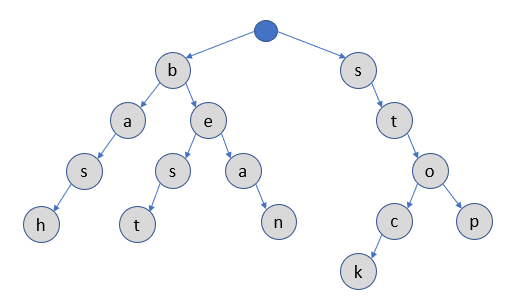
**CẤU TRÚC TRIES**

1. **Cây tiền tố tiêu chuẩn**

Cho ví dụ: ***bash, bean, best, stock, stop***



**Duyệt cây: Cây được duyệt theo thứ tự như duyệt cây nhị phân**

**Thêm: Duyệt tuần tự kí tự sau đó thêm vào khi kí tự của từ ko tồn tại trong nhánh. Vị trí thêm của kí tự dựa vào thứ tự trong bảng chữ cái, chữ đứng trước sẽ thêm vào bên trái node cha, đứng sau thêm bên phải**

Vd: bast

* Xét b -> có trong gốc -> chuyển b làm gốc
* Xét a -> có a chuyển làm gốc
* Xét s -> nhánh hiện tại kết thúc mà ko xuất hiện s -> thêm s làm node lá kế tiếp
* Xét t -> nhánh hiện tại kết thúc mà ko xuất hiện t -> thêm t làm node lá kế tiếp

**Tìm kiếm: Duyệt tuần tự từ trên xuống nếu kí tự trong từ không tím được ở node kế tiếp thông báo từ ko có trong dict**

Vd: stink

* Xét s -> có s -> chuyển qua node s
* Xét t -> có t -> chuyển qua node t
* Xét i -> danh sách node kế tiếp ko có i -> thông báo “stink” ko có trong dict

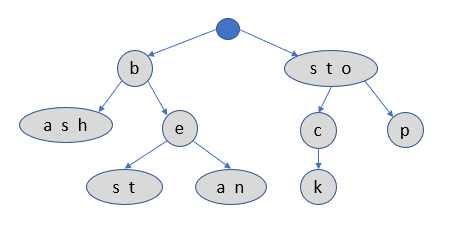
**Xóa: Đầu tiên thực hiện các bước như tìm kiếm, sau khi tới được node lá, lần lượt** **xóa node lá ngược về cho tới khi gặp node cha rẽ nhánh**

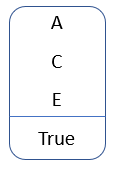
Vd: xóa “stock”

* Tìm kiếm cho ra kết quả có stock và con trỏ đang ở node “k”
* Xóa node “k”, chuyển pointer ngược về node “c”
* Xét xem “c” có phải node rẽ nhánh ko -> nếu không xóa “c”
* Con trỏ quay ngược về “o” -> “o” là node rẽ do danh sách node con của nó chưa trống -> kết thúc quá trình xóa

1. **Cây tiền tố nén**

**Quy tắc nén:** những đoạn chỉ một node đơn không rẽ nhánh có thể gom lại thành cụm

****

1. **Cấu trúc dữ liệu của cây**
2. **Node**

Gồm 2 thành phần

* List<children>: Chứa các node con tiếp theo
* Bool EndOfBranch: Cho biết nhánh kết thúc hay chưa

1. **Ứng dụng**
2. **Auto-correction**
3. **Text recommending**

Trong cây có caps thì cap append như nào ?

* Append “cap” thì tới node p biến bool sẽ từ False -> True
* Nếu tìm kiếm “cap” khi chưa append đến node p sẽ kiểm tra bool = False
* Nếu đã append node p kiểm sẽ cho biết bool = True