Đại học quốc gia TP.HCM
Trường đại học công nghệ thông tin

Tiêu đề Bài tập CS112 - Nhóm 15

Nhóm 8:
Tên SV1: Nguyễn Phạm Phương Nam – MSSV: 23520978,
Tên SV2: Phạm Huỳnh Long vũ – MSSV: 23520978,

Câu 1:

1. Kết quả tính toán

1.1. Thuật toán Greedy

- **Tuyến đường**: London → Hamburg → Falsterbo → Danzig → Tallinn → Novgorod.
- **Chi phí thực tế**: 2668.

1.2. Thuật toán UCS

- **Tuyến đường**: London → Hamburg → Lubeck → Visby → Riga → Novgorod.
- **Chi phí thực tế**: 2232.

2. Nhận xét về tối ưu hóa

- **Greedy**: Không tìm được đường đi tối ưu, vì thuật toán chỉ quan tâm đến khoảng cách Euclid tại mỗi bước. Điều này dẫn đến chi phí thực tế cao hơn.
- **UCS**: Tìm được đường đi tối ưu với tổng chi phí thực tế nhỏ nhất.

Câu 2:

1. Ý tưởng

- Để xác định xem đồ thị có chứa chu trình âm (negative cycle) hay không, ta sử dụng **thuật toán Bellman-Ford**. Thuật toán này hoạt

động trên đồ thị có trọng số, kể cả trọng số âm.

- Bellman-Ford tính toán đường đi ngắn nhất từ một đỉnh nguồn đến tất cả các đỉnh khác trong đồ thị. Trong quá trình thực hiện, nếu sau $N-1$ lần cập nhật mà vẫn có thể cải thiện được đường đi ngắn nhất, thì tồn tại chu trình âm.

2. Phương pháp thiết kế

1. **Khởi tạo**:
    - Gán giá trị ban đầu cho tất cả các đỉnh là vô cùng ($+\infty$), riêng đỉnh nguồn có giá trị bằng 0.
2. **Thực hiện cập nhật**:
    - Duyệt qua tất cả các cạnh M và cố gắng cập nhật đường đi ngắn nhất $N-1$ lần.
3. **Kiểm tra chu trình âm**:
    - Duyệt thêm một lần nữa qua các cạnh. Nếu có bất kỳ cạnh nào có thể cập nhật được giá trị đường đi, thì tồn tại chu trình âm.
4. **Truy vết chu trình âm**:
    - Nếu phát hiện chu trình âm, truy vết các đỉnh trong chu trình bằng cách lưu lại đỉnh cuối cùng trong chu trình và di chuyển ngược lại N lần.

Mã giả

```
Input: N (số đỉnh), M (số cạnh), edges (danh sách các cạnh)
Output: YES và chu trình âm nếu tồn tại, NO nếu không

1. Khởi tạo:
    - distance[u] = + với mọi đỉnh u
    - predecessor[u] = null (lưu đỉnh trước đó để truy vết chu trình)
    - distance[start] = 0
```

```
2. Thực hiện cập nhật N-1 lần:
    For i = 1 to N-1:
        For mỗi cạnh (u, v, w) trong edges:
            If distance[u] + w < distance[v]:
                distance[v] = distance[u] + w
                predecessor[v] = u

3. Kiểm tra chu trình âm:
    For mỗi cạnh (u, v, w) trong edges:
        If distance[u] + w < distance[v]:
            - Tồn tại chu trình âm
            - Tìm đỉnh thuộc chu trình:
                - curr = v
                - Duyệt ngược N lần qua predecessor để đảm bảo rơi vào chu trình
            - Lưu chu trình âm:
                - Từ curr, tiếp tục duyệt ngược qua predecessor để thu thập các đỉ
            - In chu trình và kết thúc

4. Nếu không phát hiện chu trình âm, in "NO".
```

Độ phức tạp

- **Thời gian**:
  - Thuật toán Bellman-Ford có độ phức tạp $O(N \times M)$ do thực hiện $N - 1$ vòng lặp cập nhật và một lần kiểm tra chu trình âm.
- **Không gian**:
  - Lưu trữ mảng `distance` và `predecessor`, độ phức tạp không gian là $O(N)$.