# درخت ها

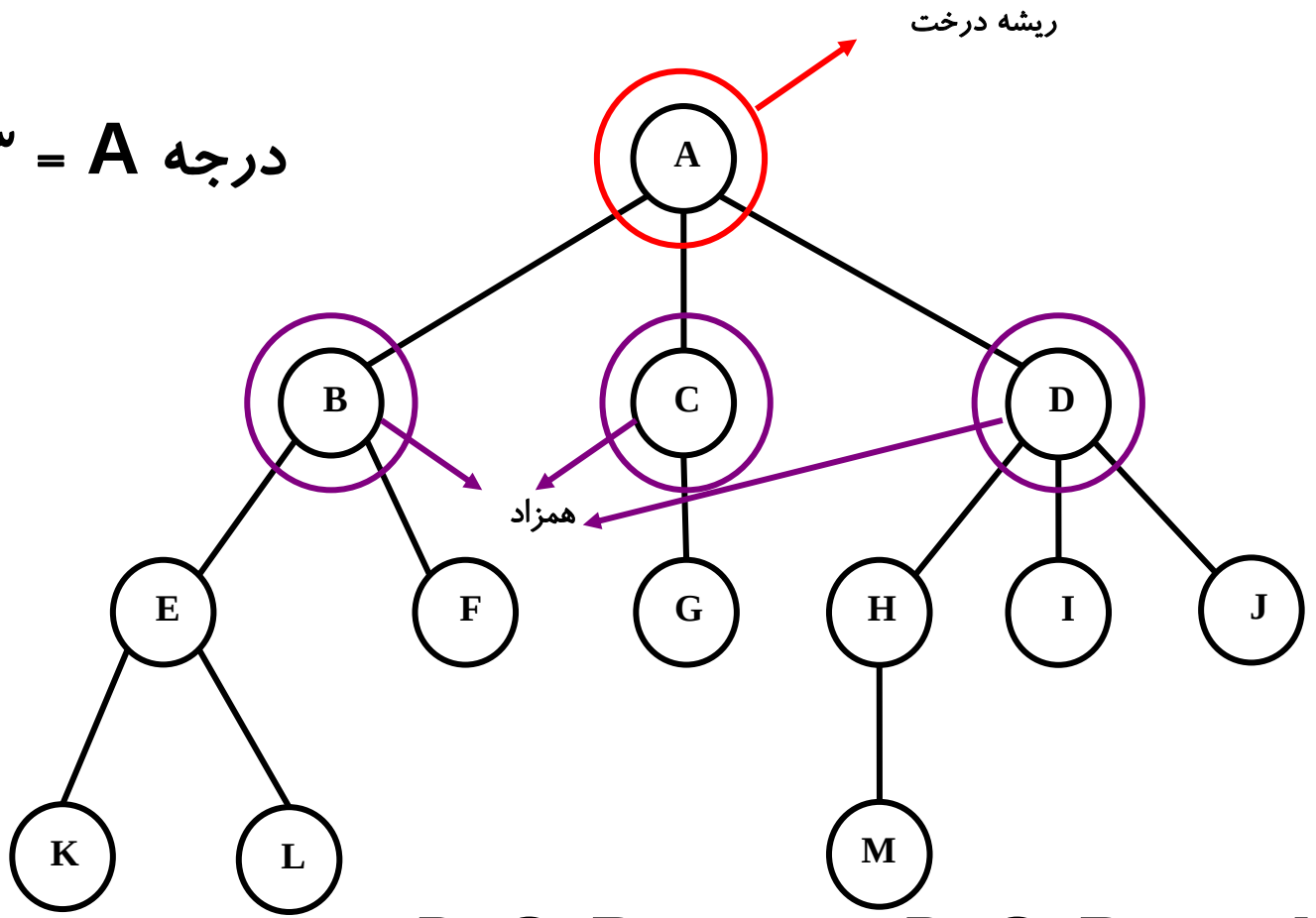
# درخت

❖ ساختار درختی یعنی مجموعه داده های سازماندهی شده ای که عناصر اطلاعاتی شان به وسیله انشعاباتی با هم رابطه داشته باشند.

❖ درخت مجموعه محدودی از یک یا چند گره به صورت زیر می باشد :

- دارای گره خاصی به نام ریشه باشد.
- بقیه گره ها به $n \geq 0$ مجموعه مجزا $T_1,....T_n$ تقسیم شده که هر یک از این مجموعه ها خود یک درخت هستند. $T_1,....T_n$ زیر درختان ریشه نامیده می شوند.

# درخت




A والد B، C، D است .  B، C، D همزادند.

# اصطلاحات درخت

**گره :** گره به عنصر حاوی اطلاعات و انشعابات به دیگر عناصر ، اطلاق می شود.

**درجه گره :** تعداد زیر درخت های یک گره را درجه آن گره می نامند.

**والد :** گره ای که دارای زیر درختانی است را والد ریشه های زیر درختان گویند .

**فرزندان گره :** ریشه های زیر درختان ، فرزندان آن گره نامیده می شوند.

# اصطلاحات درخت

گره های همزاد :فرزندان یک گره ، گره های همزاد یا هم نیا نامیده می شوند.

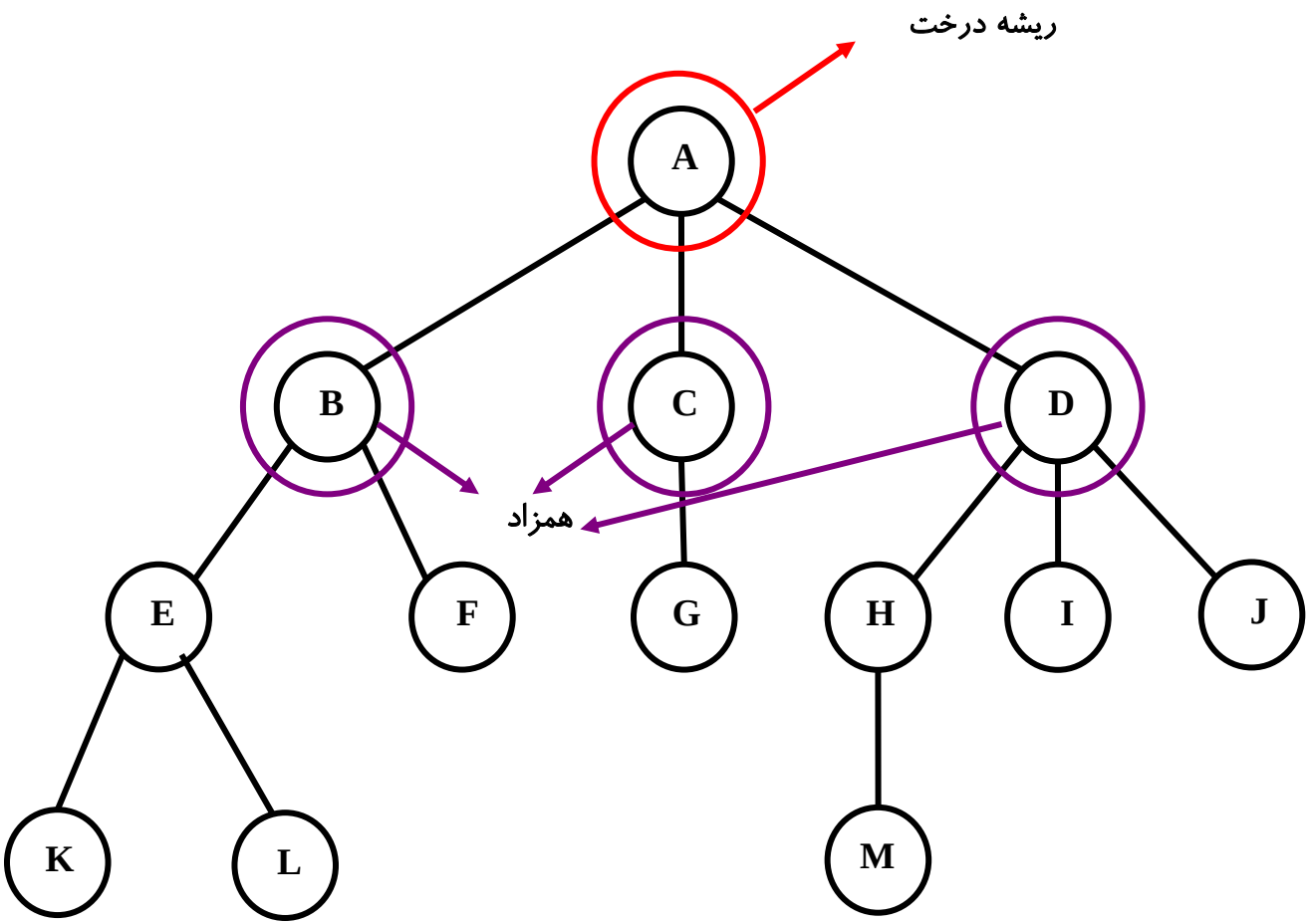
اجداد گره : اجداد یک گره ، گره هایی هستند که در مسیر طی شده از ریشه تا آن گره وجود دارند.

سطح گره : سطح یک گره  بدین صورت تعریف می شود که ریشه در سطح یک قرار می گیرد. برای تمامی گره های بعدی ، سطح گره برابر است با  سطح والد به اضافه یک .

ارتفاع درخت : ارتفاع یا عمق یک درخت به بیشترین سطح گره های آن درخت گفته می شود.

# درخت

❖ یک راه نمایش درخت ، استفاده از لیست است .




(A(B(E(K ،L)،F)،C(G)،D(H(M)،I،J)))

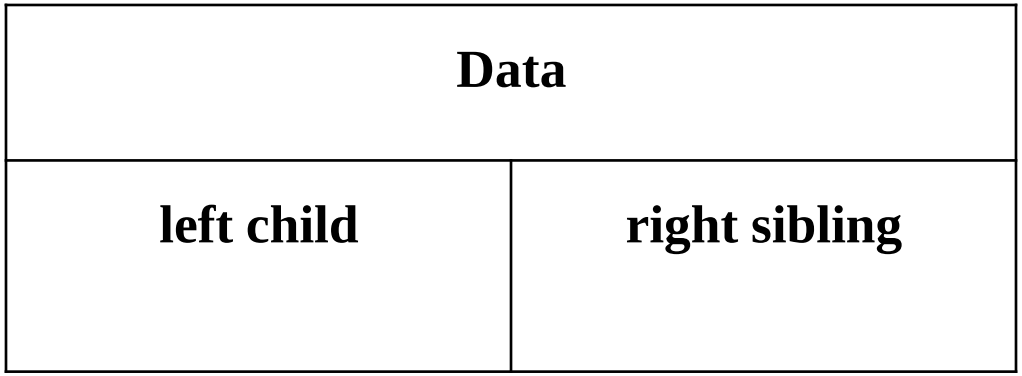
# درخت

شکل زیر یک ساختار ممکن برای نمایش لیست را نشان می دهد :

| data | link 1 | link 2 | … | Link n |
|---|---|---|---|---|

# درخت دودویی

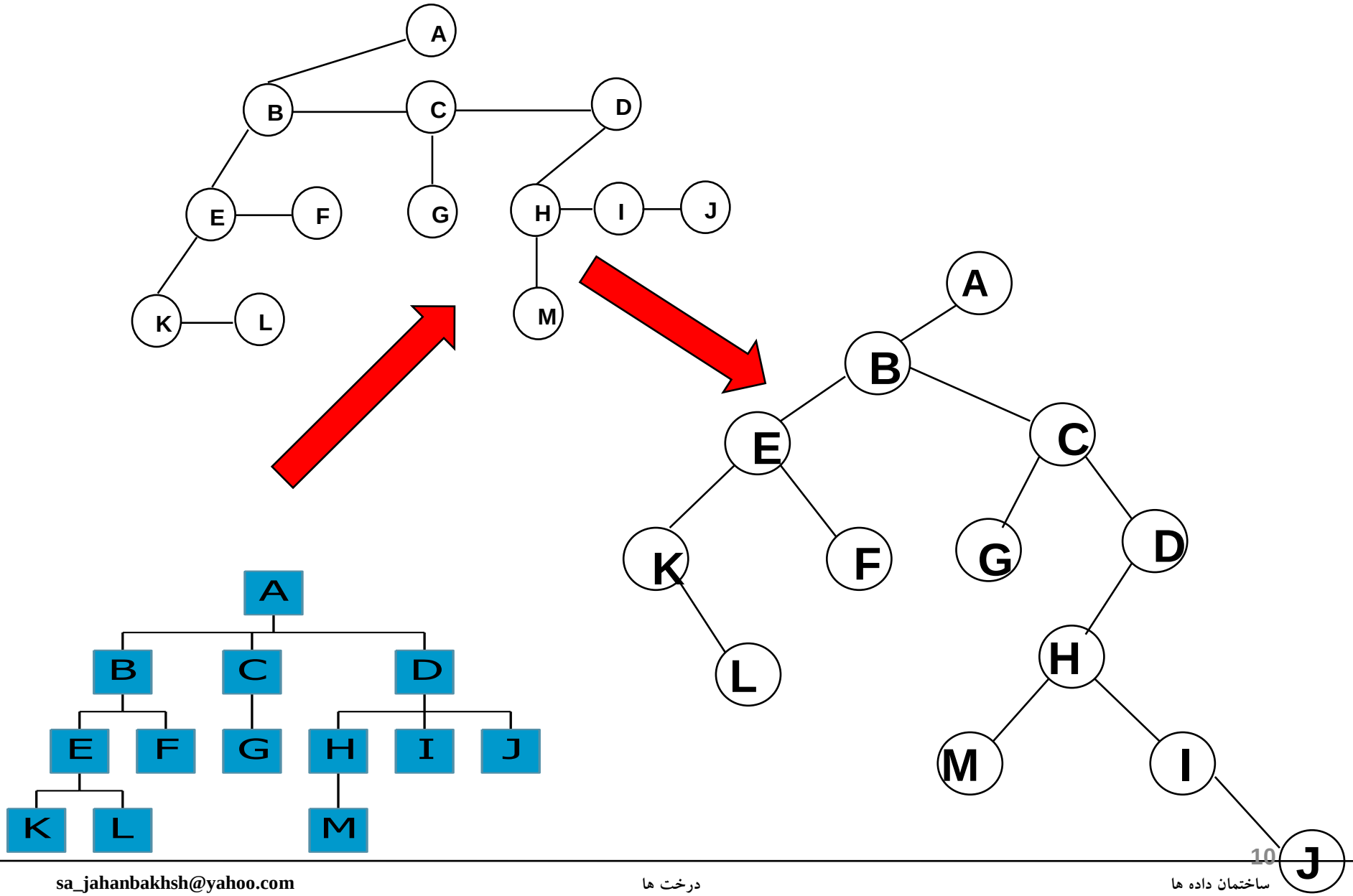
- یک درخت دودویی یا تهی است یا حاوی مجموعه ای محدود از گره ها شامل یک ریشه و دو زیر درخت دودویی است. این درخت ها زیر درخت های چپ و راست نامیده می شوند.
- مشخصه اصلی یک درخت دودویی بدین شکل است که هر گره آن حداکثر دو انشعاب دارد یعنی گره هایی که درجه ای بیشتر از دو نداشته باشند.
- برای درخت های دودویی زیر درخت سمت چپ و راست با یکدیگر متمایز است.



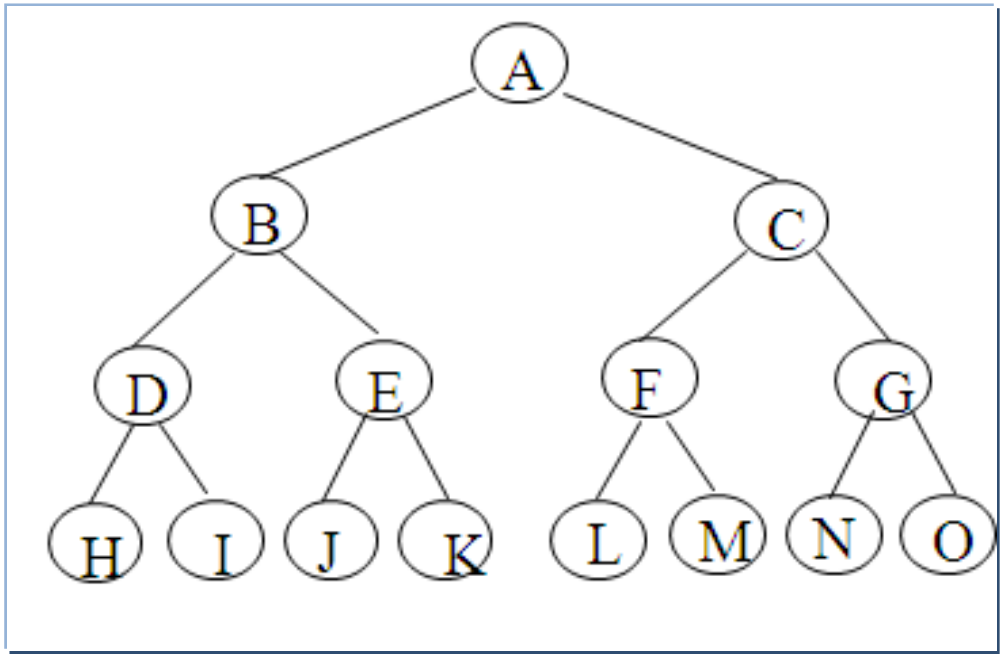
# تفاوت درخت دودویی با درخت عادی

- در هیچ درخت عادی صفر گره وجود ندارد ، اما درخت دودویی تهی وجود دارد.
- در یک درخت دودویی ترتیب فرزندان دارای اهمیت بوده در حالی که در درخت عادی به این صورت نیست.

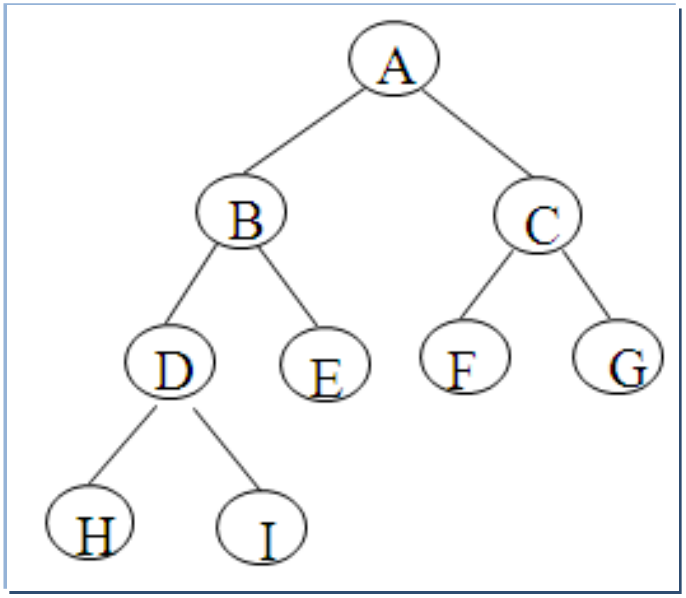
# تبدیل درخت عادی به درخت دودویی



# درختان دودویی




درخت دودویی پر



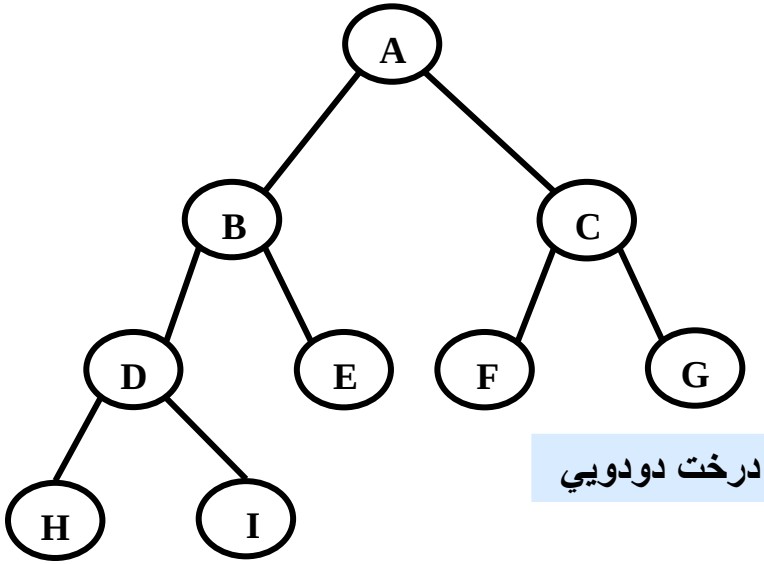

درخت دودویی کامل

# نمایش درختان دودویی

نمایش درخت دودویی به دو صورت است :

- نمایش آرایه
- نمایش لیست

# نمایش آرایه ای درختان دودویی




نمایش آرایه اي درخت دودويي

| | |
|---|---|
| [1] | A |
| [2] | B |
| [3] | - |
| [4] | C |
| [5] | - |
| [6] | - |
| [7] | - |
| [8] | D |
| [9] | - |
| . | . |
| . | . |
| . | E |
| [16] | |

# نمایش لیست پیوندی  درختان دودویی

- مادامیکه نمایش ترتیبی (آرایه ای) برای درختان دودویی کامل مناسب به نظر می رسد ، ما برای بسیاری از درختان دیگر باعث اتلاف حافظه می شود به علاوه ، این روش از نارسایی های موجود در نمایش ترتیبی نیز برخوردار است. درج یا حذف گره های یک درخت ، مستلزم جابه جایی گره هاست که خود باعث تغییر شماره سطح گره ها می شود. این مسایل می تواند با به کارگیری نمایش پیوندی به آسانی حل شود.

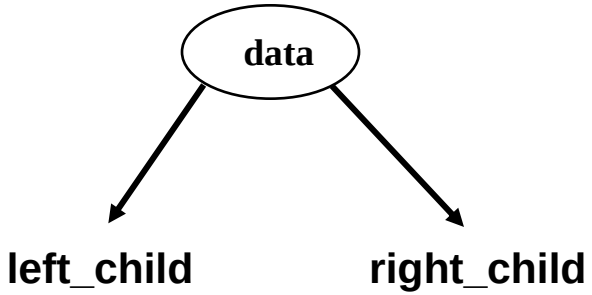
# نمایش لیست پیوندی درختان دودویی

با این روش هر گره سه فیلد خواهد داشت :

left_child، data، right_child

که در زبان C به شرح زیر تعریف می شوند :

```
typedef struct node *tree_pointer;
typedef struct node {
int data;
tree_pointer left_child ،right_child;
};
```


نمایش یک گره درخت دودویی

# پیمایش درخت دودویی

❖ به هنگام پیمایش یک درخت دودویی ، با هر گره و زیردرختانش به طرز مشابهی رفتار کنیم. اگر R، V، L به ترتیب حرکت به چپ ، ملاقات کردن یک گره ( برای مثال ، چاپ فیلد داده آن گره) و حرکت به راست باشد، آنگاه شش ترکیب ممکن برای پیمایش یک درخت خواهیم داشت :

RLV، RVL، VRL، VLR، LRV، LVR

# پیمایش درخت دودویی

❖ اگر تنها حالتی را انتخاب کنیم که ابتدا به سمت چپ و بعد به سمت راست برود ، تنها سه ترکیب VLR، LRV، LVR خواهیم داشت. این سه حالت را با توجه به موقعیت V نسبت به L و R به ترتیب preorder، postorder، inorder می نامیم.

# پیمایش درخت دودویی

- در پیمایش postorder ، یک گره موقعی ملاقات و چاپ می شود که زیردرختان چپ و راست آن قبلا ملاقات شده باشند.

- در پیمایش preorder ، یک گره قبل از پیمایش زیردرختان چپ و راست ، ملاقات می گردد.

# پیمایش Inorder

❖ هنگامی که این پیمایش انتخاب می شود ، حرکت به سمت پایین به طرف چپ انجام می شود و این عمل تا آخرین گره صورت می گیرد سپس می توان گره را بازیابی کرد و بعد به سمت راست رفته و به همین ترتیب کار را ادامه پیدا می کند.

**این متناظر با شکل infix یک عبارت است.**

# پیمایش Inorder

```
Void  inorder (tree_pointer   ptr )
/*  inorder  tree   traversal */
{
   if  (ptr)  {
         inorder  ( ptr -> left_child );
         printf  (“ % d”  ‘ptr -> data );
         inorder  (ptr -> right_child);
   }
}
```

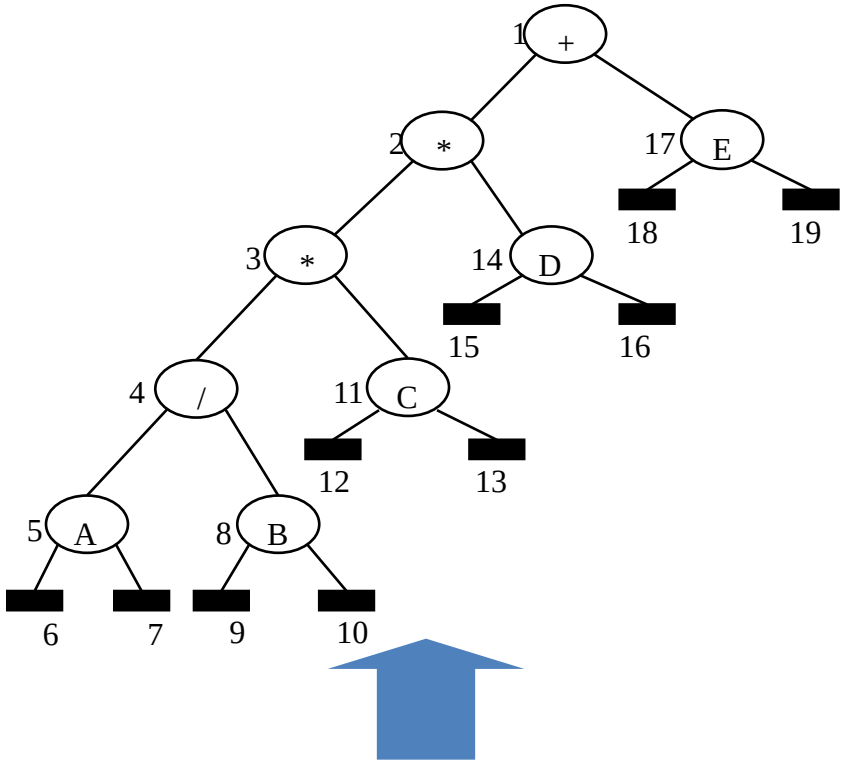
# پیمایش Preorder

❖ تابع preorder حاوی دستورات لازم برای شکل دوم پیمایش است.

❖ بر اساس این پیمایش ، گره را ابتدا بازیابی و ملاقات نموده و سپس انشعابات چپ را دنبال و تمام گره ها را بازیابی می کنیم. این فرآیند ادامه پیدا می کند تا به یک گره تهی برسیم. در این نقطه ، به نزدیکترین جدی که دارای یک فرزند راست باشد مراجعه و با این گره شروع خواهیم نمود.

# پیمایش درخت دودویی

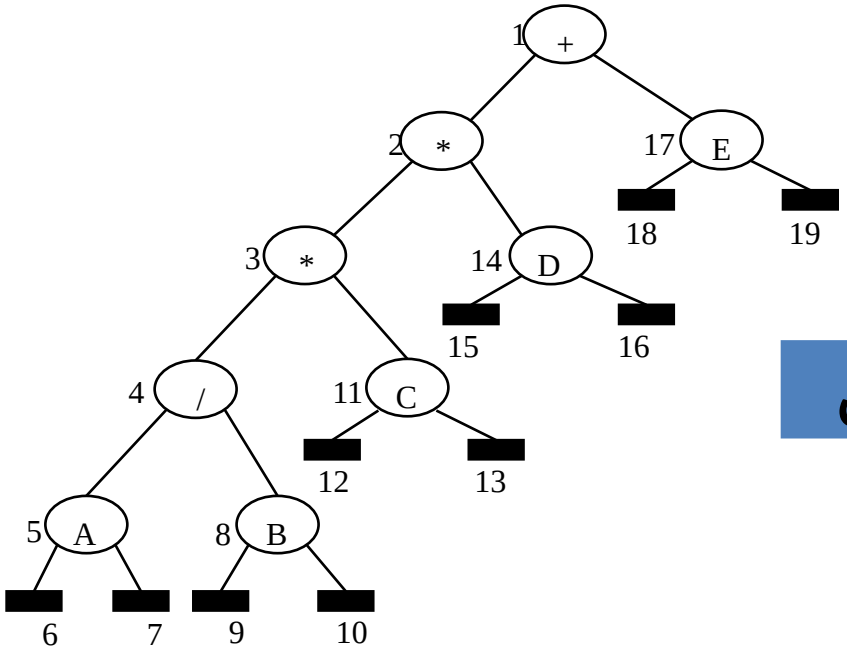
❖ درخت زیر حاوی یک عبارت ریاضی است : A/B*C*D*+E




**درخت دودویی برای یک عبارت محاسباتی**

# پیمایش Preorder

❖ با پیمایش preorder گره های درخت زیر خروجی به شکل زیر خواهند داشت :




خروجی

**+ * * / A B C D E**

**این به شکل یک عبارت prefix است.**

# پیمایش Preorder

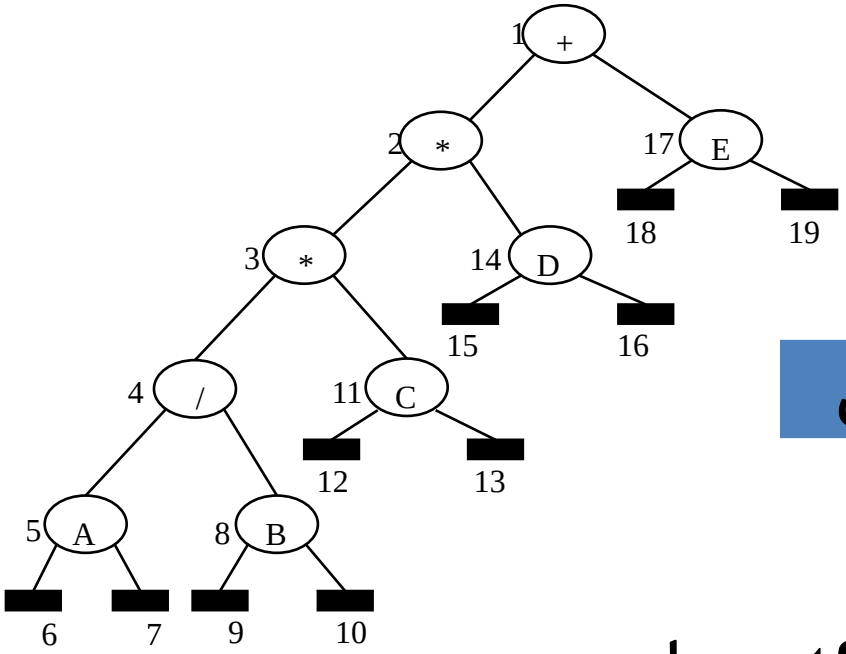
```
Void  preorder (tree_pointer   ptr )
/*  preorder  tree   traversal */
{
    if  (ptr)  {
            printf  (“ % d”  ،ptr -> data );
            preorder  ( ptr -> left_child );
            preorder  (ptr -> right_child);
    }
}
```

# پیمایش Postorder

❖ این پیمایش دو فرزند یک گره را قبل از بازیابی آن گره ملاقات و چاپ می کند. این مساله بدین مفهوم است که فرزندان یک گره قبل از خود آن گره بازیابی می گردد.

# پیمایش Postorder

❖ خروجی حاصل از پیمایش postorder شکل زیر به صورت زیر است :




خروجی

**A B/ C* D* E +**

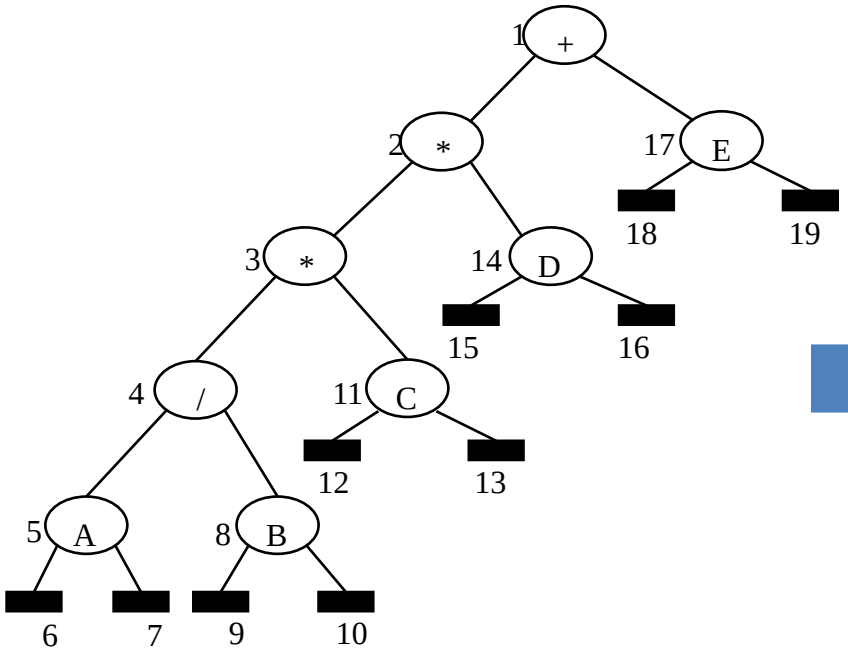
**این خروجی مانند یک عبارت postfix است.**

# پیمایش ترتیب سطحی

❖ این پیمایش ، ترتیب سطحی ، ابتدا ریشه را بازیابی ، سپس فرزند چپ ریشه و به دنبال آن فرزند راست ریشه بازیابی می گردد. این شیوه با بازیابی از گره منتهی الیه سمت چپ به سمت راست هر سطح جدید تکرار می گردد. این پیمایش از صف استفاده می کند.

# پیمایش ترتیب سطحی

❖ پیمایش ترتیب سطحی درخت زیر به صورت زیر است :




**+ *E *D / C A B**

# Heap ها

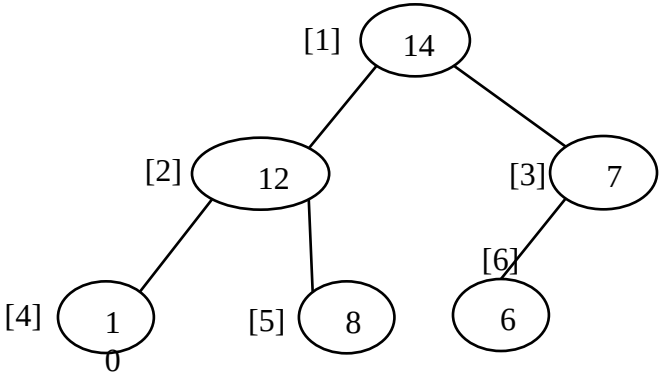
**max tree** درختی است که مقدار کلید هر گره آن کمتر از مقادیر کلیدهای فرزندانش نباشد.

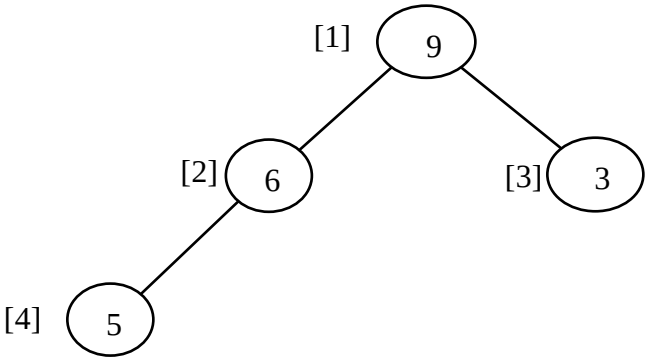
**max heap** یک درخت دودویی کامل است که یک max tree نیز می باشد.

**min tree** درختی است که مقدار کلید هر گره آن بیشتر از مقادیر کلیدهای فرزندانش نباشد.

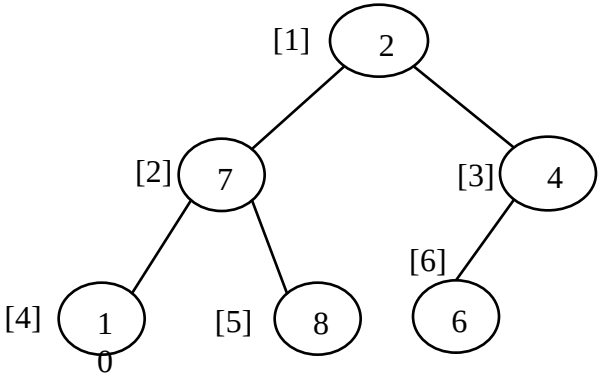
**min heap** یک درخت دودویی کامل است که در واقع یک min tree می باشد.

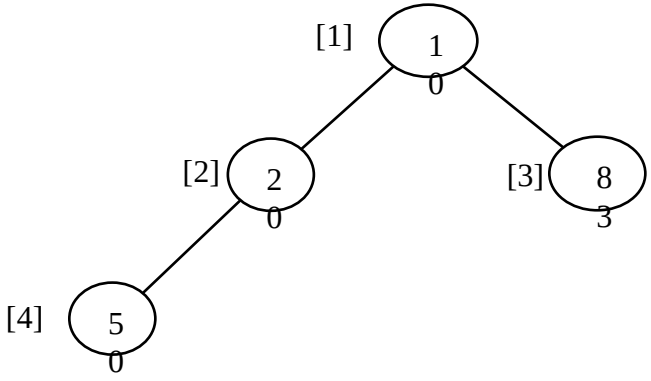
# Max Heap , Min Heap




مثال از max heap



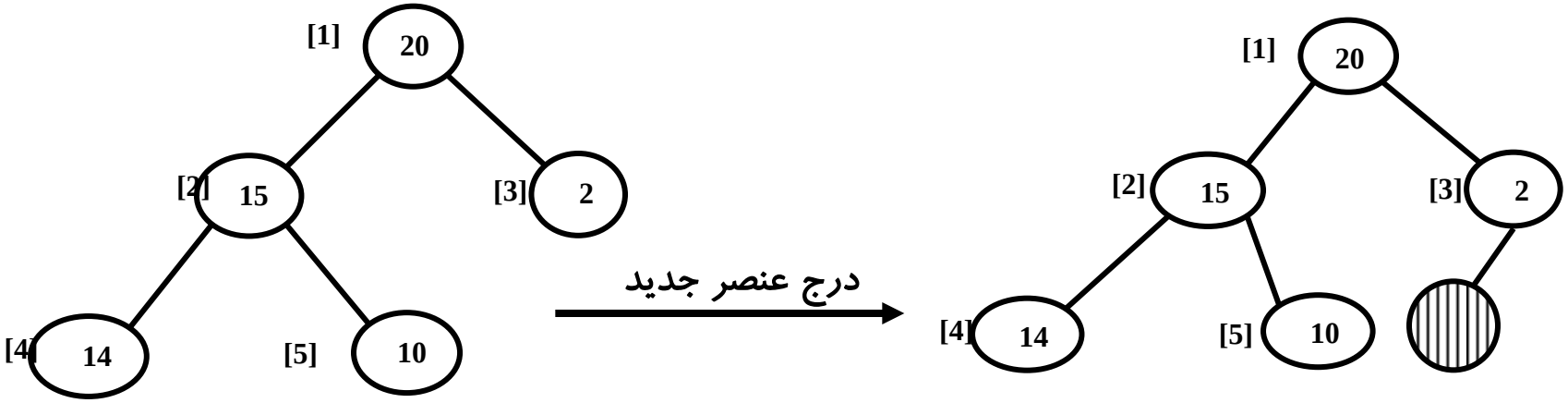




مثال از min heap

# اعمال بر روی Heapها

- ایجاد یک هرم(heap) تهی
- جایگذاری عنصر جدید به هرم (heap)
- حذف بزرگترین عنصر از هرم (heap)

# درج عناصر به داخل یک Max Heap



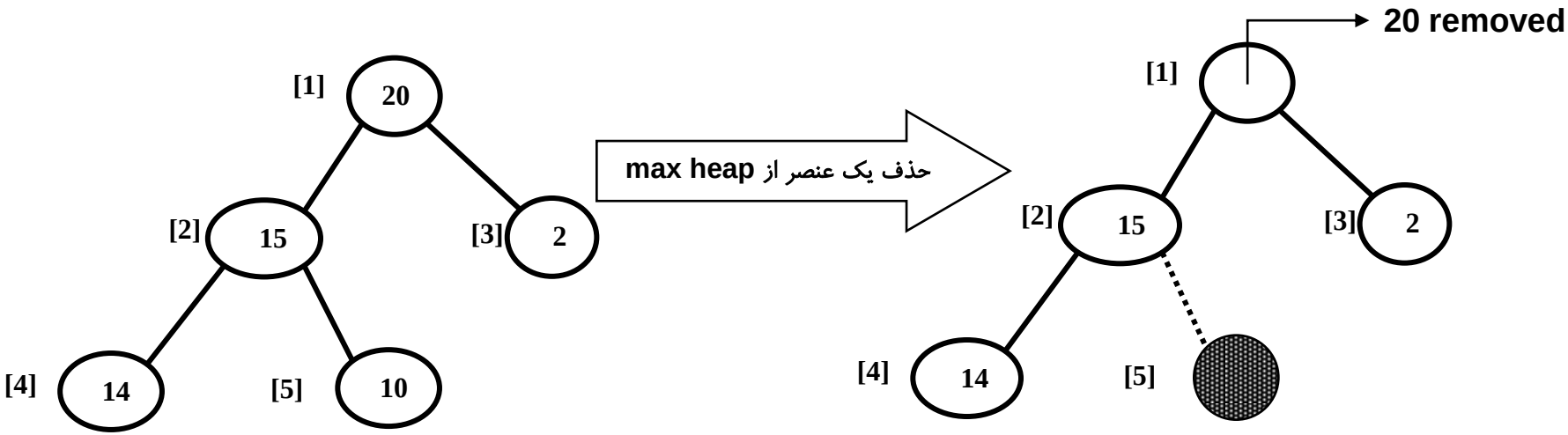

الف - درخت **heap** قبل از درج

ب - محل اولیه گره جدید

❖ اضافه کردن گره جدید در هر موقعیت دیگری ،تعریف heap را نقض می کند زیرا نتیجه یک درخت دودویی کامل نخواهد بود.

# حذف یک عنصر از یک Max Heap

❖ هنگامی که عنصری از max heap حذف می شود ، آن را از ریشه درخت heap می گیریم.

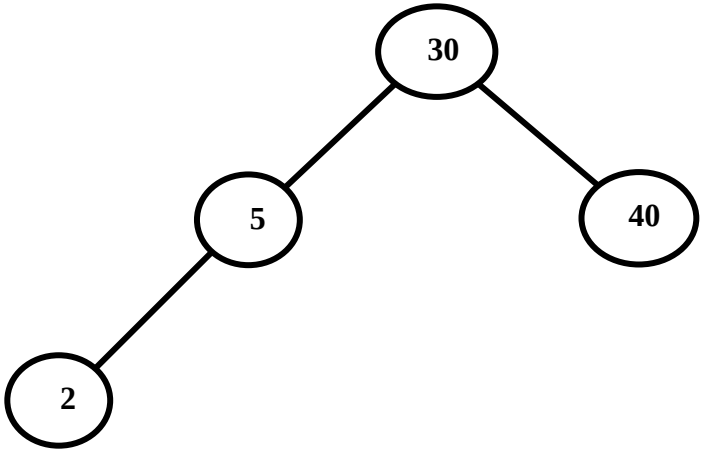


# درختان جستجوی دودویی

❖ یک درخت جستجوی یک درخت دودویی است که ممکن است تهی باشد. اگر درخت تهی نباشد خصوصیات زیر را برآورده می کند :

- هر عنصر دارای یک کلید است و دو عنصر نباید دارای کلید یکسان باشند ، در واقع کلیدها منحصر به فردند.
- کلیدهای واقع در زیردرخت غیرتهی چپ باید کمتر از مقدار کلید واقع در ریشه زیردرخت راست باشد.
- کلیدهای واقع در زیردرخت غیرتهی راست باید بزرگتر از مقدار کلید واقع در ریشه زیردرخت چپ باشد.
- زیردرختان چپ و راست نیز خود درختان جستجوی دودویی میباشند.

# درختان جستجوی دودویی




مثال

# جستجو در درخت جستجوی دودویی

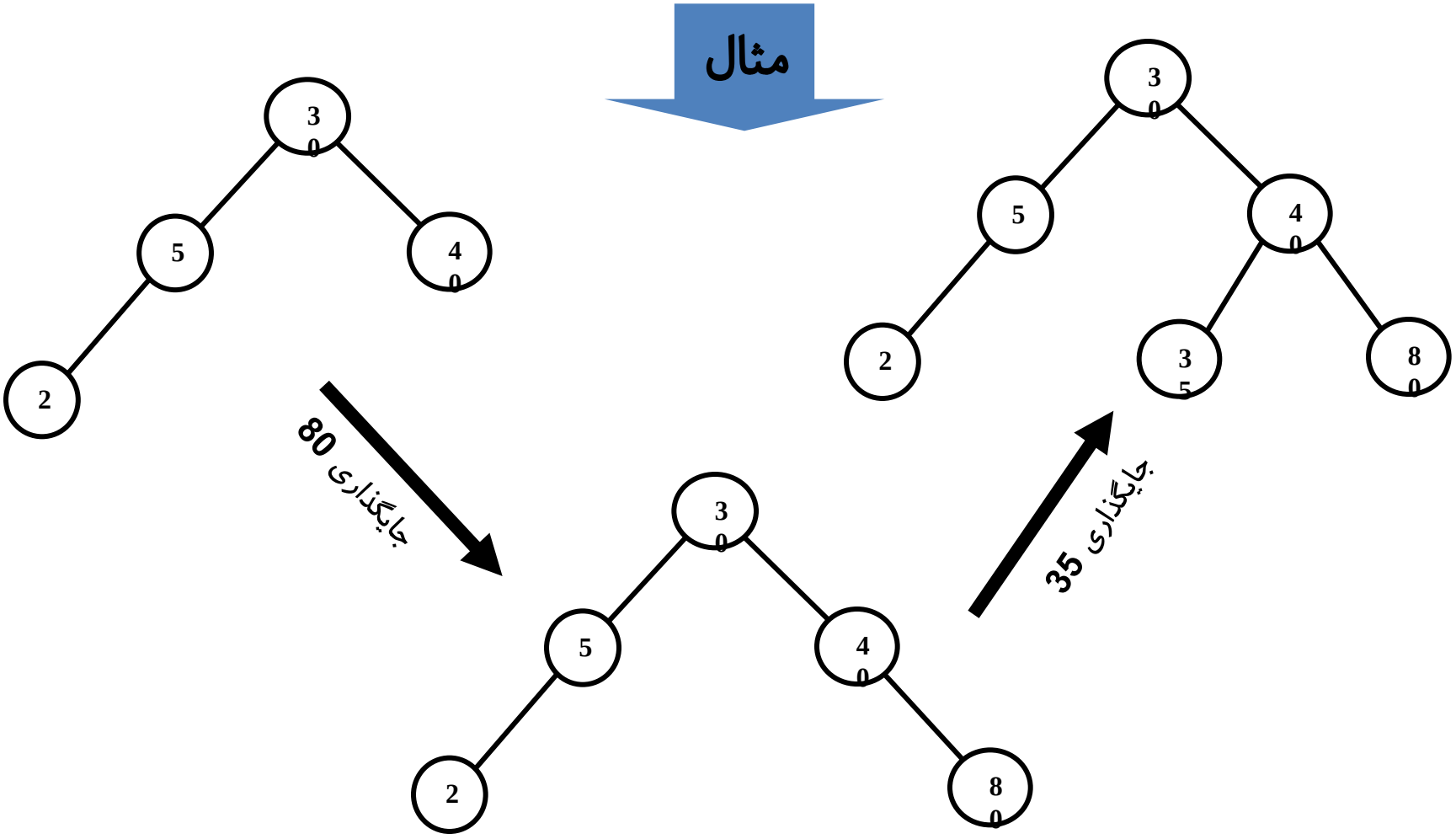
❖ فرض کنید خواسته باشیم دنبال عنصری با کلید key بگردیم. ابتدا از ریشه (root) شروع می کنیم ، اگر ریشه تهی باشد ، درخت جستجو فاقد هر عنصری بوده و جستجو ناموفق خواهد بود. در غیر این صورت keyرا با با مقدار کلید ریشه مقایسه کرده :

- اگر key کمتر از مقدار کلید ریشه باشد ، هیچ عنصری در زیردرخت راست وجود ندارد که دارای کلیدی برابر key باشد ، بنابراین زیردرخت چپ ریشه را جستجو می کنیم.
- اگر key بزرگتر از مقدار کلید ریشه باشد ، زیردرخت راست را جستجو می کنیم.

# درج عنصر به داخل یک درخت جستجوی دودویی

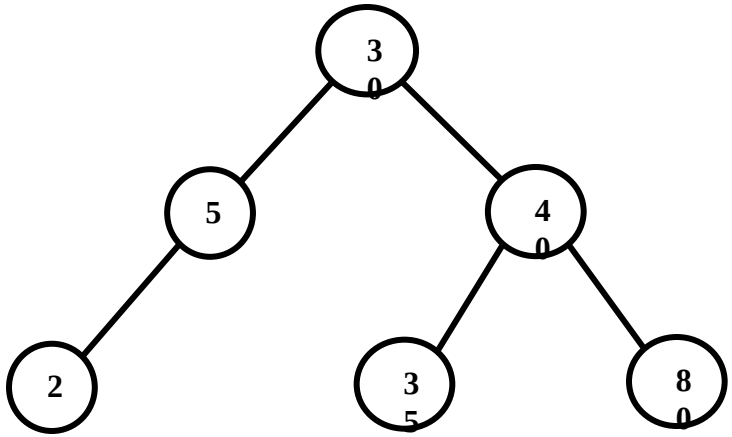
❖ برای درج عنصر جدیدی به نام key ، ابتدا باید مشخص نمود که آیا کلید با عناصر موجود متفاوت است یا خیر. برای انجام این کار باید درخت را جستجو کرد، اگرجستجو ناموفق باشد ، عنصر را در محلی که جستجو خاتمه پیدا نموده است ، درج می کنیم.

# درج عنصر به داخل یک درخت جستجوی دودویی



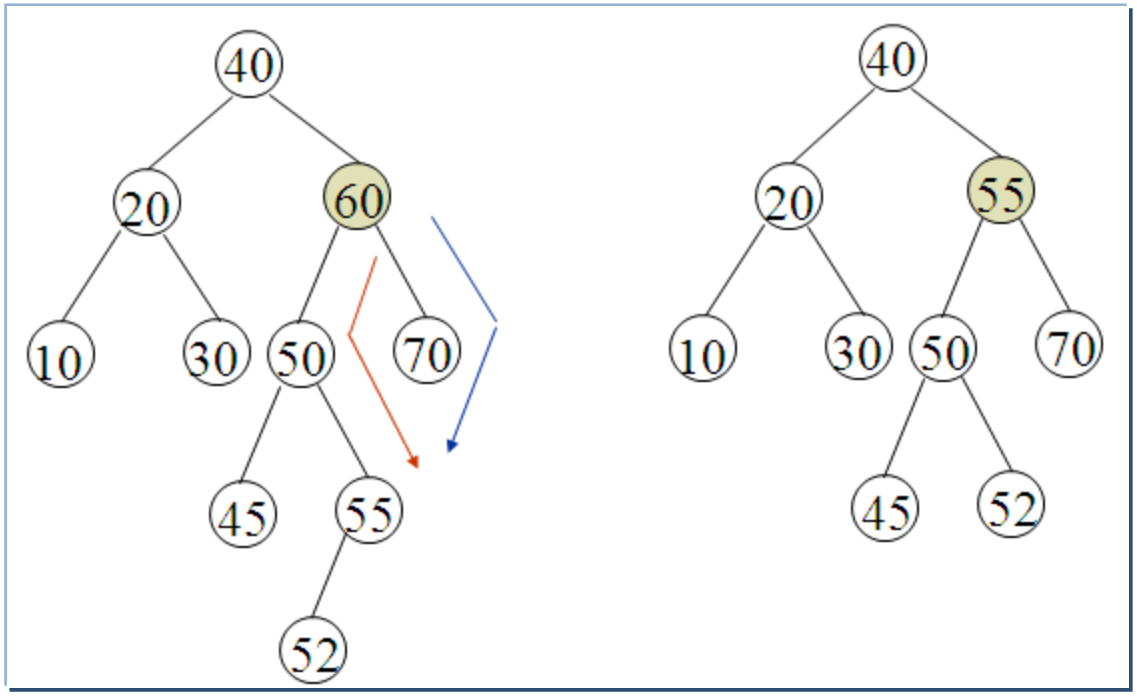
# حذف عنصر از یک درخت جستجوی دودویی

❖ برای حذف ۳۵ از درخت زیر باید فیلد فرزند چپ والد این گروه را برابر NULL قرار داده و گره را آزاد نمود :



# حذف عنصر از یک درخت جستجوی دودویی

❖ زمانی که یک گره با دو فرزند حذف می شوند ، گره را با بزرگترین عنصر در زیر درخت چپ و یا کوچکترین عنصر در زیردرخت راست آن گره جایگزین و تعویض کرد.




درخت بعد از حذف گره ۶۰