***Bài 2:*** *Trình bày Kruskal để giải bài toán tìm cây bao trùm tối thiểu của đồ thị: Ý tưởng phương pháp, lược đồ thuật toán, chứng minh tính đúng; đánh giá độ phức tạp thuật toán.*

* **Thuật toán Kruskal** là một [thuật toán](http://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) trong [lý thuyết đồ thị](http://vi.wikipedia.org/wiki/L%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B) để tìm [cây bao trùm nhỏ nhất](http://vi.wikipedia.org/wiki/C%C3%A2y_bao_tr%C3%B9m_nh%E1%BB%8F_nh%E1%BA%A5t) của một [đồ thị](http://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_th%E1%BB%8B_%28l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B%29) [liên thông](http://vi.wikipedia.org/w/index.php?title=Li%C3%AAn_th%C3%B4ng_%28l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B%29&action=edit&redlink=1) có trọng số. Nói cách khác, nó tìm một tập hợp các [cạnh](http://vi.wikipedia.org/wiki/C%E1%BA%A1nh_%28l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B%29) tạo thành một cây chứa tất cả các [đỉnh](http://vi.wikipedia.org/wiki/%C4%90%E1%BB%89nh_%28l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B%29) của đồ thị và có tổng trọng số các cạnh là nhỏ nhất
* **Ý tưởng phương pháp:**

Giả sử ta cần tìm cây bao trùm nhỏ nhất của đồ thị *G*. Thuật toán bao gồm các bước sau:

* Khởi tạo rừng *F* (tập hợp các [cây](http://vi.wikipedia.org/wiki/C%C3%A2y_%28l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B%29)), trong đó mỗi đỉnh của *G* tạo thành một cây riêng biệt
* Khởi tạo tập *S* chứa tất cả các cạnh của *G*
* Chừng nào *S* còn [khác rỗng](http://vi.wikipedia.org/w/index.php?title=Kh%C3%A1c_r%E1%BB%97ng&action=edit&redlink=1) và *F* gồm hơn một cây
  + Xóa cạnh nhỏ nhất trong *S*
  + Nếu cạnh đó nối hai cây khác nhau trong *F*, thì thêm nó vào *F* và hợp hai cây kề với nó làm một
  + Nếu không thì loại bỏ cạnh đó.

Khi thuật toán kết thúc, rừng chỉ gồm đúng một cây và đó là một cây bao trùm nhỏ nhất của đồ thị *G*.

* **Lược đồ thuật toán:**
* Mã giả:

**Procedure** **Krusal**

**Begin**

;

While() do

**Begin**

Chọn s là cạnh có độ dài nhỏ nhất trong E;

S:=S\{s};

If( không chứa chu trình) then {s};

**End**

if then Đồ thị không liên thông

**End.**

* **Chứng minh tính đúng của thuật toán:**

Chứng minh gồm hai phần: chứng minh kết quả thuật toán là một cây bao trùm và cây bao trùm đó là nhỏ nhất.

* [Cây bao trùm](http://vi.wikipedia.org/wiki/C%C3%A2y_bao_tr%C3%B9m)

*F* luôn là một rừng do việc nối hai cây bằng một cạnh luôn tạo ra một cây mới. Giả thiết phản chứng *F* gồm ít nhất hai cây *A* và *B*. Khi cạnh đầu tiên nối các đỉnh trong *A* của *F* với phần còn lại của đồ thị được xem xét (cạnh này tồn tại do *G* liên thông) thì rõ ràng thuật toán sẽ chọn nó. Vì vậy *A* không thể là một cây trong *F* khi thuật toán kết thúc. Do đó, *F* liên thông và là một cây bao trùm.

* Nhỏ nhất

Ta chứng minh mệnh đề ***P*** sau đây bằng [quy nạp](http://vi.wikipedia.org/w/index.php?title=Quy_n%E1%BA%A1p_to%C3%A1n_h%E1%BB%8Dc&action=edit&redlink=1): Nếu *F* là tập hợp các cạnh đã chọn tại bất kì thời điểm nào trong quá trình thực thi thuật toán thì tồn tại cây bao trùm nhỏ nhất chứa *F*.

* Rõ ràng ***P*** đúng khi thuật toán bắt đầu vì *F* là rỗng.
* Giả sử ***P*** là đúng cho một tập hợp *F* và giả sử *T* là một cây bao trùm nhỏ nhất chứa *F*. Nếu cạnh được thêm vào tiếp theo là *e* cũng nằm trong *T*, thì ***P*** đúng cho *F* + *e*. Nếu không, thì *T* + *e* chứa chu trình *C* và tồn tại cạnh *f* nằm trên *C* nhưng không trong *F*. (Nếu không có cạnh *f*, thì không thể thêm *e* vào *F*, do sẽ tạo ra chu trình *C* trong *F*.) Do đó *T* − *f* + *e* là một cây, và nó có cùng trọng số với *T*, do *T* có trọng số nhỏ nhất và *f* không thể nhỏ hơn *e*, vì nếu không thuật toán đã xem xét *f* trước *e* và chọn *f*. Vì vậy *T* − *f* + *e* là một cây bao trùm nhỏ nhất chứa *F* + *e* và ***P*** là đúng.
* Như vậy, ***P*** đúng khi thuật toán kết thúc và *F* là một cây bao trùm. Điều này chỉ có thể xảy ra nếu *F* là một cây bao trùm nhỏ nhất.
* **Thời gian thực hiện:**

Nếu *E* là số cạnh và *V* là số đỉnh của đồ thị thì thuật toán Kruskal chạy trong thời gian [*O*](http://vi.wikipedia.org/wiki/K%C3%AD_hi%E1%BB%87u_O_l%E1%BB%9Bn)(*E* [log](http://vi.wikipedia.org/wiki/L%C3%B4garit) *V*).

Có thể đạt được thời gian này bằng phương pháp sau: [sắp xếp](http://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_s%E1%BA%AFp_x%E1%BA%BFp) tất cả các cạnh theo trọng số trong thời gian *O*(*E* log *E*). Điều này cho phép thực hiện bước "xóa cạnh nhỏ nhất trong *S*" trong thời gian hằng số. Sau đó sử dụng [cấu trúc dữ liệu cho các tập hợp không giao nhau](http://vi.wikipedia.org/wiki/C%E1%BA%A5u_tr%C3%BAc_d%E1%BB%AF_li%E1%BB%87u_cho_c%C3%A1c_t%E1%BA%ADp_h%E1%BB%A3p_kh%C3%B4ng_giao_nhau) để lưu trữ thông tin đỉnh nào nằm ở cây nào trong *F*. Ta cần thực hiện O(*E*) thao tác, hai thao tác 'tìm' và không quá một thao tác 'hợp' cho mỗi cạnh. Ngay cả những thuật toán đơn giản cho bài toán này, chẳng hạn hợp bằng trọng số cũng có thể thực hiện O(*E*) thao tác trong thời gian *O*(*E* log *V*). Vì vậy tổng thời gian là *O*(*E* log *E*) = *O*(*E* log *V*).

***Bài 3* :** *Đề xuất bài toán người đưa hàng (TSP)*

**Bài toán:**

Một người bán hàng xuất phát từ một thành phố muốn đi qua tất cả các thành phố( n thành phố). Mỗi thành phố chỉ đi qua một lần và quay trở lại thành phố ban đầu. Từ thành phố i sang thành phố j mất chi phí là a[i,j]

Yêu cầu: Hãy tìm hành trình cho người bán hàng sao cho chi phí là thấp nhất nếu có hành trình hoặc thông báo là không tìm được.

**Ý tưởng:**

Từ điểm khởi đầu, ta liệt kê tất cả quãng đường từ điểm xuất phát cho đến n thành phố rồi chọn đi theo con đường ngắn nhất.  
  
Khi đã đi đến một thành phố , chọn đi đến thành phôs kế tiếp cũng theo nguyên tắc trên. Nghĩa là liệt kê tất cả con đường từ thành phố ta đang đứng đến những thành phố chưa đi đến. Chọn con đường ngắn nhất. Lặp lại quá trình này cho đến lúc không còn thành phố nào để đi.

**Độ phức tạp** :