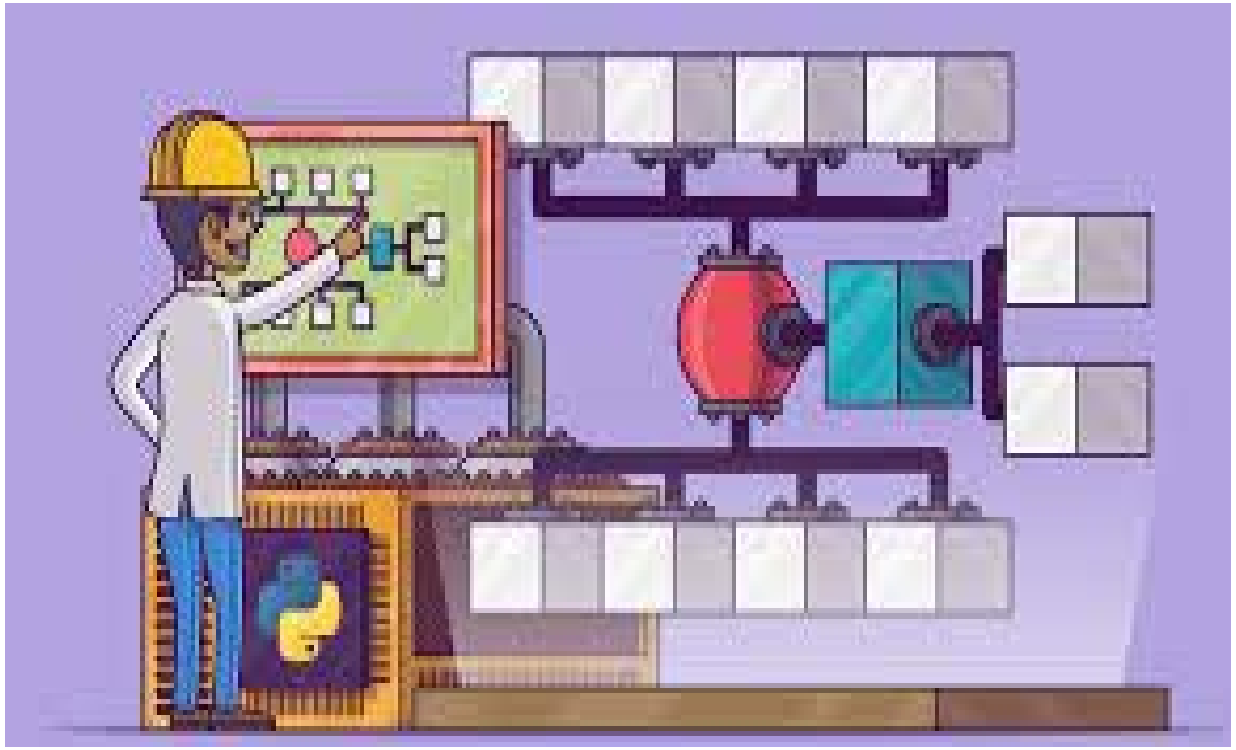
ساختمان داده ها



مدرس:
سمانه حسینی سمنانی

دانشگاه صنعتی اصفهان- دانشکده برق و کامپیوتر

تعداد درخت های متمایز دودویی

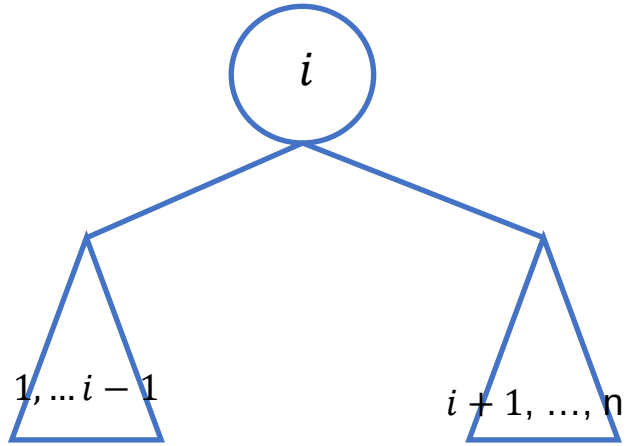
- تعداد درخت های دودویی متمایز چند تا است؟

1, 2, …., n

$$T_n = \sum_{i=1}^{n} T_{i-1} \times T_{n-i}$$

- Catalan number

$$\frac{1}{n+1}\binom{2n}{n}$$

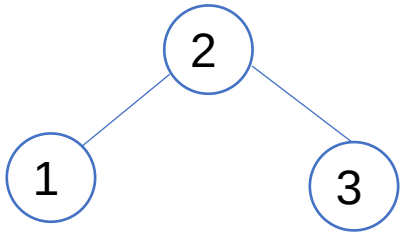


تعداد درخت های متمایز دودویی

- به ازای هر جایگشتی یک درخت یکتا وجود دارد
- n! : تعداد جایگشت های مختلف n عدد
- تعداد درخت های دودویی متمایز
- بعضی درخت ها تکراری هستند.

1, 2, …., n

$$\frac{1}{n+1}\binom{2n}{n} < n!$$



- 2 ,1, 3
- 2, 3, 1

میانگین ارتفاع درخت های ممکن

$\sigma_1 \rightarrow h_1$

$\sigma_2 \rightarrow h_2$

$\dots$

$\sigma_{n!} \rightarrow h_{n!}$

$$x_n = \frac{1}{n!}\sum_{i=1}^{n!} h_i = O(logn)$$

حداکثر ارتفاع: $n-1$

حداقل ارتفاع: $\log n$

میانگین ارتفاع؟

سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان

درخت عبارت

عبارت میانوندی

- عبارت میانوندی

$$a + (b * c)/d - e$$

- اولویت عملگرها:
  - توان، ضرب و تقسیم، جمع و تفریق
- پرانتز گذاری برای رفع ابهام

$$((a + ((b * c)/d)) - e)$$

تعریف بازگشتی عبارت میانوندی

$$E \rightarrow operand$$

$$E \rightarrow (E\ \alpha\ E)$$

$$\alpha: + \mid * \mid - \mid \% \mid / \mid \%\mid \hat{}, \ldots.$$

$$E \rightarrow (\beta\ E)$$

$$\beta : \sim \mid \sin \mid \cos \mid , \ldots.$$

مثال $((a + ((b * c)/d)) - e)$

سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان

تعریف بازگشتی عبارت پسوندی

$$E \rightarrow operand$$

$$E \rightarrow E\ E\ \alpha$$

$$\alpha: + \mid * \mid - \mid \% \mid / \mid \%\mid \hat{}, \ldots.$$

$$E \rightarrow E\beta$$

$$\beta : \sim \mid \sin \mid \cos \mid , \ldots.$$

$((a + ((b * c)/d)) - e)$ تبدیل به پسوندی $abc * d/+e-$

سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان

تعریف بازگشتی عبارت پیشوندی

$$E \rightarrow operand$$

$$E \rightarrow \alpha\ E\ E$$

$$\alpha: + \mid * \mid - \mid \% \mid / \mid \%\mid \hat{}, \ldots.$$

$$E \rightarrow \beta E$$

$$\beta : \sim \mid \sin \mid \cos \mid , \ldots.$$

$((a + ((b * c)/d)) - e)$ ← تبدیل به پیشوندی ← $- +a/* bcde$

سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان

عبارت های پیشوندی و پسوندی

مزیت نسبت به میانوندی:

عدم وجود ابهام بدون استفاده از پرانتز

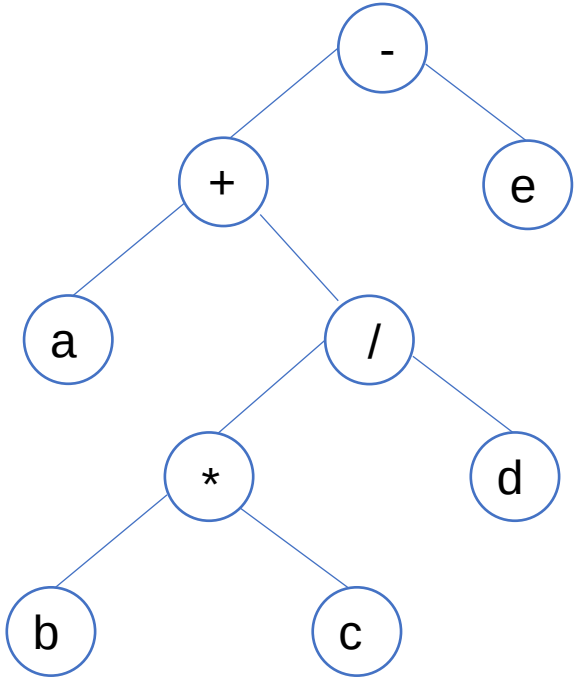
عبارت های پیشوندی و پسوندی

روش تبدیل میانوندی به پیشوندی و پسوندی:

ساخت درخت عبارت و پیمایش postorder یا preorder آن

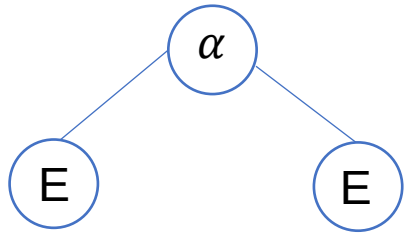
درخت عبارت

$((a + ((b * c)/d)) - e)$

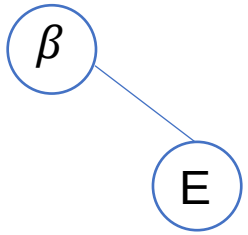


نحوه ساخت درخت

$E \rightarrow (E\ \alpha\ E)$

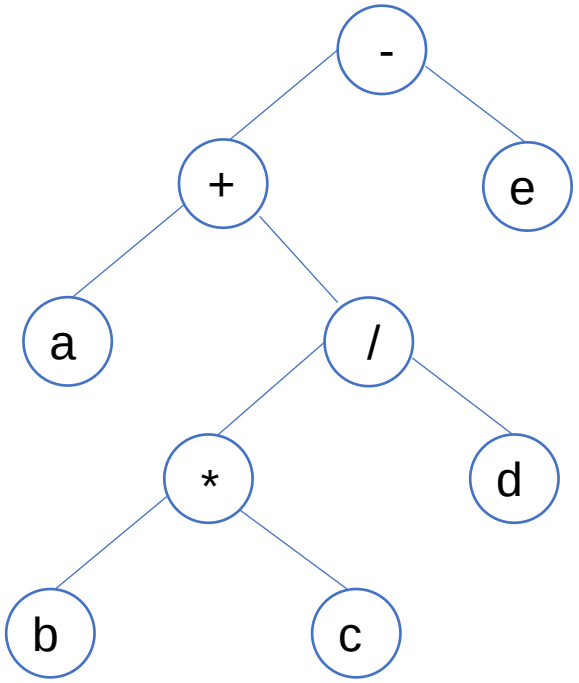


$E \rightarrow (\beta\ E)$



$operand$

کاربرد درخت عبارت

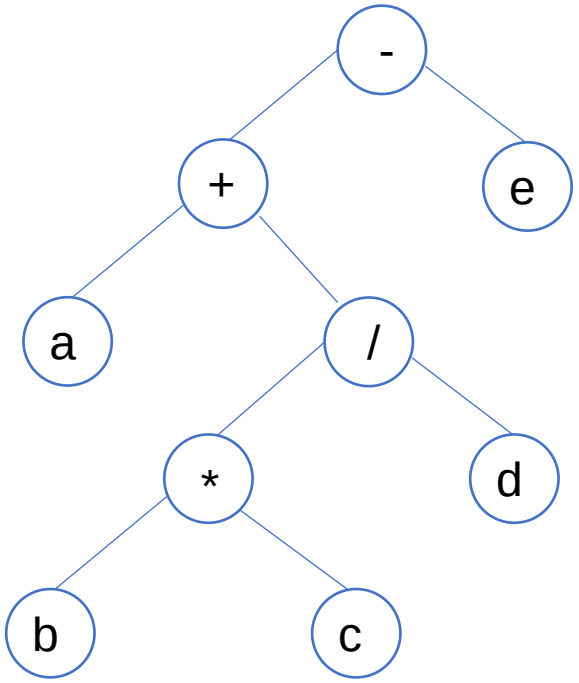


preorder tree walk — نمایش پیشوندی

inorder tree walk — نمایش میانوندی

postorder tree walk — نمایش پسوندی

Inorder tree walk




```
INORDER-TREE-WALK(x)
1  if x ≠ NIL
2      INORDER-TREE-WALK(x.left)
3      print x.key
4      INORDER-TREE-WALK(x.right)
```

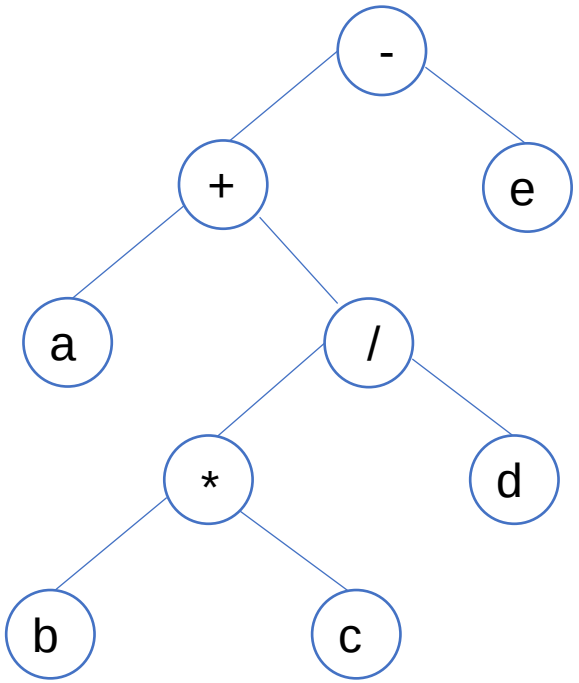
الگوریتم تبدیل عبارت میانوندی کامل به درخت عبارت

- بازگشتی

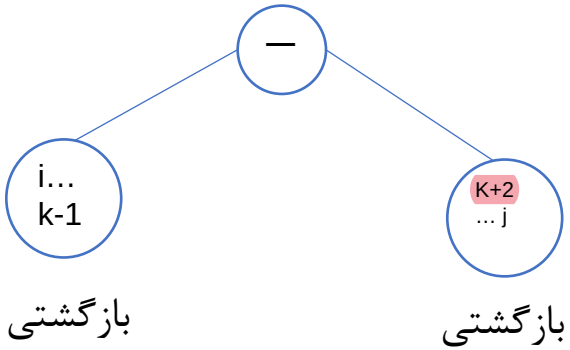
$(a + ((b * c)/d)) - e$

شمردن پرانتزها

1 2 3 2 1 0 k



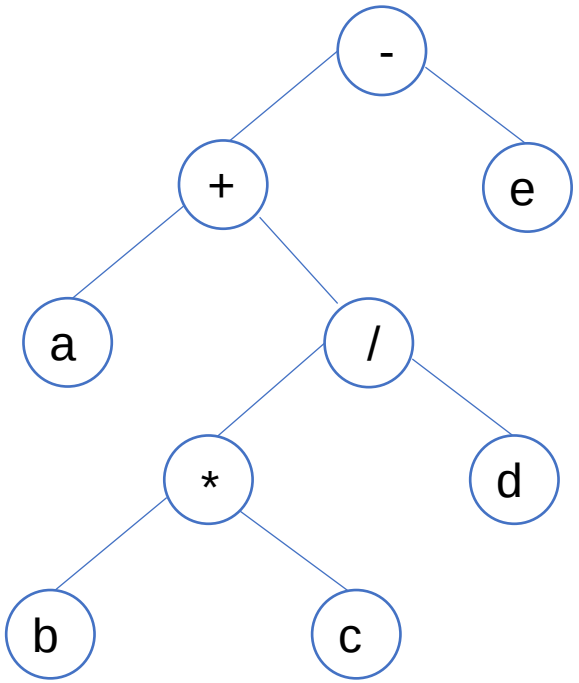
infix-to-tree (i, j)



بدترین حالت $((((a + b) + c) + d) + e)$ ⟶ $\theta(n^2)$

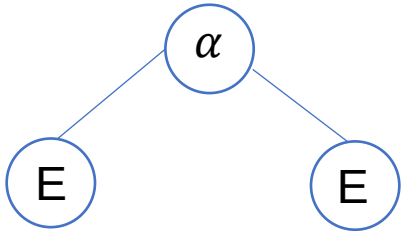
الگوریتم تبدیل عبارت میانوندی کامل به درخت عبارت

$((a + ((b * c)/d)) - e)$

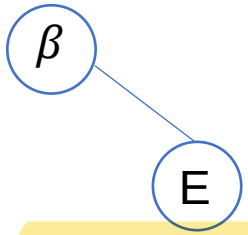


- (: شروع یک عبارت

$E \rightarrow operand$



$E \rightarrow (E\ \alpha\ E)$



$E \rightarrow (\beta\ E)$

برگشت به بالا :) یا operand

$\theta(n)$

الگوریتم تبدیل عبارت میانوندی به عبارت پسوندی

Infix to Postfix

Infix Expression: **A+ (B*C-(D/E^F)*G)*H**, where **^** is an exponential operator.

عملگر با اولویت پایین تر یا مساوی نمی تواند روی عملگر اولویت بالا در پشته قرار گیرد.

Symbol	Scanned	STACK	Postfix Expression	Description
1.		(		Start
2.	A	(	A	
3.	+	(+	A	
4.	(	(+(	A	
5.	B	(+(	AB	
6.	*	(+(*	AB	
7.	C	(+(*	ABC	
8.	-	(+(-	ABC*	'*' is at higher precedence than '-'
9.	(	(+(-(	ABC*	
10.	D	(+(-(	ABC*D	
11.	/	(+(-(/	ABC*D	
12.	E	(+(-(/	ABC*DE	
13.	^	(+(-(/^	ABC*DE	
14.	F	(+(-(/^	ABC*DEF	
15.	)	(+(-	ABC*DEF^/	Pop from top on Stack, that's why '^' Come first
16.	*	(+(-*	ABC*DEF^/	
17.	G	(+(-*	ABC*DEF^/G	
18.	)	(+	ABC*DEF^/G*-	Pop from top on Stack, that's why '^' Come first
19.	*	(+*	ABC*DEF^/G*-	
20.	H	(+*	ABC*DEF^/G*-H	
21.	)	Empty	ABC*DEF^/G*-H*+	END

Rules

1. Push "("onto Stack, and add ")" to the end of X.
2. Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack is empty.
3. If an operand is encountered, add it to Y.
4. If a left parenthesis is encountered, push it onto Stack.
5. If an operator is encountered ,then:
    1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) which has the same precedence as or higher precedence than operator.
    2. Add operator to Stack.
    [End of If]
6. If a right parenthesis is encountered ,then:
    1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) until a left parenthesis is encountered.
    2. Remove the left Parenthesis.
    [End of If]
    [End of If]
7. END.

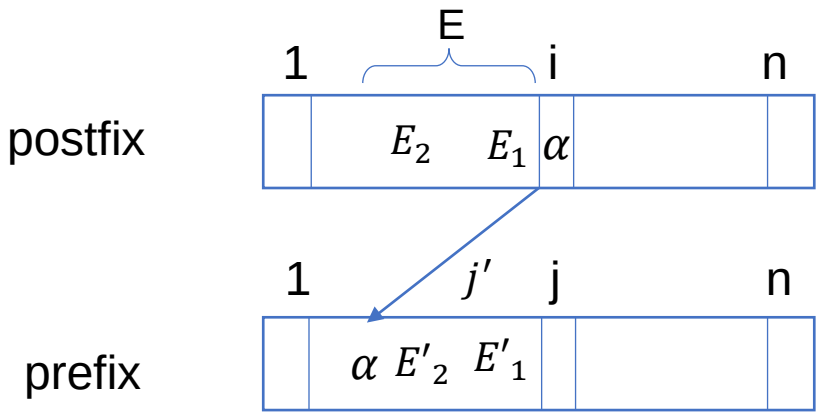
Infix to Postfix Function

```
void Postfix (Expression e)
{// Output the postfix from of the infix expression e. NextToken
// is as in function Eval (Program 3.18). It is assumed that
// the last token in e is '#'. Also, '#' is used at the bottom of the stack
    Stack<Token>stack; // initialize stack
    stack.Push('#');
    for (Token x = NextToken(e); x != '#'; x = NextToken(e))
        if (x is an operand) cout<<x;
        else if (x == ')')
            {// unstack until '('
                for (; stack.Top () != '('; stack.Pop())
                    cout << stack.Top ();
                stack.Pop (); // unstack '('
            }
        else { // x is an operator
            for (; isp (stack.Top()) <= icp (x); stack.Pop())
                cout << stack.Top();
            stack.Push(x);
        }
    // end of expression; empty the stack
    for (; !stack.IsEmpty(); cout << stak.Top(), stack.Pop());
    cout << endl;
}
```

الگوریتم تبدیل عبارت پسوندی به عبارت پیشوندی

- بازگشتی

$ab + c *$



operator = α

الگوریتم تبدیل عبارت پسوندی به عبارت پیشوندی

- بازگشتی

$ab + c *$

operator = *

operator = +

postfix

a	b	+	c	*

prefix

*	+	a	b	c

سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان