**Bài 1**: Thuật giải Prim

* Bước 0: bắt đầu từ một đỉnh u bất kỳ, và gọi u là đỉnh đang xét
* Bước 1: tìm tất cả các đỉnh v kề đỉnh đang xét, cho các cạnh này vào tập cạnh chuẩn bị xét Etemp;
* Bước 2: từ Etemp lấy ra một cạnh e, sao cho:
* ∀ ei ∈ Etemp/{e}, w(e) ≤ w(ei); (w(e) là trọng số của cạnh e)
* Edges(T) ∪ {e} => T không tạo ra chu trình;
* Bước 3: Nếu không lấy được e nào hoặc Vertices(T) = V thì dừng (T là cây khung tối tiểu), ngược lại thì gọi u ∈ e, u, u ∉ Vertices(T) là đỉnh đang xét, quay lại bước 1;

**Bài 2:** Thuật giải Kruskal

* Bước 1: từ E lấy ra một cạnh e, sao cho:
* ∀ ei ∈ E, w(e) ≤ w(ei) (w(e) là trọng số của cạnh e)
* Edges(T) ∪ {e} => T không tạo ra chu trình;
* Bước 2: Nếu không lấy được e nào hoặc V = Vertices(T) thì dừng (T là cây khung tối tiểu), ngược lại thì quay lại bước 1;
* **Bài 3**:
* **1. Thuật toán Kruskal**
* Thuật toán sẽ xây dựng tập cạnh T của cây khung nhỏ nhất H=(V,T) theo từng bước. Trước hết sắp xếp các cạnh của đồ thị G theo thứ tự không giảm của độ dài. Bắt đầu từ tập T=Æ , ở mỗi bước ta sẽ lần lượt duyệt trong danh sách cạnh đã sắp xếp, từ cạnh có độ dài nhỏ đến cạnh có độ dài lớn hơn, để tìm ra cạnh mà việc bổ sung nó vào tập T gồm n-1 cạnh.
* **2. Thuật toán Prim**
* Thuật toán Kruskal làm việc kém hiệu quả với những đồ thị dày (đồ thị với số cạnh m» n(n-1)/2). Trong trường hợp đó thuật toán Prim tỏ ra hiệu quả hơn. Thuật toán Prim còn được gọi là phương pháp lân cận gần nhất. Trong phương pháp này bắt đầu từ một đỉnh tuỳ ý của đồ thị, đầu tiên ta nối s với đỉnh lân cận gần nó nhất, chẳng hạn là đỉnh y. Nghĩa là trong số các cạnh kề của đỉnh s, cạnh (s,y) có độ dài nhỏ nhất. Tiếp theo trong số các cạnh kề với hai đỉnh s hoặc y ta tìm cạnh có độ dài nhỏ nhất, cạnh này dẫn đến đỉnh thứ ba z, và ta thu được cây bộ phận gồm 3 đỉnh và 2 cạnh. Quá trình này sẽ tiếp tục cho đến khi ta thu được cây gồm n đỉnh và n-1 cạnh sẽ chính là cây khung nhỏ nhất cần tìm.