TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

Vận dụng thiết kế thuật toán: Graph Algorithms

Môn học: Phân tích và thiết kế thuật toán

Nhóm 9

Sinh viên thực hiện:

Họ và tên	MSSV
Bùi Ngọc Thiên Thanh	23521436
Nguyễn Thái Sơn	23521356

Thành phố Hồ Chí Minh, 2024

Mục lục

1 Bài 1

1.1 Ý tưởng và phương pháp thiết kế thuật toán

Thuật toán Greedy (Tìm kiếm tham lam): Theo thuật toán Greedy, tuyến đường từ London đến Novgorod là:

$$\text{London} \rightarrow \text{Amsterdam} \rightarrow \text{Novgorod}$$

với chi phí tổng là 14 (8 từ London đến Amsterdam và 6 từ Amsterdam đến Novgorod).
Đánh giá:

- Thuật toán Greedy dựa trên giá trị heuristic (khoảng cách Euclid đến Novgorod), nên chỉ tập trung chọn thành phố gần nhất với đích tại mỗi bước, mà không xem xét đến tổng chi phí thực tế.
- Mặc dù giá trị heuristic của các điểm trong tuyến đường này thấp, nó không đảm bảo rằng chi phí thực tế cũng nhỏ nhất. Do đó, đường đi không tối ưu về chi phí di chuyển.
- Trong ví dụ này, thuật toán đã bỏ qua các đường đi khác có thể có chi phí thực tế thấp hơn.

Kết luận: *Thuật toán Greedy không đảm bảo tìm được đường đi tối ưu* do không xét đến tổng chi phí thực tế, chỉ dựa vào giá trị heuristic.

Thuật toán UCS (Uniform Cost Search): Theo thuật toán UCS, tuyến đường từ London đến Novgorod là:

$$\text{London} \rightarrow \text{Amsterdam} \rightarrow \text{Copenhagen} \rightarrow \text{Falsterbo} \rightarrow \text{Danzig}$$

$$\rightarrow \text{Visby} \rightarrow \text{Tallinn} \rightarrow \text{Novgorod}$$

với tổng chi phí thực tế là:

$$801 + 324 + 498 + 606 + 590 + 474 = 3293$$

Đánh giá:

- Thuật toán UCS mở rộng node theo chi phí thực tế thấp nhất tại mỗi bước, thay vì chỉ dựa vào giá trị heuristic như Greedy.
- UCS đảm bảo tìm được đường đi có tổng chi phí thực tế nhỏ nhất vì luôn chọn mở rộng đường đi tối ưu tại mỗi bước.
- Trong ví dụ này, mặc dù tuyến đường có thể dài hơn về số node trung gian, tổng chi phí thực tế nhỏ hơn so với các tuyến đường khác.

Kết luận: *Thuật toán UCS đảm bảo tìm được đường đi tối ưu* vì nó xét tổng chi phí thực tế, không bị ảnh hưởng bởi giá trị heuristic.

1.2 Độ phức tạp

Thuật toán Greedy:

- Độ phức tạp: $O(E \log V)$, với E là số cạnh, V là số đỉnh.
- Không đảm bảo tối ưu.

Thuật toán UCS:

- Độ phức tạp: $O(E \log V)$, đảm bảo tìm đường đi tối ưu.

2 Bài 2

2.1 Ý tưởng và phương pháp thiết kế thuật toán

Ý tưởng chính

- Sử dụng thuật toán Bellman-Ford để kiểm tra sự tồn tại của chu trình âm trong đồ thị. - Thuật toán Bellman-Ford không chỉ kiểm tra chu trình âm mà còn có thể tìm ra chu trình đó bằng cách lần ngược qua mảng cha (*parent*).

Các bước thực hiện

1. **Khởi tạo:**

- Gán giá trị khoảng cách ban đầu cho các đỉnh: $\text{dist}[u] = \infty$ (vô cùng), ngoại trừ $\text{dist}[\text{start}] = 0$.
- Đặt mảng cha ban đầu $\text{parent}[u] = -1$ cho mọi đỉnh.

2. **Cập nhật khoảng cách:**

- Lặp $N - 1$ lần (với N là số đỉnh):
  - Duyệt qua tất cả các cạnh (u, v, c):
  - Nếu $\text{dist}[u] + c < \text{dist}[v]$, cập nhật:

$$\text{dist}[v] = \text{dist}[u] + c$$

$$\text{parent}[v] = u$$

3. **Kiểm tra chu trình âm:**

- Duyệt qua tất cả các cạnh (u, v, c) thêm một lần:
- Nếu $\text{dist}[u] + c < \text{dist}[v]$, tồn tại chu trình âm.
- Lần ngược qua mảng cha (*parent*) để truy xuất chu trình.

4. **Xuất kết quả:**

- Nếu có chu trình âm, in **YES** và chu trình âm.
- Nếu không, in **NO**.

2.2 Mã giả

```
Input: Số đỉnh N, số cạnh M, danh sách cạnh edges[a, b, c]
Output: YES và chu trình âm, hoặc NO nếu không có

1. Khởi tạo:
   - dist[u] = INF với mọi u, dist[start] = 0
   - parent[u] = -1 với mọi u

2. Thực hiện cập nhật:
   - Lặp N - 1 lần:
     - Với mỗi cạnh (u, v, c):
```

```
        - Nếu dist[u] + c < dist[v]:
            dist[v] = dist[u] + c
            parent[v] = u

3. Kiểm tra chu trình âm:
   - Với mỗi cạnh (u, v, c):
     - Nếu dist[u] + c < dist[v]:
         Có chu trình âm
         - Tìm một đỉnh thuộc chu trình âm
         - Lặp qua mảng parent để tìm chu trình
         - In YES và chu trình
         - Kết thúc

4. Nếu không tìm thấy cạnh nào làm giảm chi phí: In NO
```

2.3 Độ phức tạp

- **Thời gian:** $O(N \times M)$, với N là số đỉnh và M là số cạnh.
- **Không gian:** $O(N)$, dùng để lưu mảng khoảng cách và cha.