پروژه ساختمان داده ها و الگوریتم

ترم پاییزه 1400-1401

Suffix Tree

داده ساختار درخت پسوندی

تاریخچه

Norbert weiner → Donald Knuth → Ukkonen

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؟

به عنوان مثال کلمه banana را اگر در نظر بگیریم،

m=۷

"$banana\backslash 0$"

"$anana\backslash 0$"

"$nana\backslash 0$"

"$ana\backslash 0$"

"$na\backslash 0$"

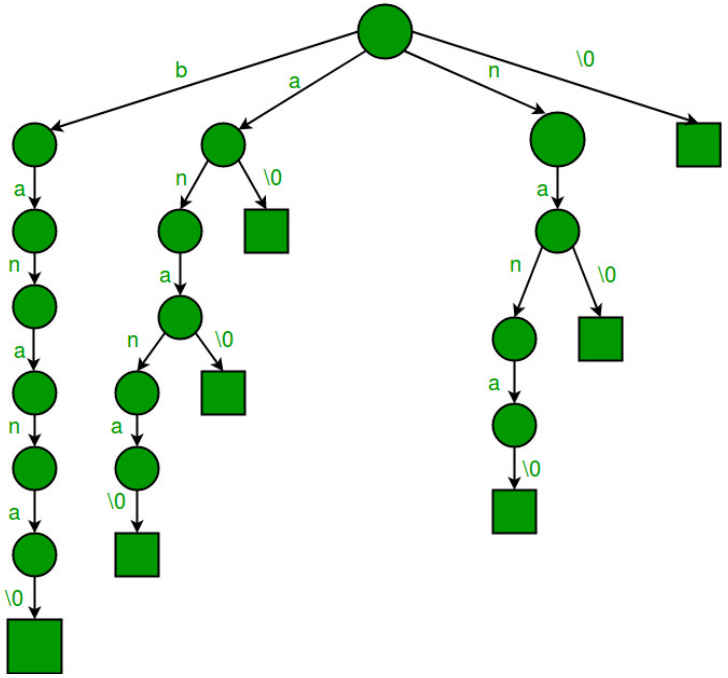
"$a\backslash 0$"

"$\backslash 0$"

Trie

Suffix tree در اصل یک trie فشرده شده است.

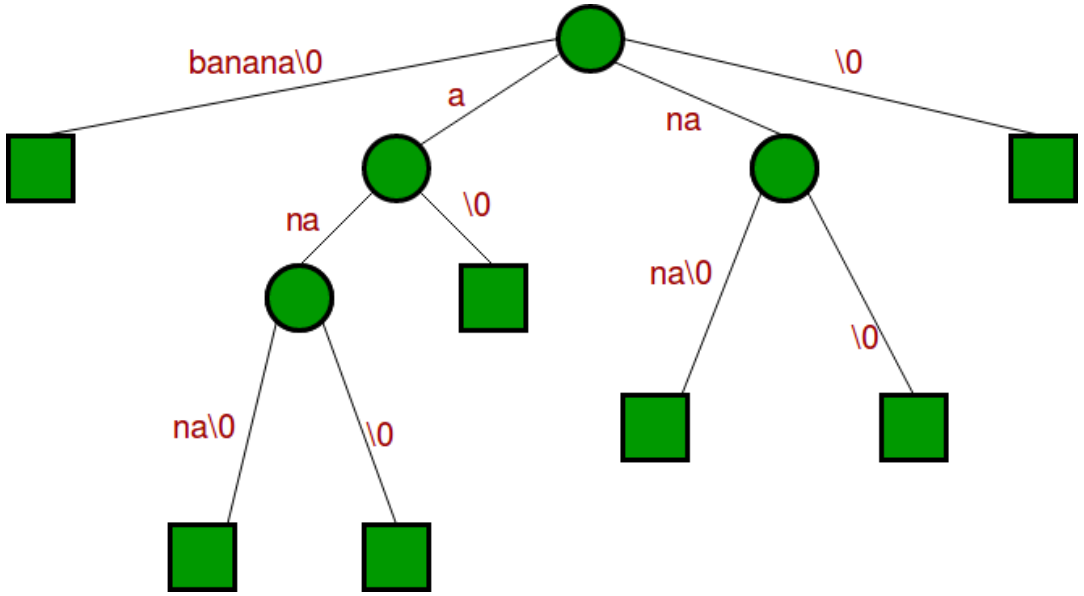
Trie چیست؟ ساختار داده ای برای ذخیره سازی مجموعه از رشته ها.



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

تبدیل trie به suffix tree

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 دقیقا m برگ هست که از ۱ تا m شماره گذاری شده اند.

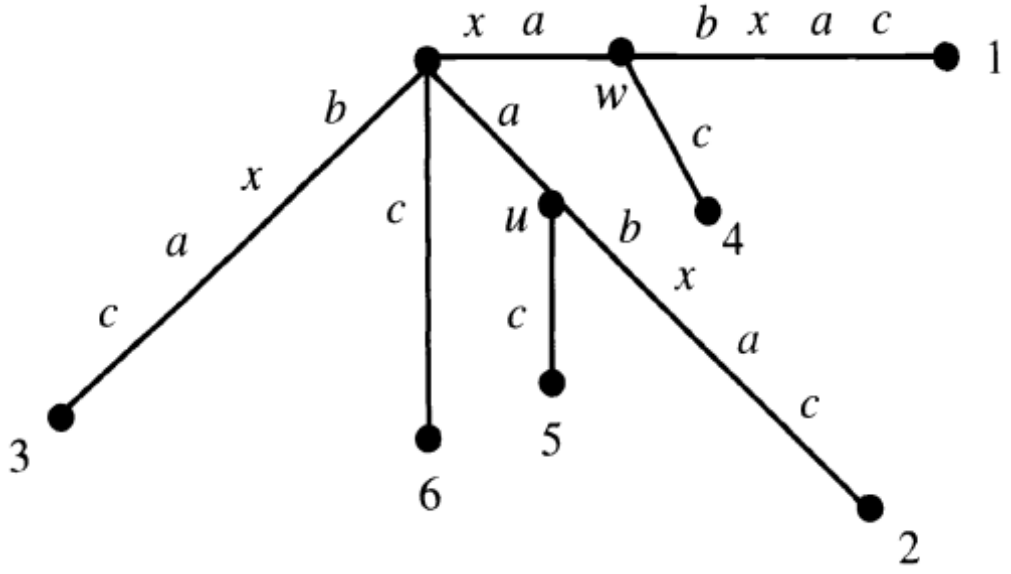
برای هر گره داخلی دارای حداقل دو گره فرزند است که یال منتهی به این گره ها با یک زیر رشته از S برچسب گذاری شده اند.

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

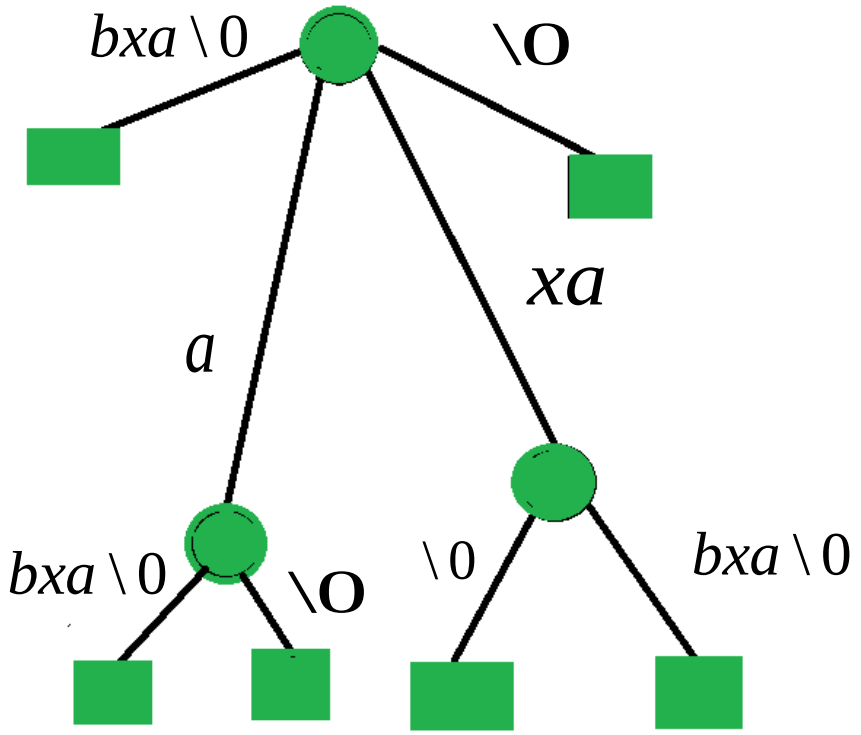
رشته xabxac را درنظر بگیرید



اما اگر رشته xabxa را در نظر بگیریم

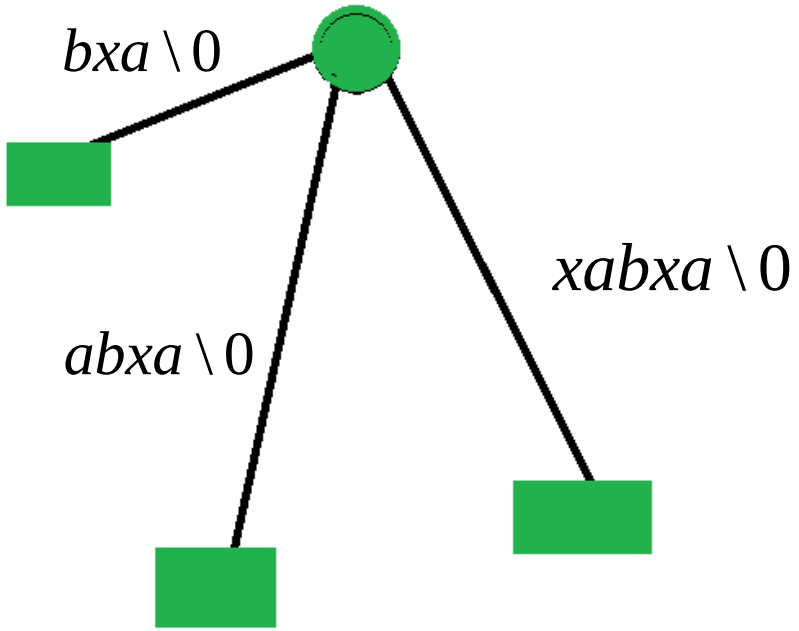
اهمیت کاراکتر termination

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 $S[1,...,i]$ میسازد.

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

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

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

به طور خلاصه داریم:

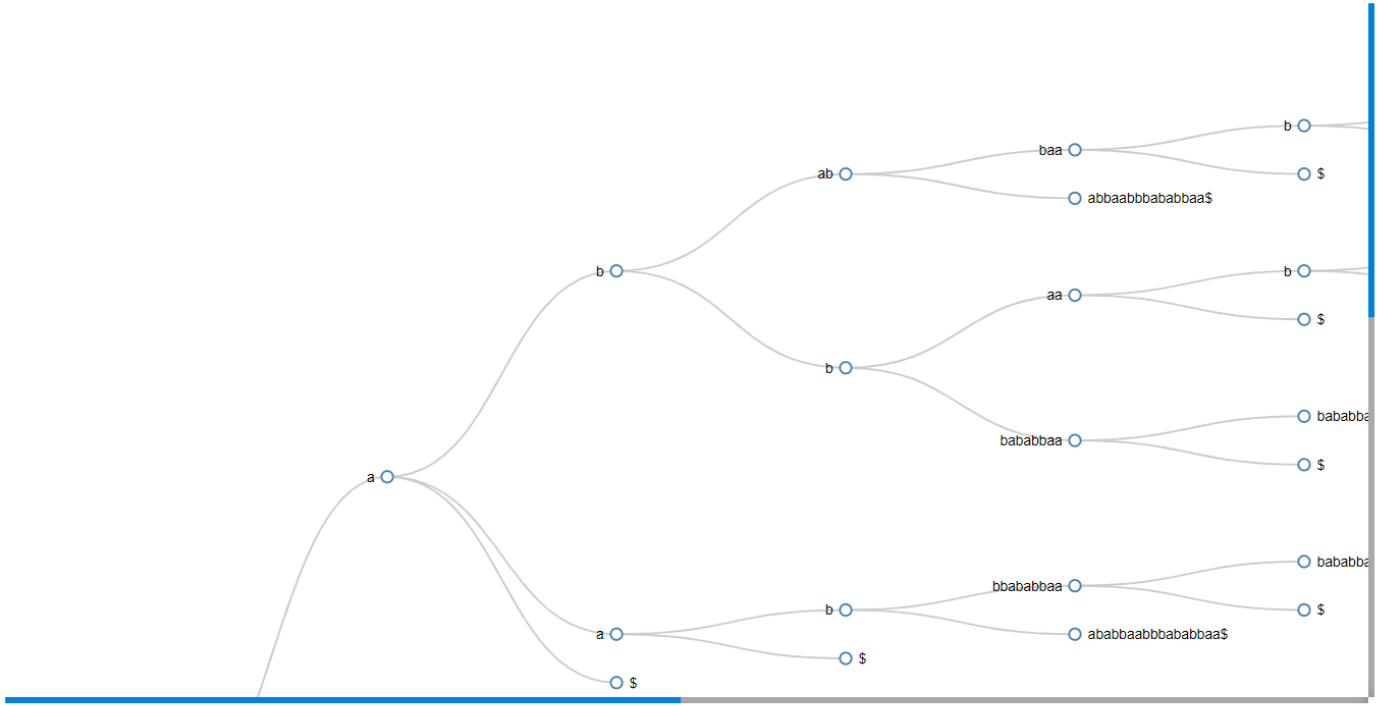
```
Construct tree I_1.
For i from 1 to m − 1 do
begin {phase i + 1}
    For j from 1 to i + 1
        begin {extension j}
        Find the end of the path from the root labeled S[j..i] in the
        current tree. If needed, extend that path by adding character S(i + 1),
        thus assuring that string S[j..i + 1] is in the tree.
        end;
end;
```

Online Suffix tree Visualizer

Builder and visualizer for Suffix Tree data structure developed as the final project of data structures and algorithms course at Shahid Beheshti University on fall 2021

contributors:

- Zahra Eftekhari
- Muhammad Karbalaee



https://zamgo.github.io/suffix-tree/website/visualizer/

پیاده سازی درخت پسوندی در C++

بخشی از کد

```
using namespace std;

int main(int argc, char *argv[])
{
    printf("what is your string?\n");
    scanf("%s",text);
    printf("---------\n");
    buildSuffixTree();
    printf("--------------\nTotal number of nodes %d\n",count);
    return 0;
}
```

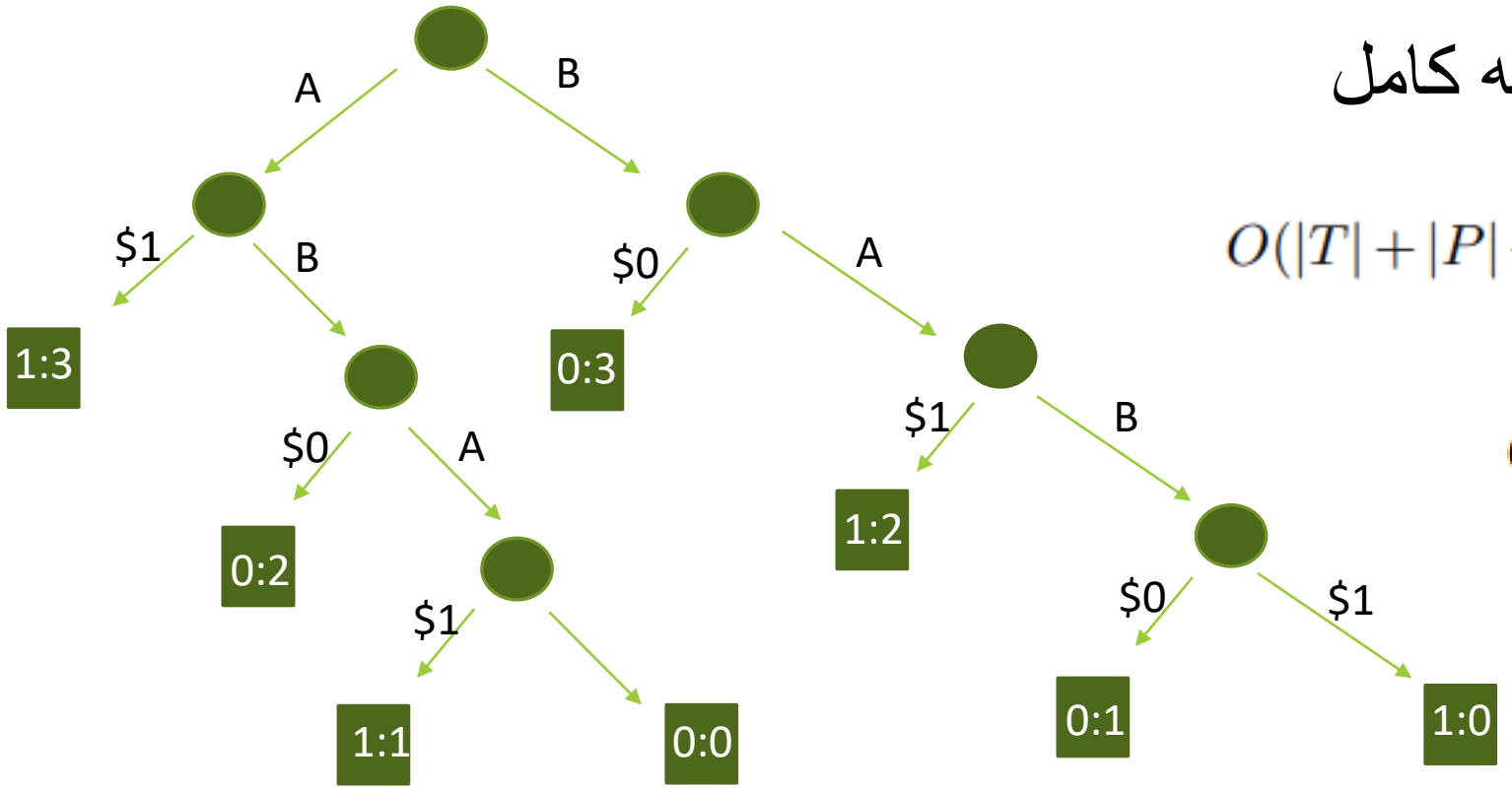
نمونه خروجی

```
what is your string?
zamgo
---------
amgo [1]
go [3]
mgo [2]
o [4]
zamgo [0]
--------------
Total number of nodes 6
```

تحلیل مرتبه زمانی اعمال در ساختمان داده درخت پسوندی

operation	Time complexity
Insert	O(n)
search	O(m)
delete	O(n)
build	O(n)

کاربرد ها و مرتبه زمانی



مسئله1: تطابق رشته کامل

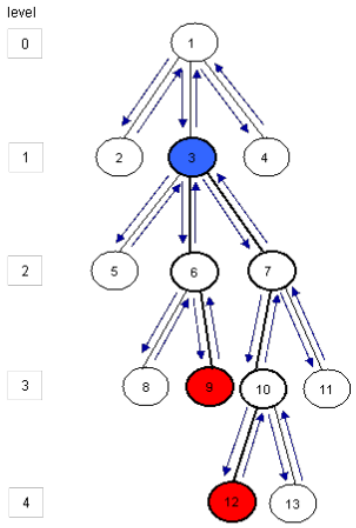
پیدا کردن همه جواب ها: $O(|T|+|P|+k)$

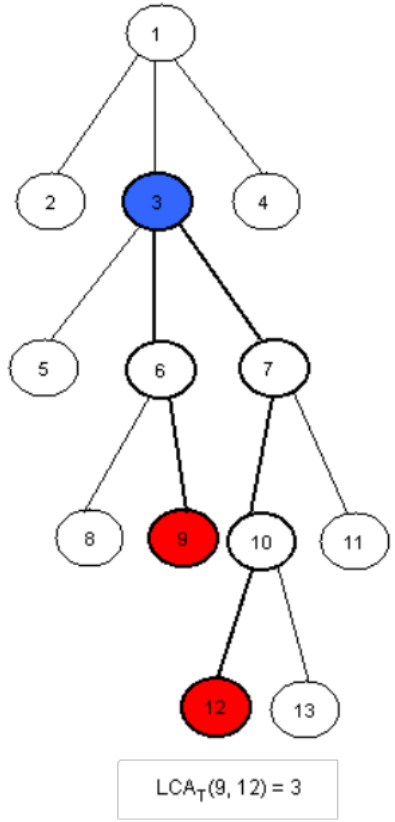
$O(|T|+\sum_{i=1}^{m}|P_i|+|k_i|)$

طولانی ترین زیر رشته مشترک

ساخت، رنگ و پیمایش: $O(|S_1| + |S_2|)$

مسئله پایین ترین جد مشترک





پرسش کمینه باز

منابع

https://fa.wikipedia.org
https://blog.faradars.org
https://en.wikipedia.org
https://www.geeksforgeeks.org
http://ce.sharif.edu

Algorithms on Strings, Trees and Sequences_ Computer Science and Computational Biology
By Dan Gusfield