Bài toán Kruskal :

**Ý tưởng:**

Thuật toán Kruskal dựa trên mô hình xây dựng cây khung bằng thuật toán hợp nhất , chỉ có điều thuật toán không phải xét các cạnh với giá trị tùy ý mà xét các cạnh với thứ tự đã được sắp xếp.

**Phân tích :**

Với đồ thị G = (V, E) có n đỉnh . Khởi tạo cây T ban đầu không có cạnh nào . Xét tất cả các cạnh của đồ thị từ cạnh có trọng số nhỏ tới cạnh có trọng số lớn , nếu việc thêm cạnh đó vào T không tạo thành chu trình đơn trong T thì kết nạp thêm cạnh đó vào T . Cứ làm vậy cho tới khi :

1. Hoặc kết nạp được n -1 cạnh vào trong T khi đó T là cây khung nhỏ nhất .
2. Hoặc chưa kết nạp đủ n -1 cạnh nhưng hễ cứ nạp thêm một cạnh bất kỳ trong số các cạnh còn lại thì sẽ tạo thành chu trình đơn . Trong trường hợp này đồ thị không liên thông việc tìm kiếm cây khung nhỏ nhất thất bại.

Như vậy có hai vấn đề quan trọng cần giải quyết là :

Thứ nhất : làm thế nào để xét các cạnh từ cạnh có trọng số nhỏ tới cạnh có trọng số lớn .

Ta có thể thực hiện bằng cách sắp xếp danh sách theo cạnh theo thứ tự không giảm của trọng số , sau đó duyệt từ đầu tới cuối danh sách cạnh . Trong trường hợp tổng quát ta sử dụng thuật giải HEAPSORT là hiệu quả nhất bời vì nó cho chọn lần lượt các cạnh có trọng số nhỏ nhất tới cạnh có trọng số lớn nhất ra khỏi Heap để có thể xử lý (bỏ qua hay thêm vào cây luôn) ngay.

Thứ hai : Làm thế nào kiểm tra xem việc thêm một cạnh có tạo thành chu trình đơn trong T hay không . Để ý rằng các cạnh trong T ở các bước sẽ tạo thành một rừng (đồ thị không có chu trình đơn) . Muốn thêm một cạnh (u,v) vào T mà không tạo thành chu trình đơn thì (u,v ) phải nối hai cây khác nhau của rừng T.

**Chứng minh tính đúng :**

Phương pháp tham lam được gọi là “giải đúng” nếu bài chạy là tối ưu.

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.m.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: 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.

**Code :**

**Đánh giá :**

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

Có thể đạt được thời gian này bằng phương pháp sau: [sắp xếp](http://vi.m.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.m.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*).