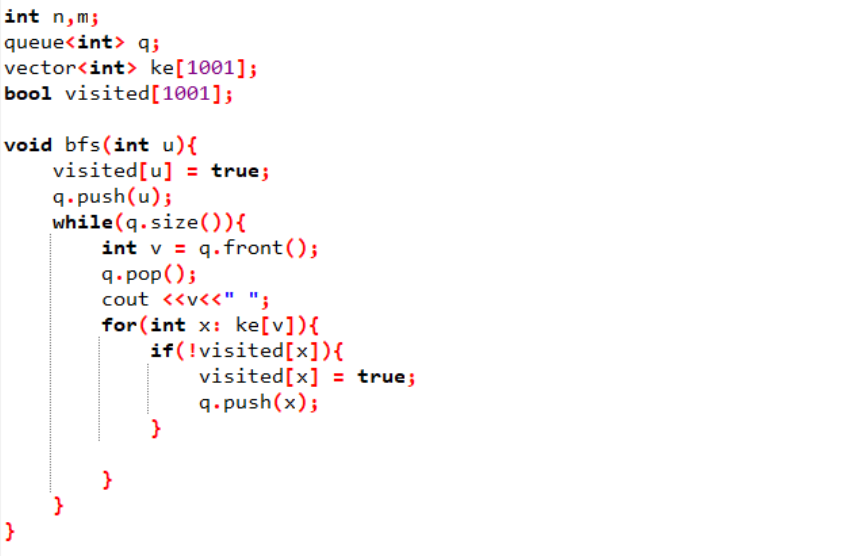
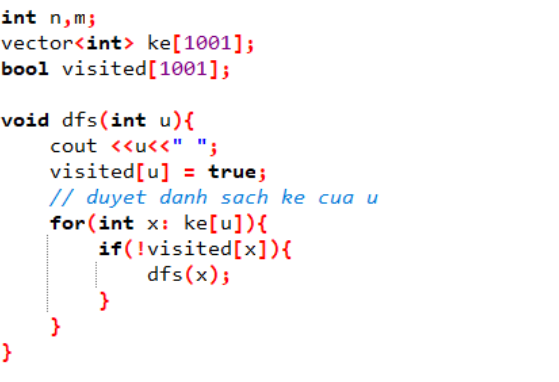
* Lý thuyết đồ thị: Graph theory

1. Một số thuật toán cơ bản trên đồ thị và mã giải:
2. BFS và DFS:
3. BFS: Duyệt theo chiều rộng, BFS luôn cho đường đi ngắn nhất ( đối với ma trận không có trọng số)

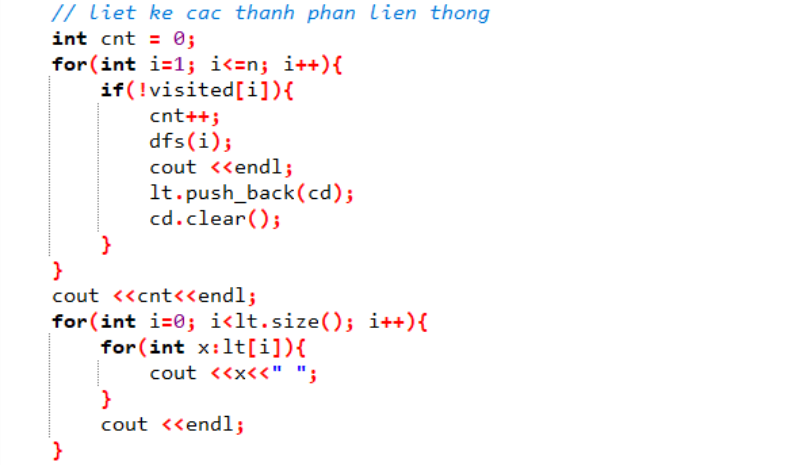


1. DFS: Duyệt theo chiều sâu



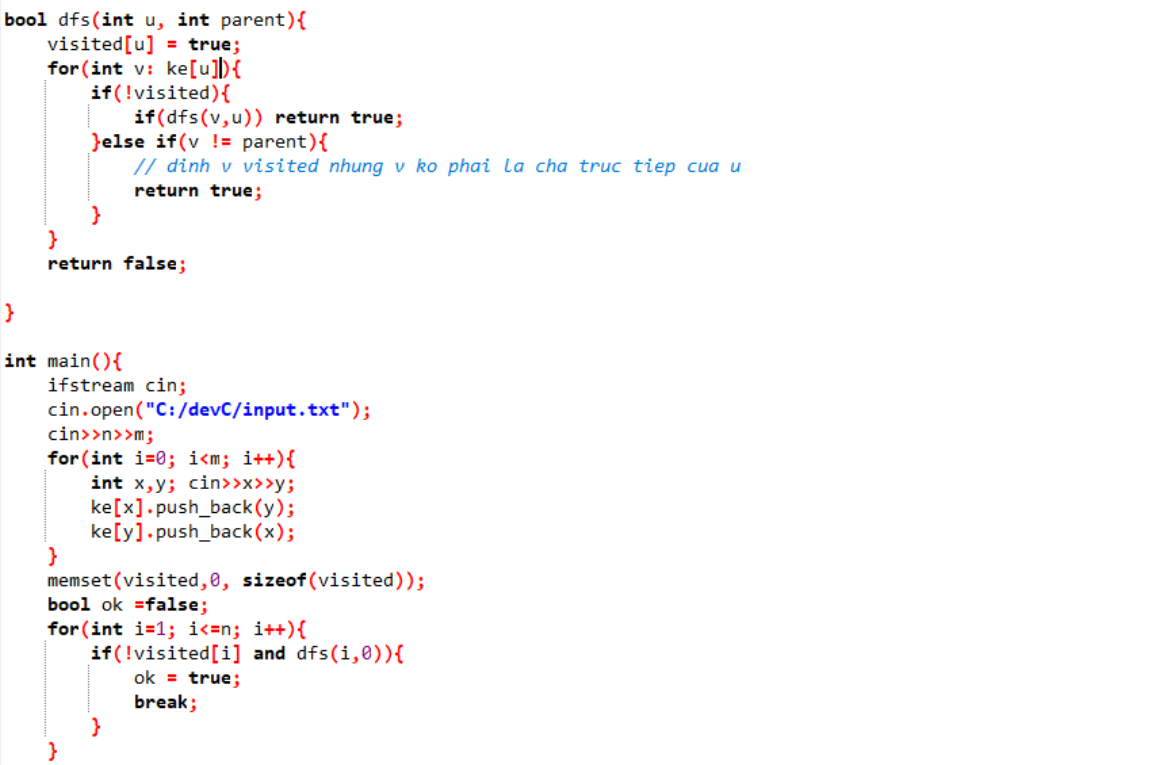
1. Check các thành phần liên thông: BFS hoặc DFS

* Liên thông là các đỉnh của đồ thị được nối với nhau, trong đồ thị VH gọi là liên thông, trong đồ thị có hướng là liên thông mạnh nếu có đường đi giữa các đỉnh với nhau, khi một đồ thị có nhiều liên thông thì gọi là các thành phần liên thông (Nếu check có 1 tplt thì đồ thị liên thông, nhiều hơn thì gọi là các thành phần liên thông)\_(Code check liên thông trong đồ thị vô hướng)

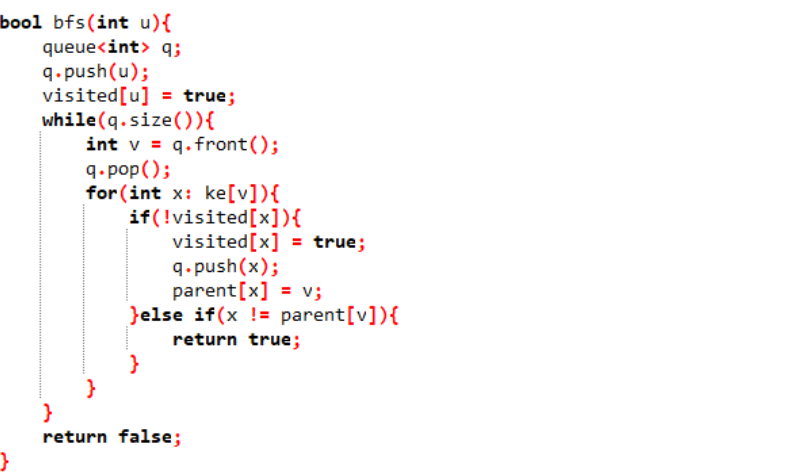


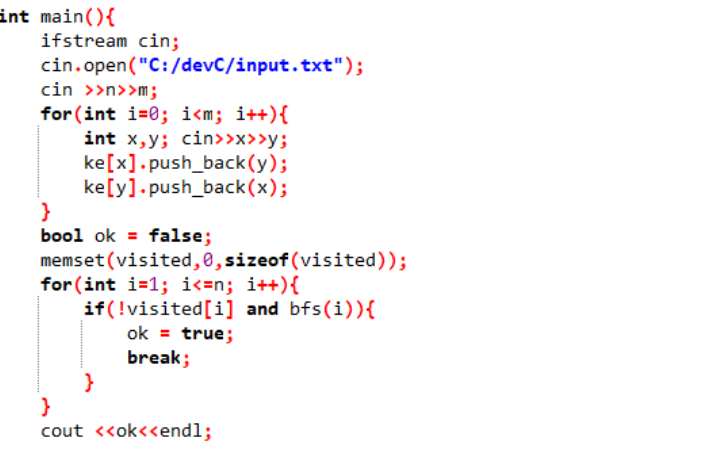
1. Chu trình: Chu trình là một đường đi khép kín xuất phát từ 1 đỉnh (cycle)

* Chu trình vô hướng bằng DFS:

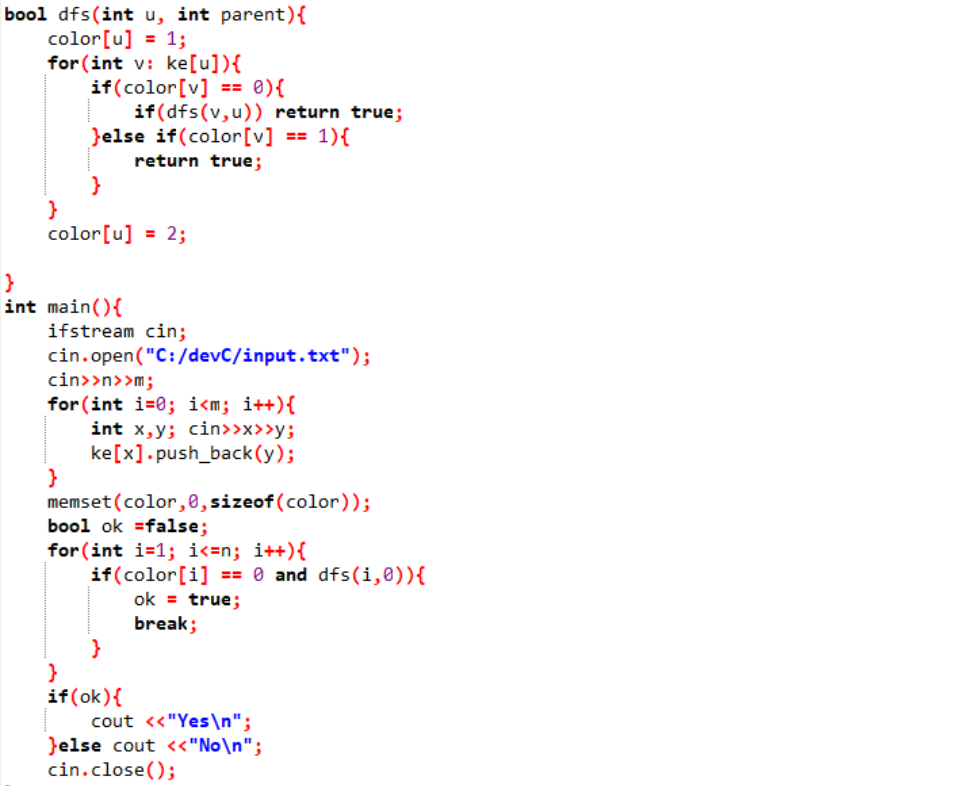


* Chu trình vô hường bằng BFS:

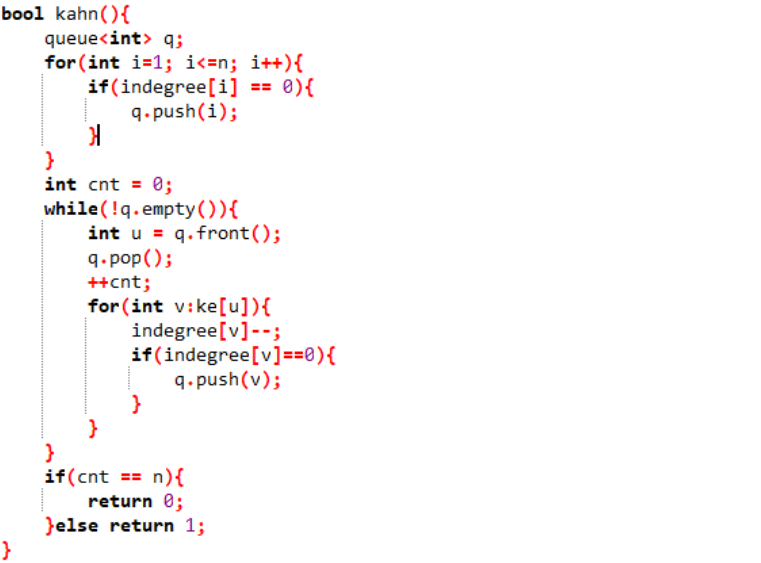




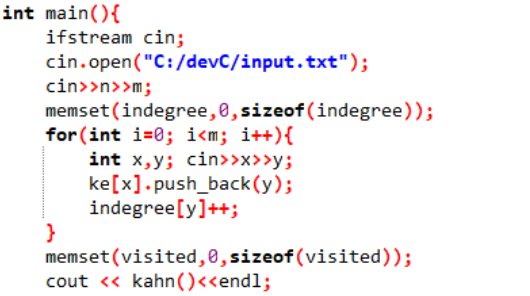
* Check chu trình trong đồ thị có hướng bằng DFS ( tô màu đỉnh): Đồ thị có chu trình khi và chỉ khi đỉnh được duyệt qua đang ở trạng thái được duyệt (1). Quy ước 1 đang duyệt, 0 chưa duyệt, 2 đã duyệt xong

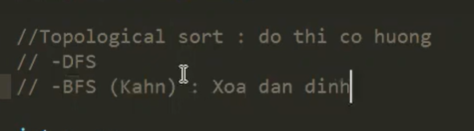


* Check chu trình trong đồ thị có hướng bằng BFS (Kahn): Thuật toán xóa dần đỉnh, giả sử đỉnh được xét ban đầu có bán bậc vào bằng 0 và xét từ đỉnh đó, các đỉnh được đi qua sẽ giảm dần bán bậc vào, bán bậc vào bằng 0 là đã duyệt, số lượng cnt == n thì chu trình không tồn tại, chu trình chỉ tồn tại khi cnt != n, thuật toán Kahn có thể dùng để check chu trình cho 1 đồ thị có hướng có nhiều thành phần liên thông.

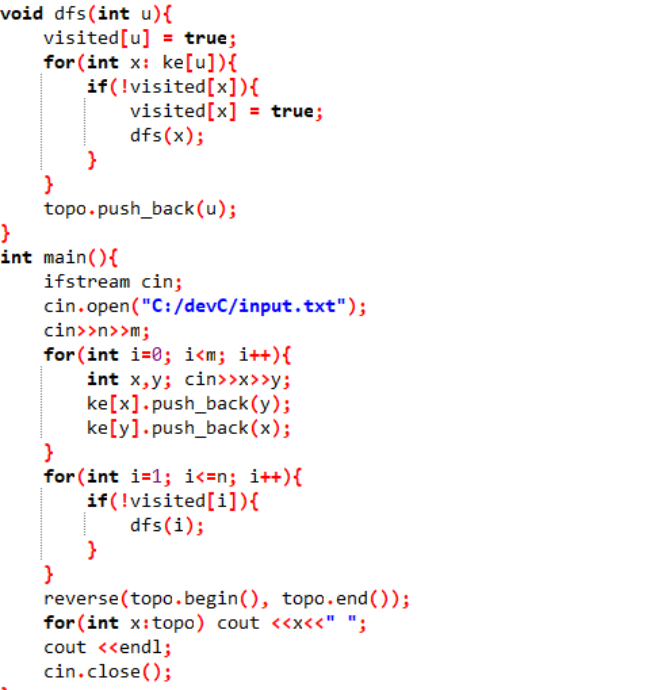


(Xem thêm phần topo dùng Kahn)



1. Toppo logical sort: Sắp xếp thứ tự các đỉnh trước sau trong đồ thị , các đỉnh đứng trước luôn được sắp trước, tuần tự. Toppo chỉ được dùng trong đồ thị có hướng. Sắp xếp toppo trong đồ thị có chu trình chỉ có thể dùng BFS.

* Toppo bằng DFS: vô hướng



* Toppo DFS có hướng



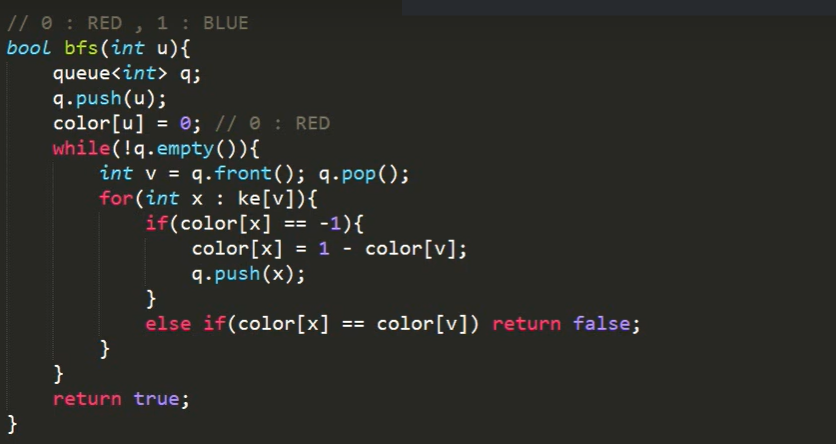
* Toppo BFS (dùng thuật toán Kahn) tính bán bậc vào, giảm dần các bán bậc vào, bán bậc vào bằng 0 nghĩa là đỉnh đó đã được thăm xong và đứng trước các đỉnh có bán bậc vào lớn hơn. (cnt <n là có chu trình)
* 

1. Đồ thị hai phía: Đồ thị hai phía là đồ thị có các tập đỉnh có thể chia làm hai tập không giao nhau thỏa mãn điều kiện không có cạnh nối giữa hai điểm bất kì thuộc cùng 1 tập (Chỉ có trong đồ thị vô hướng)

* DFS

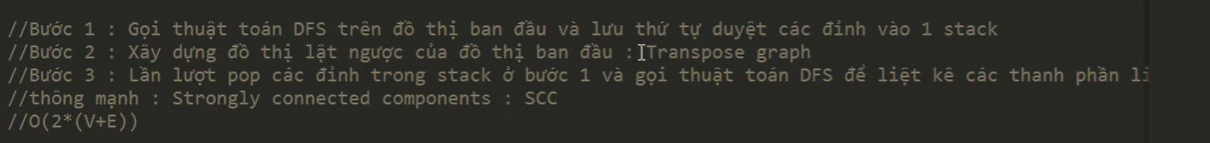


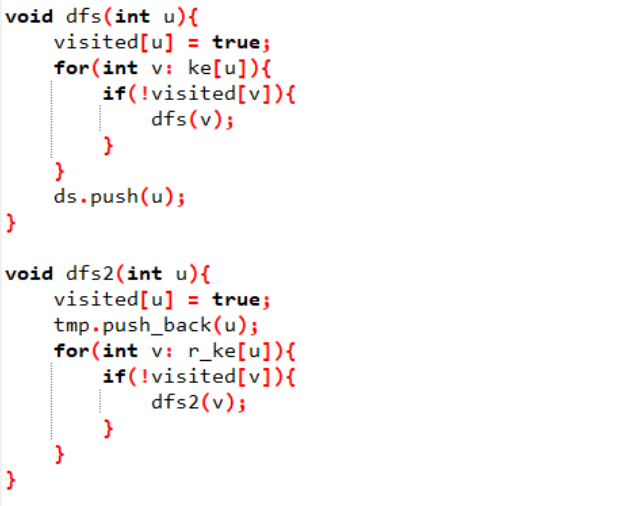
* BFS

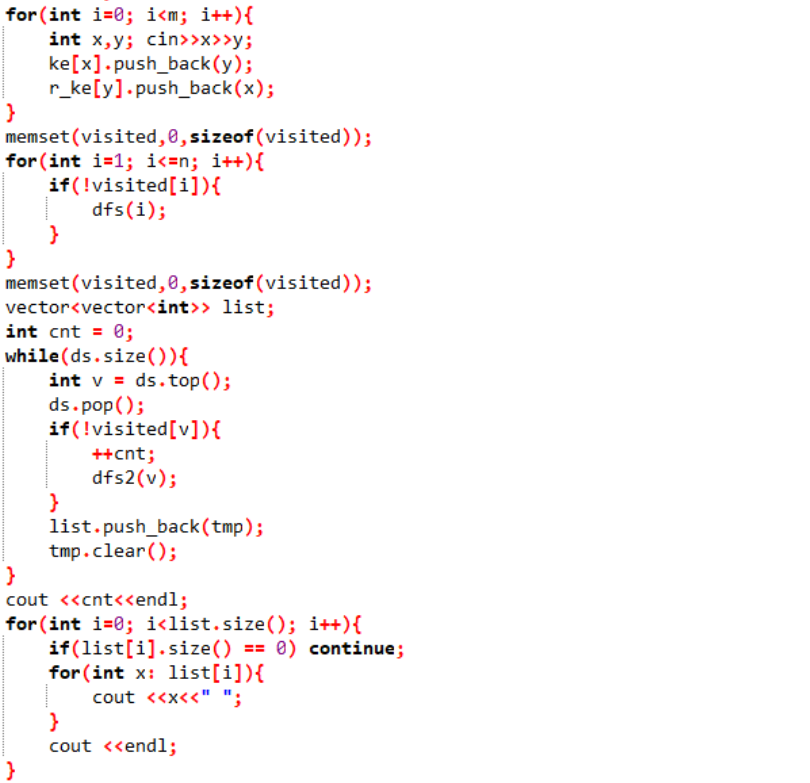


1. Một số thuật toán cần biết:
2. Kosaraju: Dùng để check liên thông mạnh (Chỉ có 1 thành phần liên thông trong đồ thị) hoặc các thành phần liên thông mạnh trong đồ thị có hướng, 1 đỉnh vẫn được xem là thành phần liên thông mạnh.

* Mã giải:

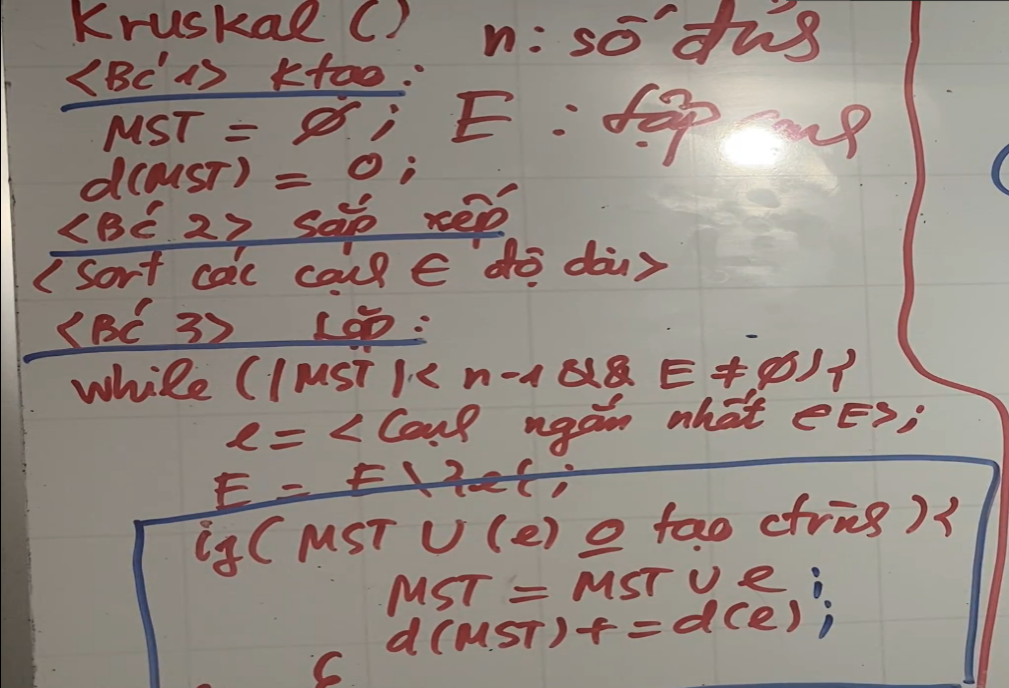


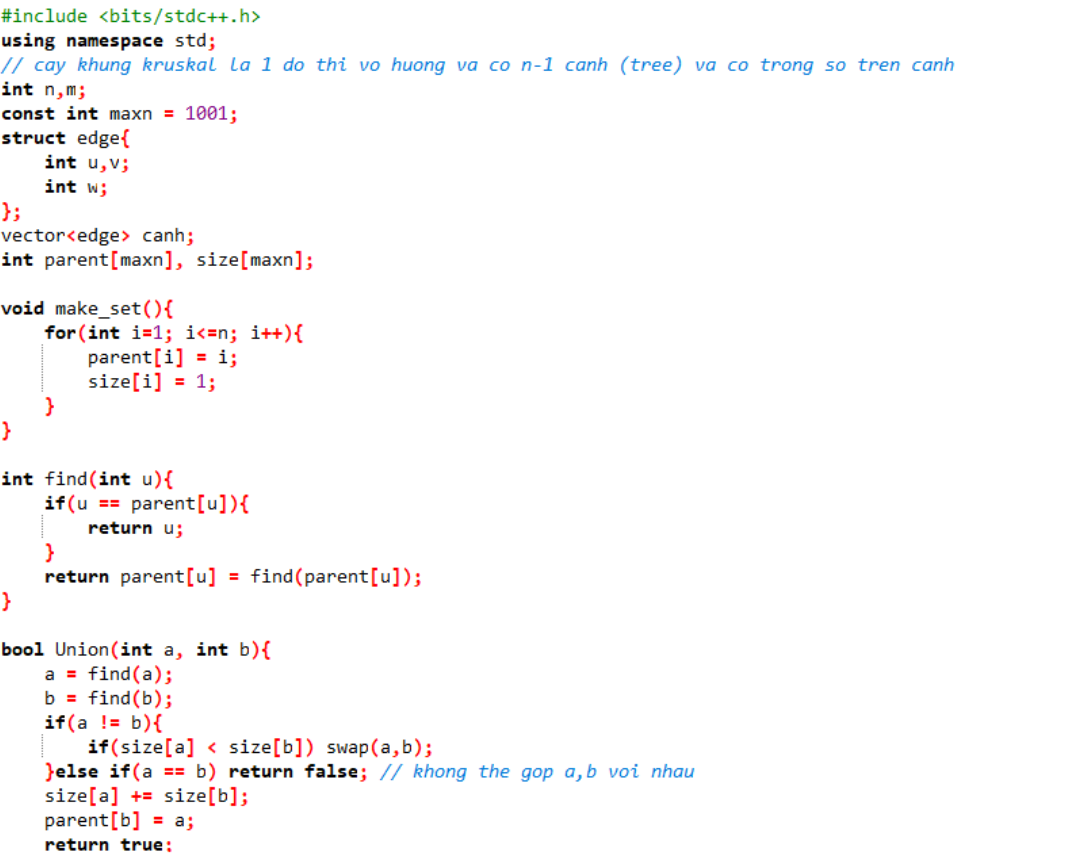


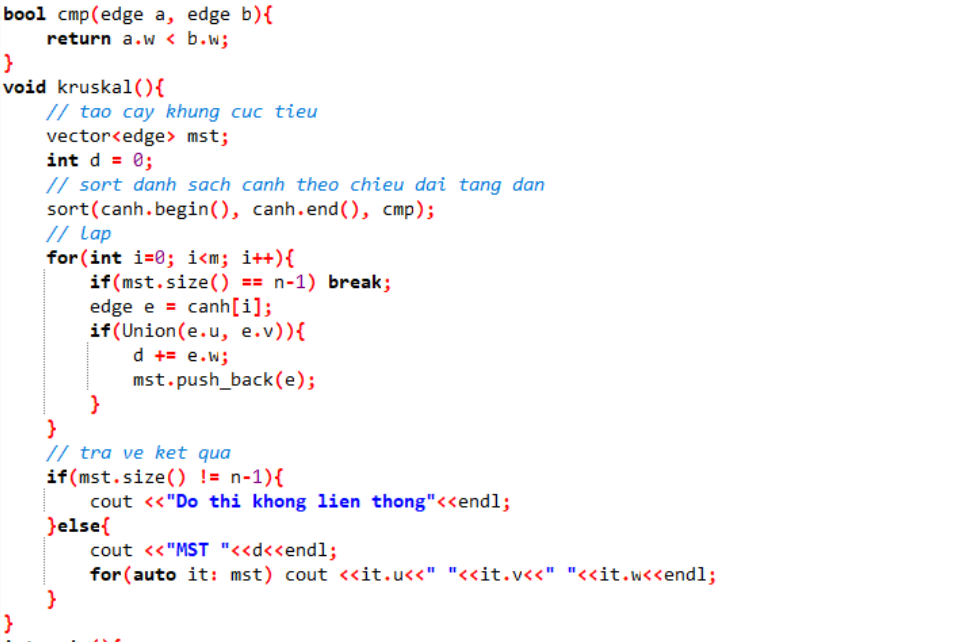


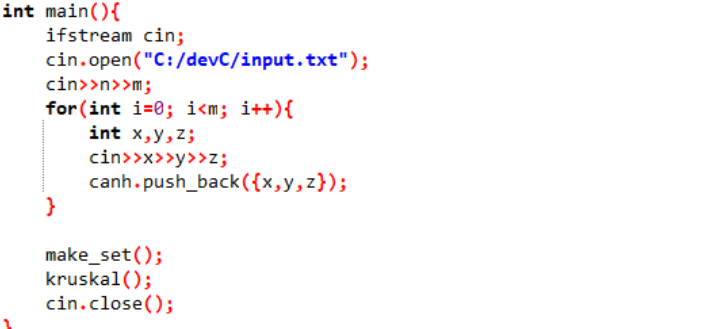
1. Disjoint Set Union (Union find – Các tập hợp rời nhau): Cho các đỉnh riêng lẻ, nối các đỉnh với nhau.
2. Kruskal (Cây khung cực tiểu)

* Cây khung – Minimum Spanning Tree (MST).
* Cây khung kết nối tất cả các đỉnh trong đồ thị, có n-1 cạnh, không được có chu trình, thỏa mãn các tính chất của cây.
* Cây khung cực tiểu là cây có trọng số nhỏ nhất.
* Mã giải:



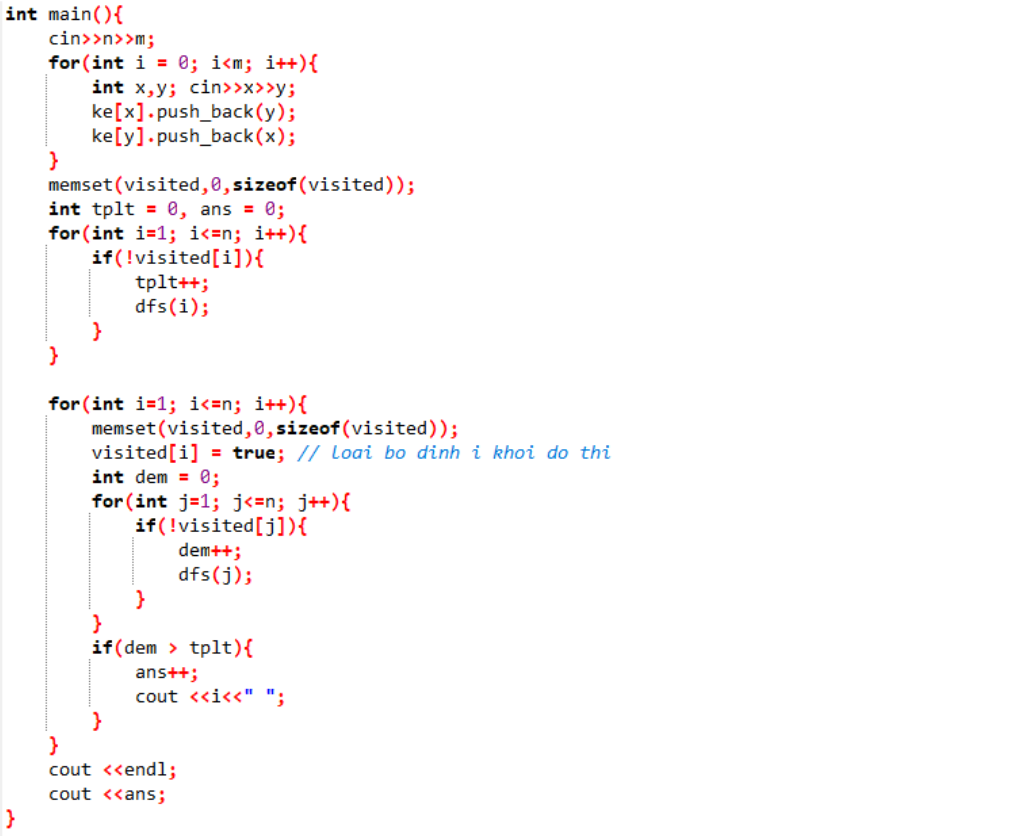




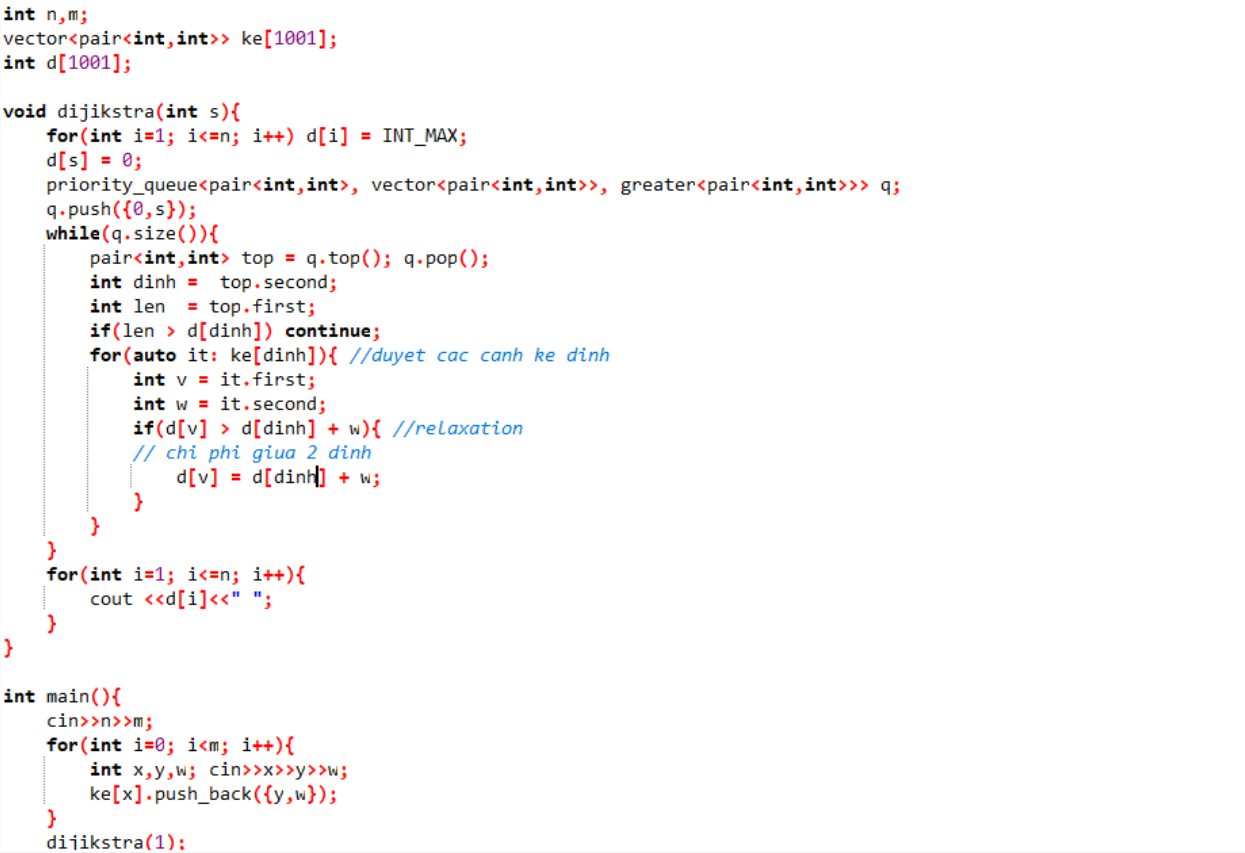


1. Đỉnh trụ: Loại bỏ 1 đỉnh thì tăng thành phần liên thông (Có thể dùng Tarjan)

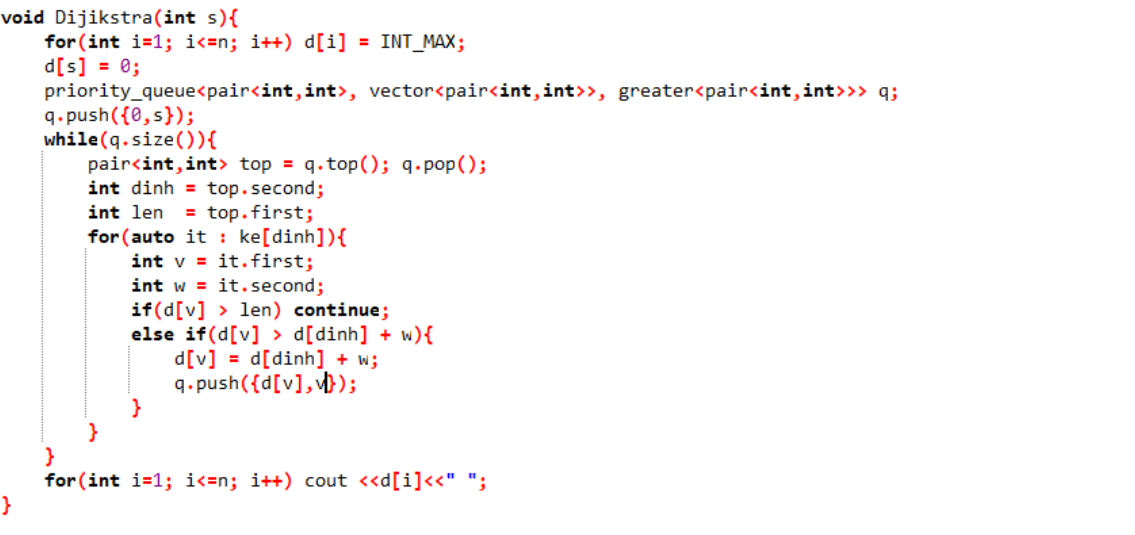
* Brute force



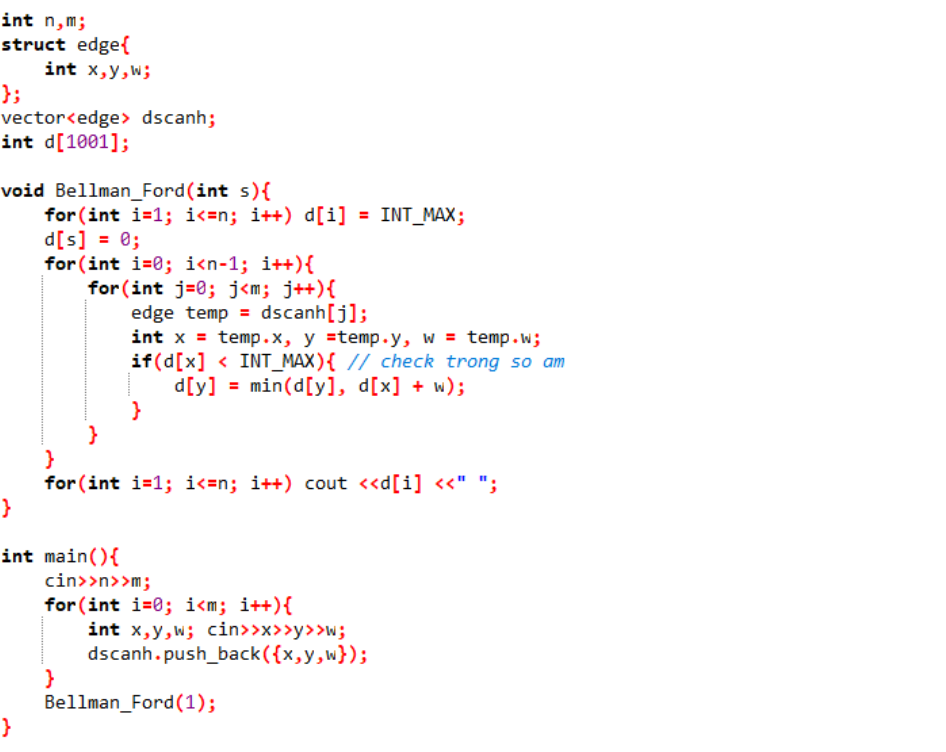
1. Cạnh cầu: Loại bỏ các cạnh nối làm tăng thành phần liên thông (Dùng Tarjan, duyệt trâu bị TLE)
2. Dijikstra: Chi phí nhỏ nhất (hoặc lớn nhất) từ 1 đỉnh tới mọi đỉnh còn lại. Áp dụng cho đồ thị vô hướng, có hướng nhưng không có trọng số âm



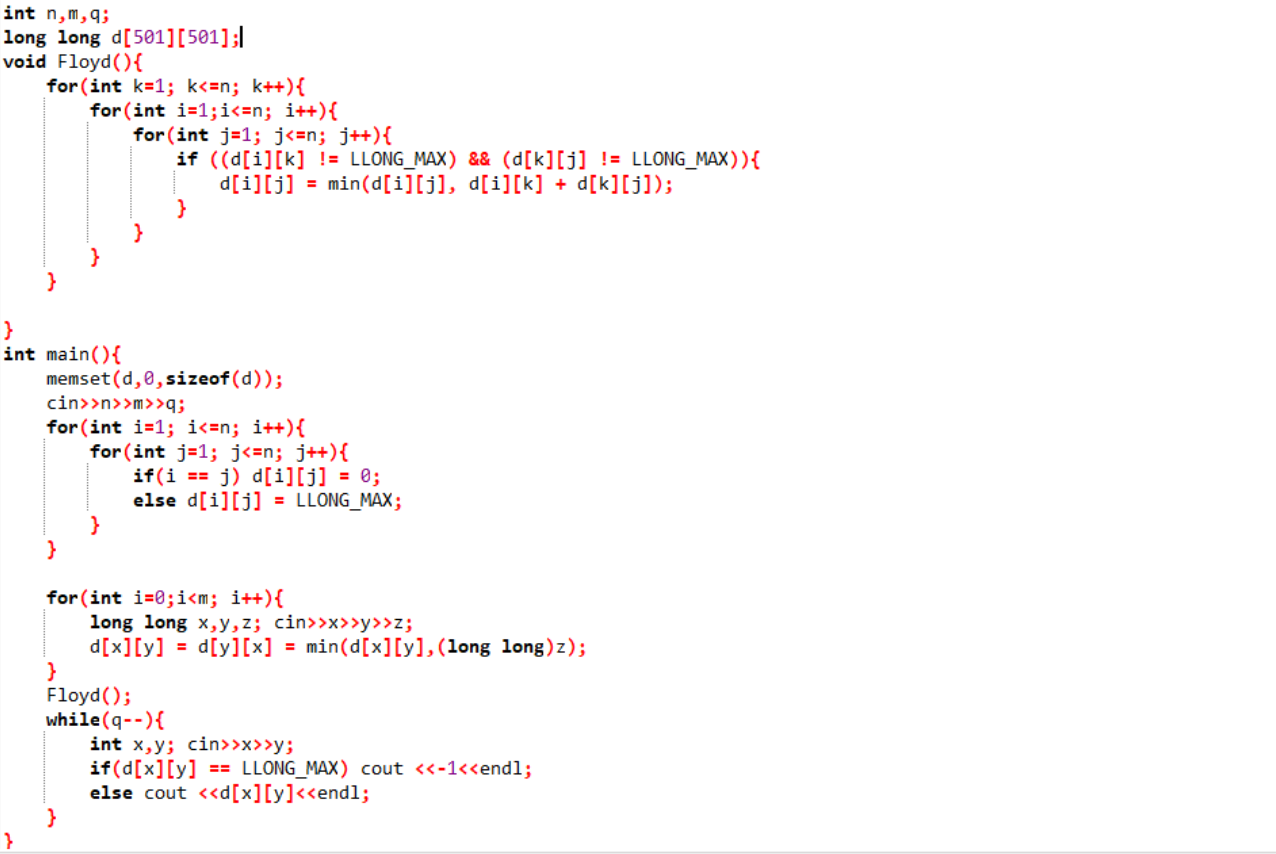
(Nhớ đẩy chi phí và đường đi mới)



1. Bellman-Ford: Giống Dijikstra nhưng có thể check trọng số âm (không có chu trình âm)



1. Floyd: Đường đi ngắn nhất giữa 2 đỉnh bất kì. Khởi tạo ma trận (Đường chéo chính = 0, các ô còn lại là INF, nhập ma trận)



1. Chu trình euler
2. Chu trình Hamiton

* Stack

1. Biểu thức trung tố, tiền tố và hậu tố:
2. Biểu thứ trung tố: các toán tử nằm giữa hai toán hạng

VD: a+b+c\*d

1. Biểu thức tiền tố: Toán tử nằm trước toán hạng

VD: +ab

1. Biểu thức hậu tố: Toán tử nằm sau toán hạng

VD: ab+