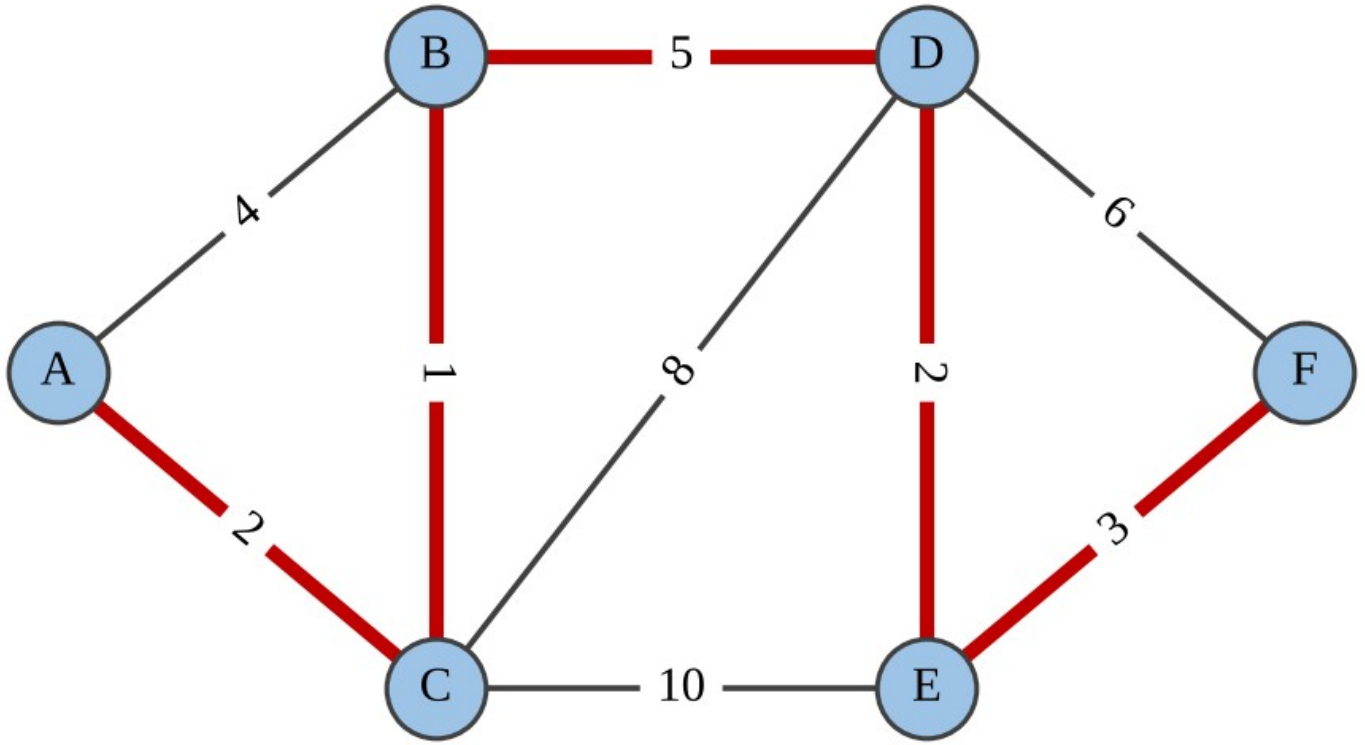
3 Задача о кратчайшем пути

- Длиной пути во взвешенном графе называется сумма весов его рёбер. Кратчайшие пути обладают свойством оптимальности подструктуры: любая часть кратчайшего пути также является кратчайшим путём между своими концами.

$$d(v) = min(\ d(v), d(u) + w(u, v)\ ).$$

- Алгоритмы различаются порядком применения этой операции.

3 Задача о кратчайшем пути (продолжение)



Взвешенная сеть и кратчайший маршрут из A в F

4 Алгоритм Дейкстры

- Алгоритм Дейкстры, опубликованный в 1959 году, применяется для графов с неотрицательными весами.
- После обработки A получаем C = 2 и B = 4. Далее обрабатывается C, и оценка B уменьшается до 3, так как путь через C короче.
- Жадный выбор корректен только при неотрицательных весах: иначе зафиксированную оценку смог бы улучшить более поздний путь.

5 Алгоритм Беллмана–Форда

- Алгоритм Беллмана–Форда выполняет |V|−1 проход, на каждом из которых применяет релаксацию ко всем рёбрам.
- В распределённой форме этот метод лежит в основе дистанционно-векторных протоколов, таких как RIP: каждый маршрутизатор обменивается с соседями таблицами расстояний.

6 Сравнение алгоритмов и другие методы

- BFS – невзвешенные графы, O(|V| + |E|).
- Дейкстра – неотрицательные веса, один источник.
- Беллман–Форд – возможны отрицательные веса, обнаружение отрицательных циклов.
- Флойд–Уоршелл – расстояния между всеми парами вершин, $O(|V|^3)$, подходит для небольших плотных сетей.
- A* – поиск от источника к заданной цели с эвристической оценкой остатка пути.

6 Сравнение алгоритмов и другие методы (продолжение)

- В NetworkX функция `shortest_path` при заданном параметре `weight` по умолчанию использует алгоритм Дейкстры, а для выбора алгоритма Беллмана–Форда применяется параметр `method="bellman-ford"`.

6 Кратчайшие пути во взвешенной сети

Листинг 7.2.

```
print("Dijkstra:", nx.dijkstra_path(G, "A", "F"),
      nx.dijkstra_path_length(G, "A", "F"))
print("Bellman-Ford:", nx.bellman_ford_path(G, "A", "F"),
      nx.bellman_ford_path_length(G, "A", "F"))
print("Все расстояния от A:", nx.single_source_dijkstra_path_length(G, "A"))

D = nx.DiGraph()
D.add_weighted_edges_from([("s", "a", 4), ("s", "b", 5),
                           ("b", "a", -3), ("a", "t", 2)])
print("С отрицательной дугой:", nx.bellman_ford_path(D, "s", "t"))

D.add_edge("a", "b", weight=1)
print("Отрицательный цикл:", nx.negative_edge_cycle(D))
```

```
Dijkstra: ['A', 'C', 'B', 'D', 'E', 'F'] 13
Bellman-Ford: ['A', 'C', 'B', 'D', 'E', 'F'] 13
Все расстояния от A: {'A': 0, 'C': 2, 'B': 3, 'D': 8, 'E': 10, 'F': 13}
С отрицательной дугой: ['s', 'b', 'a', 't']
Отрицательный цикл: True
```

7 Практические особенности маршрутизации

- В реальных протоколах вес канала задаётся настройками. Например, в OSPF стоимость интерфейса часто вычисляется как отношение опорной полосы к полосе интерфейса, поэтому маршрут с большим числом быстрых каналов может оказаться выгоднее короткого маршрута через медленный канал.
- Кроме основного маршрута, нередко требуется определить резервные варианты. Функция `shortest_simple_paths` формирует простые пути в порядке увеличения длины на основе алгоритма Йена, что удобно для анализа запасных маршрутов.

7 Поиск резервных маршрутов

Листинг 7.3.

```
from itertools import islice

def length(p):
    return nx.path_weight(G, p, "weight")

for p in islice(nx.shortest_simple_paths(G, "A", "F", "weight"), 3):
    print(p, "длина", length(p))

H = G.copy()
H.remove_edge("B", "D")
print("После отказа B-D:", nx.dijkstra_path(H, "A", "F"),
      nx.dijkstra_path_length(H, "A", "F"))
```

```
['A', 'C', 'B', 'D', 'E', 'F'] длина 13
['A', 'B', 'D', 'E', 'F'] длина 14
['A', 'C', 'B', 'D', 'F'] длина 14
После отказа B-D: ['A', 'C', 'E', 'F'] 15
```

8 Варианты задачи о кратчайших путях

- На практике задача о кратчайшем пути имеет несколько постановок. Для задачи с одним источником определяются пути до всех вершин, и результат образует дерево кратчайших путей.
- Двунаправленный поиск одновременно продвигается от источника и от получателя и завершается при встрече двух фронтов.
- Алгоритм Джонсона применяется для поиска между всеми парами вершин при наличии отрицательных весов.

9 Маршрутизация в сетях

Протоколы маршрутизации разделяются по способу распространения информации.

Протоколы состояния каналов

OSPF и IS-IS распространяют сведения о каналах по всей области. Каждый маршрутизатор располагает общей картой сети и самостоятельно вычисляет маршруты.

Дистанционно-векторные протоколы

RIP и другие подобные протоколы основаны на обмене таблицами расстояний между соседями.

Протоколы вектора путей

BGP передаёт полный путь на уровне автономных систем и применяет его для предотвращения петель.

Существенно, что протокол выполняется распределённо: каждый узел имеет собственную, возможно временно устаревшую, картину сети.

10 Разбор примера

1. Применим алгоритм Беллмана–Форда к орграфу из листинга 7.2 с дугами s → a (4), s → b (5), b → a (−3), a → t (2), считая, что исходно цикла ещё нет.
2. Начальные оценки: d(s) = 0, остальные равны бесконечности.
3. Первый проход: релаксация s → a даёт a = 4, s → b даёт b = 5, b → a даёт a = 5 − 3 = 2, a → t даёт t = 4.
4. Второй проход изменений не вносит, поэтому работа завершается.
5. Кратчайший путь s–b–a–t имеет длину 4.
6. Обратите внимание, что алгоритм Дейкстры, зафиксировав вершину a со значением 4, мог бы получить неверный результат.
7. Рассчитаем метрику OSPF при опорной полосе 100 Гбит/с: канал 10 Гбит/с имеет стоимость 10, а канал 1 Гбит/с – 100.

10 Разбор примера (продолжение)

8 Маршрут из двух каналов 10 Гбит/с стоит 20 и будет предпочтён одному каналу 1 Гбит/с стоимостью 100, несмотря на большее число переходов.

11 Вопросы для обсуждения на занятии

Метрика

Какие недостатки имеет метрика, зависящая только от полосы канала?

Сходимость

К каким последствиям приводит наличие у маршрутизаторов различных версий карты сети?

Выбор алгоритма

Какой алгоритм подходит для навигатора, использующего дорожный граф с координатами перекрёстков?

Контрольные вопросы

1. В чём различие между обходом в ширину и в глубину?
2. Почему BFS находит кратчайшие пути только в невзвешенных графах?
3. В чём заключается операция релаксации?
4. Опишите алгоритм Дейкстры и укажите его вычислительную сложность.
5. Почему алгоритм Дейкстры нельзя применять при отрицательных весах?
6. В чём преимущества и недостатки алгоритма Беллмана–Форда?
7. Какие протоколы маршрутизации основаны на рассмотренных алгоритмах?
8. Как выбрать алгоритм в NetworkX и получить все кратчайшие пути?

Список литературы

1. Dijkstra E.W. A note on two problems in connexion with graphs // Numerische Mathematik. – 1959. – Vol. 1. – P. 269–271.

2. Bellman R. On a routing problem // Quarterly of Applied Mathematics. – 1958. – Vol. 16, No. 1. – P. 87–90.
3. Yen J.Y. Finding the k shortest loopless paths in a network // Management Science. – 1971. – Vol. 17, No. 11. – P. 712–716.

4. Moy J. OSPF Version 2: RFC 2328. – IETF, 1998.
5. Malkin G. RIP Version 2: RFC 2453. – IETF, 1998.
6. Cormen T., Leiserson C., Rivest R., Stein C. Introduction to Algorithms. – 4th ed. – Cambridge, MA: MIT Press, 2022.

7. Sedgewick R., Wayne K. Algorithms. – 4th ed. – Boston: Addison-Wesley, 2011.
8. NetworkX Documentation [Электронный ресурс]. – URL: https://networkx.org/documentation/stable/