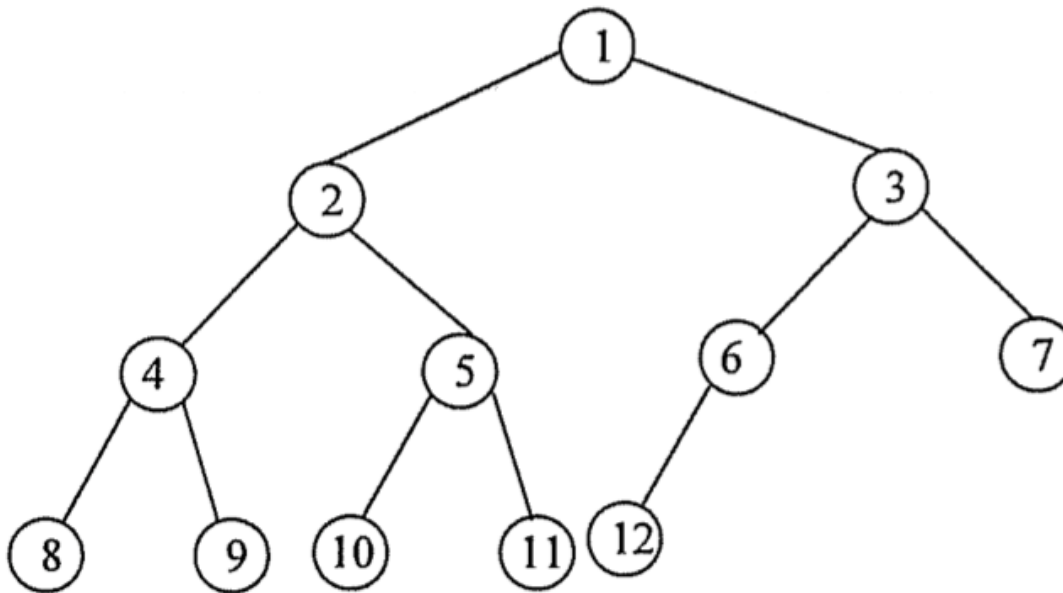


ساختمان‌های داده

Data Structures

درخت

Tree



مقدمه

□ درخت (Tree) یک ساختمان داده **سلسله مراتبی** است شامل:

- راس‌ها (گره‌ها)
- یال‌هایی است که گره‌ها را به یکدیگر متصل می‌سازند.

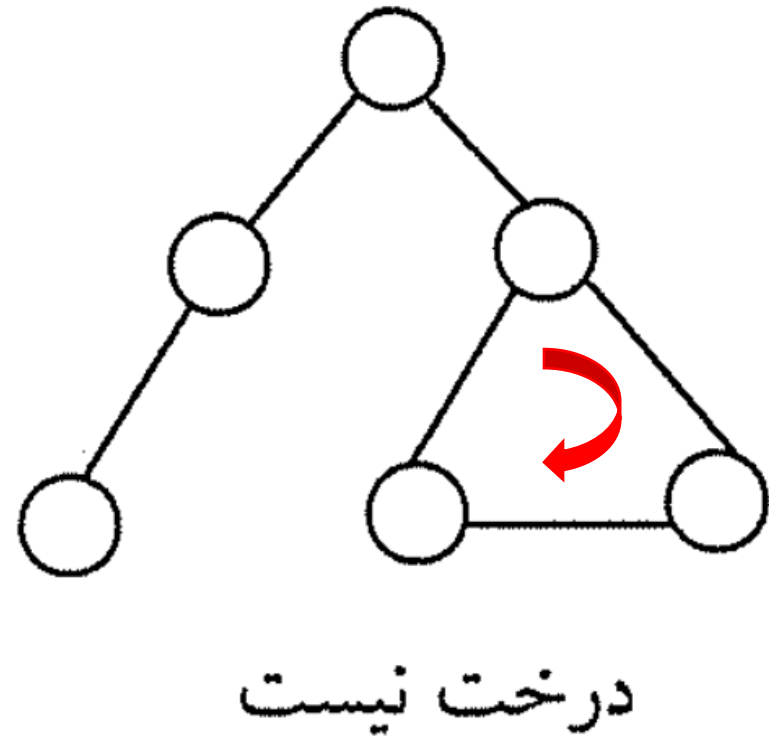
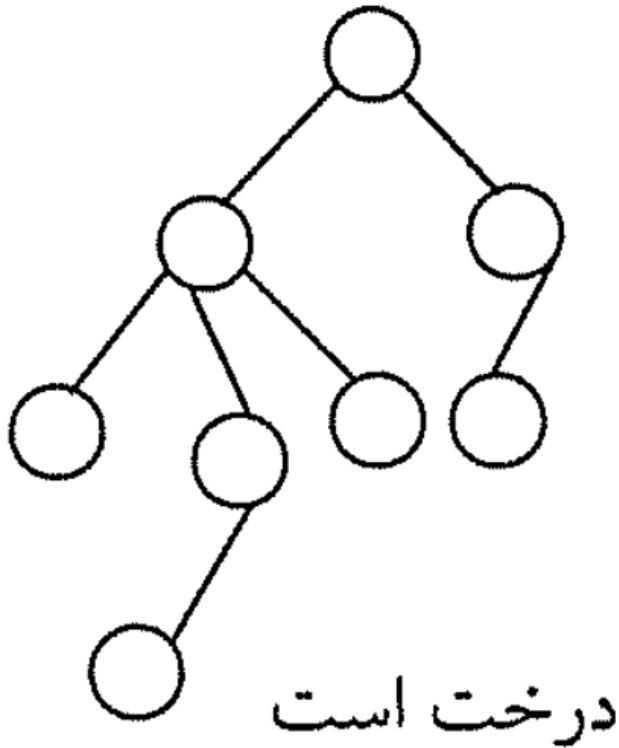
□ درخت‌ها مشابه گراف‌ها هستند با این تفاوت:

- در درخت برخلاف گراف، **دور (Cycle)** وجود ندارد.

□ درخت‌ها به طور گسترده‌ای در هوش مصنوعی و الگوریتم‌های پیچیده به منظور

فراهم کردن یک مکانیزم ذخیره سازی موثر جهت حل مساله استفاده می‌شوند.

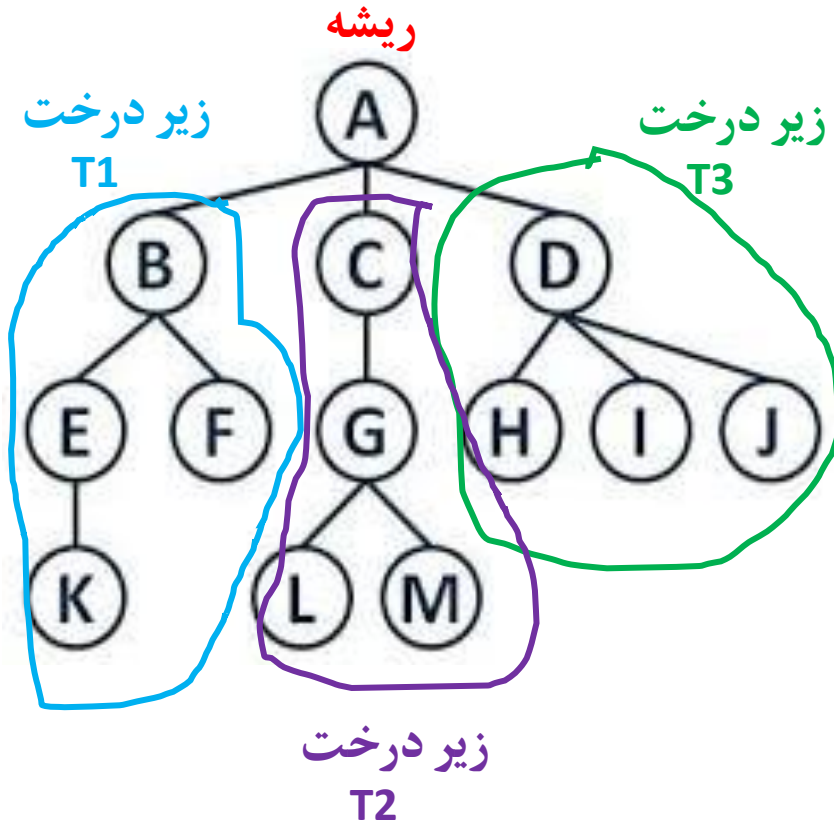
مثال



دارای دور (حلقه) می باشد

تعاریف

□ درخت دارای گره خاصی به نام ریشه است.



□ بقیه گره ها به $n \geq 0$ مجموعه مجزا T_1, \dots, T_n

تقسیم شده است که هر یک از این مجموعه ها خود یک درخت هستند.

□ T_1, \dots, T_n زیر درختان ریشه می شوند.

تعاریف

□ ریشه : گره بدون پدر

□ درجه یک گره : تعداد زیر درخت های یک گره

□ درجه یک درخت : حداکثر درجه گره های آن درخت

□ برگ (گره های پایانی) :

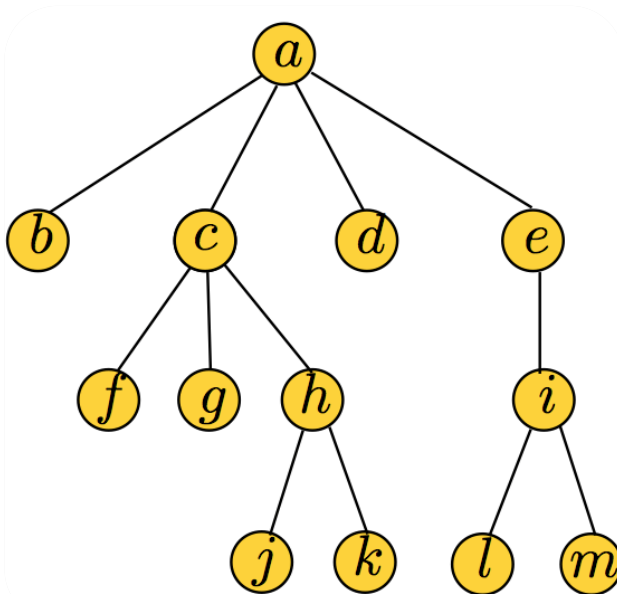
- گره هایی که درجه صفر دارند.

- گره بدون فرزند

□ گره های همزاد (هم نیا) : فرزندان یک گره

□ اجداد یک گره : گره هایی که در مسیر طی شده از ریشه تا آن گره

وجود دارد.

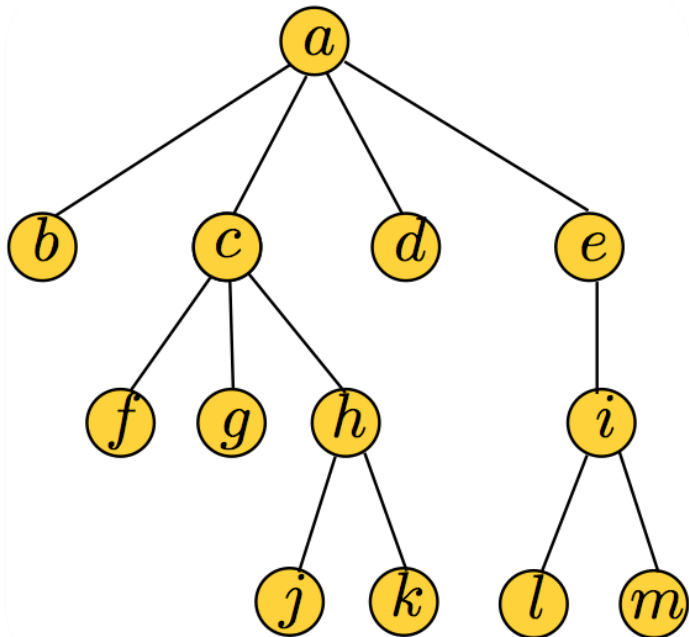


تعاریف

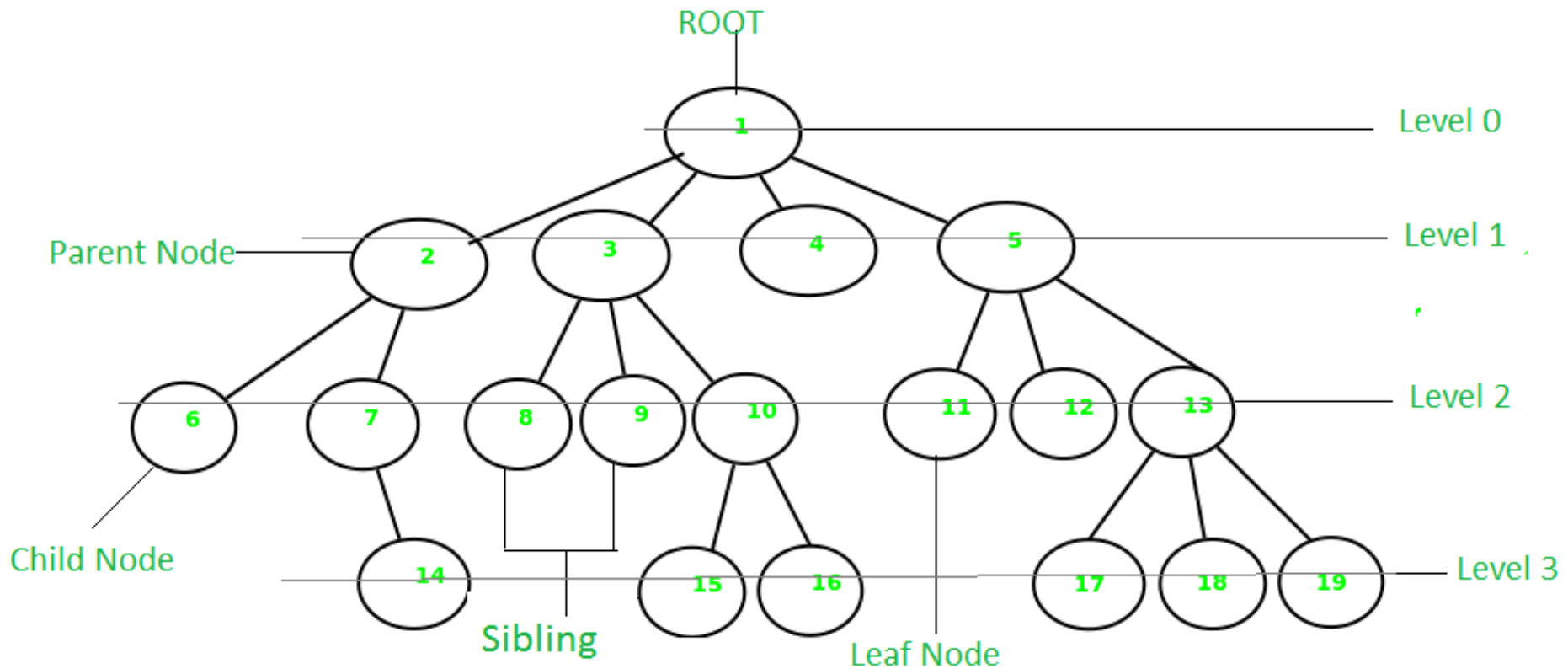
□ سطح یک گره: طول مسیر از ریشه به آن گره

• $a \rightarrow 1$ (در برخی منابع سطح ریشه 0 در نظر گرفته می شود)

• $h \rightarrow 3$ (بنابراین در این منابع سطح $h=2$ می باشد)



تعاریف

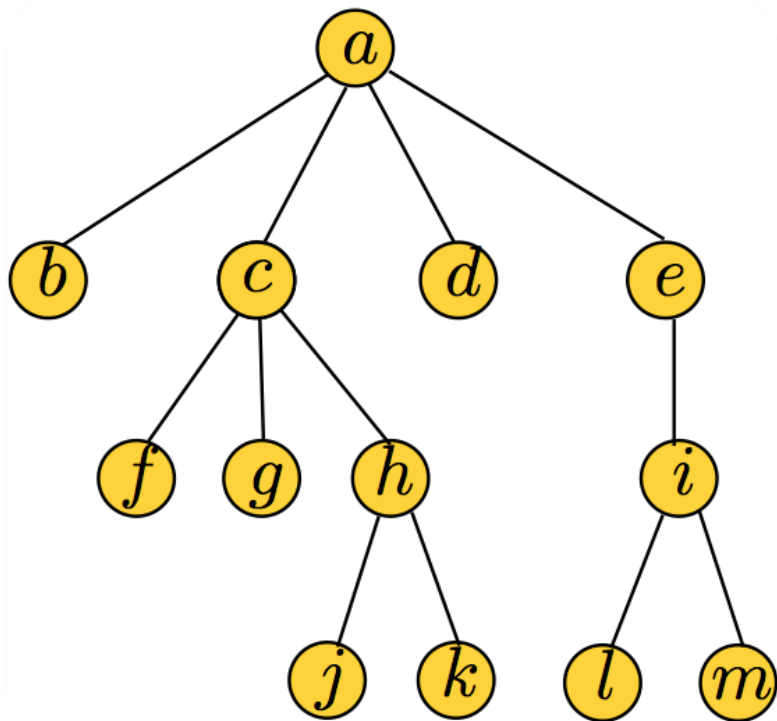


تعاریف

□ **گره داخلی** : گره غیر برگ

□ **ارتفاع (عمق) درخت** : بیشترین سطح گره ها

• در این درخت = ۴



نمایش درخت عمومی

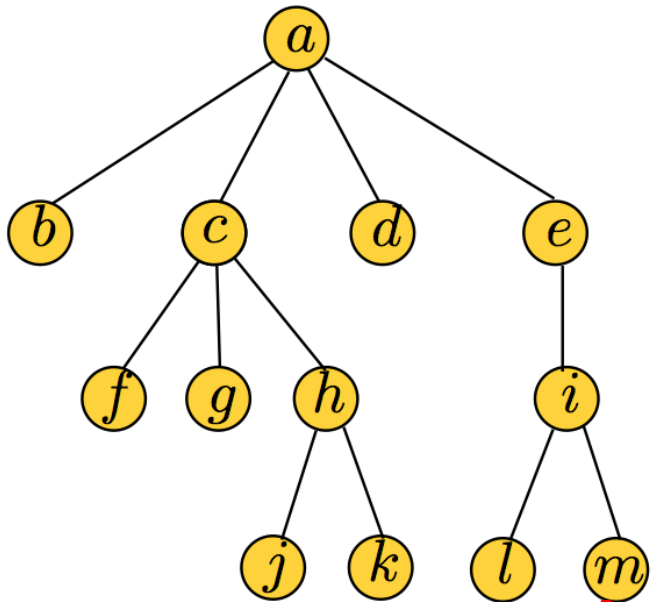
پرانتر گذاری

□ ابتدا ریشه

□ اطلاعات فرزندان ریشه در پرانتر

□ اطلاعات فرزندان از چپ به راست

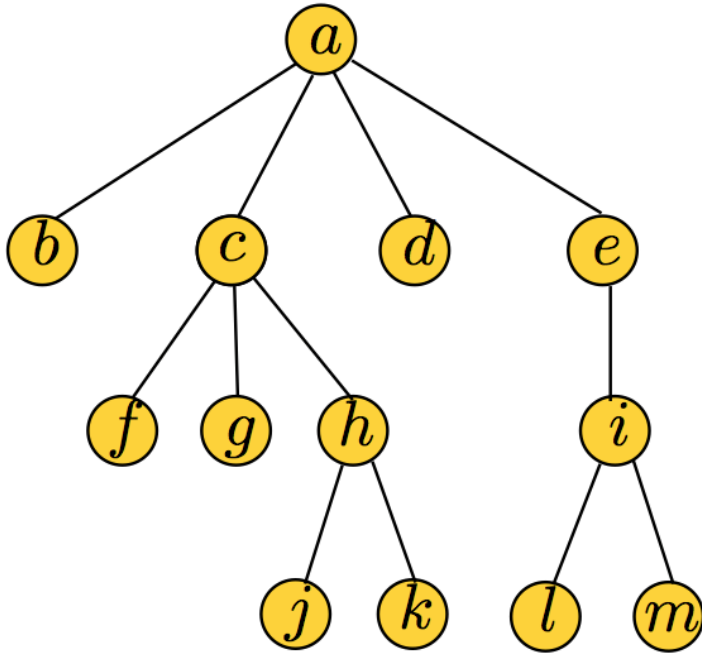
(a(b), (c(f, g, h(j, k))), (d), (e(i(l, m))))



آرایه

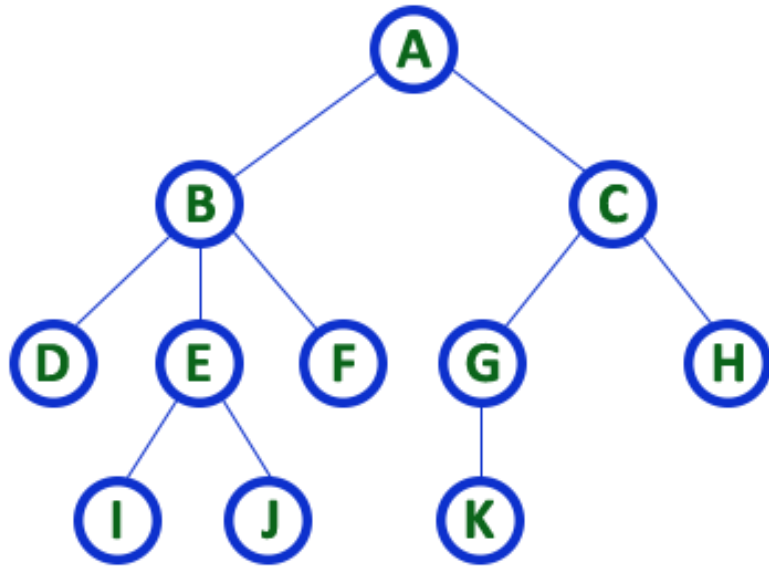
□ ذخیره درخت بصورت:

- مقدار هر گره
- شماره درایه پدر هر عنصر



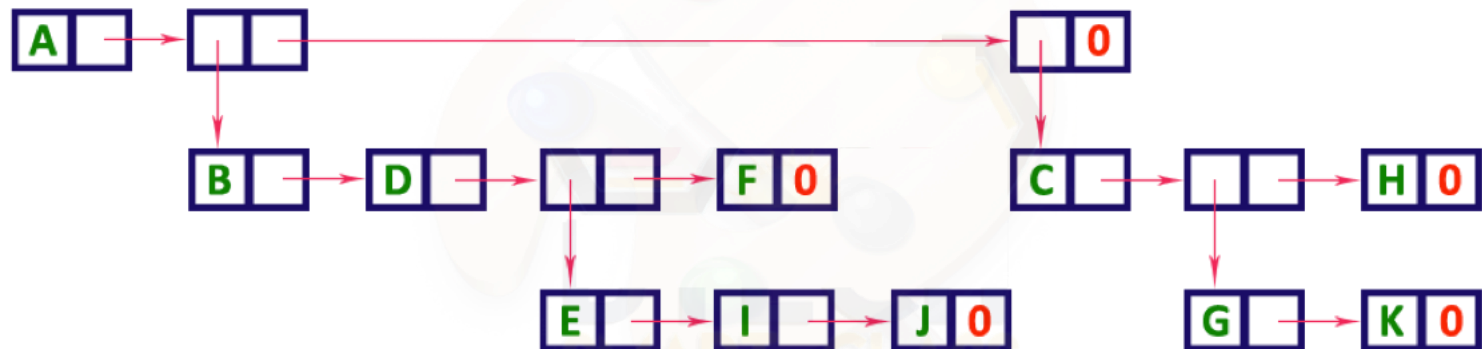
1	2	3	4	5	6	7	8	9	10	11	12	13
a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	1	1	1	3	3	3	5	8	8	9	9

لیست پیوندی



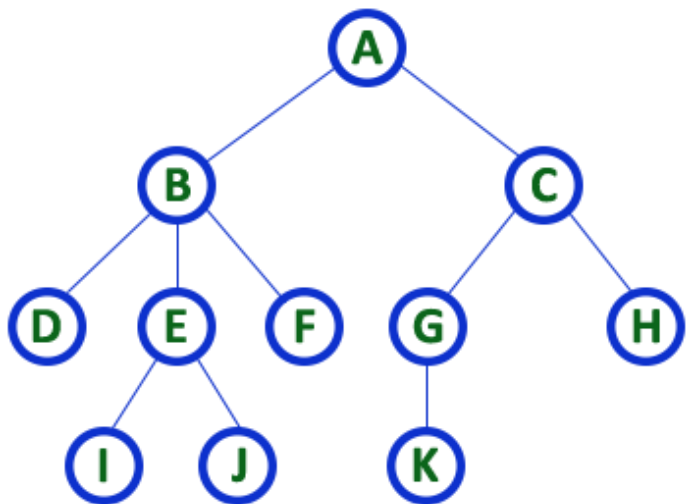
□ درخت روبرو را می توان در قالب

لیست به صورت زیر نشان داد

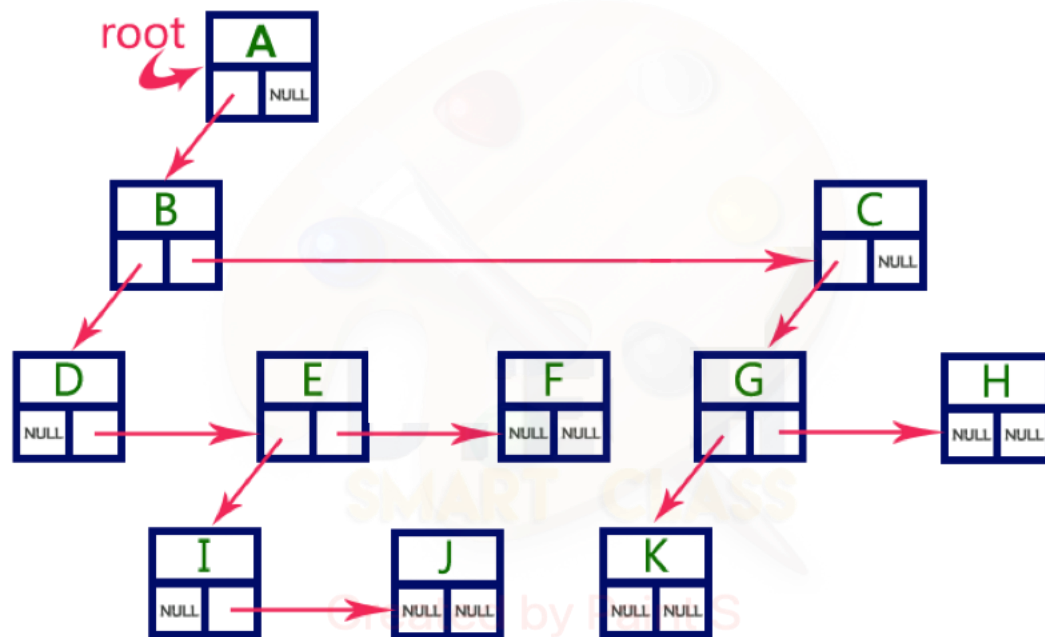


نمایش درخت عمومی با درخت دودویی

□ فرزند چپ، همزاد راست (Left Child - Right Sibling Representation)

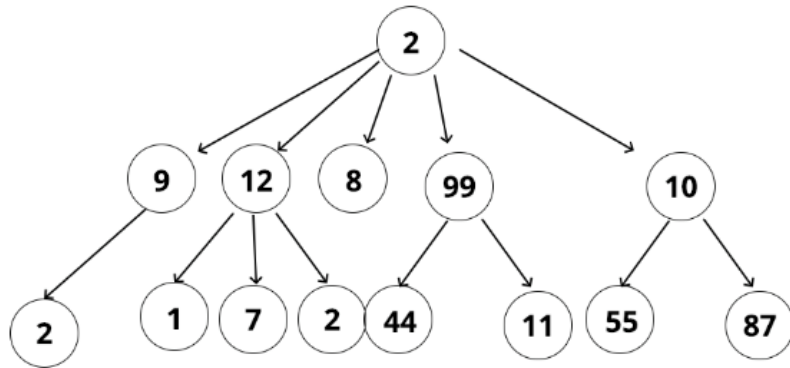


Data	
Left Child	Right Sibling



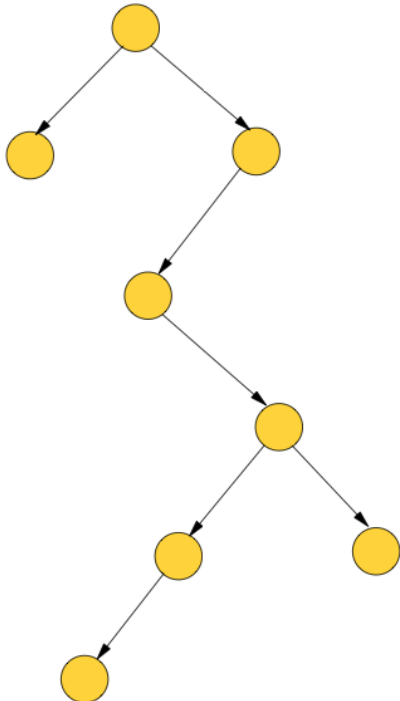
درخت دودویی (Binary Tree)

تعریف



□ درخت k تایی

- درختی که بیشینه تعداد فرزندان یک گره k باشد. (در شکل $k=5$)



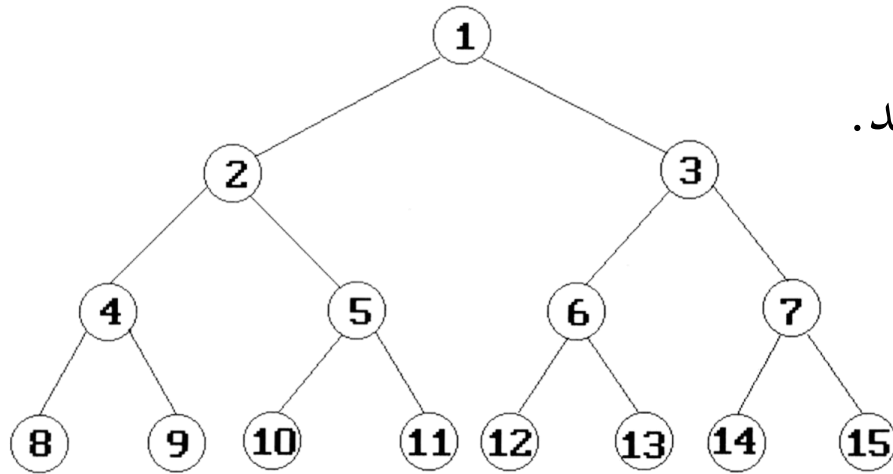
□ درخت دودویی

- حداکثر تعداد فرزندان هر گره $= 2$

□ تفاوت با درخت عادی

- می تواند دارای صفر گره باشد!
- ترتیب فرزندان اهمیت دارد.

انواع درخت دودویی



□ درخت پر (Full Binary Tree)

- همه گره ها به جز سطح آخر، ۲ فرزند دارند.
- تعداد گره سطح $i \leftarrow 2^{i-1}$
- تعداد کل گره ها $\leftarrow n = 2^h - 1$
- تعداد کل برگ ها $\leftarrow n_0 = 2^{h-1}$
- تعداد گره های داخلی $\leftarrow 2^{h-1} - 1$

• ارتفاع درخت

$$h = \lfloor \log n \rfloor + 1 = \log(n + 1)$$

انواع درخت دودویی

□ درخت کامل

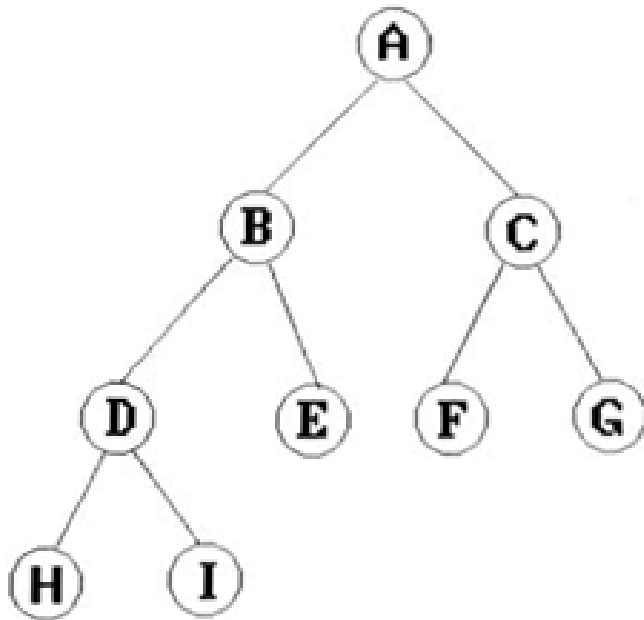
- تا ارتفاع $h-1$ پر است
- برگها در ارتفاع h از چپ به راست پر می شوند.

• ارتفاع درخت

$$h = \lfloor \log n \rfloor + 1 = \lfloor \log(n + 1) \rfloor$$

• تعداد گره ها

$$2^{h-1} \leq n \leq 2^h - 1$$



انواع درخت دودویی

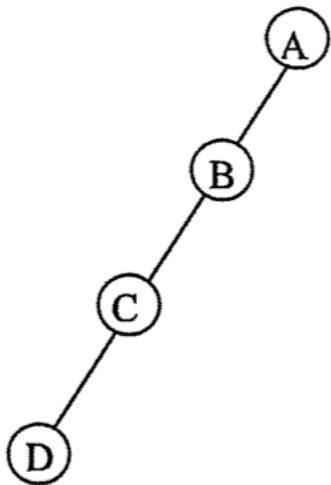
□ درخت اریب (مورب)

- درجه رئوس به جز آخری برابر ۱ است

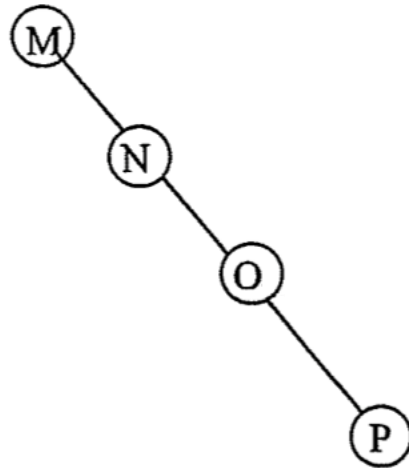
- تعداد گره تک فرزندی $n-1$

- تعداد برگها $= 1$

- ارتفاع $h = n$



درخت مورب چپ

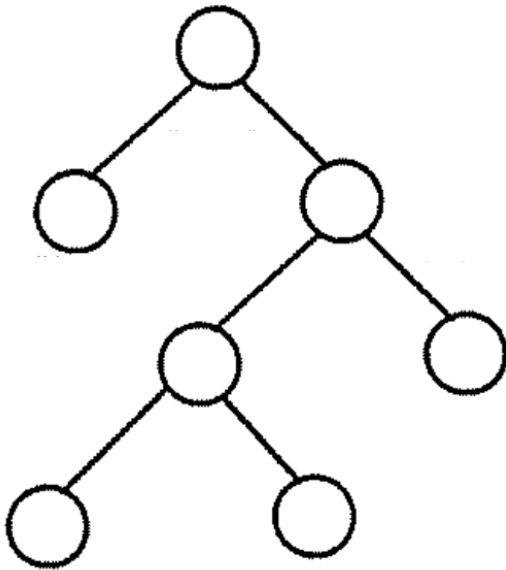


درخت مورب راست

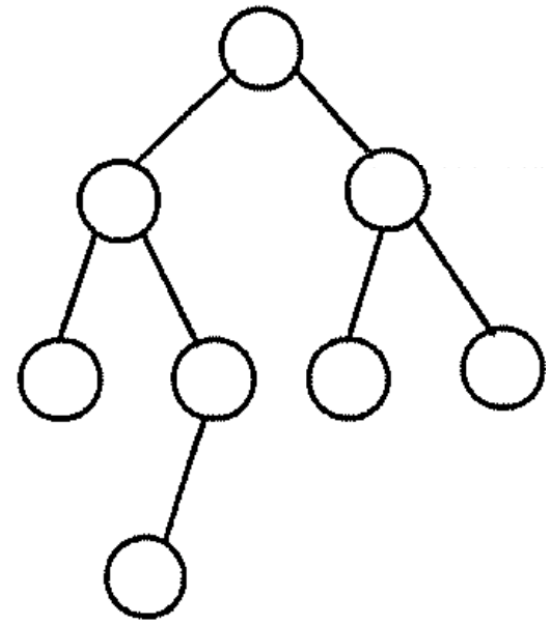
انواع درخت دودویی

□ درخت دودویی محض

- در آن هر گره ی غیر برگ، زیر درختهای چپ و راست خالی نداشته باشد.
- به عبارت دیگر تمام گره ها از درجه صفر یا ۲ است.



درخت دودویی محض هست

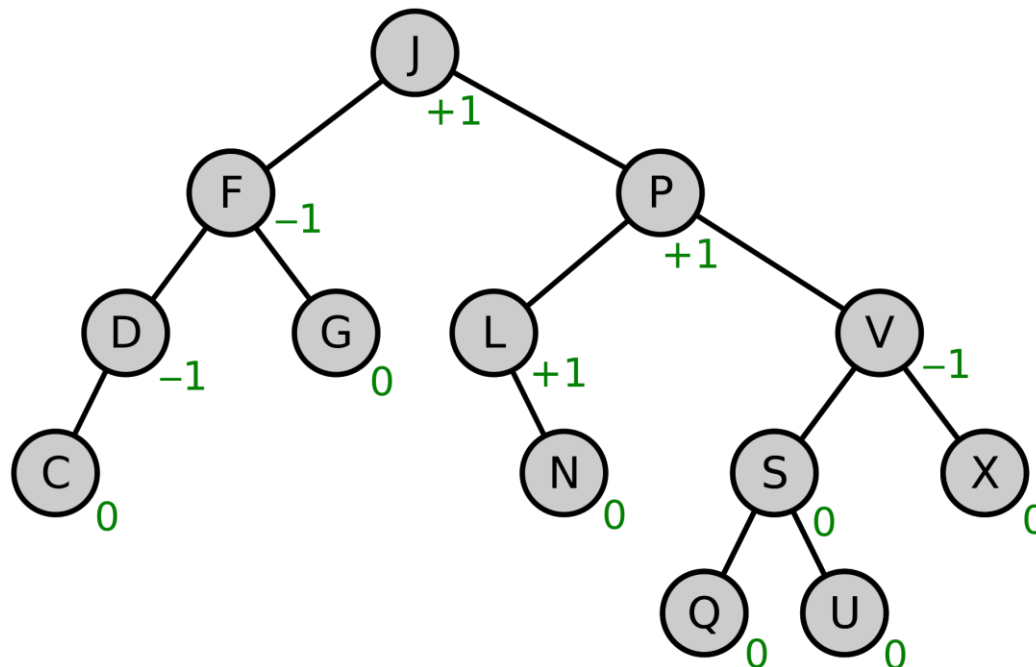


درخت دودویی محض نیست

انواع درخت دودویی

□ درخت متوازن

- درختی که در آن اختلاف در زیر درخت چپ و راست هر گره حداکثر یک باشد.



$$\text{BalanceFactor} := \text{Height}(\text{RightSubtree}(n)) - \text{Height}(\text{LeftSubtree}(n))$$

خصوصیات درخت دودویی

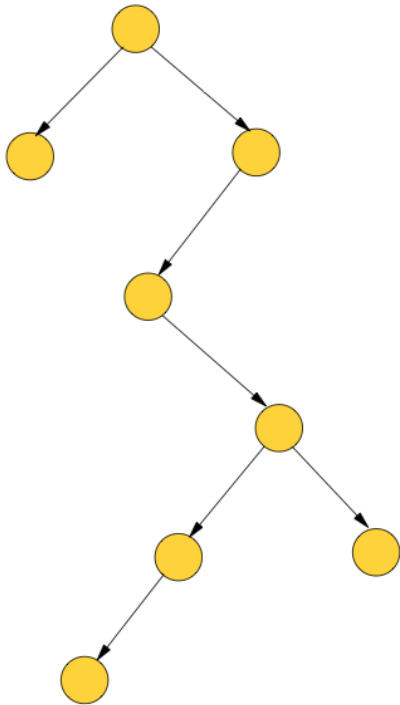
تعداد یالها

$$e = n - 1$$

$$e = 2n_2 + n_1$$

$$n_0 = n_2 + 1$$

$$n = n_0 + n_1 + n_2$$



• n_0 : تعداد گره های درجه 0 یا تعداد برگها

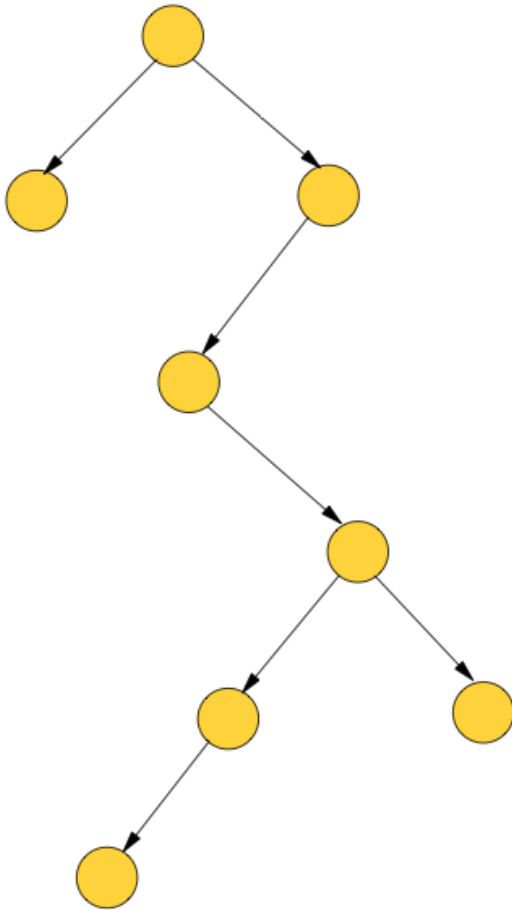
• n_1 : تعداد گره های درجه 1

• n_2 : تعداد گره های درجه 2

• e : تعداد یال ها

• n : تعداد کل گره ها

تعداد برگ‌ها



$$n_0 = n_2 + 1$$

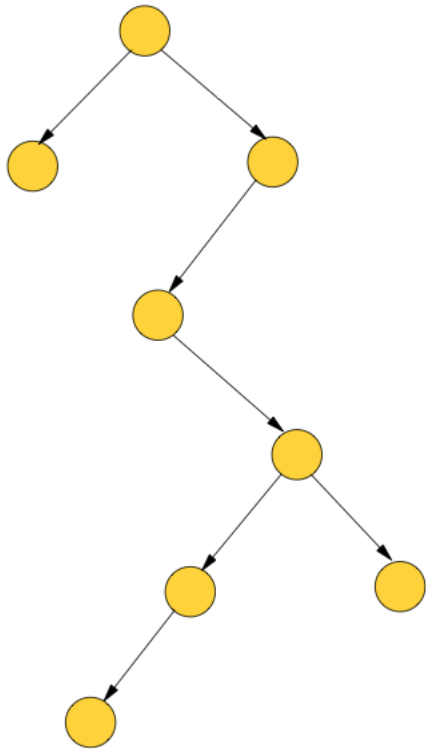
$n_0 = \left\lceil \frac{n+1}{2} \right\rceil \leftarrow$ در درخت کامل و پر

$1 \leq n_0 \leq 2^{h-1}$ → درخت پر

$$1 \leq n_0 \leq \left\lceil \frac{n+1}{2} \right\rceil$$

عمق، تعداد گره

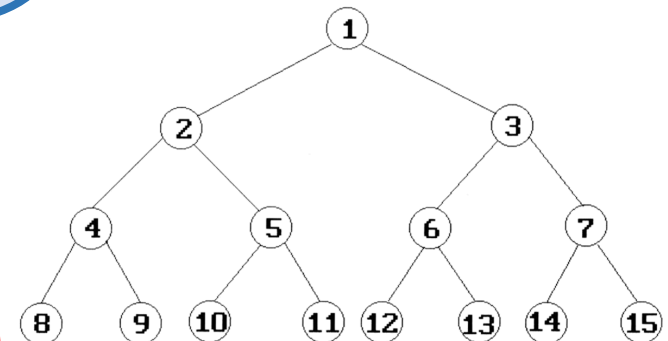
درخت اریب $\rightarrow [\log n] + 1 \leq h \leq n \leftarrow$ درخت پر یا کامل



• تعداد گره‌ها در سطح i ام $1 \leq n_i \leq 2^{i-1}$ درخت پر

تعداد کل گره‌ها

درخت پر $h \leq n \leq 2^h - 1$ درخت اریب



تعداد درخت دودویی مختلف

□ تعداد درخت‌های مختلف که با n گره می‌توان ساخت:

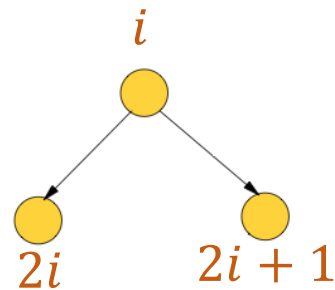
• تعداد درخت‌های دودویی کامل = تعداد برگ‌های ارتفاع $h = 2^{h-1}$

• تعداد درخت‌های متفاوت با n گره $= \frac{1}{n+1} \binom{2n}{n}$

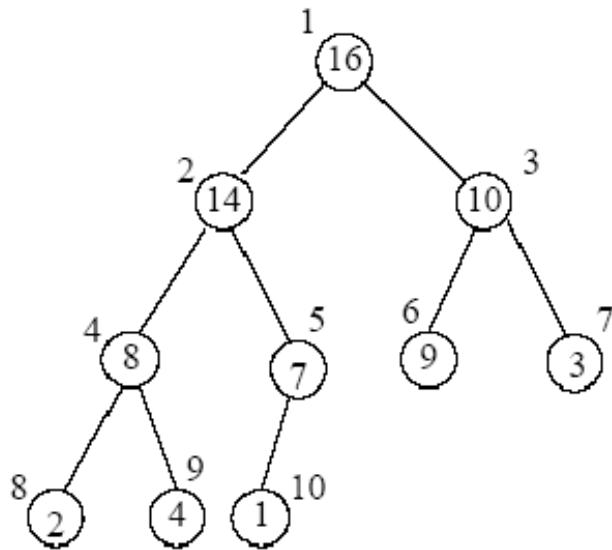
نمایش درخت دودویی

با کمک آرایه

- شماره گذاری گره‌ها به صورت زیر

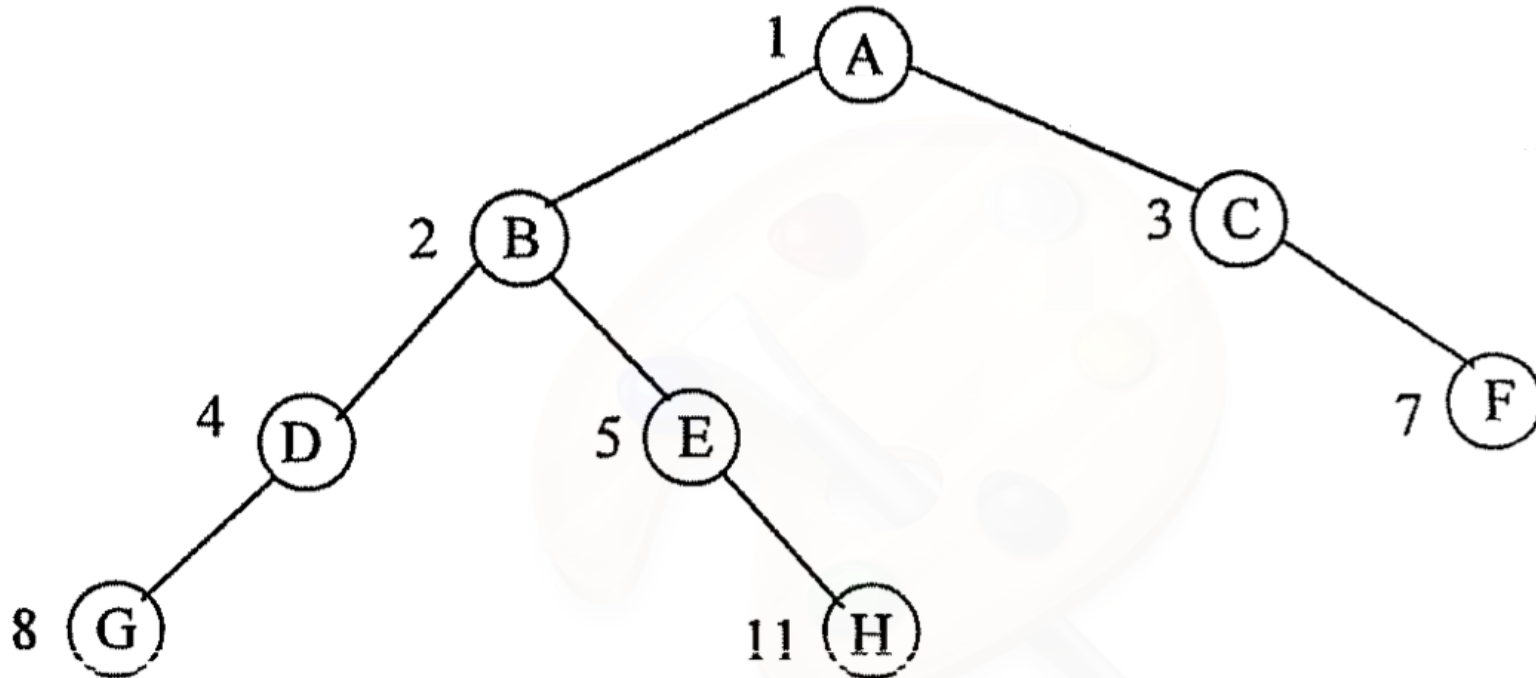


- قراردادن عناصر در آرایه براساس مکان آنها



1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

با کمک آرایه



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	C	D	E		F	G			H				

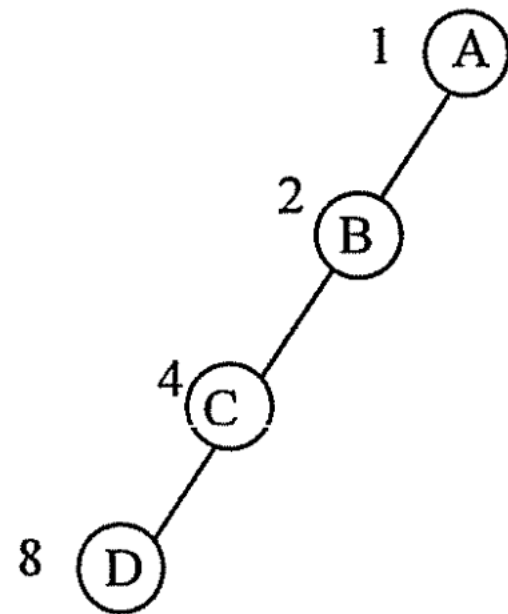
با کمک آرایه

□ استفاده از فرم آرایه برای درختهای کامل مناسب است.

- زیرا هیچ خانه ای از حافظه به هدر نمی رود.

□ اما برای درخت های مورب کاملاً نامناسب است

- چرا که فضای زیادی را به هدر می دهد.
- در واقع در بدترین حالت در یک درخت به عمق k به $2^k - 1$ خانه آرایه نیاز دارد که از این مقدار فقط k محل اشغال می شود.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B		C				D							

لیست پیوندی

□ از مشکلات نمایش درخت با کمک آرایه درج یا حذف گره می باشد

- زیرا مستلزم جابجایی گره ها در آرایه می شود.

□ برای رفع این مشکل استفاده از لیست پیوندی پیشنهاد می شود.

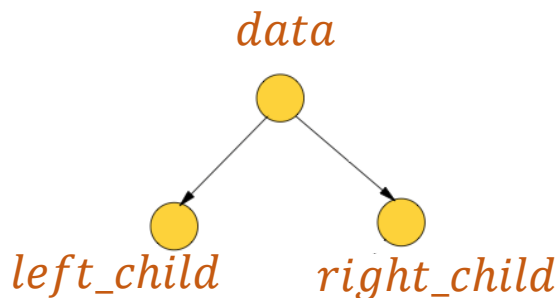
- هر گره از ۳ فیلد تشکیل می شود:



▪ فیلد داده

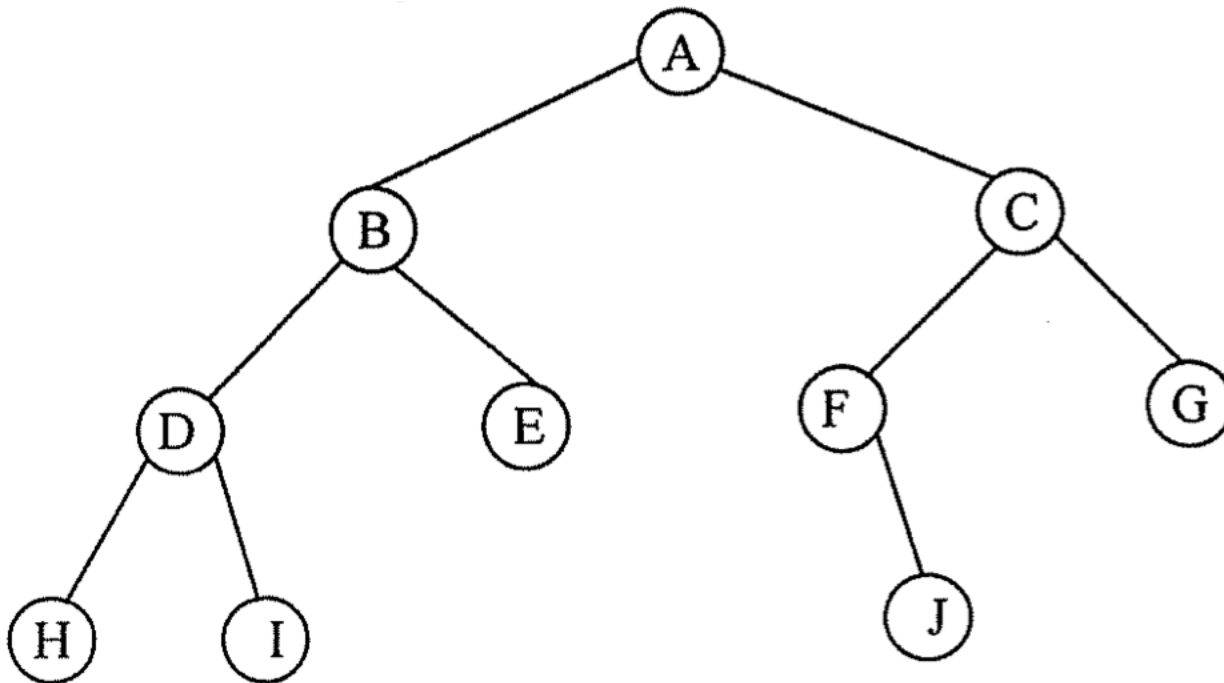
▪ فیلد اشاره به فرزند چپ

▪ فیلد اشاره به فرزند راست



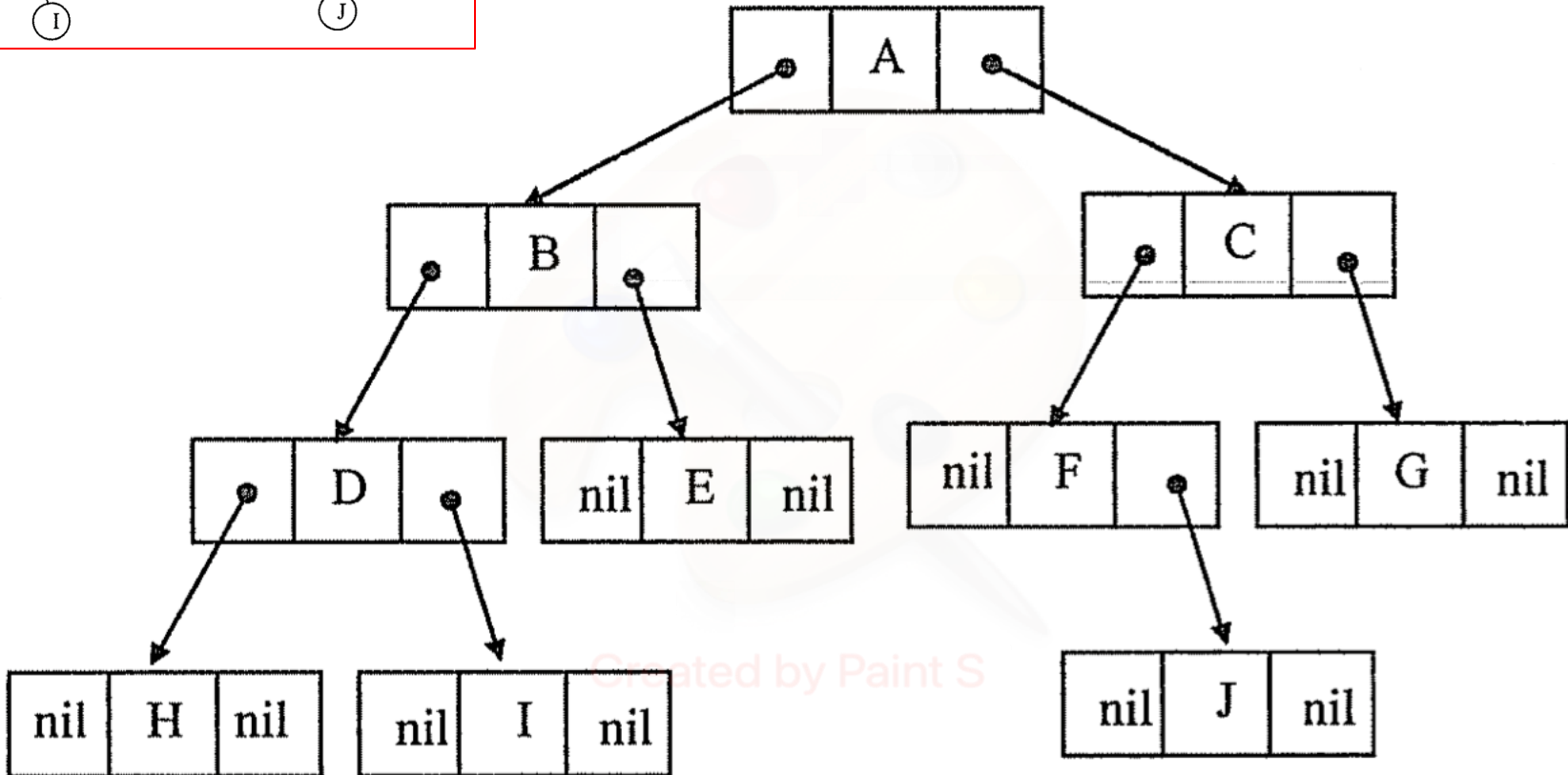
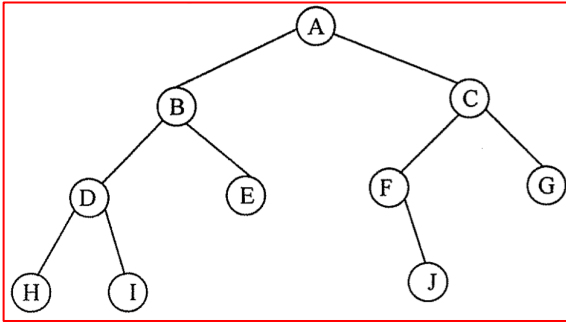
لیست پیوندی

□ مثال : نمایش لیست پیوندی درخت زیر



لیست پیوندی

□ مثال : نمایش لیست پیوندی



پیمایش درخت دودویی

تعریف پیمایش درخت

❖ می‌خواهیم با حرکت روی یالهای یک درخت، همه گره‌های آن را یک بار ملاقات کنیم.

□ پیمایش سطحی BFS

□ پیمایش عمقی DFS

• پیش ترتیب Preorder

• میان ترتیب Inorder

• پس ترتیب Postorder

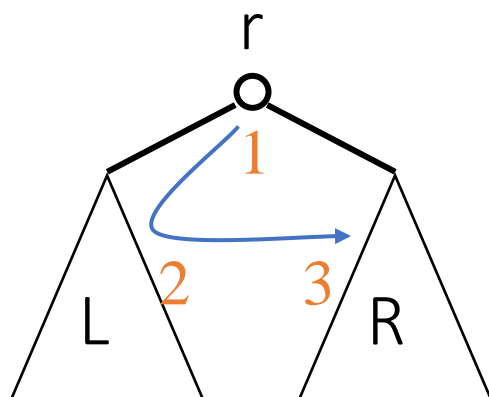
پیمایش درخت دودویی

□ طی کردن هر گره درخت يك و فقط يك بار

□ 6 ترکیب مختلف کنار هم قرار دادن $r(\text{root}), L(\text{Left}), R(\text{Right})$

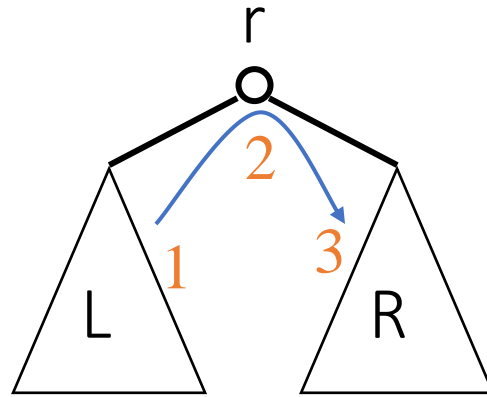
$rLR, rRL, LrR, LRR, RrL, RLr$

□ اگر زیر درخت چپ قبل از زیر درخت راست پیمایش شود، 3 حالت مختلف ایجاد می شود:



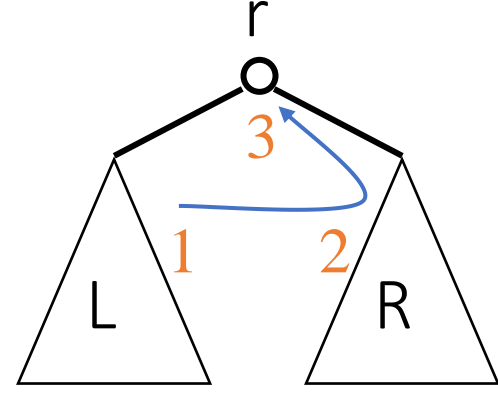
rLR

preorder



LrR

inorder

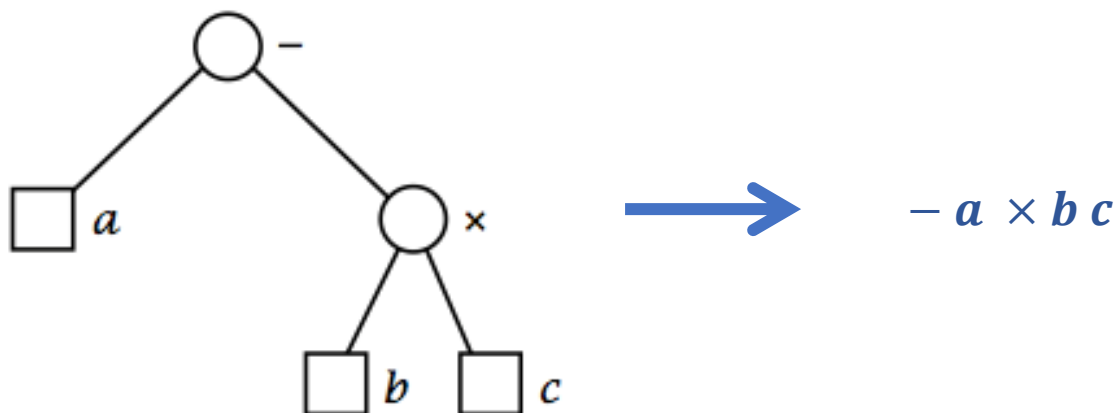


LRr

postorder

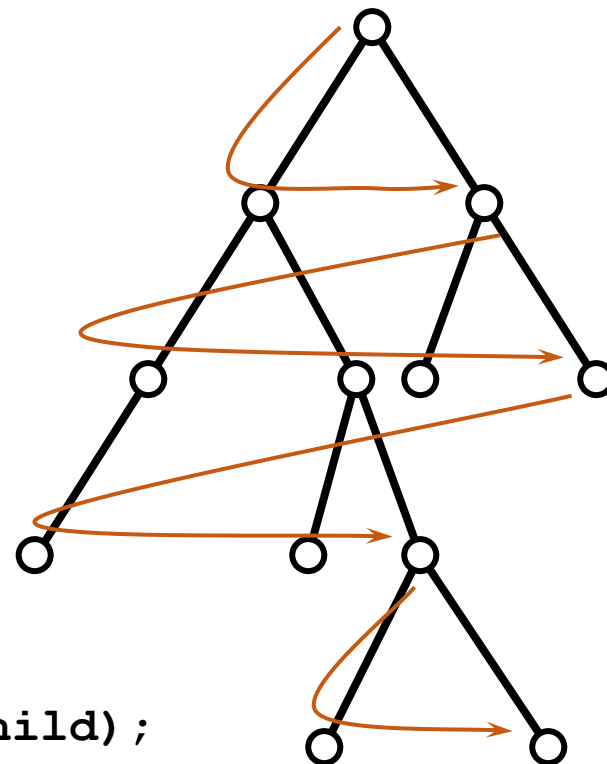
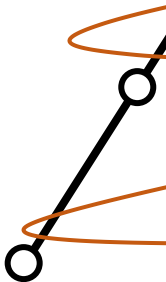
پیمایش سطحی

- حرکت از بالا به پایین
- حرکت از چپ به راست
- پیاده‌سازی به کمک صف



پیمایش سطحی

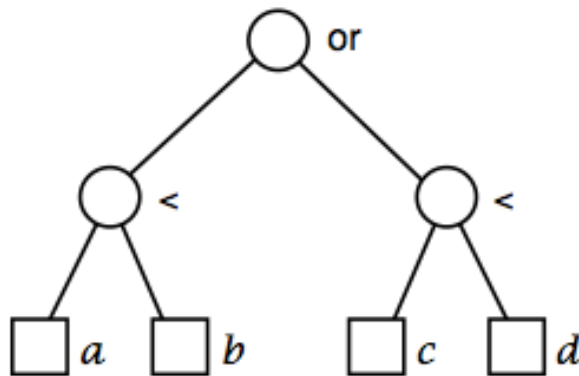
```
void level_order(tree_pointer ptr)
{
    int front = rear = 0;
    tree_pointer queue[MAX_QUEUE_SIZE];
    if (!ptr) return; /* empty tree */
    addq(front, &rear, ptr);
    for (;;) {
        ptr = deleteq(&front, rear);
        if (ptr) {
            printf(ptr.data);
            if (ptr.left_child)
                addq(front, &rear, ptr.left_child);
            if (ptr.right_child)
                addq(front, &rear, ptr.right_child);
        }
        else break;
    }
}
```



پیمایش Preorder

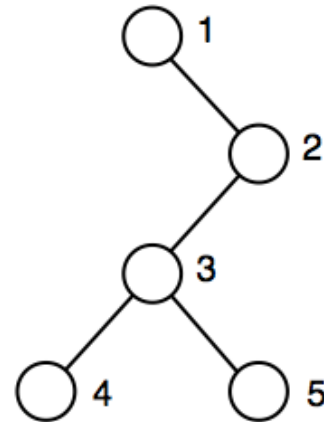
□ پیمایش Preorder با نگاه بازگشتی

- ملاقات ریشه
- اگر زیر درخت چپ داشتیم ← پیمایش چپ به روش Preorder
- اگر زیر درخت راست داشتیم ← پیمایش راست به روش Preorder



$or < a b < c \textcircled{d}$

راست ترین برگ



$\textcircled{1} 2 3 4 5$

ریشه

پیمایش Preorder

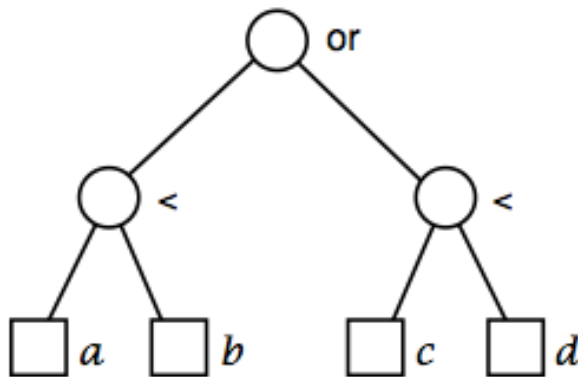
□ الگوریتم

- بازگشتی
- غیر بازگشتی به کمک پشته
- Push به تعداد فرزندان راست
- Pop به تعداد فرزندان راست

```
void preorder(tree_pointer ptr)
/* preorder tree traversal */
{
    if (ptr) {
        printf("%d", ptr->data);
        preorder(ptr->left_child);
        preorder(ptr->right_child);
    }
}
```


پیمایش Inorder

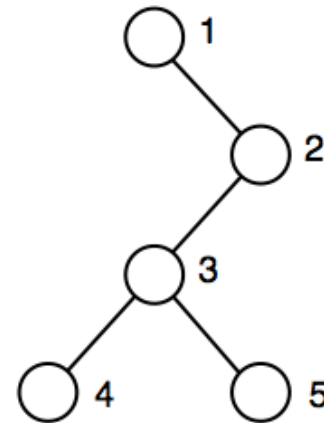
- اگر زیر درخت چپ داشتیم ← پیمایش **چپ** به روش Inorder
- ملاقات **ریشه**
- اگر زیر درخت راست داشتیم ← پیمایش **راست** به روش Inorder



$$\textcircled{a} < b \text{ or } c < d$$

↓

چپ‌ترین گره در امتداد ریشه



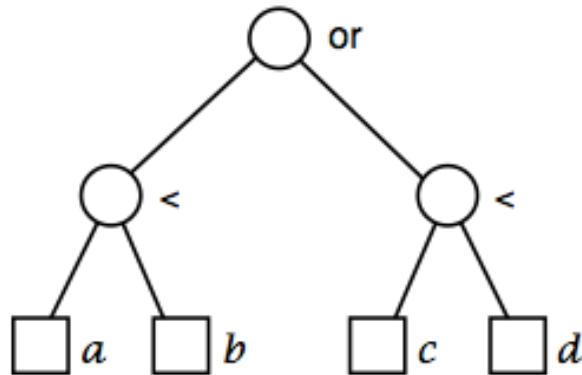
$$1 \ 4 \ 3 \ 5 \ \textcircled{2}$$

↓

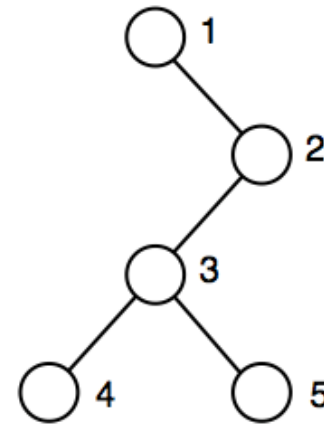
راست‌ترین گره در امتداد ریشه

پیمایش Postorder

- اگر زیر درخت چپ داشتیم ← پیمایش **چپ** بروش Postorder
- اگر زیر درخت راست داشتیم ← پیمایش **راست** بروش Postorder
- ملاقات **ریشه**



$a \ b < c \ d < or$
 ↓
 چپ ترین برگ

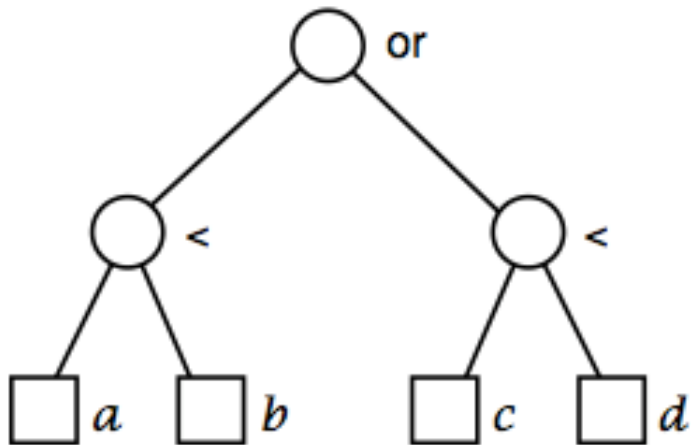


$4 \ 5 \ 3 \ 2 \ 1$
 ↓
 ریشه

نکته

ترتیب ملاقات برگ ها:

- در هر 3 پیمایش یکسان
- همواره از چپ به راست



Preorder
Inorder
Postorder

$or < a b < c d$

$a < b or c < d$

$a b < c d < or$

توابع پیمایش درخت

□ توابع بازگشتی پیمایش درخت

```
void inorder (tree_pointer ptr)
{
    if (ptr) {
        inorder(ptr.left_child);
        printf(ptr.data);
        inorder(ptr.right_child);
    }
}
```

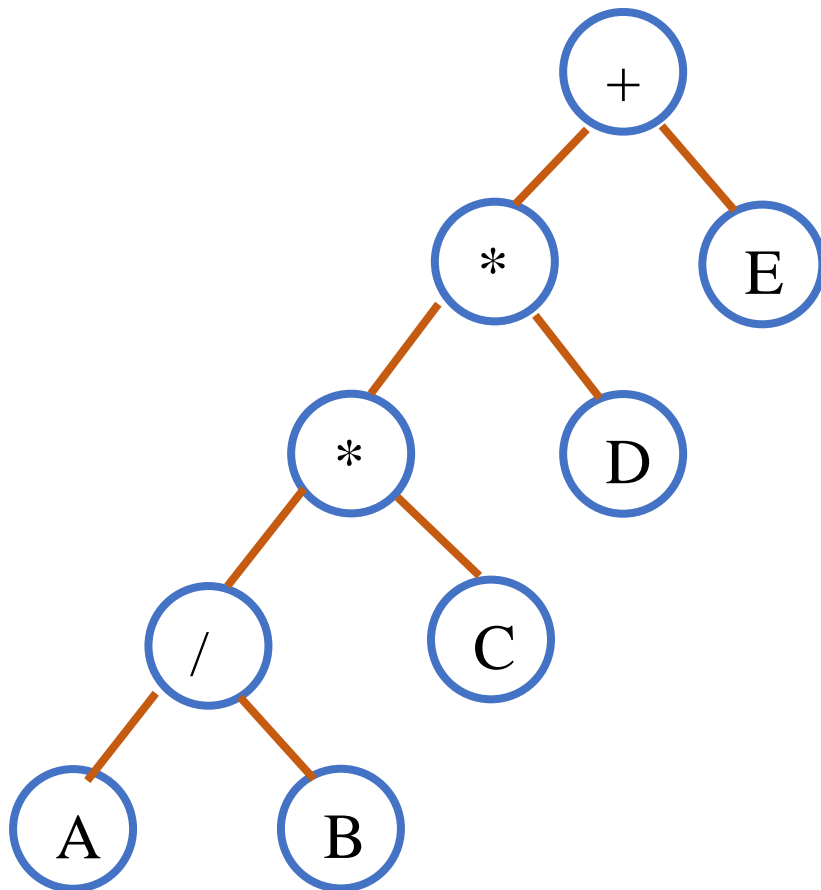
```
void preorder (tree_pointer ptr)
{
    if (ptr) {
        printf(ptr.data);
        preorder(ptr.left_child);
        preorder(ptr.right_child);
    }
}
```

پیمایش غیر بازگشتی inorder

```
void iter_inorder(tree_pointer node)
{
    int top = -1; /* initialize stack */
    tree_pointer stack[MAX_STACK_SIZE];
    for (;;) {
        for (; node; node = node.left_child)
            add(&top, node); // add to stack
        node = delete(&top); // delete from stack
        if (!node) break; // empty stack */
        printf(node.data);
        node = node.right_child;
    }
}
```

درخت عبارت

□ پیمایش‌های مختلف درخت عبارت، عبارت‌های میانوندی، پیشوندی و پسوندی را تولید خواهد نمود.



inorder traversal

$A / B * C * D + E$

infix expression

preorder traversal

$+ * * / A B C D E$

prefix expression

postorder traversal

$A B / C * D * E +$

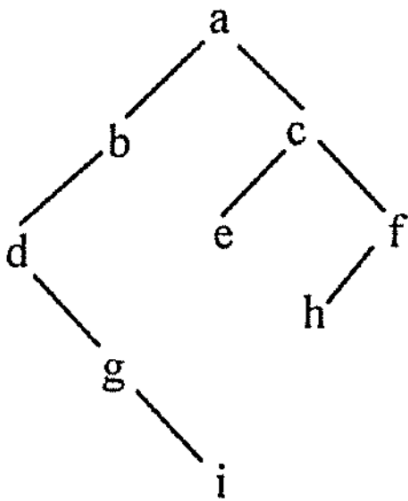
postfix expression

تمرین

(۱) درخت دودویی T که به صورت آرایه ای نمایش داده شده است را پیمایش Inorder نمایید.

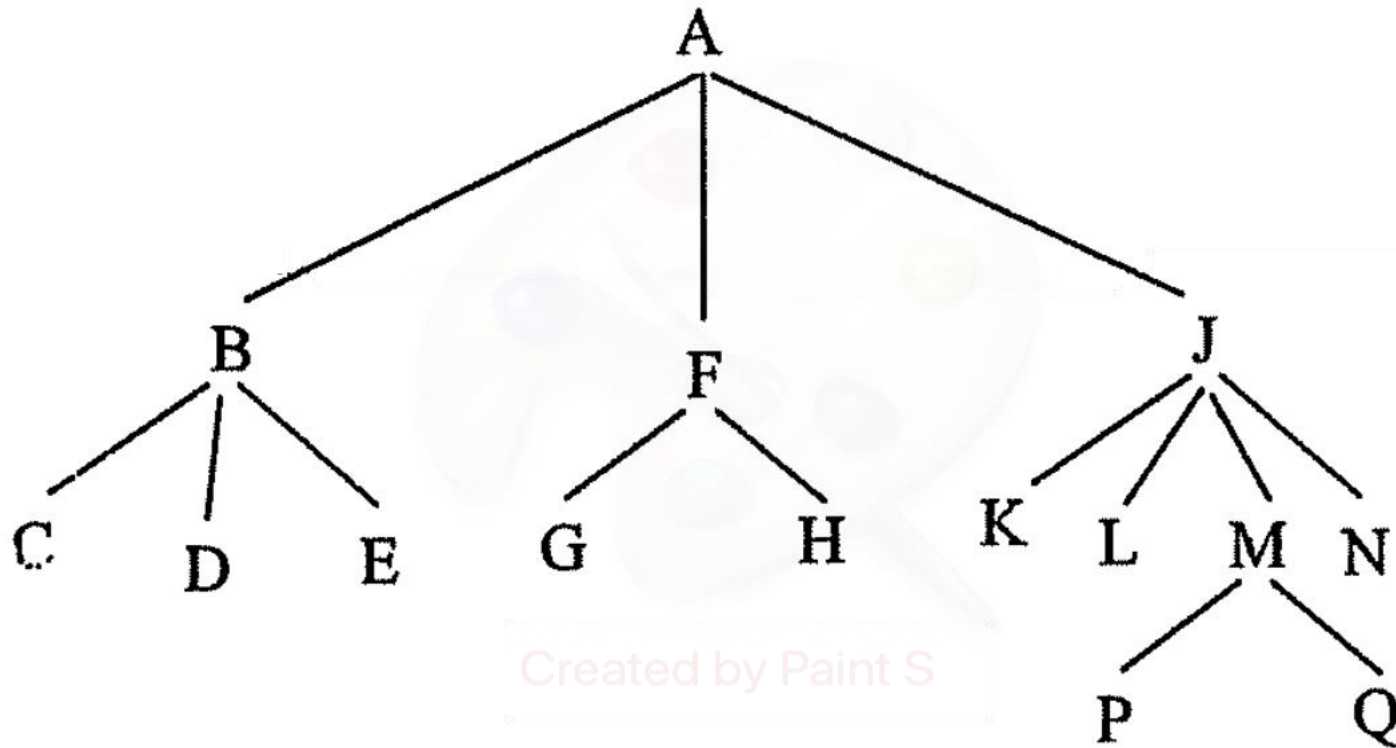
1	2	3	4	5	6	7	8	9	10				
a	b	c		d					e				

(۲) درخت زیر را به صورت Postorder پیمایش نمایید.



تمرین

۳) درخت عمومی زیر را در نظر بگیرید. درخت دودویی متناظر با آن را به دست آورید.

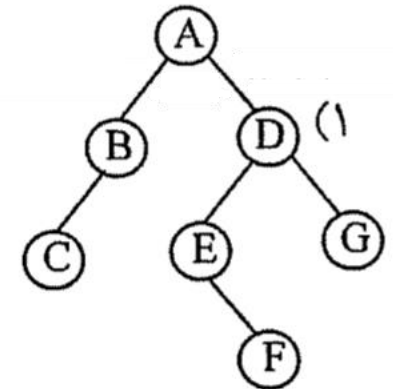
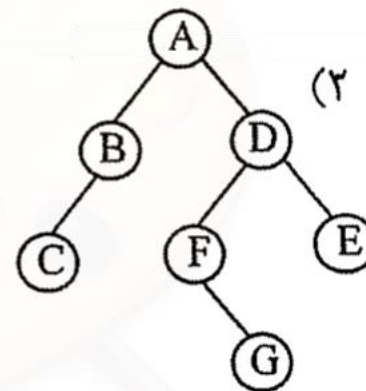
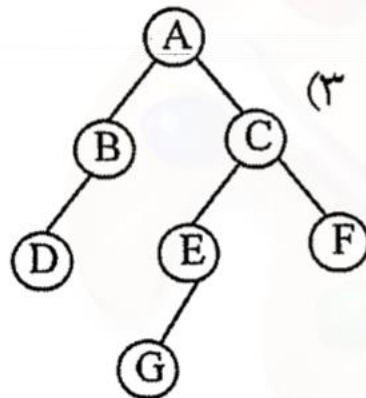
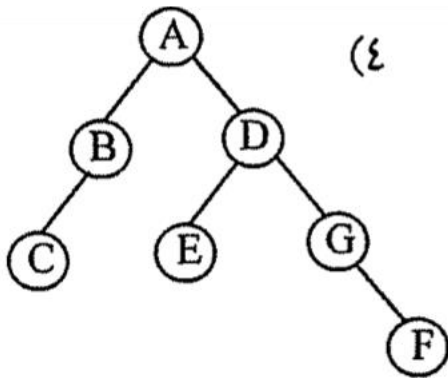


تمرین

(۴) اگر پیمایش Preorder و Inorder درخت دودویی به صورت زیر باشد، آن درخت کدام گزینه می باشد؟

Preorder : ABCDEFG

Inorder : CBAEFDG



تمرین

۵) لیست زیر مربوط به گره های درخت دودویی t را در نظر بگیرید. Root (ریشه درخت)، left (ریشه چپ) و right (ریشه راست) است. پیمایش Inorder این درخت را بنویسید.

INFO left right

root →

1	A	2	3
2	B	4	5
3	C	0	6
4	D	0	0
5	E	7	8
6	F	9	0
7	G	0	0
8	H	0	0
9	O	0	0

تمرین

۶) تابع زیر بر روی درخت دودویی T چه عملی انجام میدهد؟

پاسکال	C++ , C
<pre>Function n(T : tree) : integer; Begin if t = nil then n := 0 else if Rchild(t) = nil and Lchild (t) = nil then n:=1 else n:=n(Rchild(t)) + n (Lchild(t)) + 1; end;</pre>	<pre>int n (tree T) { if (t == NULL) return 0; else if (t->right == NULL && t->left == NULL) return 1; else return n (t -> right) + n(t -> left) + 1; }</pre>

الف) تعداد برگهای T را می شمارد.

ب) تعداد گرههای T را می شمارد.

ج) تعداد گرههای دو فرزندی T را می شمارد.

د) تعداد گرههای غیر برگ را می شمارد.

تمرین

(۷) تابع زیر بر روی درخت دودویی T چه عملی انجام میدهد؟

پاسکال

```
procedure h(root: Treeptr; var i:integer);
var
  x,y:Integer;
Begin
  if root = nil then
    i:=0
  Else Begin
    h(root ^.Left, x);
    h(root ^.Right, y);
    IF x>y then
      i:=x+1
    else
      i:=y+1
  END
END;
```

C++

```
void h(Treeptr r, int &i)
{
  int x,y;
  if (r == NULL) i = 0;
  else {
    h (r^.left, x);
    h (r^.right, y);
    if (x > y) i = x + 1;
    else i = y + 1;
  }
}
```

الف) تعداد برگهای T را می شمارد.

ب) تعداد گرههای T را می شمارد.

ج) تعداد شاخه‌های درخت T را می شمارد.

د) تعداد سطوح درخت T را می شمارد.