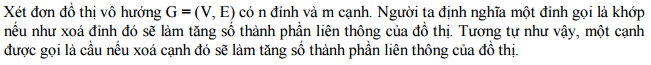
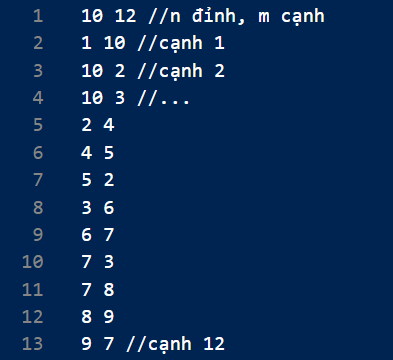
**BÀI 106. KHỚP VÀ CẦU**

**- Bài toán:**

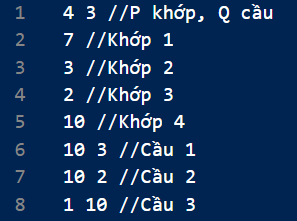


**- Hướng dẫn sử dụng chương trình:**

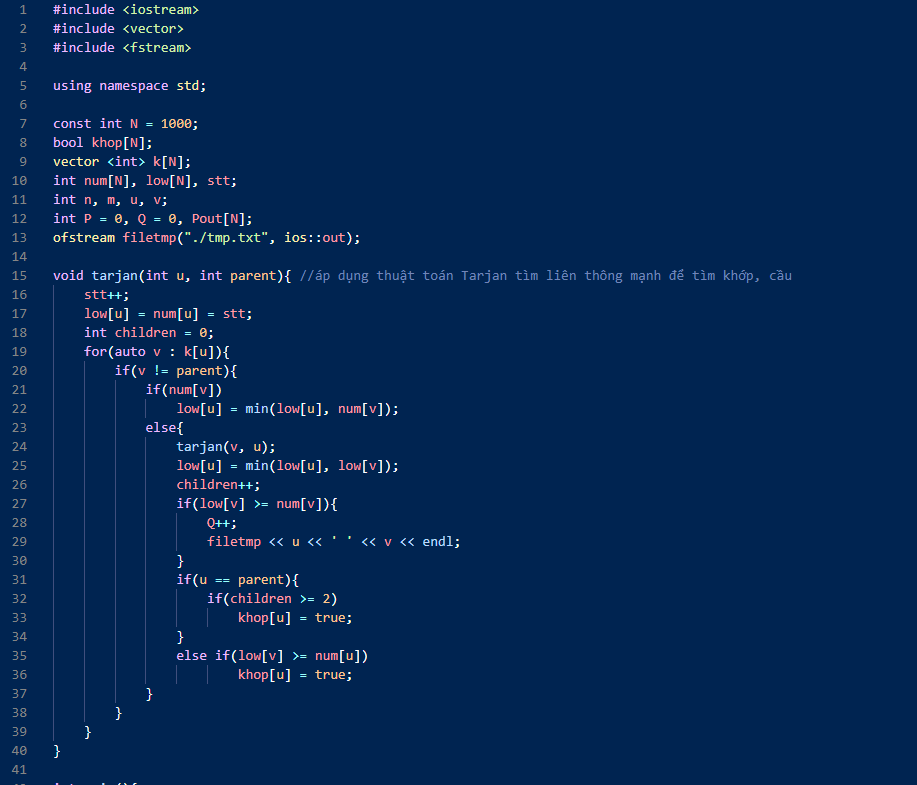
* Dùng file graph.inp để input dữ liệu vào chương trình.
* Trong file graph.inp đó chứa:
  + Dòng đầu tiên: chứa số n, m (1 ≤ n ≤ 1000; m ≤ 10000)
  + Ở m dòng tiếp theo, mỗi dòng sẽ ghi số hiệu hai đỉnh u và v: tức là đỉnh u và v nối thành một cạnh.
  + Ví dụ về file graph.inp:

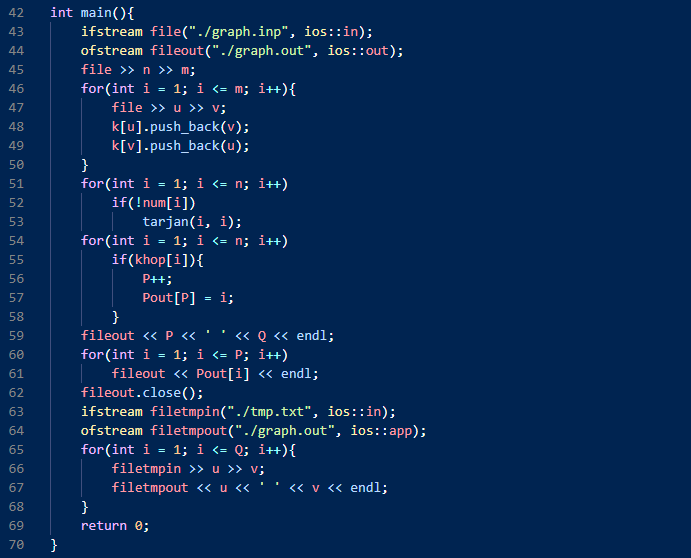


* Kết quả sẽ được xuất ra file graph.out.
* Trong file graph.out đó chứa:
* Dòng đầu tiên: chứa số P (số khớp) và Q (số cầu) tìm được trong đồ thị.
* Ở P dòng tiếp theo, mỗi dòng ghi một khớp (số hiệu đỉnh) tìm được.
* Ở Q dòng tiếp theo, mỗi dòng sẽ ghi số hiệu hai đỉnh tương ứng với một cầu liên thuộc với hai đỉnh đó
* Ví dụ về file graph.out:

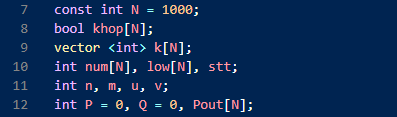


**- Code toàn bộ chương trình:**





- Phần khai báo đầu chương trình:



- Tại bài này áp dụng thuật toán Tarjan tìm thành phần liên thông mạnh để tìm khớp và cầu (từ dòng 15 đến dòng 40):

* Ở thuật toán này, sử dụng chủ yếu 2 mảng low[u], num[u] để thực hiện tìm khớp và cầu. Và một biến stt.



* Khi hàm tarjan(u, parent) được thực hiện:
* Biến “stt” sẽ tăng lên 1 mỗi khi một đỉnh mới được duyệt và đồng thời gán cho mảng low[u], num[u]. Với mảng num[u] là thứ tự được duyệt của đỉnh đó, còn low[u] dùng để so sánh tìm khớp và cầu.



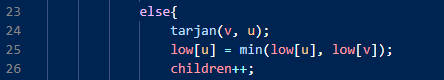
* Tìm những đỉnh kề với đỉnh đang được duyệt (đỉnh u) giống như thuật toán tìm kiếm DFS, BFS,… (không bao gồm đỉnh parent, tức là cha của đỉnh u).



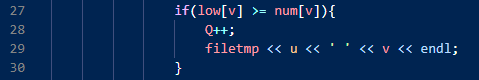
* Nếu num[v] đã được thực hiện (tức từng duyệt qua) thì sẽ chỉ cập nhật low[u] mới bằng giá trị nhỏ hơn giữa low[u] và num[v].



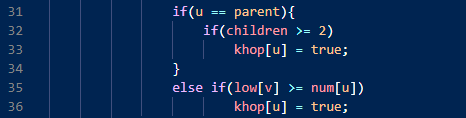
* Ngược lại đỉnh v chưa được duyệt thì sẽ gọi thực hiện lại hàm tarjan(v, u) với đỉnh u là cha của v. Đồng thời cập nhật low[u] bằng min(low[u], low[v]) và tăng biến đếm children (với mỗi đỉnh kề với u tìm được không tính cha của u).



* Nếu low(v) >= num[v] thì đỉnh u và v sẽ là 2 đỉnh cầu. Mỗi lần tìm được một cầu sẽ tăng biến đếm “cầu” Q lên một và lưu vết 2 đỉnh tương ứng.

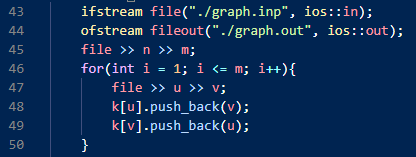


* Nếu đỉnh u vừa là đỉnh đang duyệt và vừa là đỉnh cha của chính nó và có ít nhất 2 đỉnh v kề (children >= 2) thì đỉnh u chính là một khớp (điều này hiển nhiên). Ngược lại nếu low[v] >= num[u] (tức v là một thành phần liên thông mạnh của đồ thị) thì đỉnh u cũng là một khớp.



- Tại hàm main (từ dòng 42 đến 70) sẽ thực hiện:

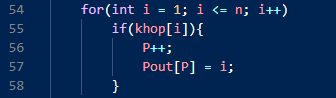
* Mở file, đọc biến n (số đỉnh), m (số cạnh) của đồ thị. Và lệnh for chạy từ 1 đến m cạnh để lưu danh sách các cạnh của đồ thị bằng vector k[] (sử dụng push\_back giữa 2 đỉnh)



* For chạy từ đỉnh i = 1 đến n. Nếu đỉnh i chưa được duyệt thì thực hiện tarjan(i, i).



* Sau khi chạy xong xuôi thuật toán tarjan tìm khớp và cầu thì sử dụng một vòng for để tìm số lượng khớp (biến P) và lưu vết khớp.



* In P (số lượng khớp), Q (số lượng cầu), các khớp và cầu (2 đỉnh tương ứng của cầu) vào file graph.out



