HỌC VIỆN KỸ THUẬT QUÂN SỰ



BÁO CÁO CUỐI KỲ

***Đề số 13:*** Với bài toán tìm các thành phần liên kết mạnh (**strongly connected components**) trên đồ thị và thuật toán **Kosaraju** hãy trình bày các nội dung sau:

1. Nêu bài toán;
2. Mô tả chi tiết thuật toán;
3. Đánh giá độ phức tạp thuật toán;
4. Tự xác định 2 bộ dữ liệu (với số phần tử N>=5), với mỗi bộ dữ liệu hãy thực hiện từng bước thuật toán đã mô tả ở mục 2 và ghi ra kết quả mỗi bước;
5. Viết chương trình trên C/C++
6. Viết báo cáo trình bày các nội dung từ 1-4 và kết quả thực hiện chường trình (5).

|  |  |  |
| --- | --- | --- |
|  | **Môn:**  **Giáo viên hướng dẫn:**  **Sinh viên:**  **Lớp:** | Phân tích và thiết kế giải thuật.  Nguyễn Mậu Uyên.  Nguyễn Thị Nhung Tâm.  KTPM 14 |
|  |  |  |

Năm học: 2017-2018

MỤC LỤC:

[1. **Nêu bài toán:** 2](#_Toc501149404)

[a. Cơ sở lý thuyết: 2](#_Toc501149405)

[b. Nêu bài toán: 3](#_Toc501149420)

[**2.** **Mô tả chi tiết thuật toán:** 3](#_Toc501149424)

[ *Với bài toán này sử dụng thuật toán Kosaraju sẽ gồm 2 bước chính như sau:* 3](#_Toc501149425)

[ Mô tả thuật toán: 4](#_Toc501149437)

[**3. Độ phức tạp của thuật toán: 5**](#_Toc501149446)

[4. **Áp dụng với bộ dữ liệu:** 5](#_Toc501149447)

[a. Bộ dữ liệu số 1: 5](#_Toc501149448)

[b. Bộ dữ liệu số 2: 7](#_Toc501149486)

[**5**. **Mã nguồn chương trình**: 9](#_Toc501149529)

[**6. Kết quả chương trình: (Áp dụng với bộ dữ liệu số 2)** 12](#_Toc501149547)

1. **Nêu bài toán:**
2. Cơ sở lý thuyết:

* Một số lý thuyết về tìm kiếm theo chiều sâu (DFS - Depth-first search):
* Giải thuật duyệt hoặc tìm kiếm trên một cây hoặc một đồ thị và sử dụng stack (ngăn xếp) để ghi nhớ đỉnh liền kề để bắt đầu việc tìm kiếm khi không gặp được đỉnh liền kề trong bất kỳ vòng lặp nào.
* Giải thuật tiếp tục cho tới khi gặp được đỉnh cần tìm hoặc tới một nút không có con. Khi đó giải thuật quay lui về đỉnh vừa mới tìm kiếm ở bước trước.
* Giải thuật này tuân theo 3 qui tắc sau:
* **Qui tắc 1**: Duyệt tiếp tới đỉnh liền kề mà chưa được duyệt. Đánh dấu đỉnh mà đã được duyệt. Hiển thị đỉnh đó và đẩy vào trong một ngăn xếp (stack).
* **Qui tắc 2**: Nếu không tìm thấy đỉnh liền kề, thì lấy một đỉnh từ trong ngăn xếp (thao tác pop up). (Giải thuật sẽ lấy tất cả các đỉnh từ trong ngăn xếp mà không có các đỉnh liền kề nào)
* **Qui tắc 3**: Lặp lại các qui tắc 1 và qui tắc 2 cho tới khi ngăn xếp là trống.
* Một số lý thuyết về liên thông trong đồ thị:
* Một đồ thị được gọi là liên thông (connected) nếu có đường đi giữa mọi cặp đỉnh phân biệt của đồ thị.
* Thành phần liên thông: một đồ thị không liên thông sẽ bao gồm nhiều đồ thị con liên thông, các đồ thị con này được gọi là các [thành phần liên thông](https://vi.wikipedia.org/wiki/Th%C3%A0nh_ph%E1%BA%A7n_li%C3%AAn_th%C3%B4ng" \o "Thành phần liên thông) ([connected component](https://vi.wikipedia.org/w/index.php?title=Connected_component&action=edit&redlink=1" \o "Connected component (trang chưa được viết))). Đồ thị liên thông khi và chỉ khi có một thành phần liên thông.
* Đồ thị liên thông có hướng:
* *Liên thông mạnh* (strongly connected): Đồ thị có hướng gọi là liên thông mạnh nếu có đường đi từ a tới b và từ b tới a với mọi cặp đỉnh a và b của đồ thị.
* *Liên thông yếu* (weakly connected): Đồ thị có hướng gọi là liên thông yếu nếu có đường đi giữa 2 đỉnh bất kỳ của đồ thị vô hướng tương ứng với đồ thị đã cho. Tức là hủy bỏ các hướng của các cạnh trong đồ thị
* *Liên thông một phần* (unilaterally connected): Đồ thị có hướng gọi là liên thông một phần nếu với mọi cặp đỉnh a, b bất kỳ, có ít nhất một đỉnh đến được đỉnh còn lại.

1. Nêu bài toán:

* Bài toán tìm các thành phần liên kết mạnh (strongly connected components) trên đồ thị:
* Input: Do đồ thị sử dụng đồ thị không quá lớn nên ta sử dụng nhập danh sách kề từ bàn phím.
* Output: In ra từng thành phần liên kết mạnh của đồ thị đã nhập.

⇒ Sử dụng thuật toán Kosaraju với bài toán này. Trong thuật toán Kosaraju ta duyệt đồ thị bằng thuật toán DFS.

1. **Mô tả chi tiết thuật toán:**

* *Với bài toán này sử dụng thuật toán Kosaraju sẽ gồm 2 bước chính như sau:*

1. Duyệt đồ thị, sử dụng duyệt theo chiều sâu (DFS):

(Một số cơ sở lý thuyết về DFS được nêu ở mục a.)

* Bài toán sử dụng 1 ngăn xếp S, mỗi đỉnh v sau khi duyệt sẽ được đẩy vào ngăn xếp S.
* Một đỉnh được gọi là duyệt xong khi mà mọi hàng xóm của đỉnh đó được thăm và được đánh dấu là đã duyệt.
* Hoặc có thể hiểu là một đỉnh được duyệt xong khi mọi đỉnh nằm trong cây con DFS của nó đã được duyệt.

1. Ở bước 2 này ta in ra các thành phần liên thông dựa vào thao tác trên đồ thị ngược với đồ thị ban đầu.

* Đồ thị ngược hiểu đơn giản là đồ thì mà ta đổi hướng tại tất cả các đỉnh của đồ thị ban đầu.
* Ta bắt đầu từ đỉnh v, đỉnh đầu tiên của ngăn xếp S có được sau bước 1.
* Từ đỉnh v ta tìm các đỉnh u sao cho tồn tại đường đi có hướng từ v đến u trong đồ thị.
* Tập các đỉnh u tìm được sẽ là một thành phần liên thông mạnh trong đồ thị ban đầu.

⇒ Sau khi tìm được tập đỉnh thuộc thành phần liên thông mạnh rồi, ta sẽ xóa tập đỉnh đó khỏi ngăn xếp S.

* Mô tả thuật toán:

B1: Nhập số cạnh, số đỉnh, các đỉnh và hàng xóm của nó ta được đồ thị G.

B2: Với đỉnh i, ta tạo 1 danh sách chứa các đỉnh lân cận với i. Nếu danh sách này trống ta đẩy i vào ngăn xếp S, ngược lại thực hiện DFS với đỉnh lân cận i là v. Sau đó trỏ đến đỉnh lân cận tiếp theo của i và tiếp tục thực hiện DFS. => Ta được ngăn xếp S.

B3: Đảo ngược đồ thị G => đồ thị H.

B4: Với H, nếu S không rỗng, ta lấy vị trí top của S là v

=> In ra đỉnh v và xóa v khỏi S.

* Nếu v có danh sách đỉnh lân cận khác NULL => kiểm tra đỉnh lận cận u.
* Nếu u chưa được thăm và chưa xóa trong S thì thực hiện tương tự v với u ở bước này (đệ quy) và tiếp tục trỏ đến đỉnh lân cận tiếp theo.
* Mỗi lần ta xóa v thì top thay đổi. Thực hiện cho đến khi S là NULL ta được các thành phần liên kết mạnh trong đồ thị đã cho.

1. Độ phức tạp của thuật toán:

* Thuật toán này về cơ bản sẽ duyệt qua đồ thị 2 lần, do đó có thời gian tiệm cận O(V+E).

1. Áp dụng với bộ dữ liệu:
2. Bộ dữ liệu số 1:

|  |  |  |
| --- | --- | --- |
| Bước | Thực hiện | Kết quả |
| 1 | Số đỉnh: 5  Số cạnh: 6  Danh sách lân cận:   * Đỉnh A: B, D, E * Đỉnh B: C * Đỉnh C: D * Đỉnh D: B * Đỉnh E: |  |
| 2 | * Đỉnh A: * Lân cận: B, D, E * DFS đỉnh B, B→C→D tại D lân cận là B do B đã đi qua → push(S,D) * Quay lại C, không còn lân cận → push(S,C) * Quay lại B, không còn lân cận → push(S,B) * Xét lân cận E của A → push(S,E) và push(S,A) |  |
| 3 | Đảo ngược đồ thị G ta được đồ thị H. |  |
| 4 | * Trên đồ thị H: * Lấy v = pop (S) = A, A không có lân cận   → in ra A, xóa A khỏi S.   * Lấy v = pop (S) = E, E không có lân cận   → in ra E, xóa E khỏi S.   * Lấy v = pop (S) = B, B có lân cận là D, D có lân cận là C   → in ra B, D, C, xóa B, D, C khỏi S.  ⇒ S lúc này là NULL | Thành phần liên kết mạnh:   1. A 2. E 3. B, D, C |

1. Bộ dữ liệu số 2:

|  |  |  |
| --- | --- | --- |
| Bước | Thực hiện | Kết quả |
| 1 | Số đỉnh: 9  Số cạnh: 16  Danh sách lân cận:   * Đỉnh 0: 1, 2, 4 * Đỉnh 1: 3 * Đỉnh 2: 5, 4 * Đỉnh 3: 8, 7, 2, 0 * Đỉnh 4: 5 * Đỉnh 5: 8, 6 * Đỉnh 6: 4 * Đỉnh 7: * Đỉnh 8: 7, 2 |  |
| 2 | * Đỉnh 0: * Lân cận: 1, 2, 4 * DFS đỉnh 0, 0→4→5→8→7 tại 7 không còn lân cận → push(S,7) * Quay lại 8, 8→2, tại 2 lân cận với 4 do 4 đã đi qua → push(S,2), 8 lúc này không còn lân cận → push(S,8) * Quay lại 5, 5→ 6, 6 lân cận với 4, do 4 đã đi qua → push(S,6), 5 lúc này cũng không còn lân cận → push(S,5) * Quay lại 4, không còn lân cận → push(S,4) * Quay lại 0, 0→1→3, 3 lân cận với 7, 8, 2 - đều đã xét → push(S,3) * Quay lại 1, không còn lân cận → push(S,1) và push(S,0) |  |
| 3 | Đảo ngược đồ thị G ta được đồ thị H. |  |
| 4 | * Trên đồ thị H: * Lấy v = pop (S) = 0, 0→3→1, tại 1 không còn lân cận mới.   → in ra 0, 3, 1, xóa 0, 3, 1 khỏi S.   * Lấy v = pop (S) = 4, 4→6→5→2→8, tại 8 có lân cận là 3, do 3 đã xét → in ra 4, 6, 5, 2, 8, xóa 4, 6, 5, 2, 8 khỏi S. * Lấy v = pop (S) = 7, 7 có lân cận là 8, do 8 đã xét   → in ra 7, xóa 7 khỏi S.  ⇒ S lúc này là NULL | Thành phần liên kết mạnh:   1. 0, 3, 1 2. 4, 6, 5, 2, 8 3. 7 |

1. Mã nguồn chương trình:

|  |  |
| --- | --- |
| * Một số cấu trúc dữ liệu sử dụng trong bài. |  |
| * Hàm thêm đỉnh vào đồ thị. |  |
| * Xây dựng đồ thị G.   Nhập vào thông tin của đồ thị: số cạnh, số đỉnh. Danh sách kề của đồ thị. |  |
| * In ra thông tin của đồ thị đã nhập và danh sách các đỉnh lân cận với đỉnh bất kỳ của đồ thị. |  |
| * Sử dụng thuật toán duyệt đồ thị theo chiều sâu cho đồ thị G. |  |
| * Đảo ngược đồ thị G =>H |  |
| * In ra thành phần liên thông. |  |
| * Một số hàm sử dụng cho stack. |  |

1. Kết quả chương trình: (Áp dụng với bộ dữ liệu số 2)

