# Cấu trúc dữ liệu và giải thuật

Graph Algorithms (Kruskal, Prim, Dijkstra)

## Nội dung

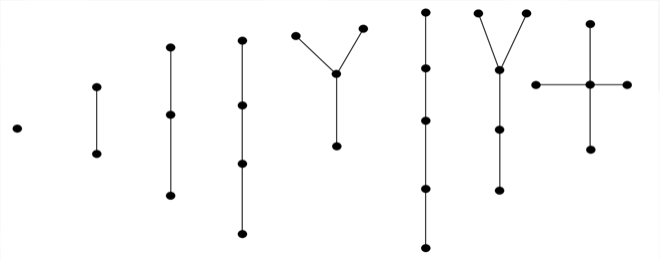
- ❑ Cây và cây khung của đồ thị
- ❑ Thuật toán Kruskal
- ❑ Thuật toán Prim
- ❑ Thuật toán Dijkstra

## Nội dung

- **Cây và cây khung của đồ thị**
- Thuật toán Kruskal
- Thuật toán Prim
- Thuật toán Dijkstra

## Cây

- Cây là một đồ thị vô hướng liên thông, không chứa chu trình.
- Một đồ thị vô hướng không chứa chu trình và có ít nhất hai đỉnh gọi là một rừng.
  - Trong một rừng, mỗi thành phần liên thông là một cây.

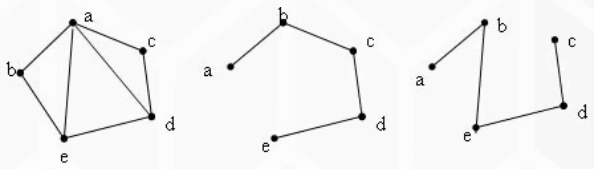


## Tính chất cơ bản

- Với T là một đồ thị có n ≥ 2 đỉnh. Các mệnh đề sau là tương đương:
  - T là một cây.
  - T liên thông và có n−1 cạnh.
  - T không chứa chu trình và có n−1 cạnh.
  - T liên thông và mỗi cạnh là cầu.
  - Giữa hai đỉnh phân biệt bất kỳ của T luôn có duy nhất một đường đi đơn.
  - T không chứa chu trình nhưng khi thêm một cạnh mới thì có được một chu trình duy nhất.

## Cây khung của đồ thị

- Trong đồ thị liên thông G, nếu loại bỏ cạnh nằm trên chu trình nào đó thì sẽ được đồ thị vẫn là liên thông. Nếu cứ loại bỏ các cạnh ở các chu trình khác cho đến khi nào đồ thị không còn chu trình (vẫn liên thông) thì thu được một cây nối các đỉnh của G. Cây đó gọi là cây khung hay cây bao trùm của đồ thị G.



## Cây khung nhỏ nhất

- Cho $\boldsymbol{G(V, E)}$ là đồ thị vô hướng liên thông có trọng số, mỗi cạnh $\mathbf{e} \in \boldsymbol{E}$ có trọng số $m(e) \geq 0$. Giả sử $\boldsymbol{T} = (\boldsymbol{V_T}, \boldsymbol{E_T})$ là cây khung của đồ thị $\boldsymbol{G}$ ($V_T = V$). Ta gọi độ dài $m(T)$ của cây khung $\boldsymbol{T}$ là tổng trọng số của các cạnh của nó:

$$m(T) = \sum_{e \in E_T} m(e)$$

- Yêu cầu đặt ra là trong số tất cả các cây khung của đồ thị $\boldsymbol{G}$, hãy tìm cây khung có độ dài nhỏ nhất. Cây khung như vậy được gọi là cây khung nhỏ nhất của đồ thị.

## Nội dung

- *Cây và cây khung của đồ thị*
- **Thuật toán Kruskal**
- Thuật toán Prim
- Thuật toán Dijkstra

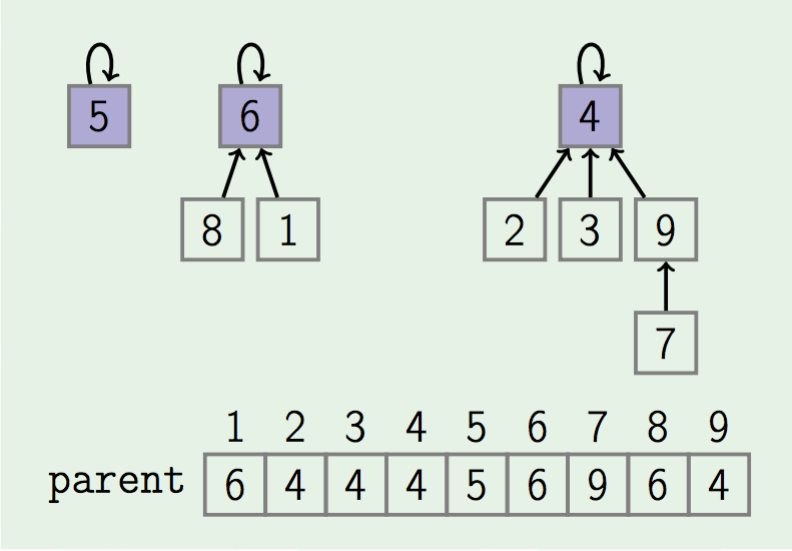
## Thuật toán Kruskal

- Thuật toán sẽ xây dựng tập cạnh $E_T$ của cây khung nhỏ nhất $T = (V_T, E_T)$ theo từng bước:
  - Bắt đầu từ đồ thị rỗng T có n đỉnh.
  - Sắp xếp các cạnh của G theo thứ tự tăng dần của trọng số.
  - Bắt đầu từ cạnh đầu tiên của tập đã được sắp xếp, thêm dần các cạnh của tập đã được sắp xếp vào T theo nguyên tắc cạnh thêm vào không được tạo thành chu trình trong T.
  - Lặp cho đến khi nào số cạnh trong T bằng n−1, ta thu được cây khung nhỏ nhất cần tìm.s

## Cấu trúc Union-Find

- Còn gọi là cấu trúc Disjoint Set, sử dụng để kiểm tra 1 đồ thị vô hướng có chu trình hay không.
- Disjoint set là một cấu trúc dữ liệu theo dõi (tracking) một tập các phần tử được phân chia thành các tập con khác nhau không chồng chéo nhau (non-overlapping).
- Ví dụ: Cho tập hợp gồm N phần tử A = {1, 2, 3, 4 ,5, 6, 7, 8, 9} và có các liên kết (6, 1) (6, 8) (4, 2) (4, 3) (4, 9) (9, 7). Tìm các tập hợp con mà giữa 2 tập hợp không có liên kết trực tiếp hoặc gián tiếp.

# Ý tưởng cấu trúc Disjoint set



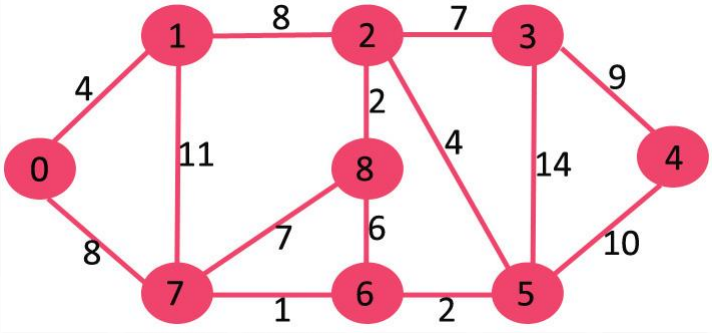

# Giải thuật

- Thực hiện bởi 2 hàm là:
  - Find: tìm tập hợp chứa phần tử (tìm root).
  - Union: hợp 2 tập hợp vào với nhau (gắn root cây này vào cây kia dựa vào rank hoặc size).

## Độ phức tạp của thuật toán

- Chi phí của thuật toán **Kruskal** tìm cây khung nhỏ nhất cho đồ thị $G(V, E)$ là: $O(V^2)$.
- Nếu sử dụng cấu trúc **UnionFind** thì độ phức tạp thuật toán sẽ giảm và đạt: $O(E \log V)$.

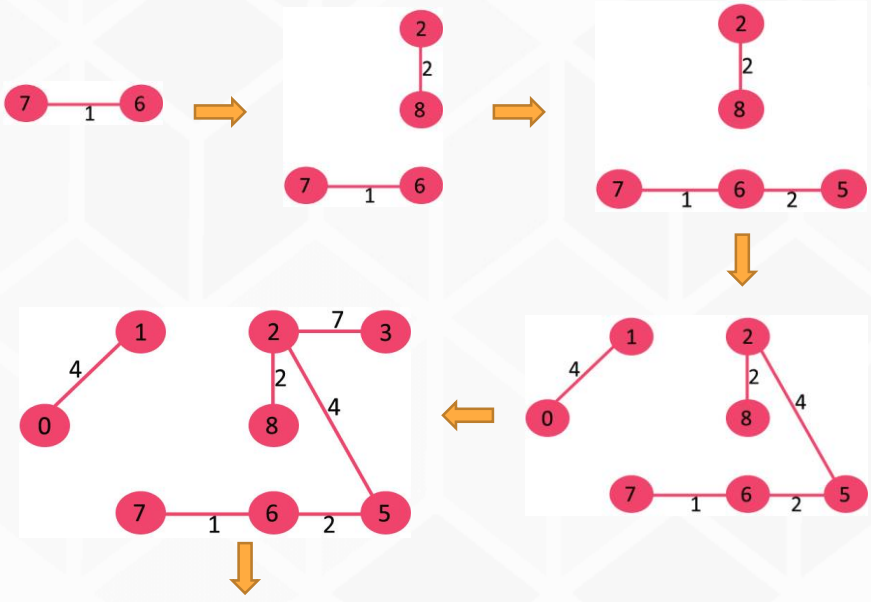
## Ví dụ

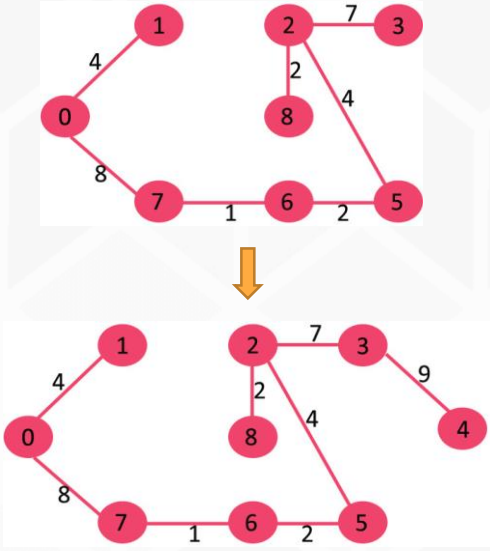


# Sắp xếp các cạnh

| Trọng số | Đỉnh đầu | Đỉnh cuối |
|---|---|---|
| 1 | 7 | 6 |
| 2 | 8 | 2 |
| 2 | 6 | 5 |
| 4 | 0 | 1 |
| 4 | 2 | 5 |
| 6 | 8 | 6 |
| 7 | 2 | 3 |
| 7 | 7 | 8 |
| 8 | 0 | 7 |
| 8 | 1 | 2 |
| 9 | 3 | 4 |
| 10 | 5 | 4 |
| 11 | 1 | 7 |
| 14 | 3 | 5 |

# Lấy lần lượt các cạnh






## Nội dung

- ❑ *Cây và cây khung của đồ thị*
- ❑ *Thuật toán Kruskal*
- ❑ **Thuật toán Prim**
- ❑ Thuật toán Dijkstra

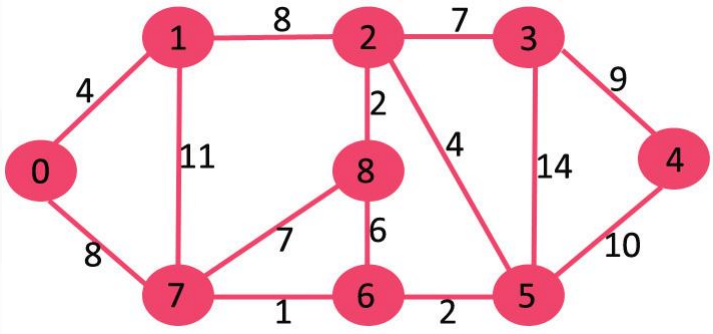
## Thuật toán Prim

- Còn được gọi là phương pháp lân cận gần nhất, thực hiện như sau với đồ thị G có n đỉnh:
  - Khởi tạo T là tập rỗng chứa các cạnh của cây khung nhỏ nhất.
  - Gán nhãn cho tất cả đỉnh của đồ thị với giá trị INFINITE. Gán nhãn giá trị 0 cho đỉnh xét đầu tiên (chọn bất kỳ).
  - Trong khi T chưa đủ n đỉnh thì thực hiện lặp:
    - Chọn đỉnh hiện chưa có trong **T** và có nhãn giá trị nhỏ nhất **u** cho vào **T**.
    - Cập nhật giá trị nhãn cho các đỉnh liền kề **v** với đỉnh **u** mới thêm vào **T** theo nguyên tắc nếu giá trị trọng số ***u-v*** nhỏ hơn giá trị nhãn hiện tại của **v** thì cập nhật giá trị nhãn hiện tại của v là giá trị trọng số ***u-v***.

## Độ phức tạp của thuật toán

- Chi phí thực hiện thuật toán Prim cho đồ thị $\boldsymbol{G(V,E)}$ là: $\boldsymbol{O(V^2)}$.

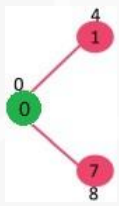
# Ví dụ



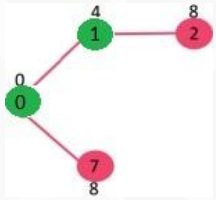

# Thực hiện

| Đỉnh | Nhãn | Đã lấy |
|---|---|---|
| 0 | 0 | |
| 1 | INFINITE | |
| 2 | INFINITE | |
| 3 | INFINITE | |
| 4 | INFINITE | |
| 5 | INFINITE | |
| 6 | INFINITE | |
| 7 | INFINITE | |
| 8 | INFINITE | |

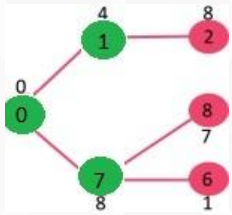
| Đỉnh | Nhãn | Đã lấy |
|---|---|---|
| 0 | 0 | x |
| 1 | 4 | |
| 2 | INFINITE | |
| 3 | INFINITE | |
| 4 | INFINITE | |
| 5 | INFINITE | |
| 6 | INFINITE | |
| 7 | 8 | |
| 8 | INFINITE | |



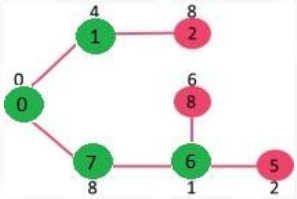
| Đỉnh | Nhãn | Đã lấy |
|---|---|---|
| 0 | 0 | x |
| 1 | 4 | x |
| 2 | 8 | |
| 3 | INFINITE | |
| 4 | INFINITE | |
| 5 | INFINITE | |
| 6 | INFINITE | |
| 7 | 8 | |
| 8 | INFINITE | |




| Đỉnh | Nhãn | Đã lấy |
|---|---|---|
| 0 | 0 | x |
| 1 | 4 | x |
| 2 | 8 | |
| 3 | INFINITE | |
| 4 | INFINITE | |
| 5 | INFINITE | |
| 6 | 1 | |
| 7 | 8 | x |
| 8 | 7 | |



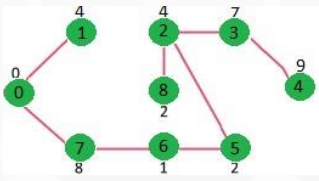
| Đỉnh | Nhãn | Đã lấy |
|---|---|---|
| 0 | 0 | x |
| 1 | 4 | x |
| 2 | 8 | |
| 3 | INFINITE | |
| 4 | INFINITE | |
| 5 | 2 | |
| 6 | 1 | x |
| 7 | 8 | x |
| 8 | 7 -> 6 | |




| Đỉnh | Nhãn | Đã lấy |
|---|---|---|
| 0 | 0 | x |
| 1 | 4 | x |
| 2 | 8 -> 4 | |
| 3 | 14 | |
| 4 | 10 | |
| 5 | 2 | x |
| 6 | 1 | x |
| 7 | 8 | x |
| 8 | 6 | |

| Đỉnh | Nhãn | Đã lấy |
|---|---|---|
| 0 | 0 | x |
| 1 | 4 | x |
| 2 | 4 | x |
| 3 | 14 -> 7 | |
| 4 | 10 | |
| 5 | 2 | x |
| 6 | 1 | x |
| 7 | 8 | x |
| 8 | 6 -> 2 | |

| Đỉnh | Nhãn | Đã lấy |
|---|---|---|
| 0 | 0 | x |
| 1 | 4 | x |
| 2 | 4 | x |
| 3 | 7 | x |
| 4 | 10 -> 9 | x |
| 5 | 2 | x |
| 6 | 1 | x |
| 7 | 8 | x |
| 8 | 2 | x |



## Nội dung

- *Cây và cây khung của đồ thị*
- *Thuật toán Kruskal*
- *Thuật toán Prim*
- **Thuật toán Dijkstra**

## Đường đi ngắn nhất trong đồ thị

- Xét đồ thị $\boldsymbol{G(V, E)}$.
- Với mỗi cạnh $u, v \in \mathrm{E}$, có một giá trị trọng số $\mathrm{W}[u, v]$.
- Đặt $\mathrm{W}[u, v] = \infty$ nếu $u, v \notin \mathrm{E}$.
- Nếu dãy $v_0, v_1, \dots, v_k$ là một đường đi trên $\boldsymbol{G}$ thì $\sum_{i=1}^{k} W[v_{i-1}, v_i]$ được gọi là độ dài của đường đi.
- Yêu cầu đặt ra là tìm đường đi ngắn nhất từ đỉnh **s** đến đỉnh **t** của đồ thị $\boldsymbol{G}$.

## Thuật toán Dijkstra

- Được đề xuất năm 1959 bởi E. Dijkstra (nhà toán học Hà Lan). Thuật toán tìm đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại được Dijkstra đề nghị áp dụng cho trường hợp đồ thị với trọng số không âm.
- Thuật toán được thực hiện trên cơ sở gán tạm thời cho các đỉnh. Nhãn của mỗi đỉnh cho biết cận trên của độ dài đường đi ngắn nhất tới đỉnh đó. Các nhãn này sẽ được biến đổi (tính lại) nhờ một thủ tục lặp, mà ở mỗi bước lặp một số đỉnh sẽ có nhãn không thay đổi, nhãn đó chính là độ dài đường đi ngắn nhất từ s đến đỉnh đó.

## Giải thuật

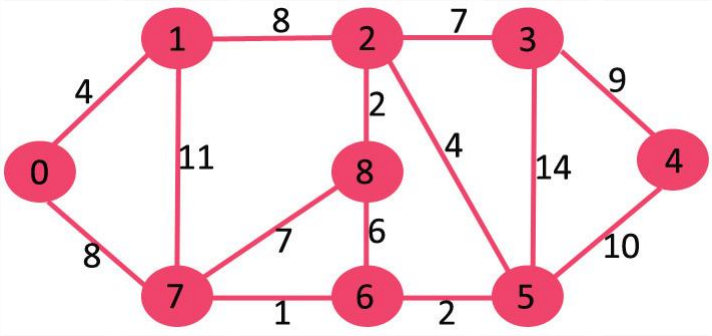
- Các bước thực hiện:
  - Khởi tạo tập V rỗng chứa các đỉnh.
  - Gán giá trị độ dài $d$ cho tất cả các đỉnh của đồ thị. Khởi tạo giá trị $d = INF$ cho tất cả các đỉnh và $d = 0$ cho đỉnh xuất phát.
  - Thực hiện lặp trong khi tập V chưa chứa hết các đỉnh:
    - Lấy 1 đỉnh $u$ chưa có trong V và có giá trị $d$ nhỏ nhất.
    - Đưa $u$ vào V.
    - Cập nhật giá trị $d$ cho tất cả các đỉnh liền kề với $u$ theo nguyên tắc lấy giá trị nhỏ nhất của: $d_{(u)} + w_{(u-v)}$ và $d_{(v)}$ (trong đó: $w_{(u-v)}$ là trọng số của cạnh $u - v$).

## Độ phức tạp của thuật toán

- Độ phức tạp trong trường hợp xấu nhất: $\mathbf{O}(\boldsymbol{V^2})$.
- Nếu sử dụng **Heap** cho tập hợp đỉnh chưa xét thì độ phức tạp đạt được là: $\mathbf{O}((\boldsymbol{V}+\boldsymbol{E})\boldsymbol{logV})$.
- Nếu sử dụng **Fibonacci Heap** cho tập hợp đỉnh chưa xét thì độ phức tạp đạt được là: $\mathbf{O}(\boldsymbol{VlogV}+\boldsymbol{E})$.

## Ví dụ

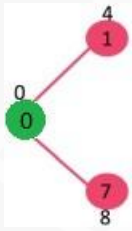
- Tìm đường đi ngắn nhất của đồ thị sau xuất phát từ đỉnh 0:



# Thực hiện

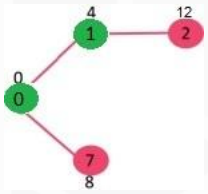
| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | | -1 |
| 1 | INF | | -1 |
| 2 | INF | | -1 |
| 3 | INF | | -1 |
| 4 | INF | | -1 |
| 5 | INF | | -1 |
| 6 | INF | | -1 |
| 7 | INF | | -1 |
| 8 | INF | | -1 |

| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | | 0 |
| 2 | INF | | -1 |
| 3 | INF | | -1 |
| 4 | INF | | -1 |
| 5 | INF | | -1 |
| 6 | INF | | -1 |
| 7 | 8 | | 0 |
| 8 | INF | | -1 |



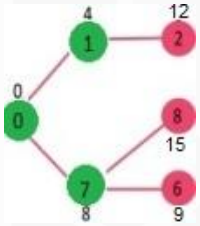

| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | | 0 |
| 2 | INF | | -1 |
| 3 | INF | | -1 |
| 4 | INF | | -1 |
| 5 | INF | | -1 |
| 6 | INF | | -1 |
| 7 | 8 | | 0 |
| 8 | INF | | -1 |

| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | x | 0 |
| 2 | 12 | | 1 |
| 3 | INF | | -1 |
| 4 | INF | | -1 |
| 5 | INF | | -1 |
| 6 | INF | | -1 |
| 7 | 8 | | 0 |
| 8 | INF | | -1 |



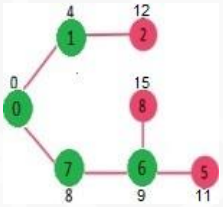
| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | x | 0 |
| 2 | 12 | | 1 |
| 3 | INF | | -1 |
| 4 | INF | | -1 |
| 5 | INF | | -1 |
| 6 | INF | | -1 |
| 7 | 8 | | 0 |
| 8 | INF | | -1 |

| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | x | 0 |
| 2 | 12 | | 1 |
| 3 | INF | | -1 |
| 4 | INF | | -1 |
| 5 | INF | | -1 |
| 6 | 9 | | 7 |
| 7 | 8 | x | 0 |
| 8 | 15 | | 7 |




| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | x | 0 |
| 2 | 12 | | 1 |
| 3 | INF | | -1 |
| 4 | INF | | -1 |
| 5 | INF | | -1 |
| 6 | 9 | | 7 |
| 7 | 8 | x | 0 |
| 8 | 15 | | 7 |

| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | x | 0 |
| 2 | 12 | | 1 |
| 3 | INF | | -1 |
| 4 | INF | | -1 |
| 5 | 11 | | 6 |
| 6 | 9 | x | 7 |
| 7 | 8 | x | 0 |
| 8 | 15 | | 6 |

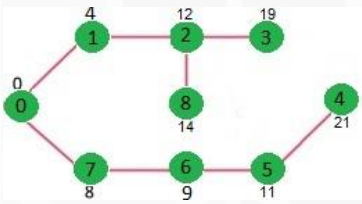


| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | x | 0 |
| 2 | 12 | | 1 |
| 3 | INF | | -1 |
| 4 | 21 | | 5 |
| 5 | 11 | x | 6 |
| 6 | 9 | x | 7 |
| 7 | 8 | x | 0 |
| 8 | 15 | | 6 |

| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | x | 0 |
| 2 | 12 | x | 1 |
| 3 | 19 | | 2 |
| 4 | 21 | | 5 |
| 5 | 11 | x | 6 |
| 6 | 9 | x | 7 |
| 7 | 8 | x | 0 |
| 8 | 14 | | 2 |

| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | x | 0 |
| 2 | 12 | x | 1 |
| 3 | 19 | | 2 |
| 4 | 21 | | 5 |
| 5 | 11 | x | 6 |
| 6 | 9 | x | 7 |
| 7 | 8 | x | 0 |
| 8 | 14 | x | 2 |

| Đỉnh | d | V | Trước |
|---|---|---|---|
| 0 | 0 | x | -1 |
| 1 | 4 | x | 0 |
| 2 | 12 | x | 1 |
| 3 | 19 | x | 2 |
| 4 | 21 | x | 5 |
| 5 | 11 | x | 6 |
| 6 | 9 | x | 7 |
| 7 | 8 | x | 0 |
| 8 | 14 | x | 2 |

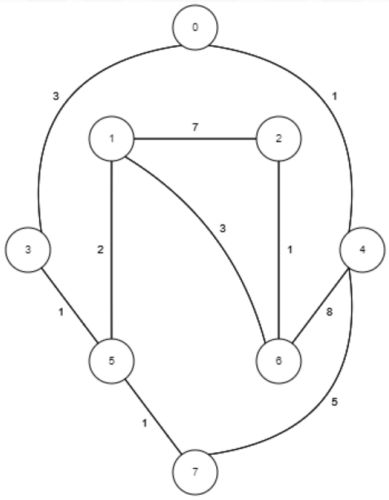


## Nội dung

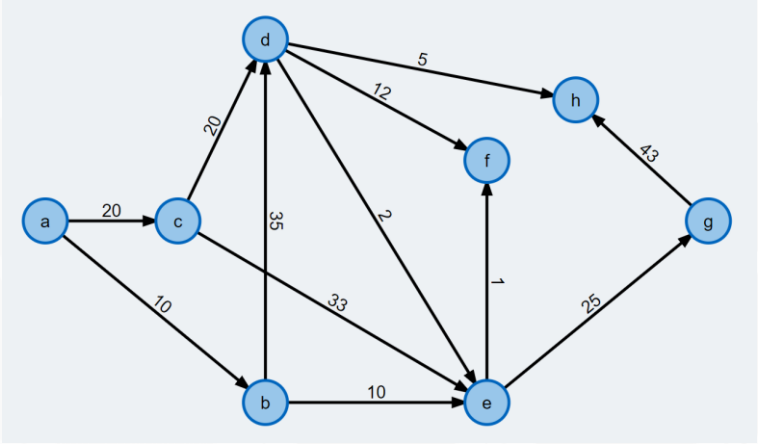
- *Cây và cây khung của đồ thị*
- *Thuật toán Kruskal*
- *Thuật toán Prim*
- *Thuật toán Dijkstra*
- **Bài tập**

## Bài tập

- Tìm cây khung theo thuật toán Kruskal, Prim của đồ thị sau:



❑ Tìm đường đi ngắn nhất xuất phát từ đỉnh a theo thuật toán Dijkstra của đồ thị sau:




❑ Tìm đường đi ngắn nhất xuất phát từ đỉnh a theo thuật toán Dijkstra của đồ thị sau: