

Data Structures and Algorithms

Mohammad GANJTABESH

*Department of Computer Science,
School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.*

mgtabesh@ut.ac.ir



*Dept. of Computer Science
University of Tehran*

Trees

Data Structures and Algorithms
Undergraduate course

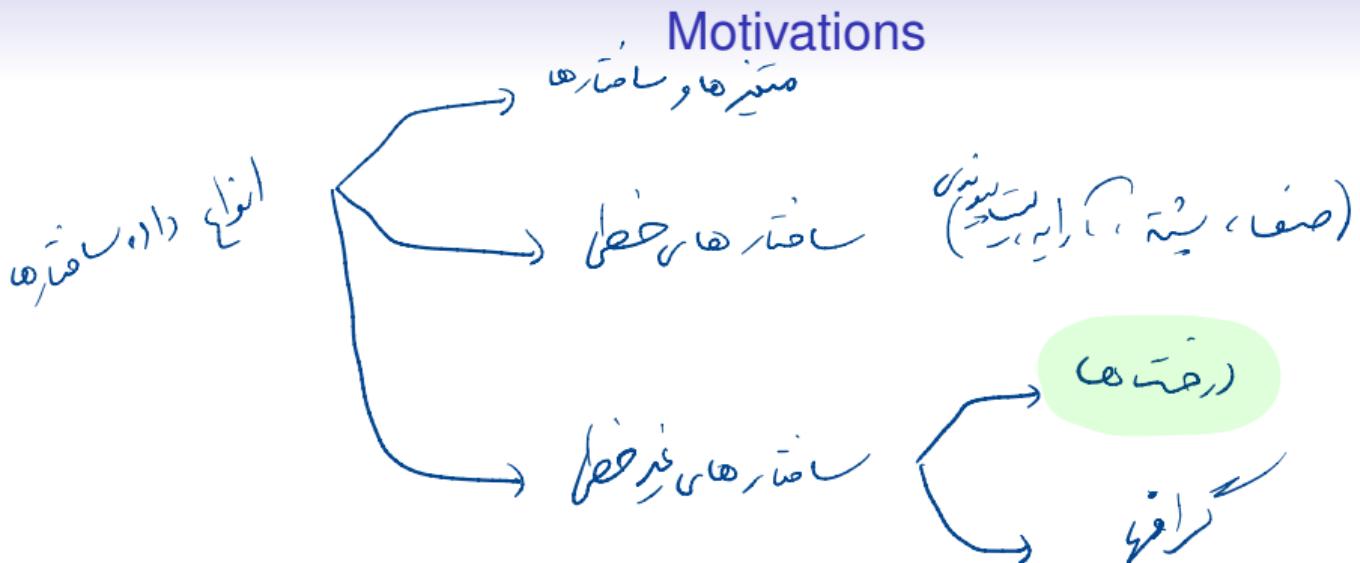


Mohammad GANJTABESH
mgtabesh@ut.ac.ir



Dept. of Computer Science
University of Tehran

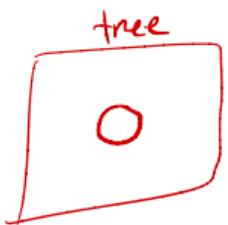
Motivations



Basic Definitions

- درخت (tree) یعنی درخت بدون دور.

برهان بازسش:



- گره به تفاهی پر درخت است.

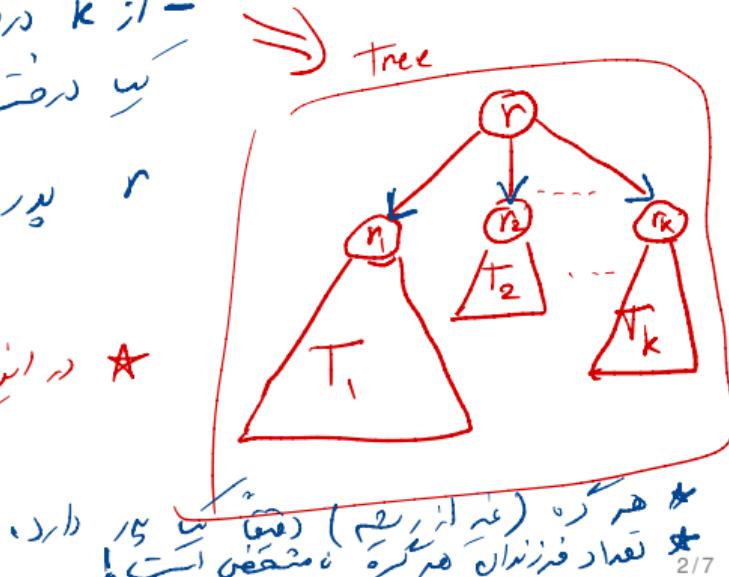
$r_k \in \mathbb{N}$ و هر کمی $T_k \subseteq T_1$ داشته باشد $k \geq 1$

دست زیر را برای T_1 درخت مسازید

نیز $r_k \in \mathbb{N}$ و هر کوچکتر از r_k باشد

T_1 درختی \star و $T_k \subseteq T_1$ حالا T_1 در این حالت

خواهد بود.



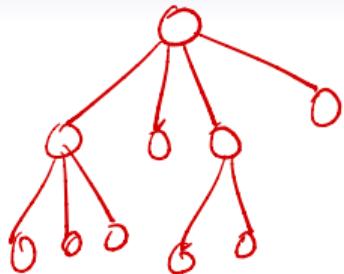
* هر رده (نمای از درخت) (نمای از درخت) است!

* تعداد فرزندان هر گره نه متساوی است!

Basic Definitions

- ریشه (root) : دره ای که خالص پدر است (اگر درخت جسته دار باشد، ریشه یعنی است).
 - برگ (leaf) : دره بدون فرزند.
 - برادر (sibling) : دره هایی که پدر ملکیت دارند.
 - دره داخل (internal node) : دره غیر برگ.
 - ارتفاع دره (height) : طول برازیر میز از دره و بزرگتر از کدهای آن درگاهان لذت گرفته باشد (از دره و همچنین از دره هایی که از آن زیر داشته باشند).
 - ارتفاع درخت : ارتفاع دره ریشه.
 - عمق یک دره (depth) : طول مسیر لذت گرفته شده تا آن دره.
-

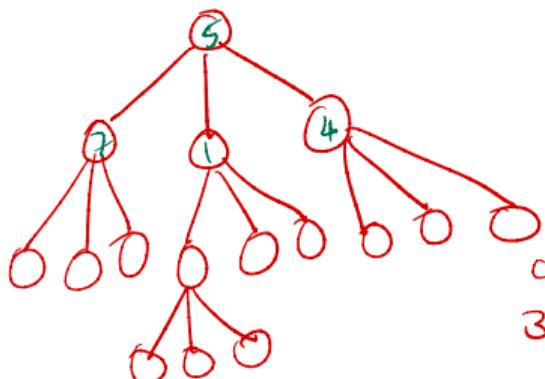
Basic Definitions



4-ary tree

: (k-ary tree) درخت کوئی -
درخت مکانی تعداد فرزندان هرگره برابر کم نباشد.
اگر $k=2$ ، درخت حاصل می‌شود دوی خود (binary tree)

: (complete k-ary tree)
درخت کامل : درخت کوئی -
درخت کامل در آن تعداد فرزندان هرگره برابر k است.

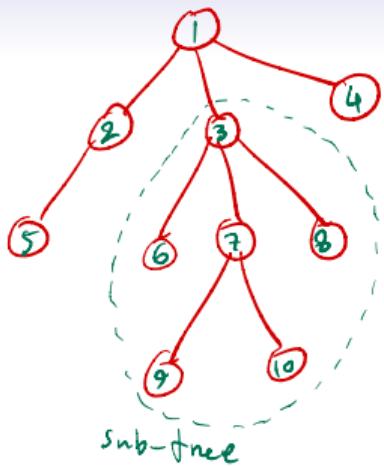


complete 3-ary tree

: (ordered tree)
درخت مرتب : درخت کوئی ترتیب فرزندان هرگره مشخص است.

- درخت برچسب (labeled tree)
درخت کوئی هرگره دارای یک
برچسب است.

Basic Definitions



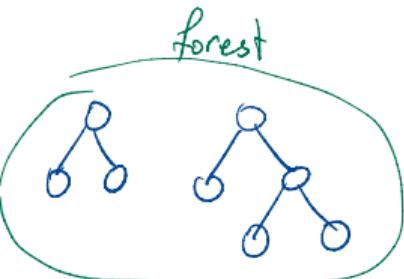
- اولاد سر (descendants) : روزه های موجود در زیر درخت به رتبه \forall

$$\text{اولاد } 3 \Rightarrow \{6, 7, 8, 9, 10, 3\}$$

- اجداد سر (ancestors) : روزه های موجود در مسیر از ریشه تا کنون روزه

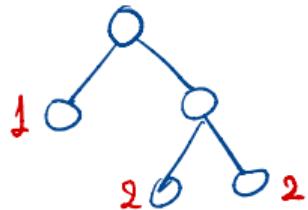
$$8 \Rightarrow \{3, 1, 8\}$$

- زیر درخت (sub-tree) : مجموعه ای از روزه های درخت و جمله ای از روزه های درخت



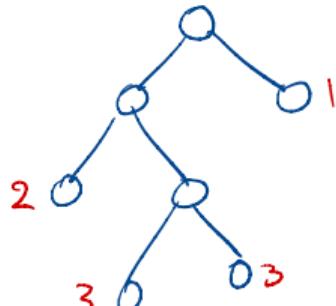
Basic Definitions

- درخت معادن (balanced tree) :
درخت در کل عمق برگ ها مغایر نیست و این بین اختلاف را نمایه می شود.

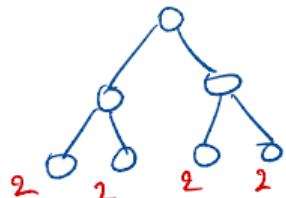


$$2 - 1 = 1 \leq 1$$

balanced



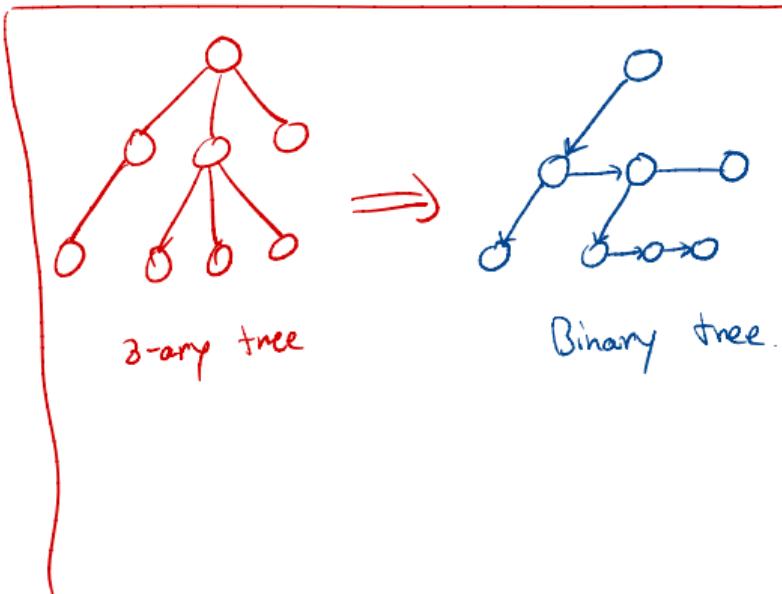
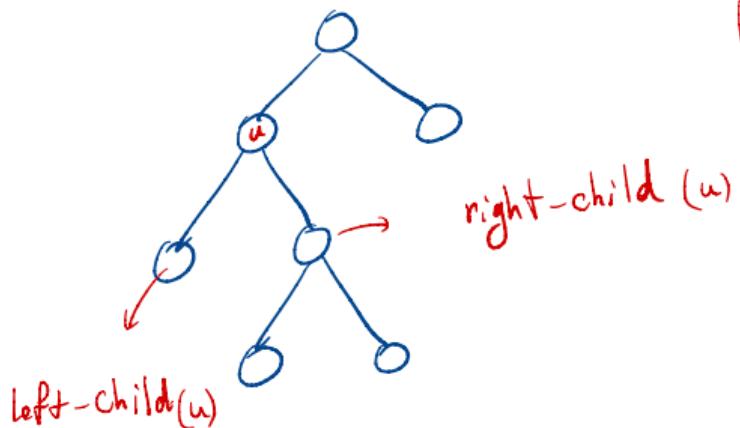
not balanced



- درخت کامل معادن (completely balanced tree) :
درخت کامل عمق برگ ها مغایر نباشند.

Basic Definitions

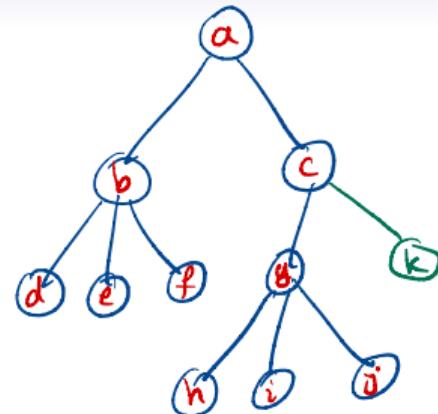
- درخت دوگانه (binary tree) : درخت مترتب که هر کوئن صدالت دارد
و فرزنه ایست (فرزنه چپ و فرزنه راست).



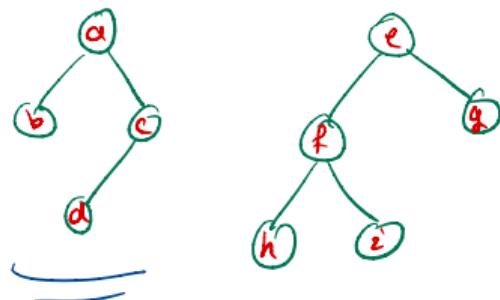
Tree Representations

1) Array :

	1	2	3	4	5	6	7	8	9	10	11	12	---
key	a	b	c	d	e	f	g	h	i	j	k		
parent	0	1	1	2	2	2	3	7	7	7	3		



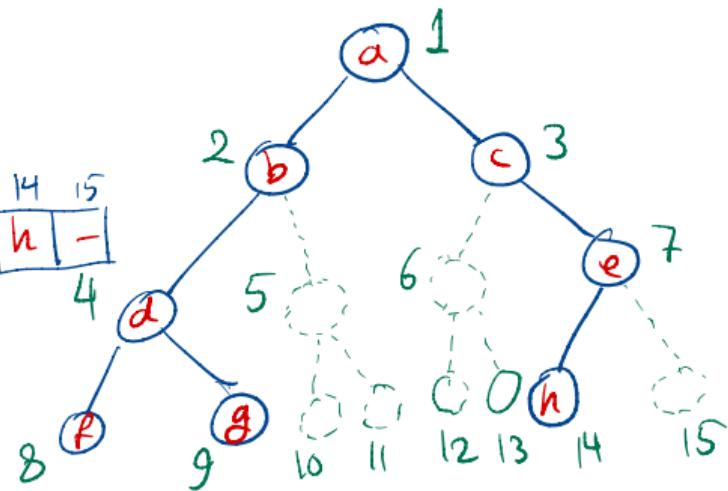
	1	2	3	4	5	6	7	8	9
key	a	b	e	g	c	d	f	h	i
parent	0	1	0	3	1	5	3	7	7



Tree Representations

2) Array (\checkmark , \checkmark , \checkmark)

1	2	3	4	5	6	7	8	9	-	14	15
a	b	c	d	-	-	e	f	g	-	h	-



$$\text{left-child}(i) = 2i$$

$$\text{right-child}(i) = 2i + 1$$

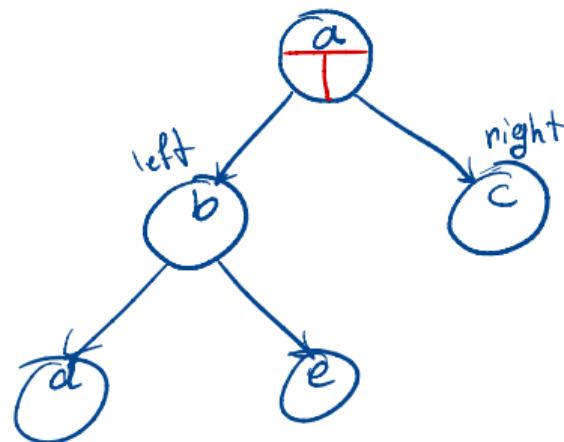
$$\text{Parent}(i) = \lfloor i/2 \rfloor$$

$h \Rightarrow \frac{h}{2} - 1$
 عدد الگویی که متوالی
 صفت درخت را دارد

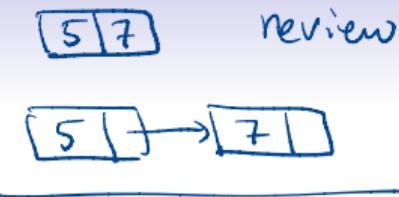
Tree Representations

3) Using Pointers:

```
struct TreeNode {  
    int key;  
    TreeNode *left;  
    r     *right;  
}
```



for Kary tree:



For k-ary tree:

Tree Representations

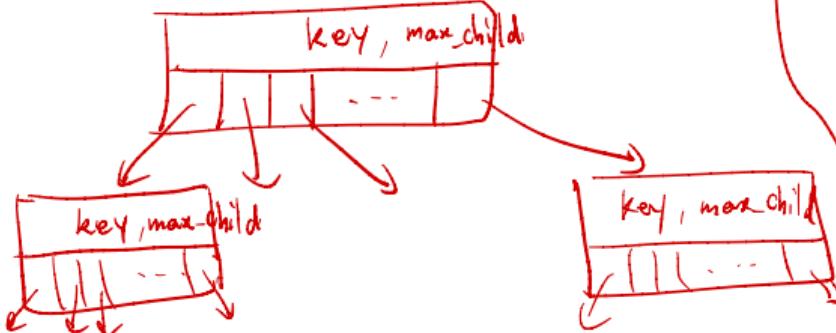
```
struct TreeNode {
```

```
    int key;
```

```
    TreeNode ** childs;
```

```
    int max_child;}
```

```
}
```



For k-ary tree converted to binary tree.

```
struct TreeNode {
```

```
    int key;
```

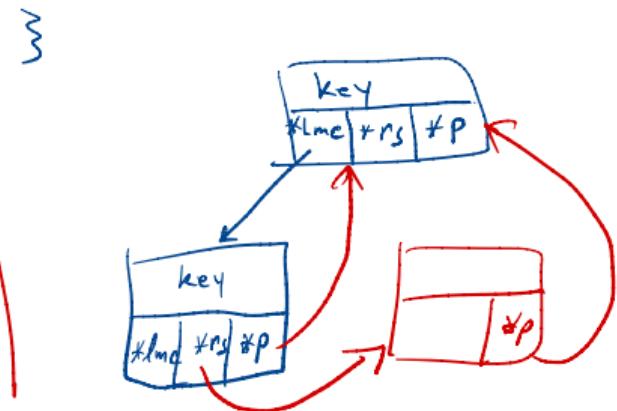
```
    TreeNode * leftmost_child;
```

```
    ~~~~~~
```

```
    * right_sibling;
```

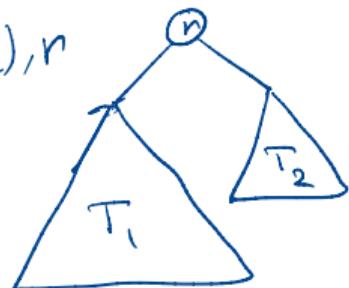
```
    ~~~~~~
```

```
    * parent;
```

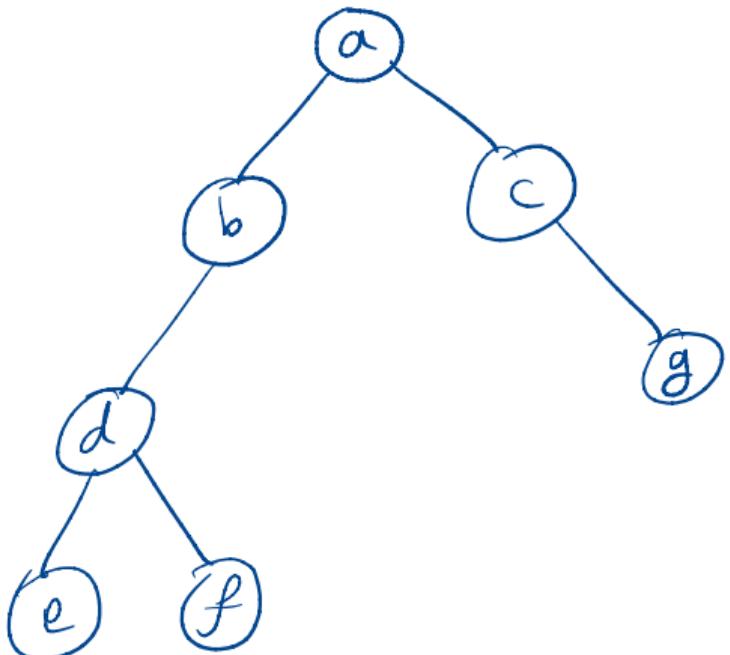


Tree Traversals

- Breadth First Search (BFS) : $r \rightarrow T \rightarrow \text{Left Subtree} \rightarrow \text{Right Subtree}$
- Preorder(T) : $r, \text{Preorder}(T_1), \text{Pre-order}(T_2)$
جستجوی پیش
- In-order(T) : $\text{In-order}(T_1), r, \text{In-order}(T_2)$
جستجوی درون
- Post-order(T) : $\text{Post-order}(T_1), \text{Post-order}(T_2), r$
جستجوی پس
- level-order(T) : $r, \text{Level Order of Left Subtree}, \text{Level Order of Right Subtree}$
جستجوی طبقه



Tree Traversals



left - sub tree right
 sub-tree

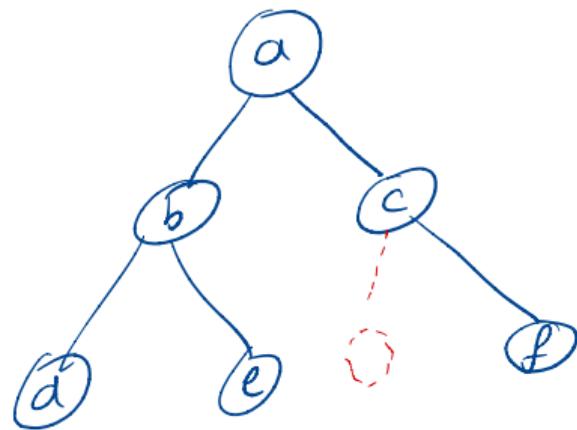
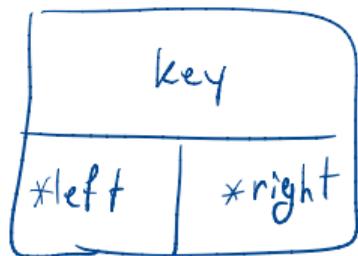
pre-order : a, b, d, e, f, c, g

In-order : e, d, f, b, a, c, g
left - sub-tree right
 sub-tree

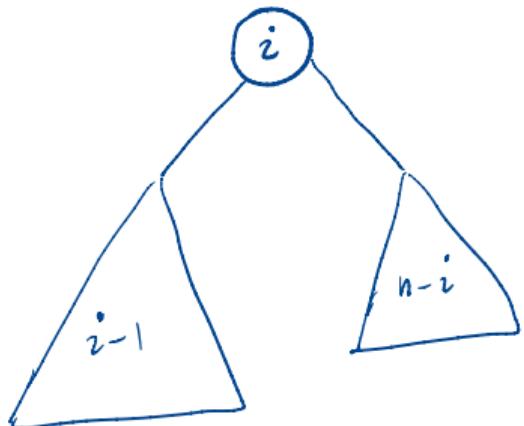
post-order : e, f, d, b, g, c, a
left sub-tree right
 sub-tree

Level-order : a, b, c, d, g, e, f

Binary Tree



Number of Binary Trees with n Nodes



$$T(n) = \begin{cases} 1 & \text{if } n=0 \\ \sum_{i=1}^n T(i-1)T(n-i) & n \geq 1 \end{cases}$$

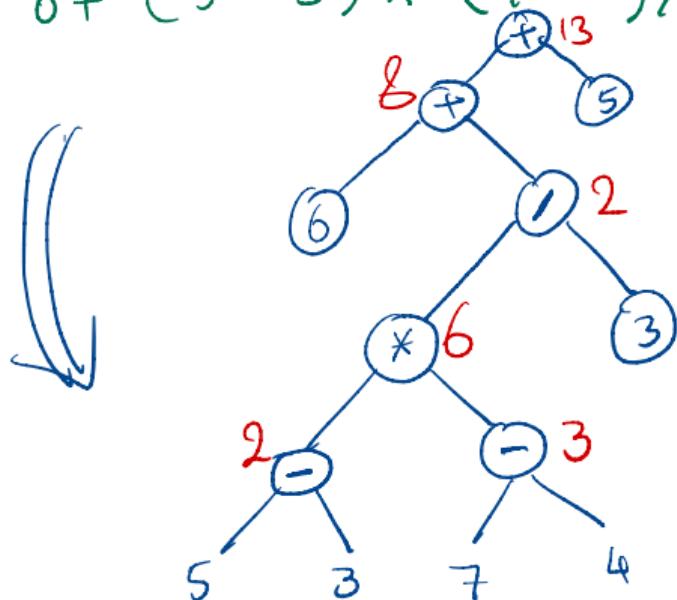


$$T(n) = \frac{1}{n+1} \binom{2n}{n}$$

(Jack 1-n 2x)

Expression Tree

$6 + (5 - 3) * (7 - 4) / 3 + 5$



(infix)
↓

post fix

6 5 3 - 7 4 - * 3 / + 5 +
 | | | | |
 2 3 6 2 8 13

post-order :

6 5 3 - 7 4 - * 3 / + 5 +