# Euler

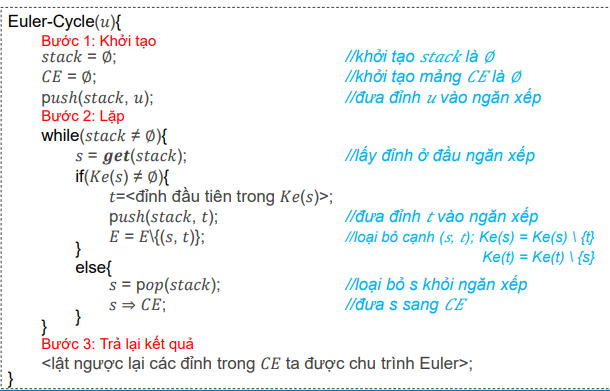
## Định nghĩa:

* Chu trình đơn trong đồ thị 𝐺 đi qua tất cả các cạnh của nó được gọi là chu trình Euler.
* Đường đi đơn trong đồ thị G đi qua tất cả các cạnh của nó được gọi là đường đi Euler
* Đồ thị được gọi là đồ thị Euler nếu nó có chu trình Euler
* Đồ thị được gọi là đồ thị nửa Euler nếu nó có đường đi Euler

## Đk cần và đủ

* Với đồ thị vô hướng: là đồ thị Euler khi và chỉ khi mọi đỉnh đều có bậc chẵn
* Với đồ thị có hướng: là đồ thị Euler khi và chỉ khi tất cả các đỉnh của nó đều có bán bậc ra bằng bán bậc vào

## Thuật toán tìm chu trình euler:

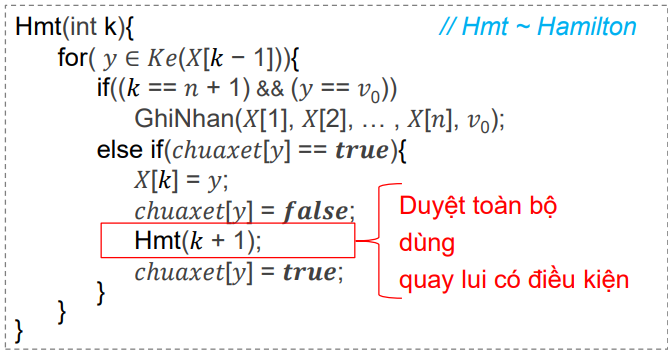


# Hamilton

## Định nghĩa:

* Đường đi qua tất cả các đỉnh của đồ thị, mỗi đỉnh đúng một lần được gọi là đường đi Hamilton
* Chu trình bắt đầu tại một đỉnh 𝑣 nào đó, qua tất cả các đỉnh còn lại mỗi đỉnh đúng một lần, sau đó quay trở lại 𝑣, được gọi là chu trình Hamilton
* Đồ thị được gọi là đồ thị Hamilton nếu có chu trình Hamilton.
* Đồ thị được gọi là đồ thị nửa Hamilton nếu có đường đi Hamilton.

## Thuật toán



# Cây khung của đồ thị

* Cây là 1 đồ thị vô hướng liên thông, ko có chu trình

## Các tính chất của cây:

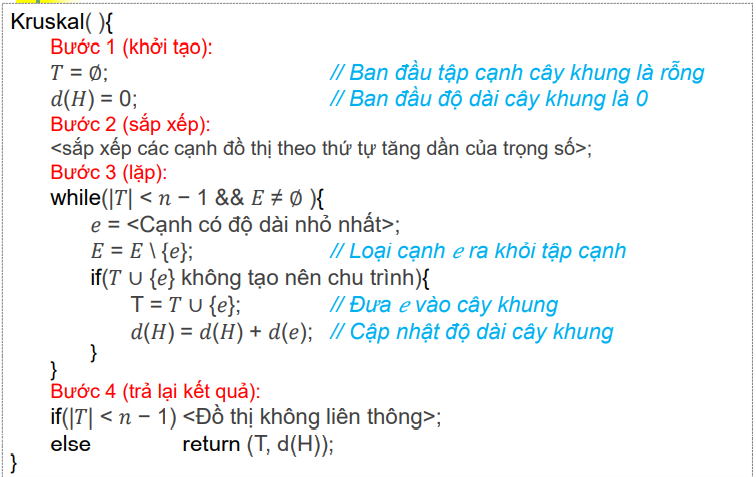
Giả sử T là đồ thị vô hướng n đỉnh, khi đó các khẳng định sau là tương đương:

1. T là 1 cây
2. T ko có chu trình và có n-1 cạnh
3. T liên thông và có đúng n-1 cạnh
4. T liên thông và mỗi cạnh của nó đều là cầu
5. Giữa 2 đỉnh bất kì của T được nối với nhau bởi đúng 1 đường đi đơn
6. T ko chứa chu trình nhưng hễ cứ thêm vào nó một cạnh ta được đúng 1 chu trình

Số cây khung của đồ thị đầy đủ n đỉnh Kn là n^(n-2)

# Thuật toán Kruskal

1. Mỗi bước chọn cạnh có trọng số nhỏ nhất chưa nằm trong khung
2. Nếu việc thêm cạnh này vào khung không tạo thành chu trình thì thêm cạnh này vào
3. Thuật toán dừng lại khi có đủ n-1 cạnh



# Thuật toán Prim

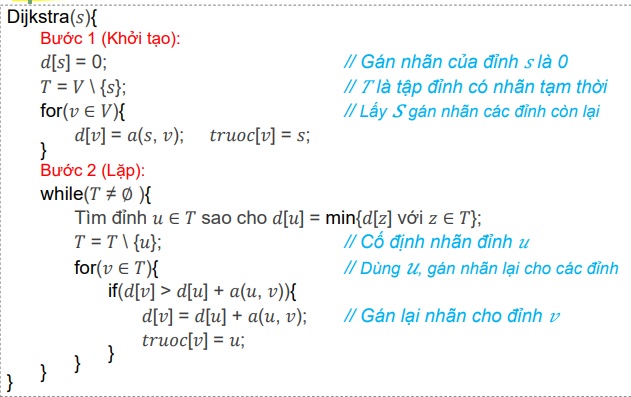
1. Chọn 1 cạnh có trọng số nhỏ nhất đặt vào cây
2. Bổ sung vào cây các cạnh có trọng số nhỏ nhất kề với 1 đỉnh đã có trong cây mà ko tạo ra

1 chu trình đơn = những cạnh đã có trong cây\

# Thuật toán Dijkstra

Áp dụng cho đơn đồ thị liên thông trọng số không âm

* Gán nhãn tạm thời cho các đỉnh
* Nhãn của mỗi đỉnh cho biết cận trên của độ dài đường đi ngắn nhất tới đỉnh đó
* Biến đổi các nhãn(tính lại) theo một thủ tục lặp
* Ở mỗi bước lặp sẽ có một nhãn tạm thời trở thành nhãn cố định- chính là độ dài đường đi ngắn từ s đến đỉnh đó



# Thuật toán Bellman-Ford

Áp dụng cho đồ thị có hướng và không có chu trình âm nhưng có thể có cạnh âm

* Gán nhãn tạm thời cho các đỉnh
* Nhãn của mỗi đỉnh cho biết cận trên của độ dài đường đi ngắn nhất tới đỉnh đó
* Tính lại các nhãn nhờ 1 thủ tục lặp
* Mỗi khi phát hiện d[v] > d[u] + a(u,v) => cập nhật : d[v] = d[u] + a(u,v)

