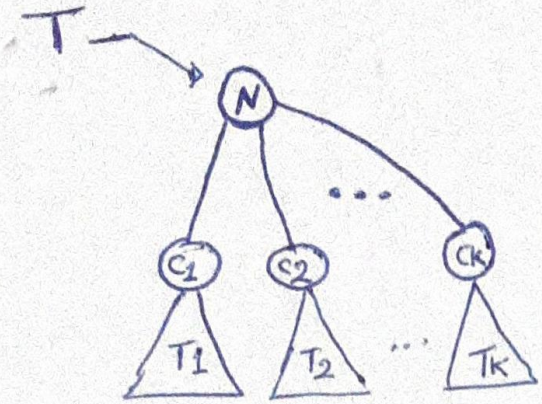
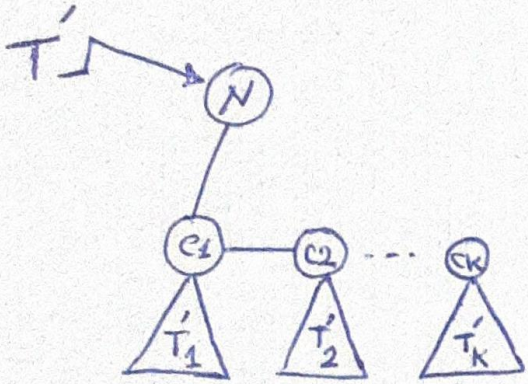
سمودانیات پیش ترتیب



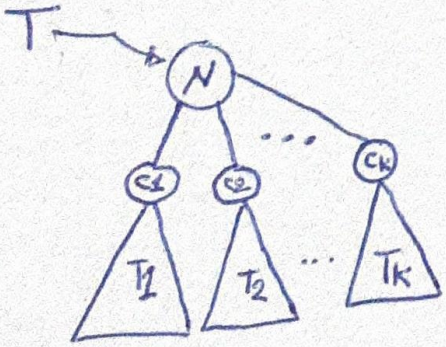
Preorder(T) = $N C_1 T_1 C_2 T_2 \cdots C_k T_k$



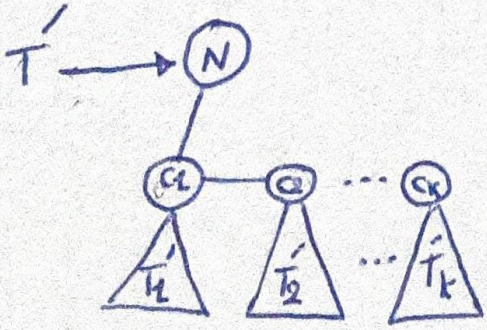
Preorder(T') = $N C_1 T'_1 C_2 T'_2 \cdots C_k T'_k$

واضح است که جایگاه نودهای ملاقات شده در پیمایش پیش ترتیب آو T' یکسان است.

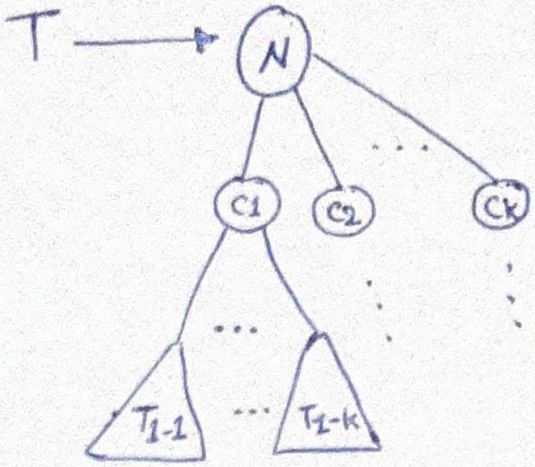
می‌توان از ایده بالا برای رد یکسان بودن پیمایش میان ترتیب و پس ترتیب درخت آو T' نیز استفاده کرد:



Postorder(T') = $T_1 C_1 T_2 C_2 \cdots T_k C_k N$

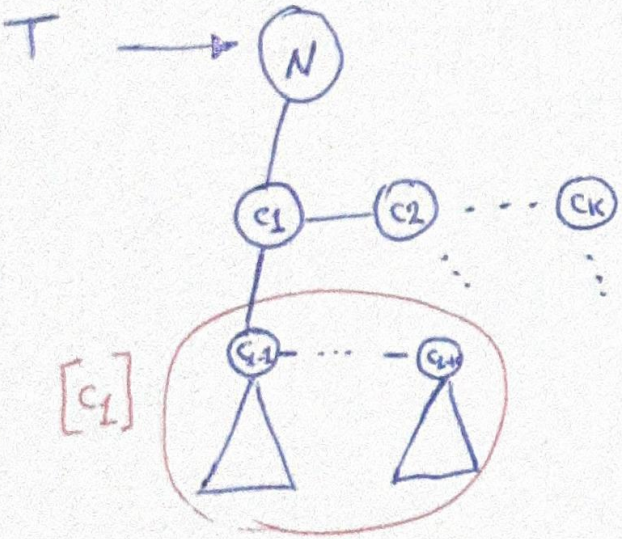


Postorder(T') = $T'_1 T'_2 \cdots T'_k C_k \cdots C_2 C_1 N$



$$Inorder(T) = T_{1-1}\, c_1\, T_{1-2} \cdots T_{1-k}\, N$$

$$T_{2-1}\, c_2\, T_{2-2} \cdots T_{2-k} \cdots T_{k-1}\, c_k\, T_{k-2} \cdots T_{k-k}$$



$$Inorder(T') = [c_1]\, c_1\, [c_2]\, c_2 \cdots [c_k]\, c_k\, N$$

IV

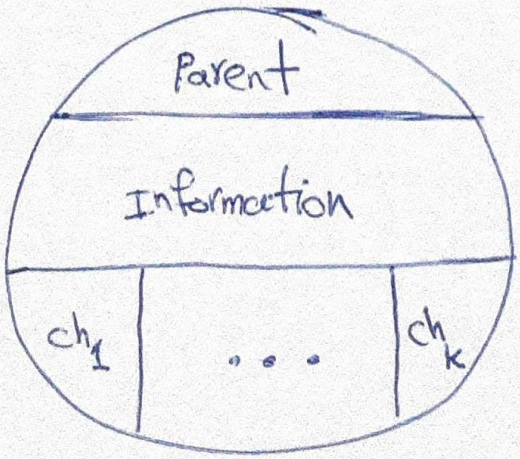
درخت ترای (Trie):

همانطور که در بخش‌های قبلی گفتیم، درخت k-تایی بطور بالقوه‌ای پتانسیل هدررفت حافظه را دارا می‌باشد. با این حال، این داده ساختار برای برخی کاربردها مناسب بوده و بطور گسترده‌ای از آنها استفاده می‌شود. در ادامه سعی داریم، درخت ترای را به عنوان یک درخت k-تایی معرفی کنیم.

درخت ترای که با نام‌های رشته‌ای معمول درخت دیجیتال و درخت پیشوندی نیز شناخته می‌شود، نوعی درخت k تایی هست که برای بازیابی کلیدهای مشخص در داخل یک مجموعه مورد استفاده قرار می‌گیرد.

کلیدها در این ساختار بیشتر از نوع رشته‌ای هستند و به طور کامل در گره‌ها نگهداری نمی‌شوند بلکه با پیمایش درخت و جابجایی بین گره‌ها از مسیر یال بین آنها، کاراکترهای تشکیل دهنده کلید به مرور مشخص می‌شوند.

نمایش گرافیکی گره‌های درخت ترای در حالت کلی به صورت زیر است:

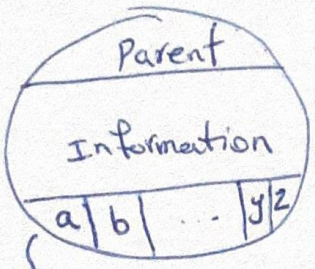


اشاره‌گرهای Parent و ch_1 تا ch_K شبیه به نودهای دیگر درخت در حالت کلی می‌باشد. با اینحال بخش Information می‌تواند شامل هر نوع اطلاعات مفیدی متناسب با کاربرد داده ساختار باشد، مثل: تعداد تکرار رشته‌ها در هر نود، معنای کلید یا توضیحات در رابطه با آن و ... .

(رشته‌های موجود در)

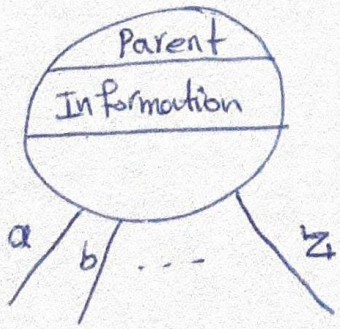
مثال: درخت trie برای کلیدهای book، ball، thin، the، tester، good را رسم کنید.

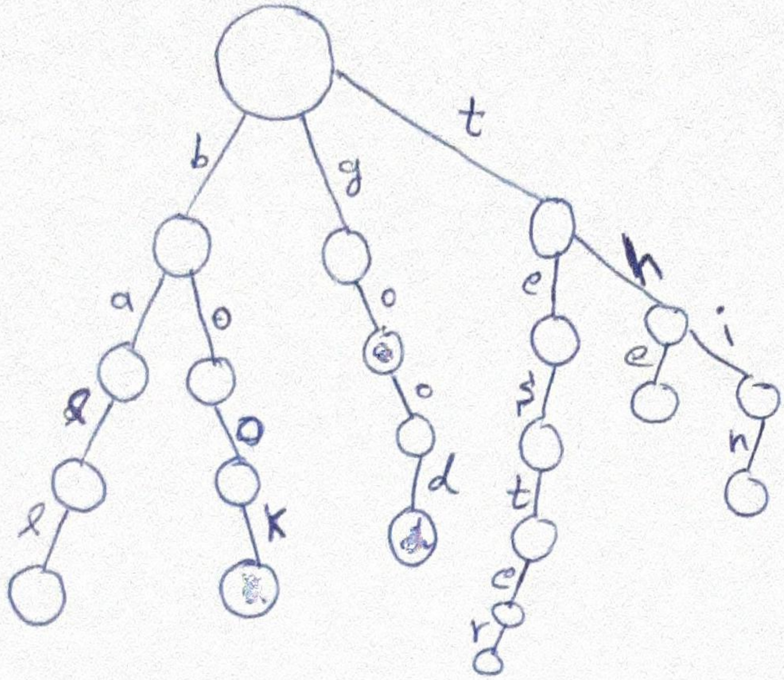
برای این ترسیم نودهای درخت را به صورت زیر در نظر می‌گیریم:



که به ترتیب بچه‌ها از چپ به راست با حروف لاتین برچسب می‌خورند. در واقع اولین اشاره‌گر از راست نود که برای حرف a است، همین طور الی آخر.

برای راحتی کار و ساده‌تر در ترسیم روی هر یال درخت، حرف مربوطه نوشته می‌شود، بنابراین هر نود درخت به صورت زیر در می‌آید:





با توجه به مثال بالا، پر واضح است که داده ساختار درخت ترای، گزینه مناسبی برای ساخت یک دیکشنری است. همچنین از آنجایی که در پیمایش درخت در هر نود، پیشوند کلیدها را داریم می‌تواند گزینه مناسبی برای تکمیل خودکار (Auto complete) در زبان‌های برنامه‌نویسی یا جست و جوی گوگل (موتورهای جست و جو) باشد.

در ادامه، مثال قبل را برای پشتیبانی از دیکشنری با استفاده از درخت ترای کامل می‌کنیم.

برای قسمت اطلاعات (Information) هر نود می‌توان داده‌های زیر را در نظر گرفت:

۱. متغیر flag؛ که نشان دهد مسیر پیموده شده تا نود کنونی یک واژه معتبر است یا خیر،

۲. معنی کلمه نهایی،

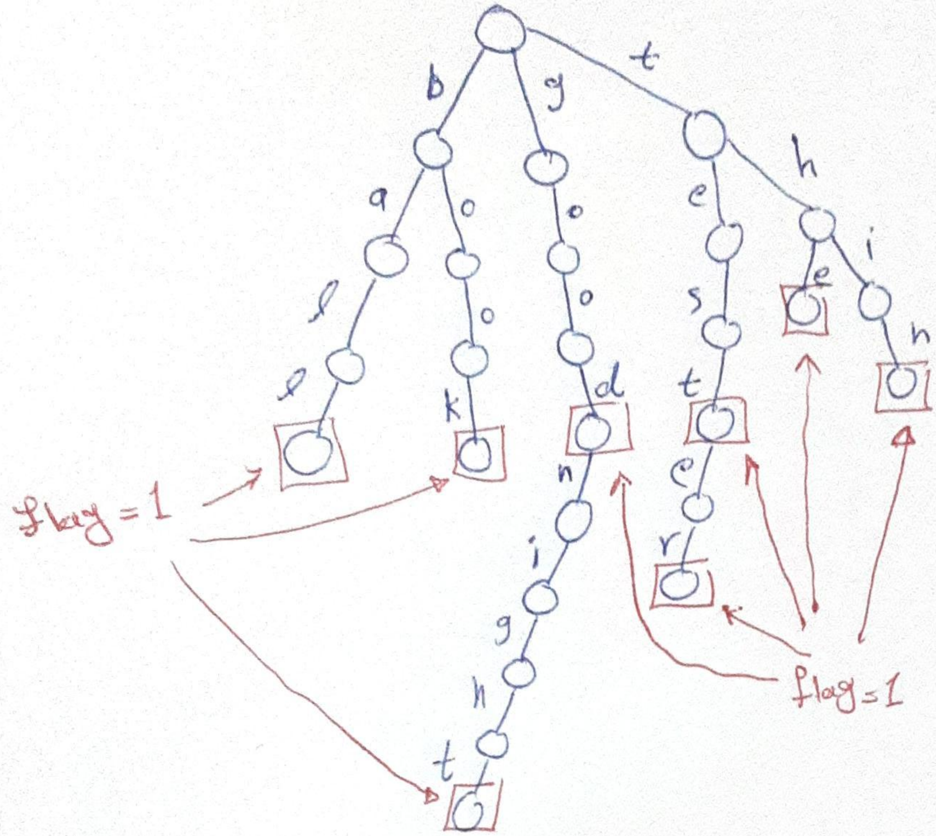
۳. تعداد کلماتی که حاوی پیشوند پیمایش شده است،

۴. کلمات مترادف؛

۵. حتی پر یا بودن کنونی نودها – تعداد جستجوهای کلید مورد نظر

۶. و ...

مثال: برای سادگی تنها از اعلام flag استفاده کرده و با اضافه کردن کلیدهای test ، goodnight به مثال قبل، درخت ترای آن را باز ترسیم می‌کنیم.



● flag مابقی نودها صفر است (flag = 0).

عملیات روی درخت ترای:

۱- عمل جستجو: برای جستجوی یک کلید در درخت ترای نیاز است درحداکثر به تعداد کاراکترهای آن رشته، درخت را پیمایش کنیم و بنابراین پیچیدگی جستجو از مرتبه $O(m)$ است (جایی که m تعداد کاراکترهای رشته مورد نظر است).

۲- عمل درج: برای عمل درج یک کلید می‌بایست گام‌های زیر را دنبال کنیم:

- از ریشه شروع به پیمایش کرده و اگر کلید مورد نظر در پیمایش روشن شد، دو حالت پیش می‌آید: