به نام خدا

ساختمان داده ها و الگوریتم ها

محمدمهدی گیلانیان صادقی

دانشگاه آزاد اسلامی واحد قزوین

نیمسال دوم ۱۴۰۱-۱۴۰۲

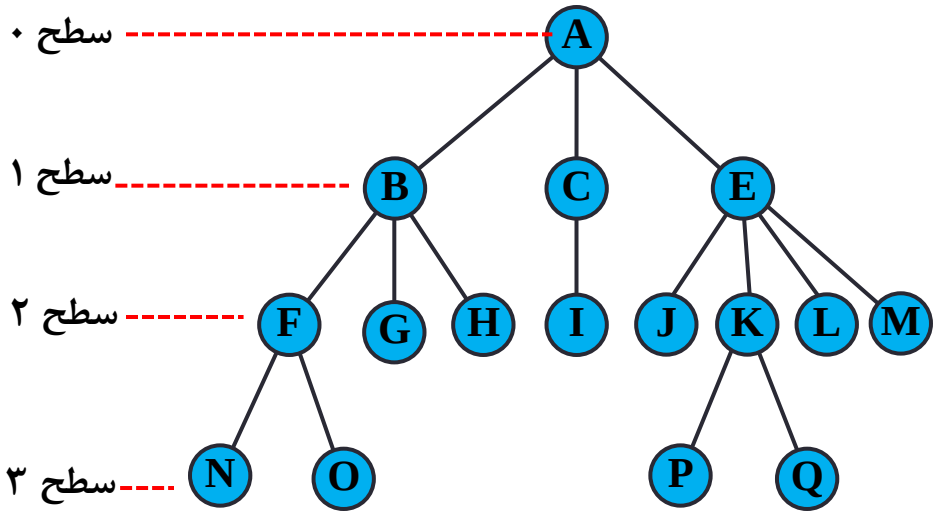
- **درخت (Tree):**

درخت مجموعه محدودی از یک یا چند گره است که دارای شرایط زیر می باشند:

۱- دارای گره ای به نام ریشه است.

۲- بقیه گره ها به مجموعه هایی مجزا تقسیم شوند که هر یک از این مجموعه ها خود یک درخت هستند.

- **اصطلاحات مربوط به درخت:**
  - گره، مسیر و طول مسیر:
  - درخت عمومی و درخت دودویی:
  - سطح درخت:
  - عمق گره و عمق درخت:
  - درجه گره و درجه درخت:
  - اجداد گره:
  - نسل های گره:
  - والد گره و فرزند گره:
  - گره برگ:
  - گره های هم زاد:



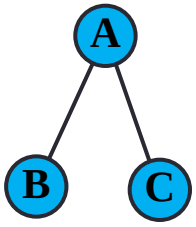
- **درخت دودویی:**

  - درخت دودویی یا تهی است یا حاوی مجموعه ای از گره ها شامل یک ریشه و دو زیر درخت دودویی چپ و راست است.

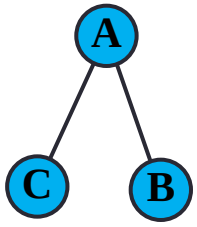
تفاوت درخت عمومی و دودویی:

۱- در درخت دودویی هر گره حداکثر دو فرزند دارد.
۲- در درخت عمومی ترتیب فرزندان مهم نیست، ولی در درخت دودویی ترتیب مهم است.
۳- درخت عمومی تهی وجود ندارد، ولی درخت دودویی تهی وجود دارد.

درختان دودویی الف و ب یکسان نیستند



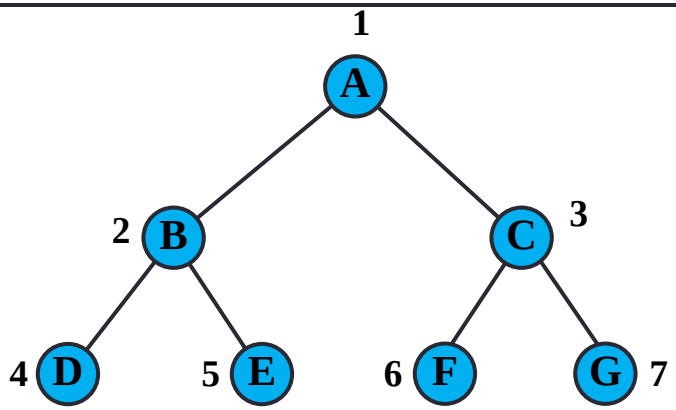
الف



ب

- **پیاده سازی درخت دودویی:**
  - با استفاده از آرایه
  - با استفاده از اشاره گره ها

- **پیاده سازی درخت دودویی با آرایه:**



0	1	2	3	4	5	6	7
-	A	B	C	D	E	F	G

نکات مهم:

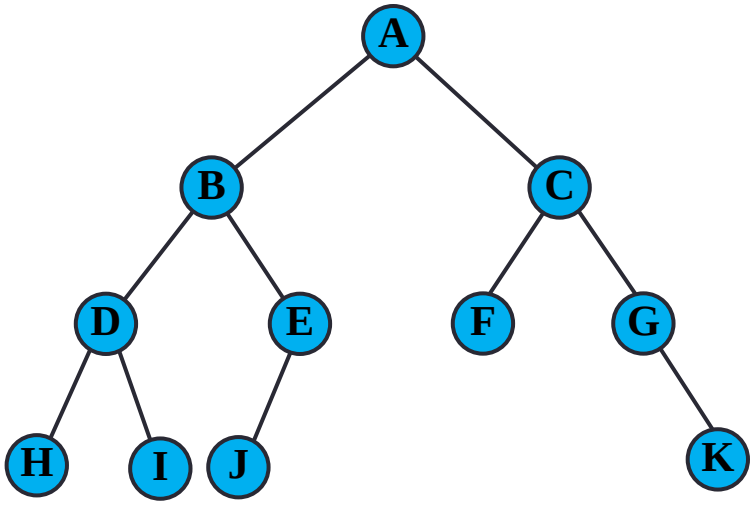
حداکثر تعداد گره ها در سطح iام یک درخت دودویی 2^i گره است.

حداکثر تعداد گره ها (n) در یک درخت دودویی به عمق d، $2^{d+1}-1$ است.

اگر شماره گره ای i (i!=1) باشد والد آن دارای شماره [i/2] است.

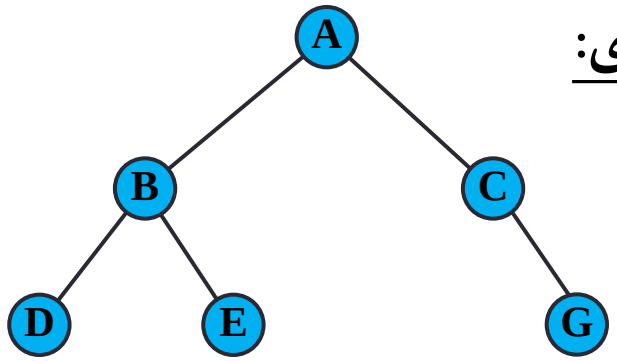
اگر شماره گره ای i باشد، فرزندان چپ و راست آن به ترتیب دارای شماره های 2i و 2i+1 (<=n) هستند.

- پیاده سازی درخت دودویی با آرایه:

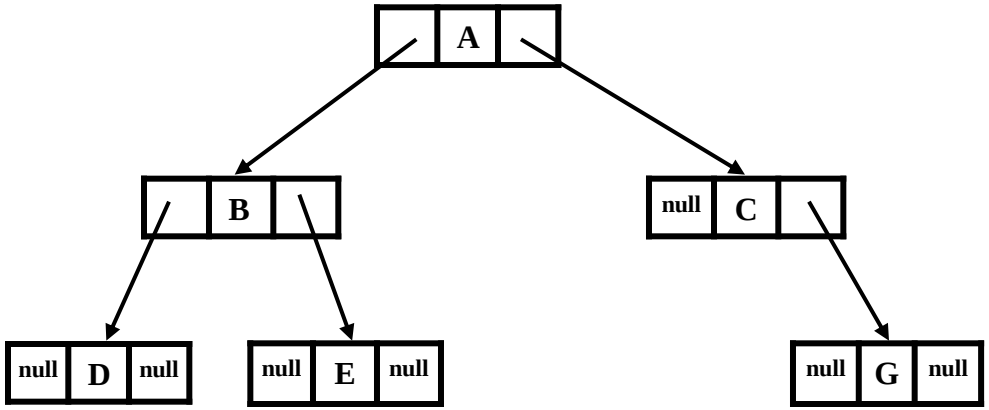


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-	A	B	C	D	E	F	G	H	I	J	-	-	-	-	K

- پیاده سازی درخت دودویی با لیست پیوندی:

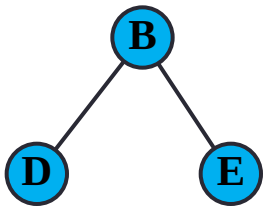



```
class treenode{
    friend class tree;
    private:
        treenode *left;
        int info;
        treenode *right;
};
class tree{
    public:
       // member functions
    private:
       treenode *root;
 };
```



- **پیمایش درخت دودویی:**

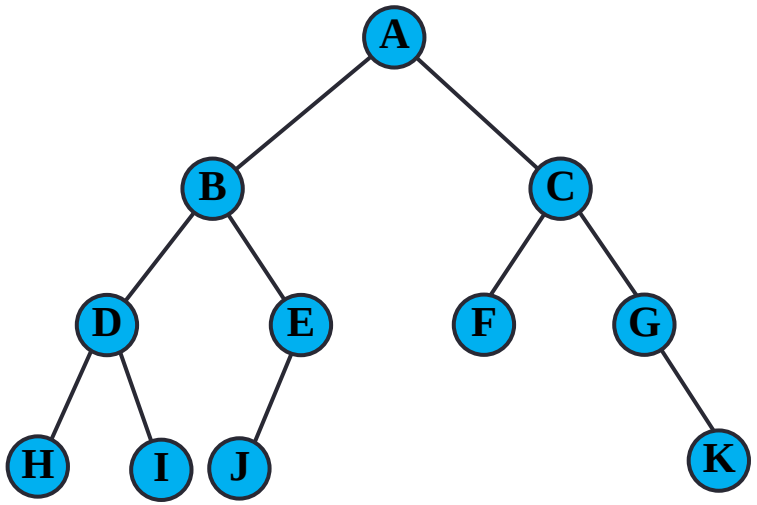
۱) روش پیشوندی یا preorder: ریشه، زیر درخت چپ، زیر درخت راست
۲) روش میانوندی یا inorder: زیر درخت چپ، ریشه، زیر درخت راست
۳) روش پسوندی یا postorder: زیر درخت چپ، زیر درخت راست، ریشه



preorder: B D E
inorder: D B E
postorder: D E B

- روش پیمایش پیشوندی یا **preorder**:

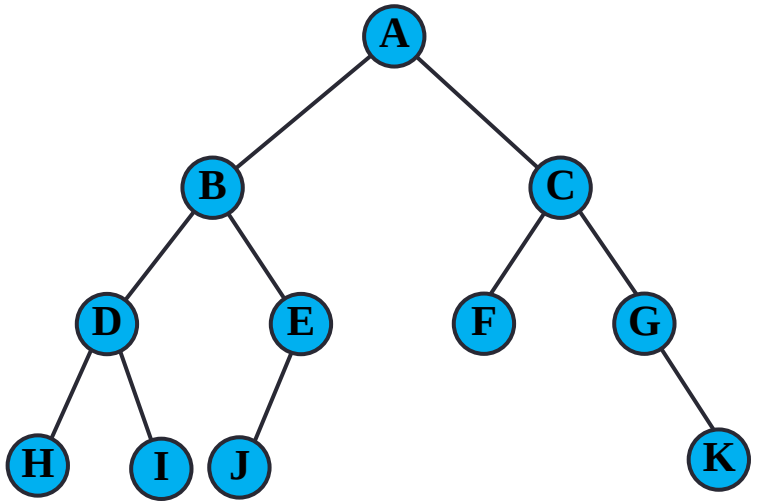
```
void  tree::preorder(tree *s)
 {
   if(s)
    {
      cout<<s->info;
      preorder(s->left);
      preorder(s->right);
     }
 }
```



preorder: A B D H I E J C F G K

- روش پیمایش میانوندی یا **inorder**:

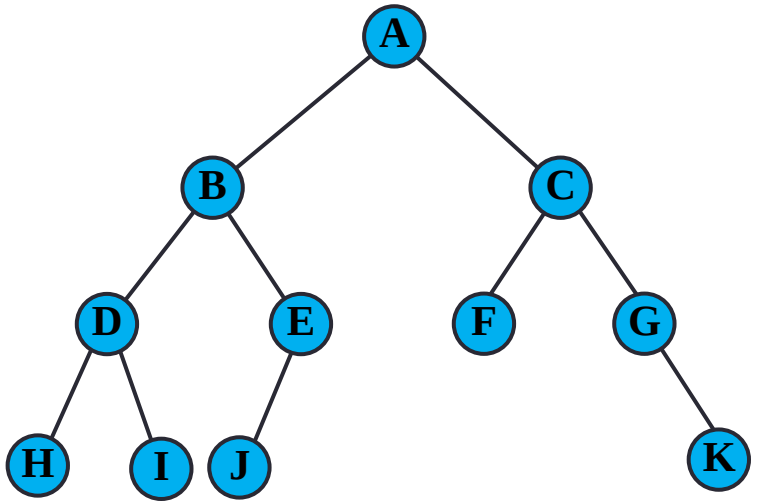
```
void  tree::inorder(tree *s)
 {
   if(s)
    {
      inorder(s->left);
      cout<<s->info;
      inorder(s->right);
     }
 }
```



inorder : H D I B J E A F C G K

- روش پیمایش پسوندی یا **postorder**:

```
void  tree::postorder(tree *s)
 {
   if(s)
    {
      postorder(s->left);
      postorder(s->right);
      cout<<s->info;
     }
 }
```



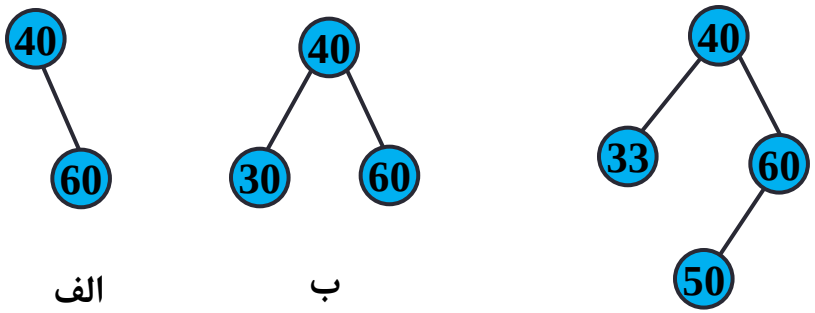
postorder :H I D J E B F K G C A

- **درخت جستجوی دودویی(Binary Search Tree- BST):**

درخت دودویی است که در آن برای هر گره X :
مقدار فرزند چپ X کوچکتر یا مساوی مقدار X، و مقدار فرزند راست X بزرگتر از X باشد.

عملیات اصلی:
- ایجاد درخت
- بررسی خالی بودن درخت
- درج مقدار در درخت
- جستجو در درخت
- حذف گره ای از درخت

....



- **عمل ایجاد درخت BST خالی:**

root=null

- **عمل بررسی خالی بودن درخت BST:**

```
if( root = = null )
/// درخت خالی است
```

درج گره در درخت BST:

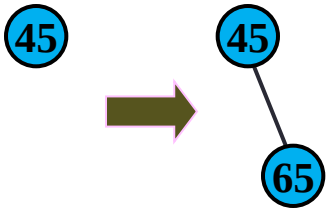
درخت BST با مقادیر زیر ایجاد کنید.

45, 65, 52, 40, 59, 15 , 49

درج گره در درخت BST:

درخت BST با مقادیر زیر ایجاد کنید.

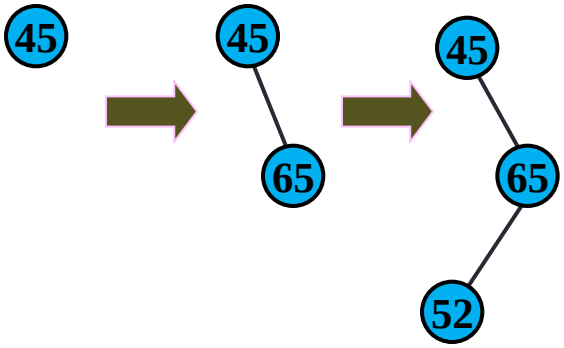
45, 65, 52, 40, 59, 15 , 49



درج گره در درخت BST:

درخت BST با مقادیر زیر ایجاد کنید.

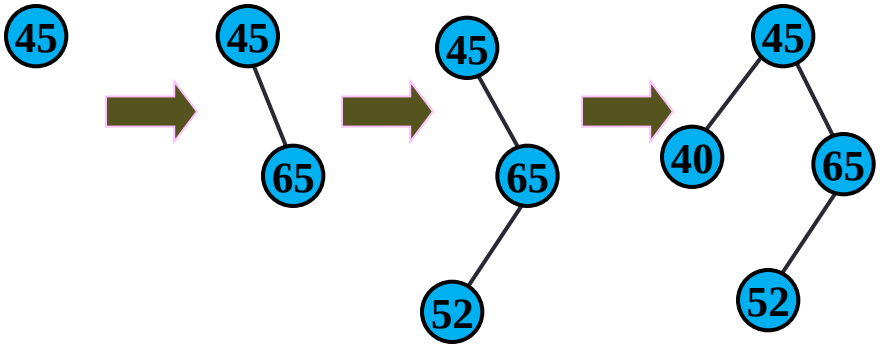
45, 65, 52, 40, 59, 15 , 49



درج گره در درخت BST:

درخت BST با مقادیر زیر ایجاد کنید.

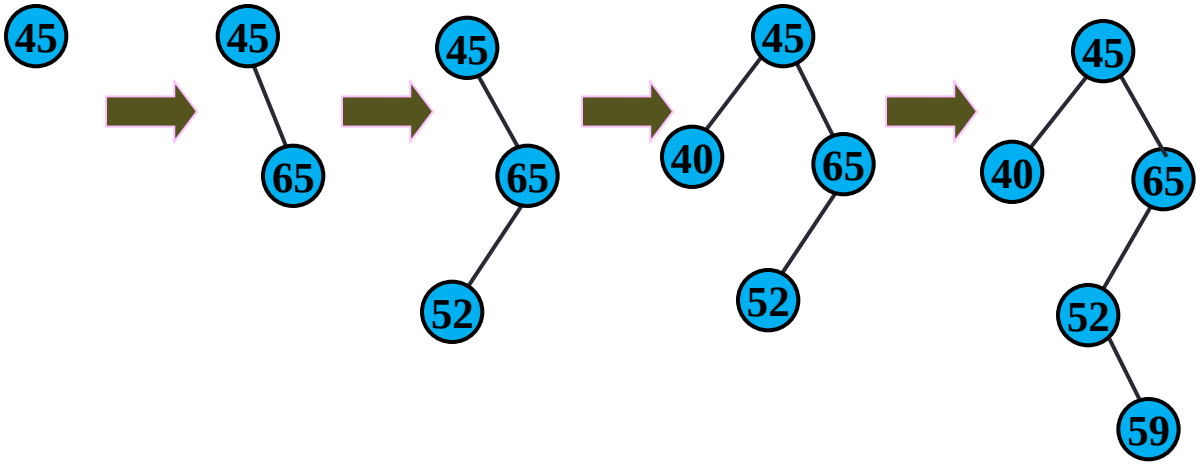
45, 65, 52, 40, 59, 15 , 49



درج گره در درخت BST:

درخت BST با مقادیر زیر ایجاد کنید.

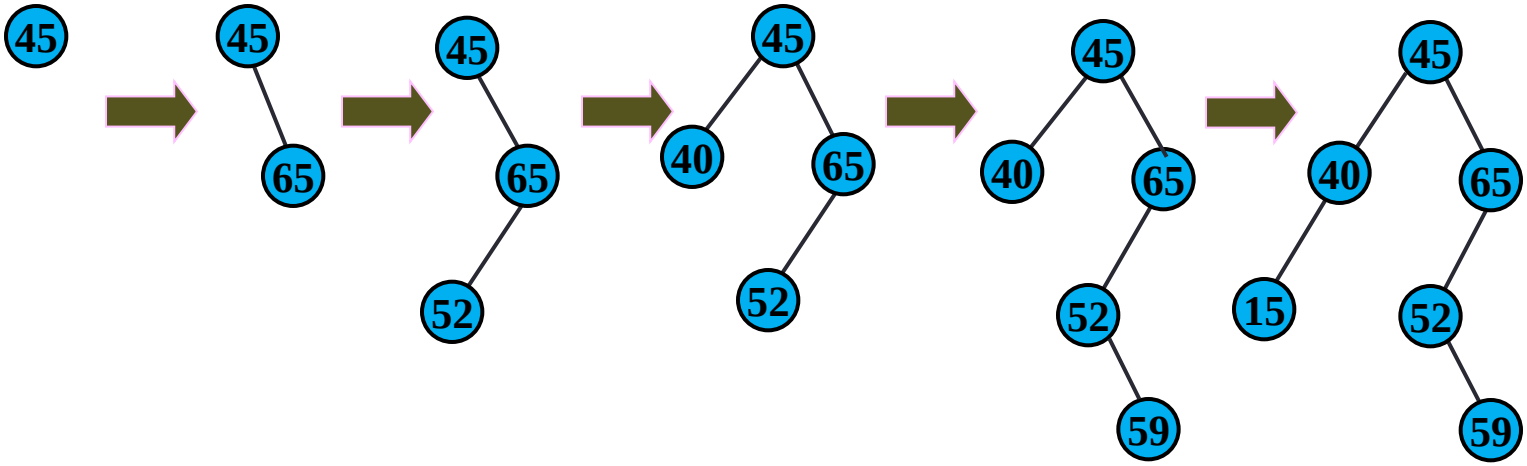
45, 65, 52, 40, 59, 15 , 49



درج گره در درخت BST:

درخت BST با مقادیر زیر ایجاد کنید.

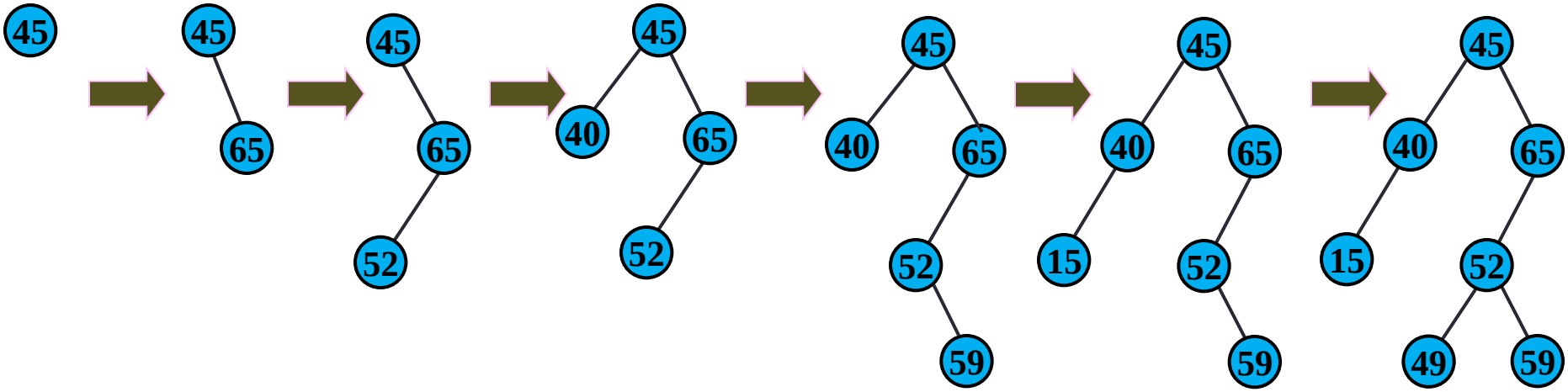
45, 65, 52, 40, 59, 15 , 49



درج گره در درخت BST:

درخت BST با مقادیر زیر ایجاد کنید.

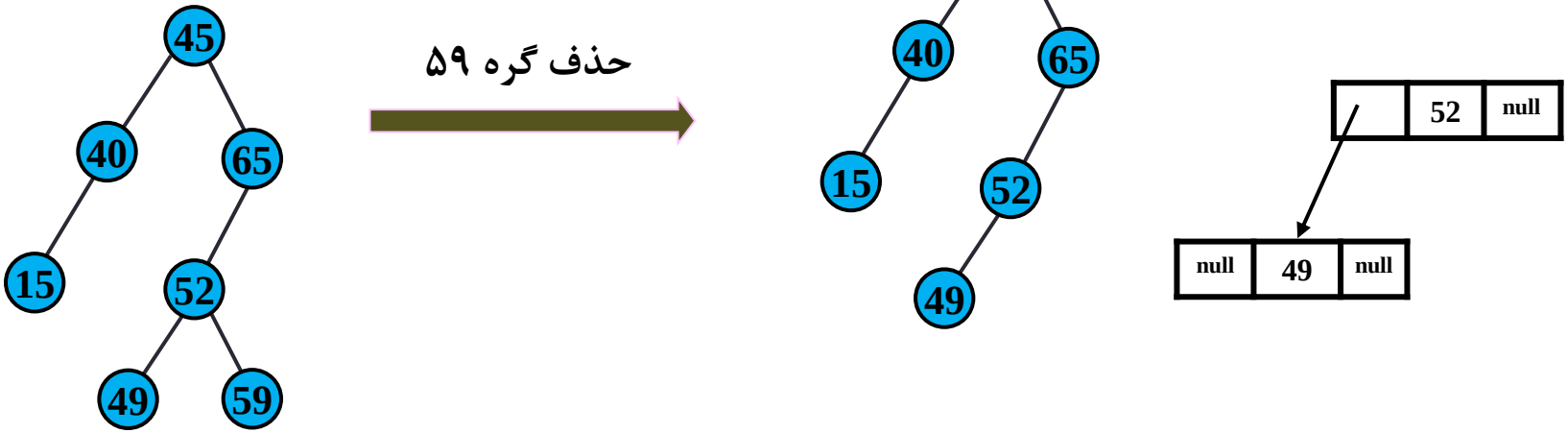
45, 65, 52, 40, 59, 15 , 49



حذف گره از درخت BST:

برای حذف گره ای مثل X سه حالت را در نظر می گیریم:

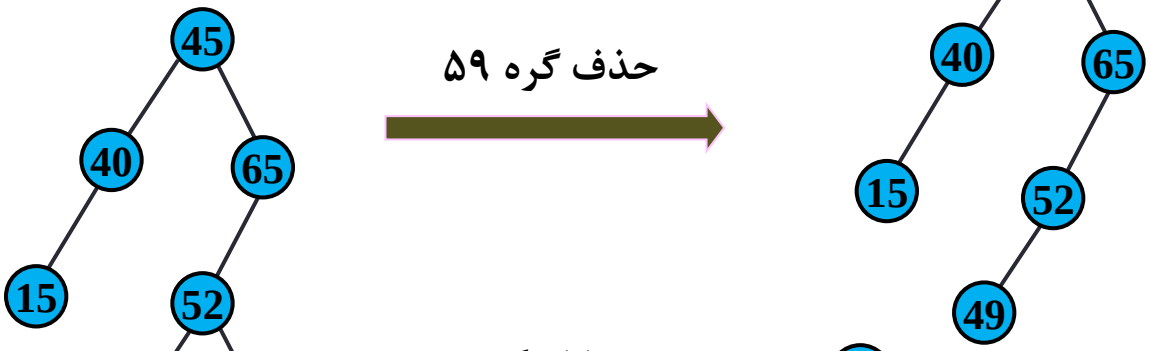
۱) X برگ است

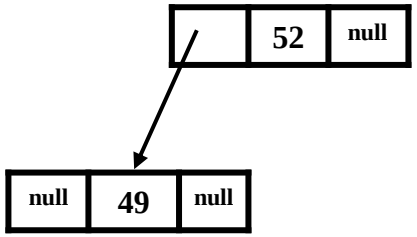


حذف گره از درخت BST:

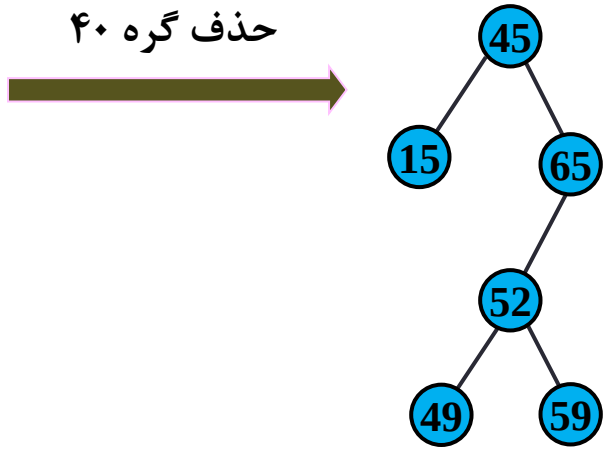
برای حذف گره ای مثل X سه حالت را در نظر می گیریم:

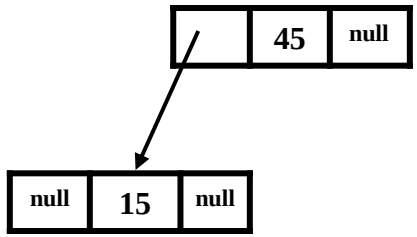
۱) X برگ است





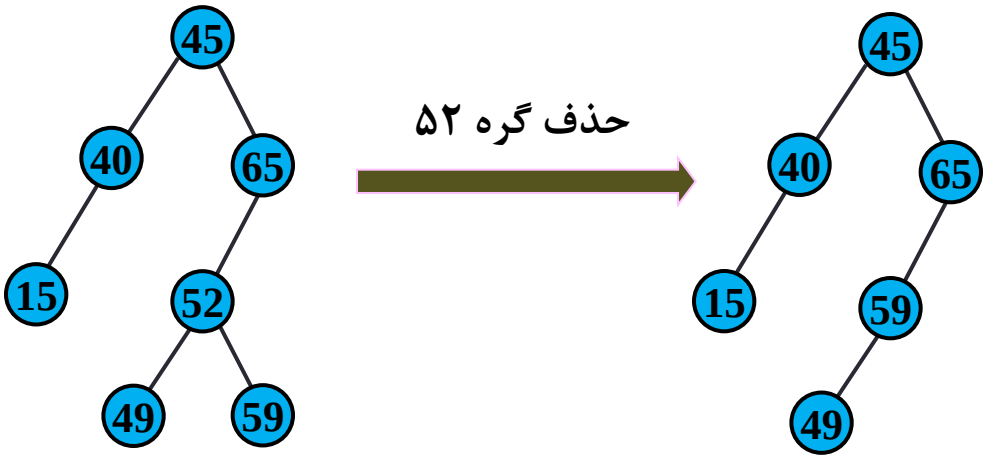
۲) X یک فرزند دارد





حذف گره از درخت BST:

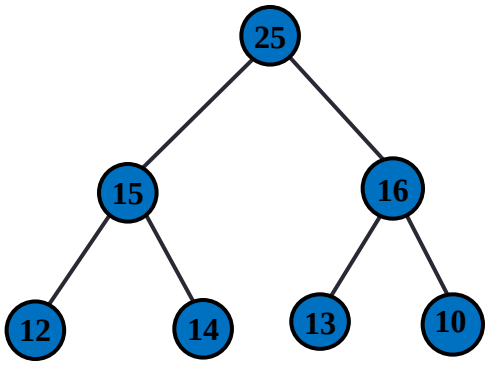
۳) x دو فرزند دارد



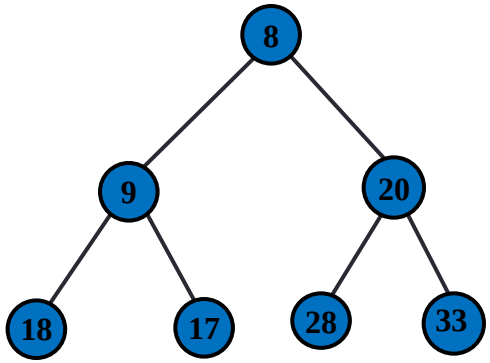
inorder:15, 40, 45, 49, 52, 59, 65

درخت Heap (هرم- نیمه مرتب):

maxtree: درختی است که مقدار هر گره آن بزرگتر یا مساوی مقدار فرزندانش باشد.
mintree: درختی است که مقدار هر گره آن کوچکتر یا مساوی مقدار فرزندانش باشد.



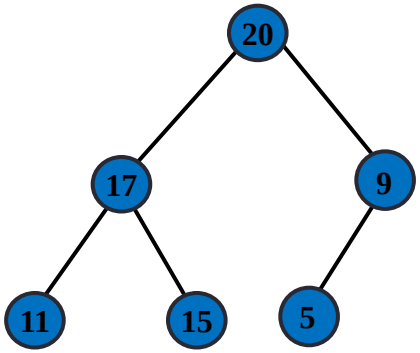
maxtree



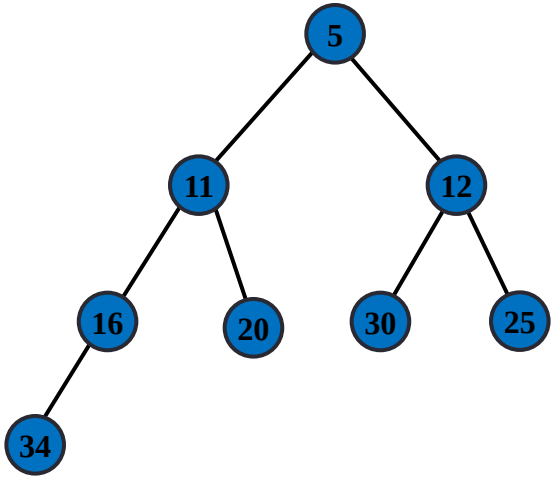
mintree

درخت maxheap: درخت دودویی کاملی است که یک maxtree باشد.

درخت minheap: درخت دودویی کاملی است که یک mintree باشد.



maxheap



minheap

درج عنصر در maxheap:

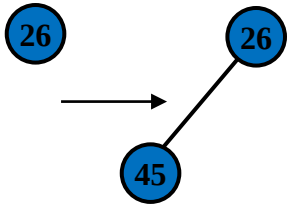
اعمال درج و حذف بگونه ای انجام می گیرد که درخت بصورت نیمه مرتب باقی بماند.
همواره گره در درخت heap از چپ به راست و در سطح آخـر درج مـی شـود و سـپس عمل مرتب سازی درخت انجام می شود.

26 , 45, 27, 50, 90

درج عنصر در maxheap:

اعمال درج و حذف بگونه ای انجام می گیرد که درخت بصورت نیمه مرتب باقی بماند.
همواره گره در درخت heap از چپ به راست و در سطح آخـر درج مـی شـود و سـپس عمل مرتب سازی درخت انجام می شود.

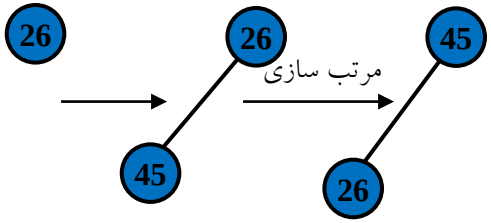
26 , 45, 27, 50, 90



درج عنصر در maxheap:

اعمال درج و حذف بگونه ای انجام می گیرد که درخت بصورت نیمه مرتب باقی بماند.
همواره گره در درخت heap از چپ به راست و در سطح آخر درج می شود و سپس عمل مرتب سازی درخت انجام می شود.

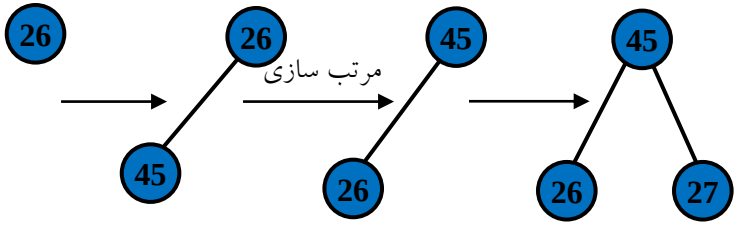
26 , 45, 27, 50, 90



درج عنصر در maxheap:

اعمال درج و حذف بگونه ای انجام می گیرد که درخت بصورت نیمه مرتب باقی بماند.
همواره گره در درخت heap از چپ به راست و در سطح آخر درج می شود و سپس عمل مرتب سازی درخت انجام می شود.

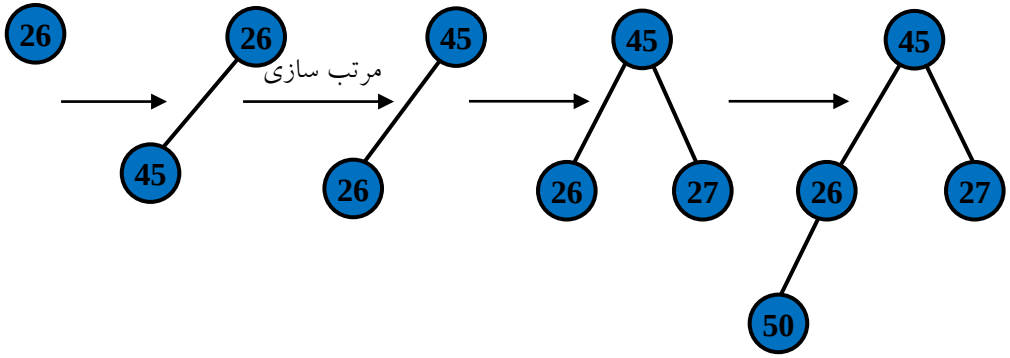
26 , 45, 27, 50, 90



درج عنصر در maxheap:

اعمال درج و حذف بگونه ای انجام می گیرد که درخت بصورت نیمه مرتب باقی بماند.
همواره گره در درخت heap از چپ به راست و در سطح آخر درج می شود و سپس عمل مرتب سازی درخت انجام می شود.

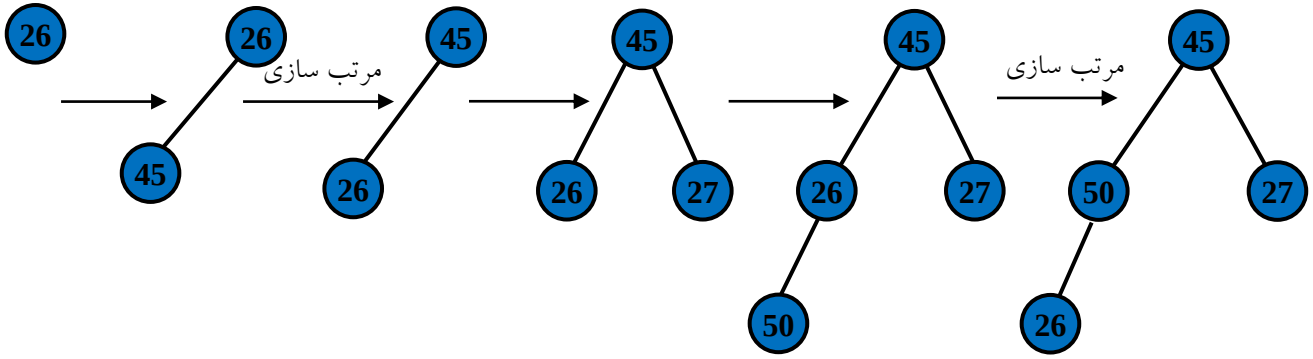
26 , 45, 27, 50, 90



درج عنصر در maxheap:

اعمال درج و حذف بگونه ای انجام می گیرد که درخت بصورت نیمه مرتب باقی بماند.
همواره گره در درخت heap از چپ به راست و در سطح آخر درج می شود و سپس عمل مرتب سازی درخت انجام می شود.

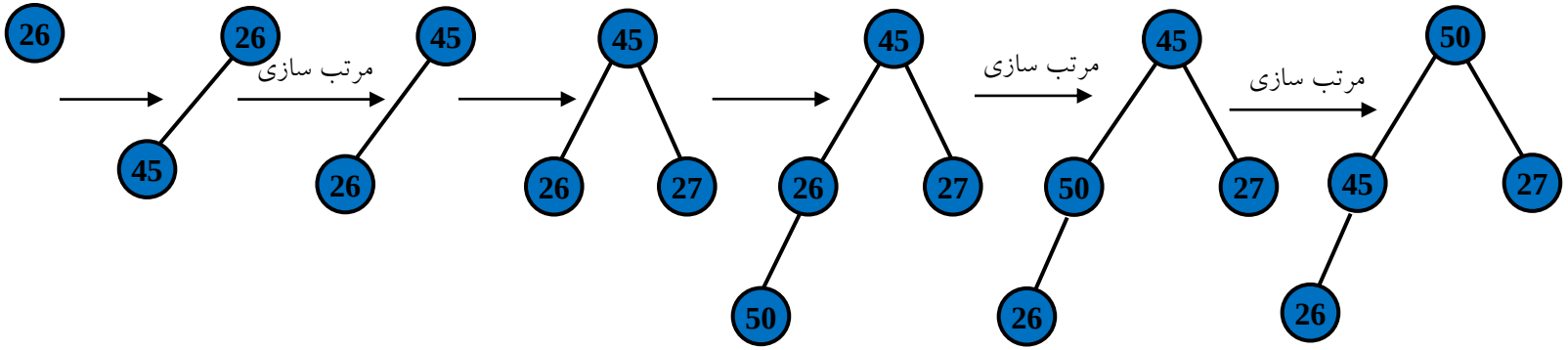
26 , 45, 27, 50, 90



درج عنصر در maxheap:

اعمال درج و حذف بگونه ای انجام می گیرد که درخت بصورت نیمه مرتب باقی بماند.
همواره گره در درخت heap از چپ به راست و در سطح آخر درج می شود و سپس عمل مرتب سازی درخت انجام می شود.

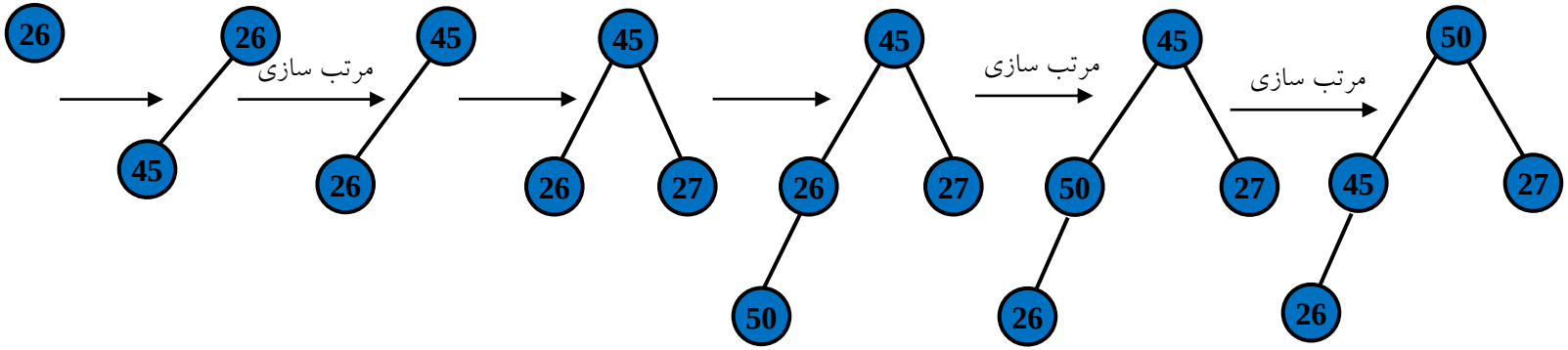
26 , 45, 27, 50, 90

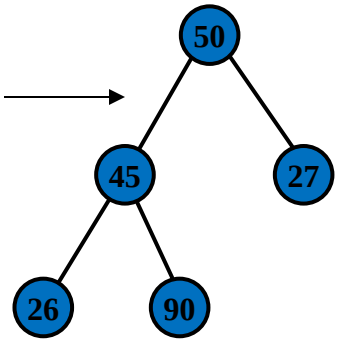


درج عنصر در maxheap:

اعمال درج و حذف بگونه ای انجام می گیرد که درخت بصورت نیمه مرتب باقی بماند.
همواره گره در درخت heap از چپ به راست و در سطح آخر درج می شود و سپس عمل مرتب سازی درخت انجام می شود.

26 , 45, 27, 50, 90

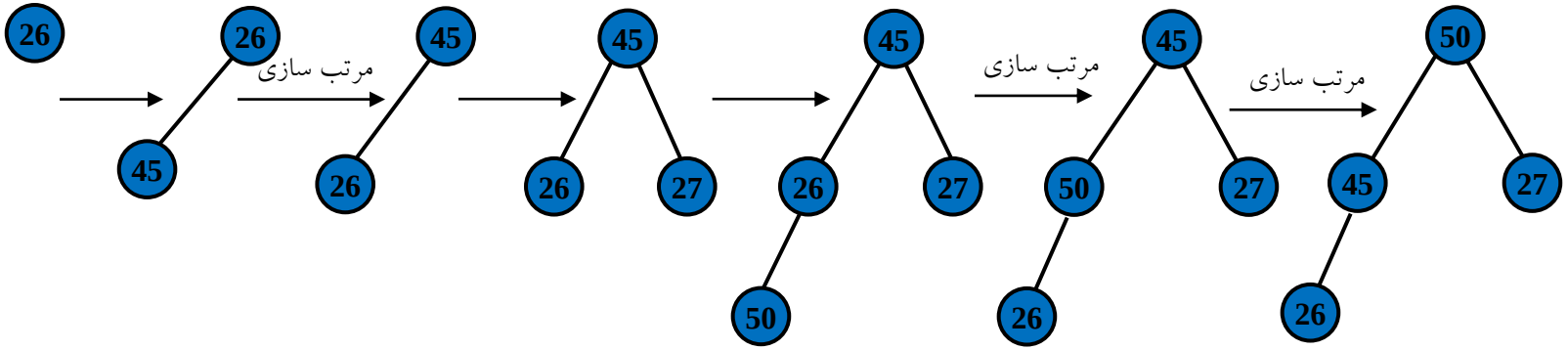


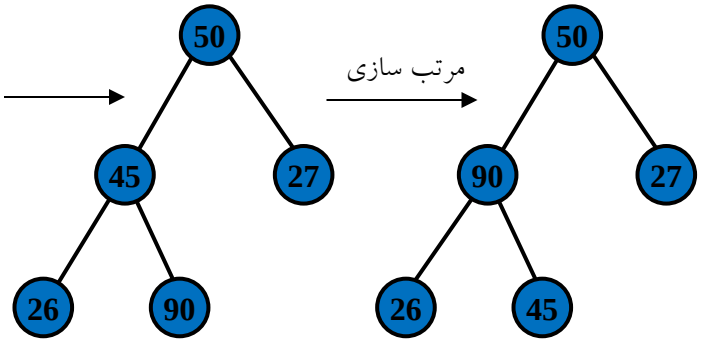


درج عنصر در maxheap:

اعمال درج و حذف بگونه ای انجام می گیرد که درخت بصورت نیمه مرتب باقی بماند.
همواره گره در درخت heap از چپ به راست و در سطح آخر درج می شود و سپس عمل مرتب سازی درخت انجام می شود.

26 , 45, 27, 50, 90

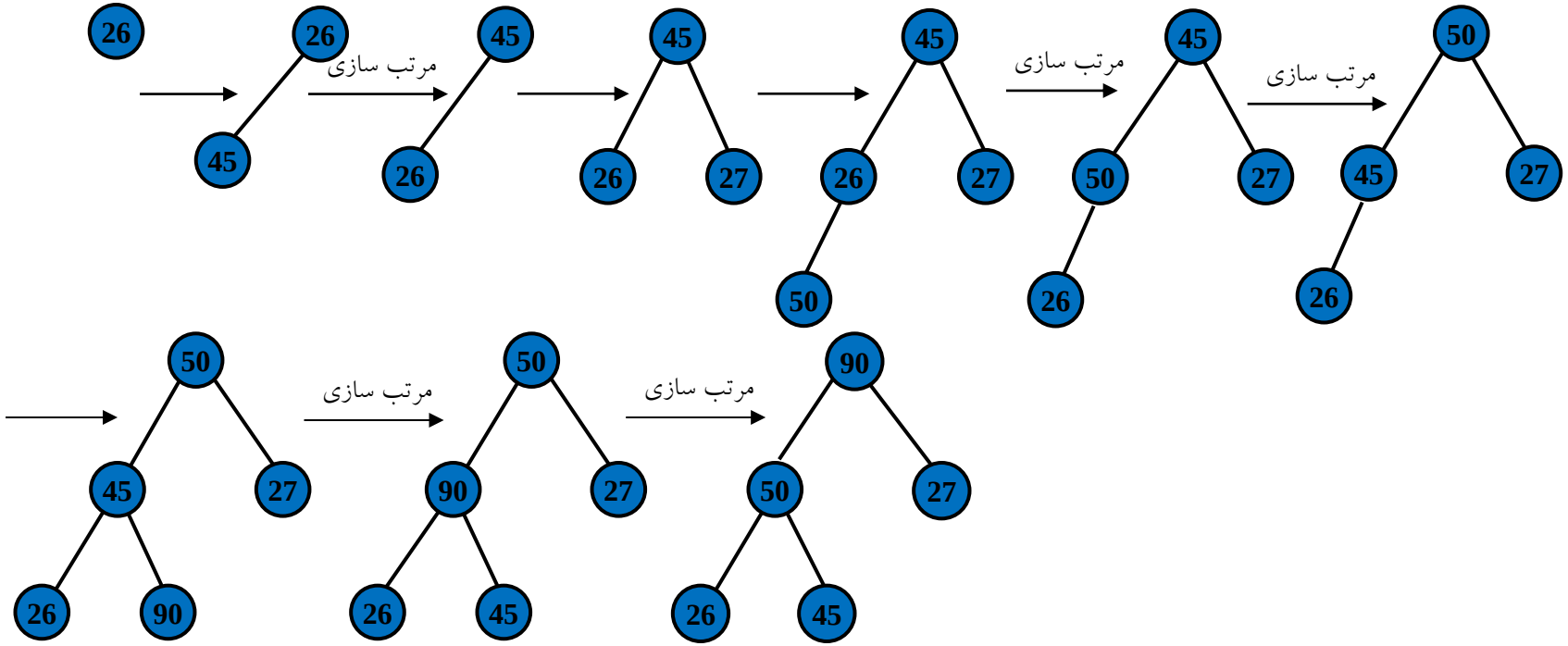




درج عنصر در maxheap:

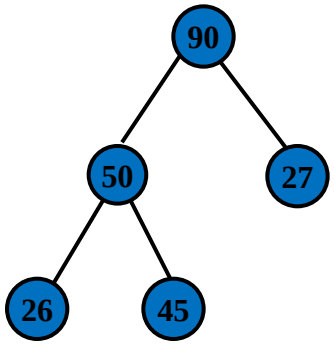
اعمال درج و حذف بگونه ای انجام می گیرد که درخت بصورت نیمه مرتب باقی بماند.
همواره گره در درخت heap از چپ به راست و در سطح آخر درج می شود و سپس عمل مرتب سازی درخت انجام می شود.

26 , 45, 27, 50, 90



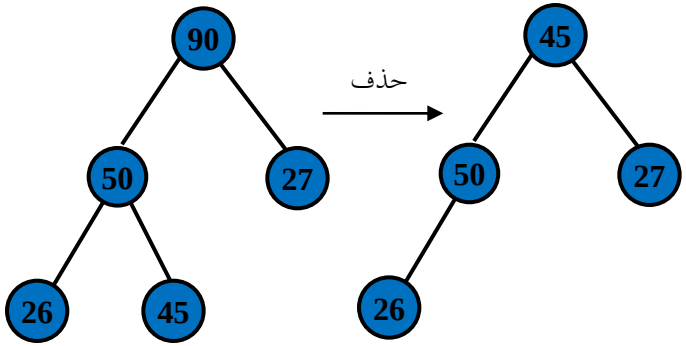
حذف عنصر از maxheap:

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



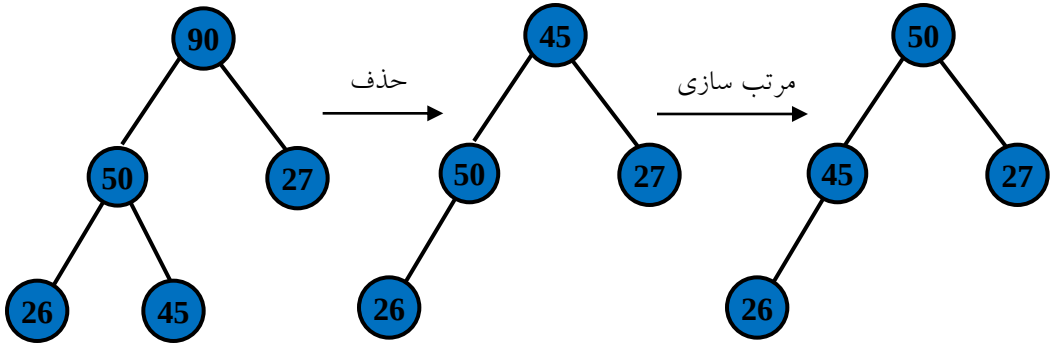
حذف عنصر از maxheap:

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



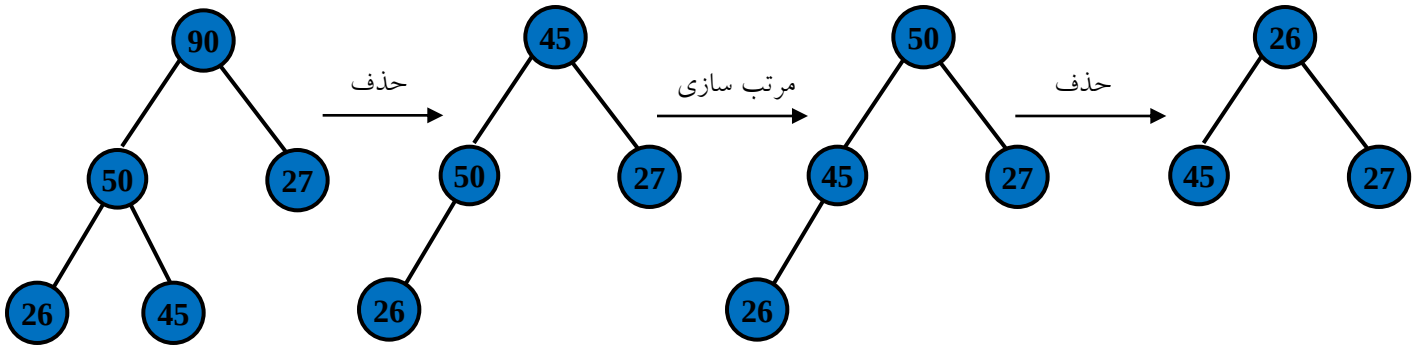
حذف عنصر از maxheap:

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



حذف عنصر از maxheap:

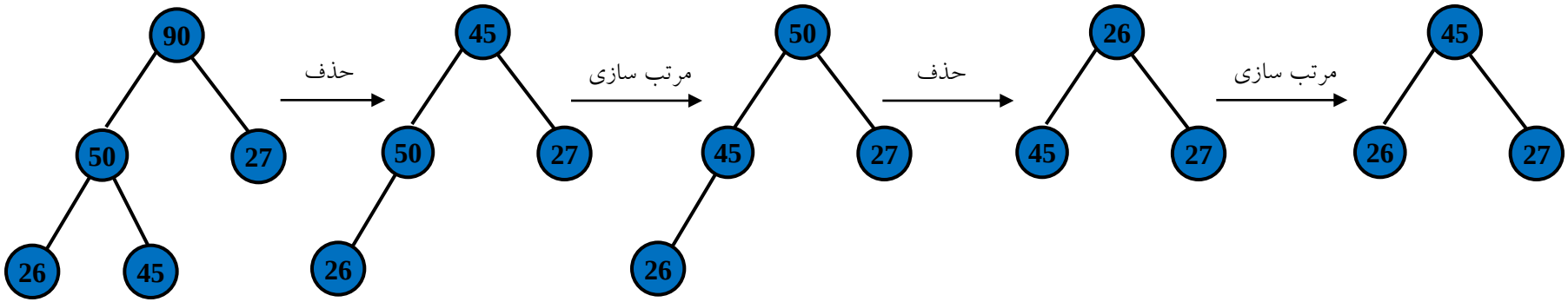
هنگامی که عنصری از درخت maxheap حذف می شود، آن را از ریشه درخت heap می گیریم.



90, 50

حذف عنصر از maxheap:

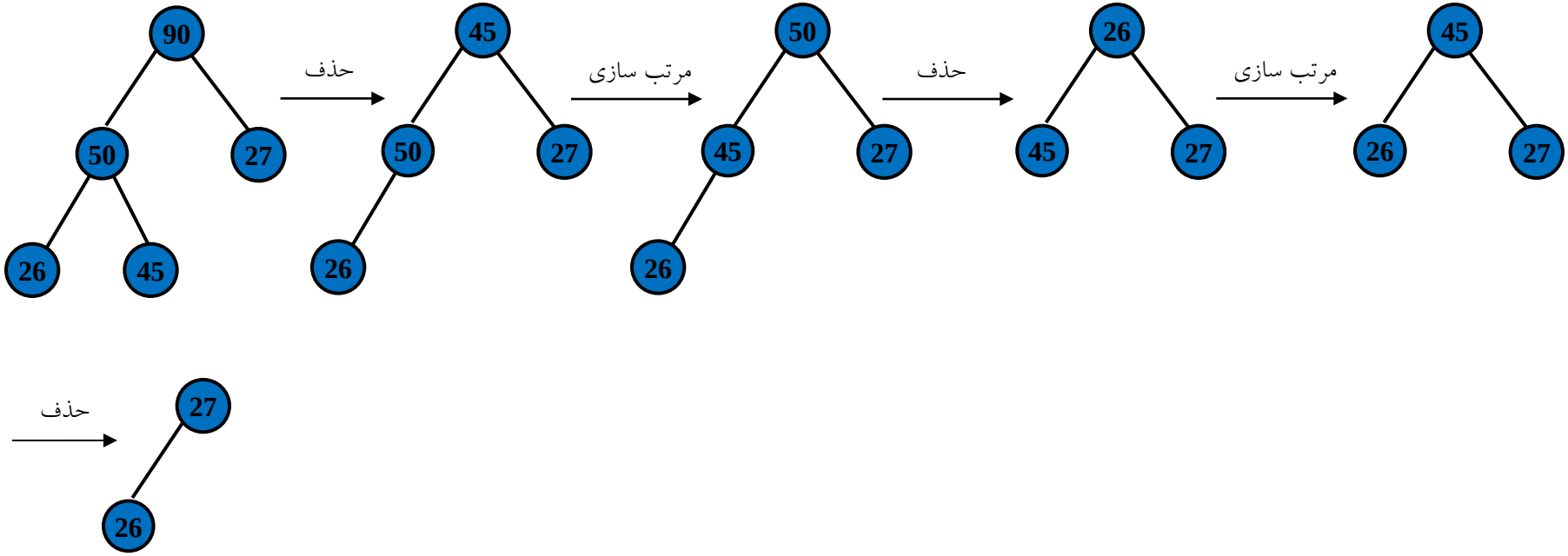
هنگامی که عنصری از درخت maxheap حذف می شود، آن را از ریشه درخت heap می گیریم.



90, 50

حذف عنصر از maxheap:

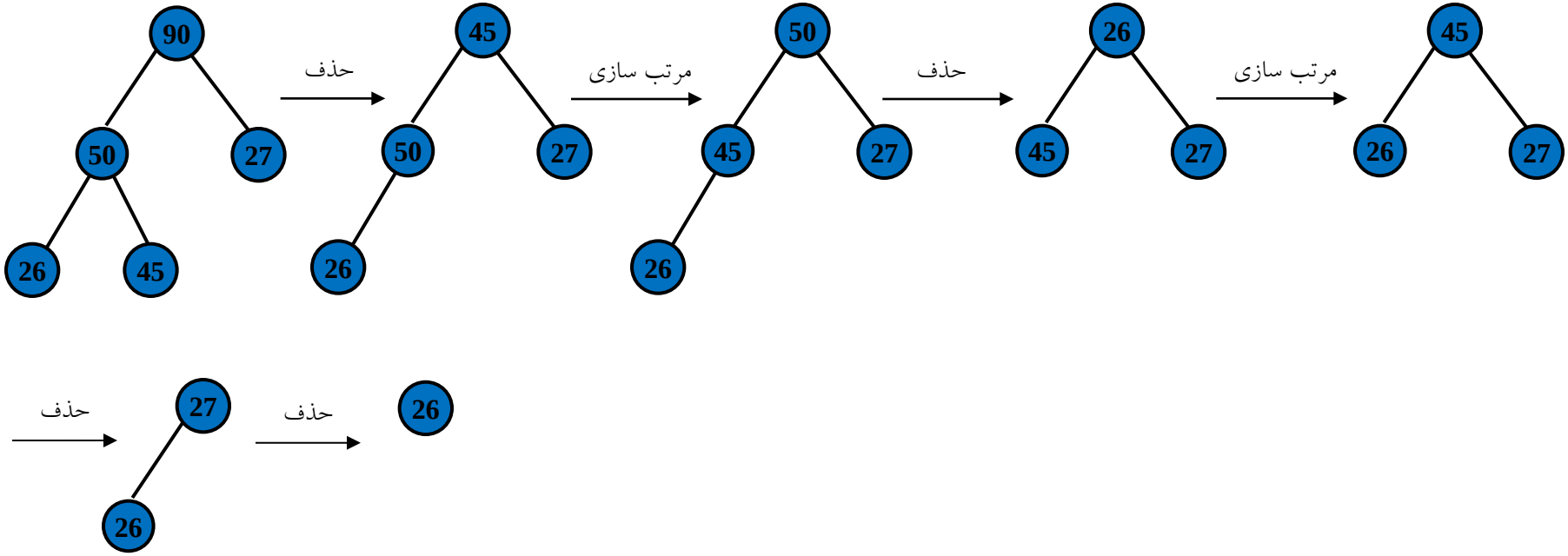
هنگامی که عنصری از درخت maxheap حذف می شود، آن را از ریشه درخت heap می گیریم.



90, 50, 45

حذف عنصر از maxheap:

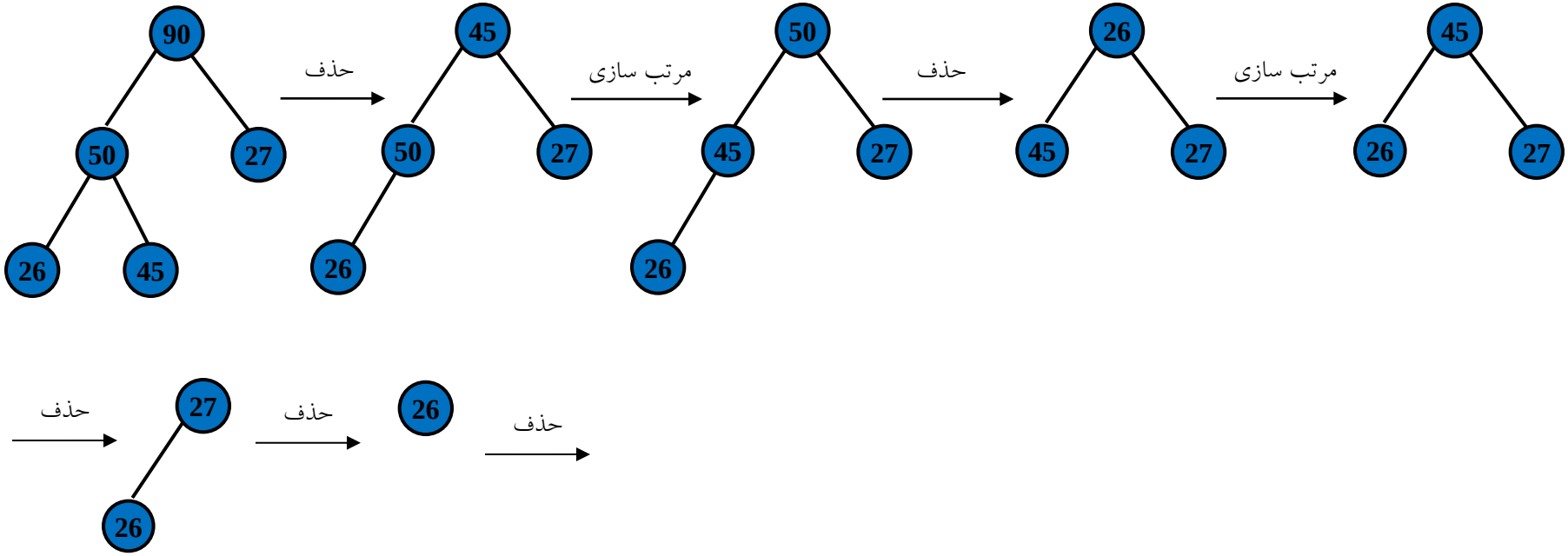
هنگامی که عنصری از درخت maxheap حذف می شود، آن را از ریشه درخت heap می گیریم.



90, 50, 45, 27

حذف عنصر از maxheap:

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



90, 50, 45, 27, 26

پایان