

Suffix Tree

داده ساختار درخت پسوندی

معرفی اجمالی داده ساختار درخت پسوندی

درخت پسوندی یک ساختمان داده است برای به نمایش درآوردن ساختار درونی یک رشته کاراکتری.

منظور از پسوند های یک رشته چیست؟

"banana \ 0"

"anana \ 0"

"nana \ 0"

"ana \ 0"

"na \ 0"

"a \ 0"

"\ 0"

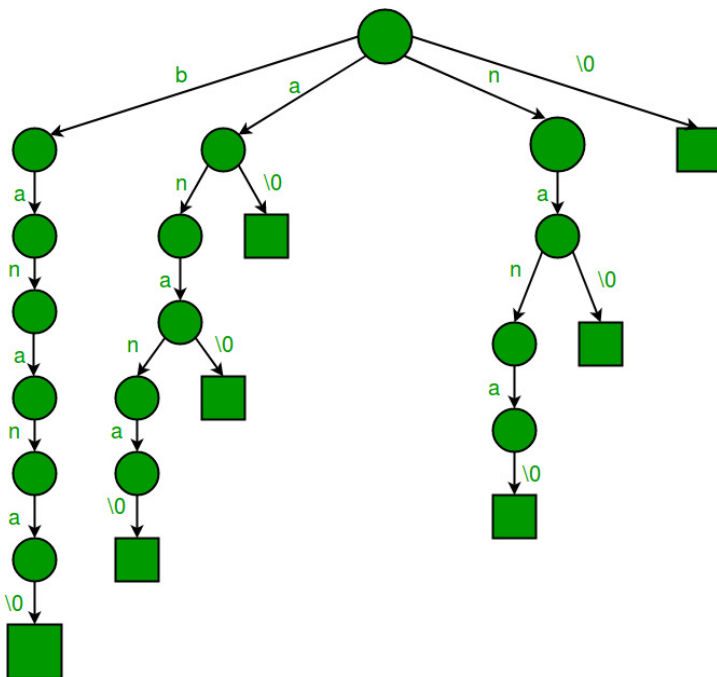
به عنوان مثال کلمه banana را اگر در نظر بگیریم،

m=۷

Trie

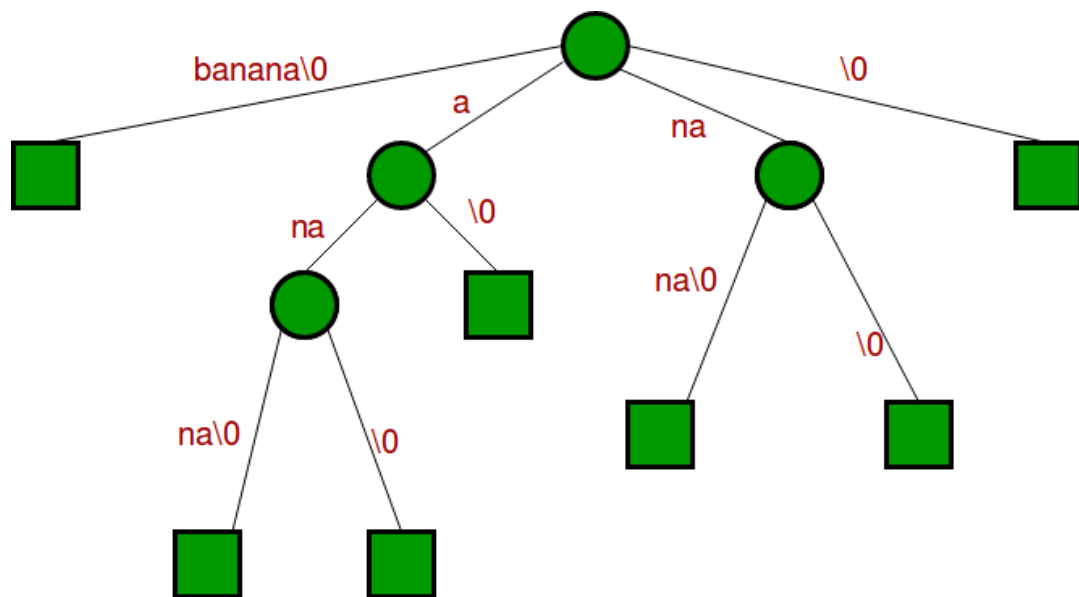
Suffix tree در اصل یک trie فشرده شده است.

Trie چیست؟ ساختار داده ای برای ذخیره سازی مجموعه از رشته ها.


$$\{ "banana \setminus 0", "anana \setminus 0", "nana \setminus 0", "ana \setminus 0", "na \setminus 0", "a \setminus 0", "" \setminus 0" \}$$

تبدیل suffix tree به trie

همونطور که گفتیم suffix tree به بیان ساده یک trie فشرده سازی شده است.



تعریف دقیق داده ساختار suffix tree

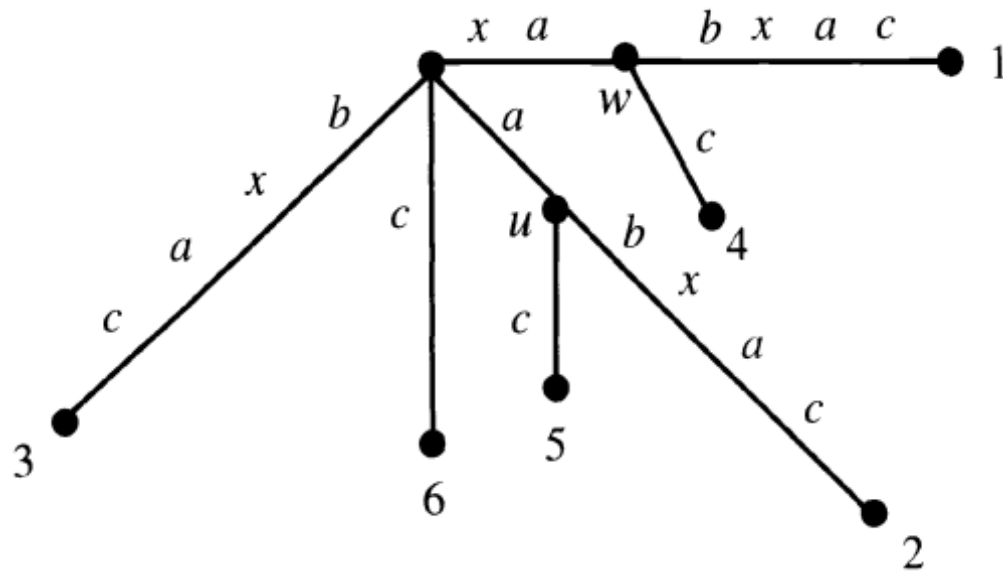
یک درخت پسوندی T برای یک رشته m کاراکتری S یک درخت ریشه دار با دقیقاً m برگ هست که از ۱ تا m شماره گذاری شده اند.

برای هر گره داخلی دارای حداقل دو گره فرزند است که یال منتهی به این گره ها با یک زیر رشته از S برچسب گذاری شده اند. برچسب هیچ دو یال از هر گره نباید با کاراکتر یکسان آغاز شود.

ویژگی اساسی suffix tree این هست که اگر همه کاراکتر ها از ریشه تا یک برگ کنار هم چیده بشن یکی از پسوند ها را تشکیل میدهند.

اهمیت کاراکتر termination

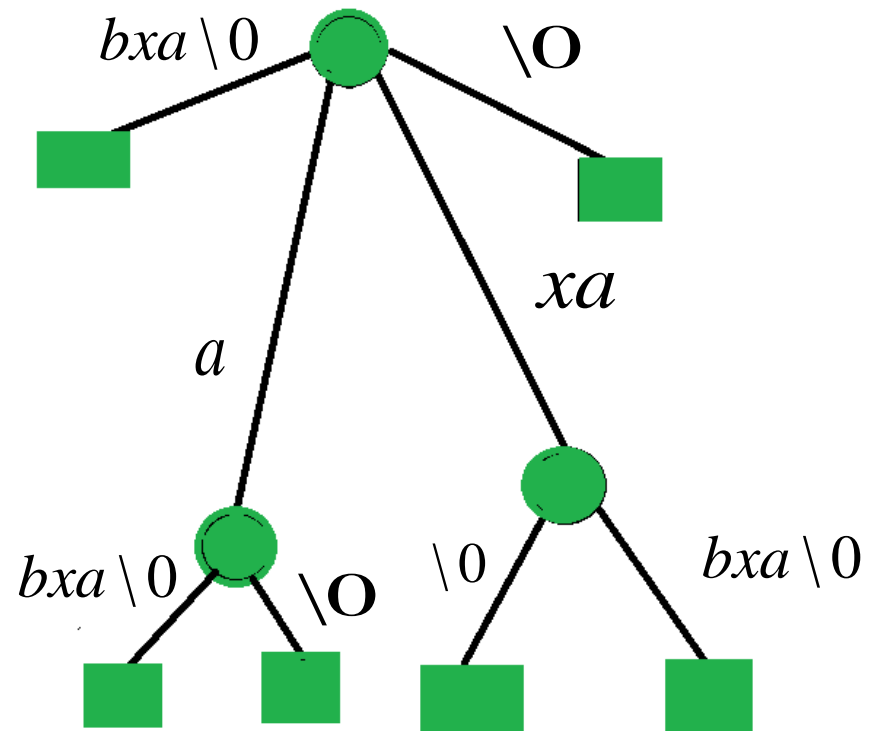
رشته $xabxac$ را در نظر بگیرید



اما اگر رشته $xabxa$ را در نظر بگیریم

اهمیت کاراکتر termination

حال اگر یک کاراکتر termination به استرینگ $xabxa$ اضافه کنیم این مشکل رفع میشود.



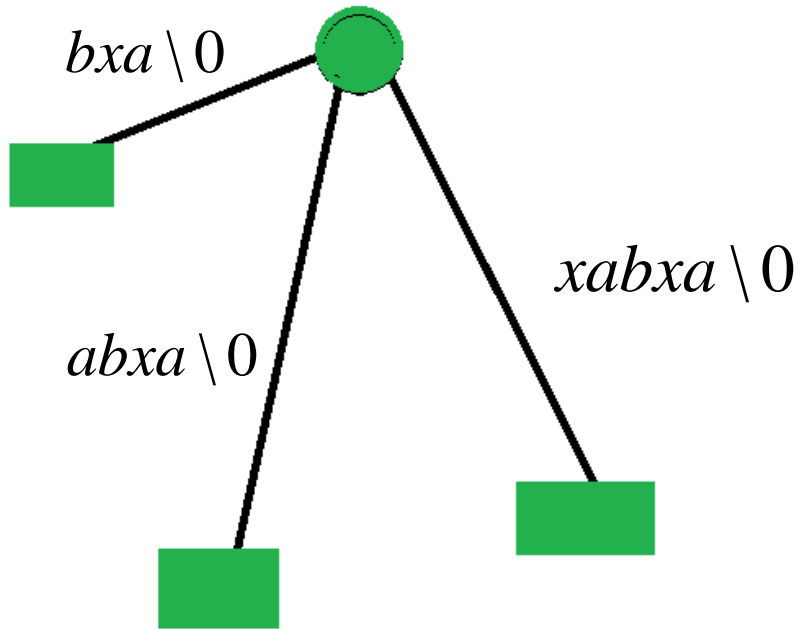
ساخت suffix tree در مرتبه زمانی خطی

از Ukkonen's algorithm برای ساخت یک suffix tree استفاده میکنیم.

مفهوم implicit suffix tree (درخت پسوندی مجازی)

* هیچ یالی با کاراکتر نال وجود ندارد

* هیچ گره داخلی با یک یال متصل وجود ندارد



توضیح کلی Ukkonen's algorithm

این الگوریتم درخت پسوندی مجازی T_i را برای زیر رشته $S[1, \dots, i]$ میسازد.

و T_{i+1} از T_i ساخته میشود و نهایتاً درخت پسوندی اصلی از T_m و با اضافه کردن کاراکتر نال به S و دوباره ساختن درخت پسوندی مجازی نهایتاً درخت پسوندی اصلی را تولید میکند.

در افزونه j ام از فاز $i + 1$ ام. اول انتهای مسیر برچسب گذاری شده با $S[j, \dots, i]$

و در ادامه کاراکتر $S(i + 1)$ را به این برچسب اضافه میکند.

Construct tree \mathcal{T}_1 .

For i from 1 to $m - 1$ do

begin {phase $i + 1$ }

For j from 1 to $i + 1$

begin {extension j }

Find the end of the path from the root labeled $S[j..i]$ in the current tree. If needed, extend that path by adding character $S(i + 1)$, thus assuring that string $S[j..i + 1]$ is in the tree.

end;

end;

به طور خلاصه داریم: