ساختمان داده‌ها

فصل پنجم

درختها

E-mail: Hadi.khademi@gmail.com

مهر ماه ۹٤- دانشگاه علم و فرهنگ

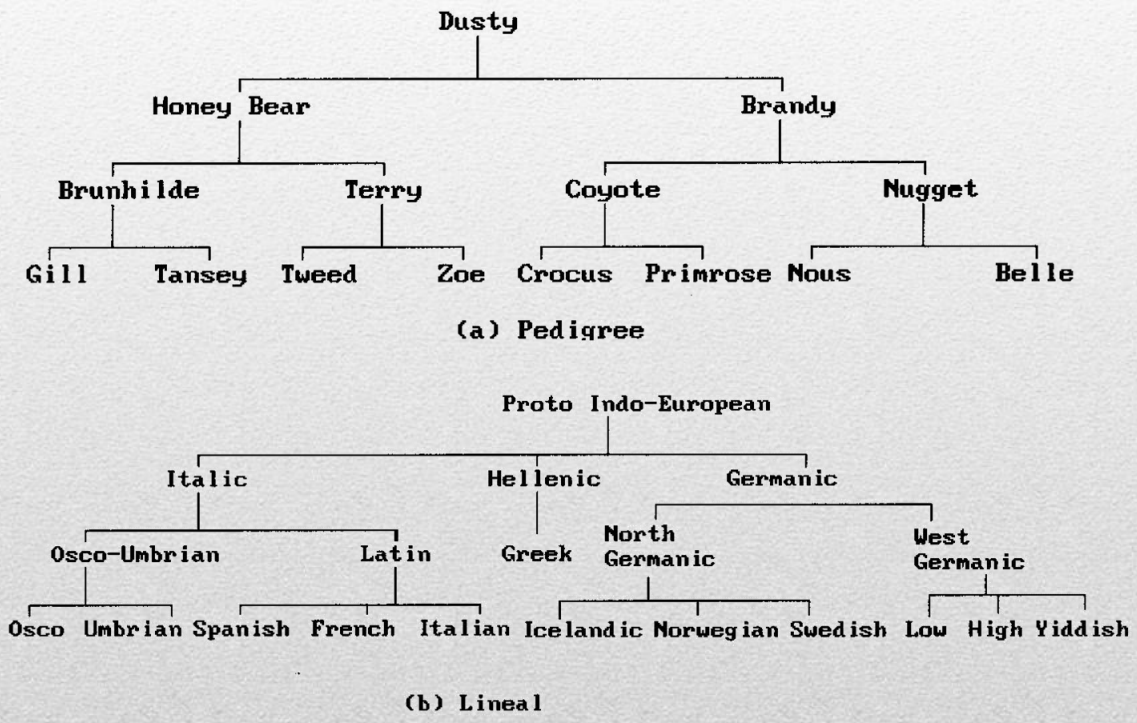
- مقدمه
  - بازنمایی درخت ها
- درخت های دودویی
- پیمایش درختهای دودویی
- عملیات دیگر روی درختهای دودویی
- درختهای دودویی نخ کشی شده
- Heap ها
- درختان جستجوی دودویی
- درختهای انتخاب
- جنگل ها
- نمایش مجموعه ها
- شمارش درخت های دودویی متمایز

درختها

E-mail: Hadi.khademi@gmail.com

- ساختار درختی، مجموعه داده های سازمان دهی شده ای است که اقلام اطلاعاتی آن از طریق شاخه ها با هم ارتباط دارند.

مثال



درختها

E-mail: Hadi.khademi@gmail.com

- تعریف: درخت مجموعه ای متناهی و غیر تهی از یک یا چند گره است به طوریکه:
- دارای گره خاصی به نام ریشه باشد.
- بقیه گره ها به $n \geq 0$ مجموعه مجزا $T_1,\ldots,T_n$ تقسیم شده که هر یک از این مجموعه ها خود یک درخت هستند. $T_1,\ldots,T_n$ زیر درختان ریشه نامیده می شوند.
- هر نود در درخت ریشه مجموعه ای از زیر درختان است

تعریف درخت

E-mail: Hadi.khademi@gmail.com

- درجه گره : تعداد زیر درخت های یک گره را درجه آن گره می نامند.
- گره های پایانی (برگ): گره هایی که درجه صفر دارند برگ یا گره پایانی نامند.
- درجه درخت: بزرگترین درجه گره های موجود در آن است
- فرزندان گره : ریشه های زیر درختهای گره ای مانند X ، فرزندان X نامیده می شوند.
- والد : X والد بچه هایش است.
- همزاد (برادر): بچه هایی که والد یکسان دارند هم زاد نامیده می شوند .

اصطلاحات مربوط به درختها

E-mail: Hadi.khademi@gmail.com

- اجداد یک گره: تمامی گره هایی که در مسیر ریشه تا یک گره قرار دارند اجداد آن گره نامیده می شوند.
- نسل های گره: اگر گره y جد گره x باشد، می گوییم x نسل y است. تمام گره ها (به جز گره ریشه)، نسل های گره ریشه اند.
- 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 بزرگترین سطح گره های آن درخت است.

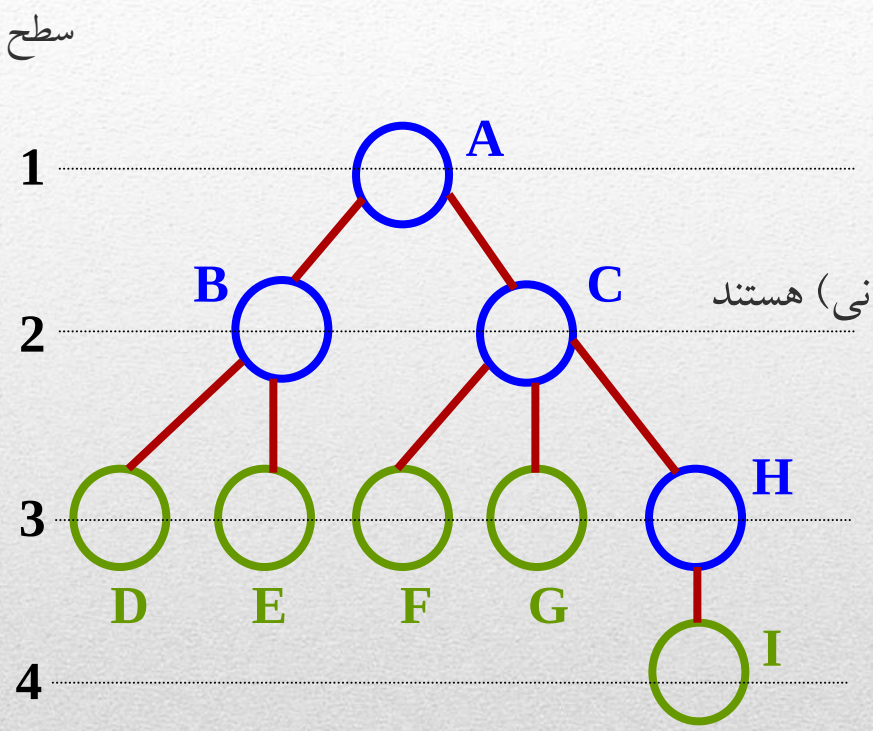
اصطلاحات مربوط به درختها

E-mail: Hadi.khademi@gmail.com

مثال

- A گره ریشه است.
- B والد D و E است.
- C همزاد B است.
- D و E فرزندان B هستند.
- D، E، F، G و I گره های برگ هستند.
- A، B، C و H گره های داخلی (غیر پایانی) هستند
- سطح E برابر ۳ است.
- عمق (ارتفاع) درخت برابر ۴ است.
- درجه نود B برابر ۲ است.
- درجه درخت برابر ۳ است.
- اجداد I برابر A, C, H است.
- نسل های C برابر F, G, H, I است.

۱–تعداد گره =تعداد شاخه

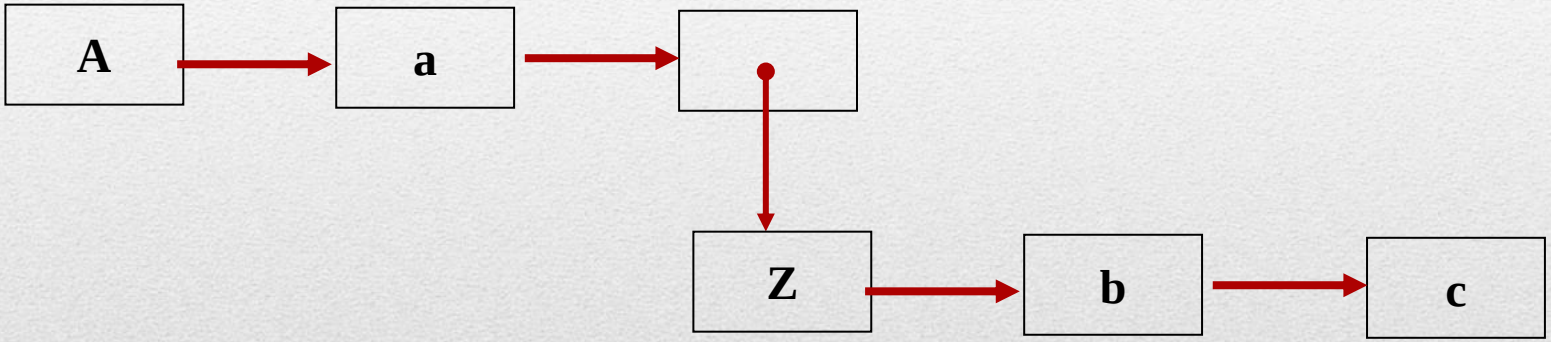


اصطلاحات مربوط به درخت ها

E-mail: Hadi.khademi@gmail.com

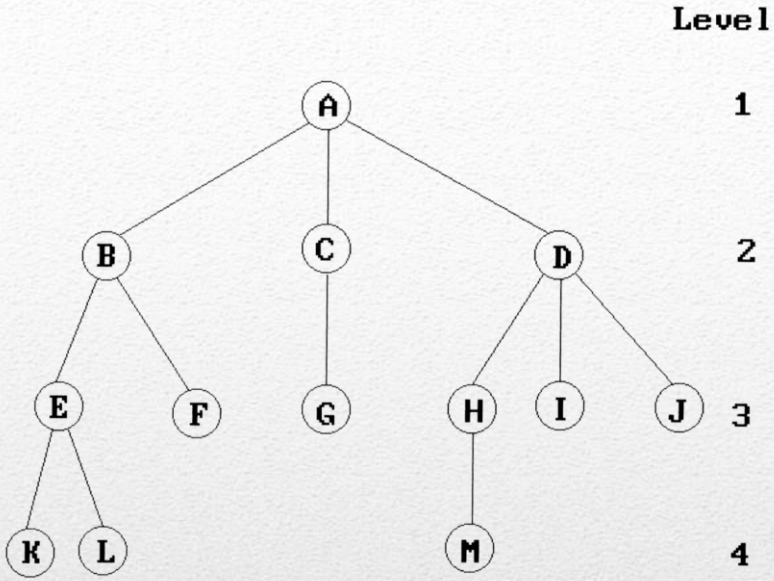
A=(a, (b,c))

A(a, Z(b,c))



یاد اوری (لیست تعمیم یافته)

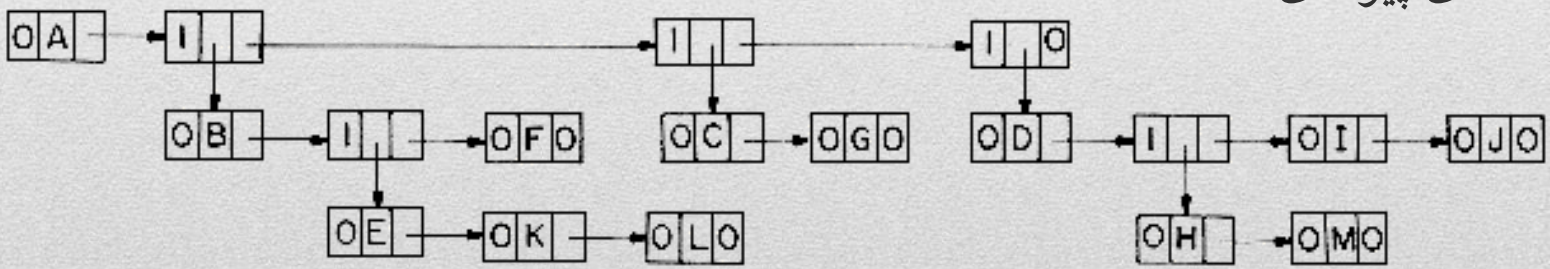
E-mail: Hadi.khademi@gmail.com



- درخت روبرو را می توان در قالب لیست به صورت زیر نشان داد

(A(B(E(K ،L)،F)،C(G)،D(H(M)،I،J)))

- نمایش درخت ها به صورت لیست های پیوندی



بازنمایی درختها

E-mail: Hadi.khademi@gmail.com

- نمایش خاص برای درخت ها با گره های به طول ثابت
- برای نمایش هر گره درخت، از یک گره در حافظه استفاده شود که فیلدهایی برای داده ها و اشاره گرهایی برای بچه های این گره دارد.

data	*link 1*	*link 2*	· · ·	*link n*

- عیب: هدر رفت حافظه
- اگر T درختی از درجه k با n گره به صورت بالا باشد آنگاه از nk اشاره گر بچه تعداد n(k-1)+1 تا صفر است که در آن n>=1 است.
- بنابر این دو نمایش دیگر با استفاده از گره های به طول ثابت ارائه می دهیم.

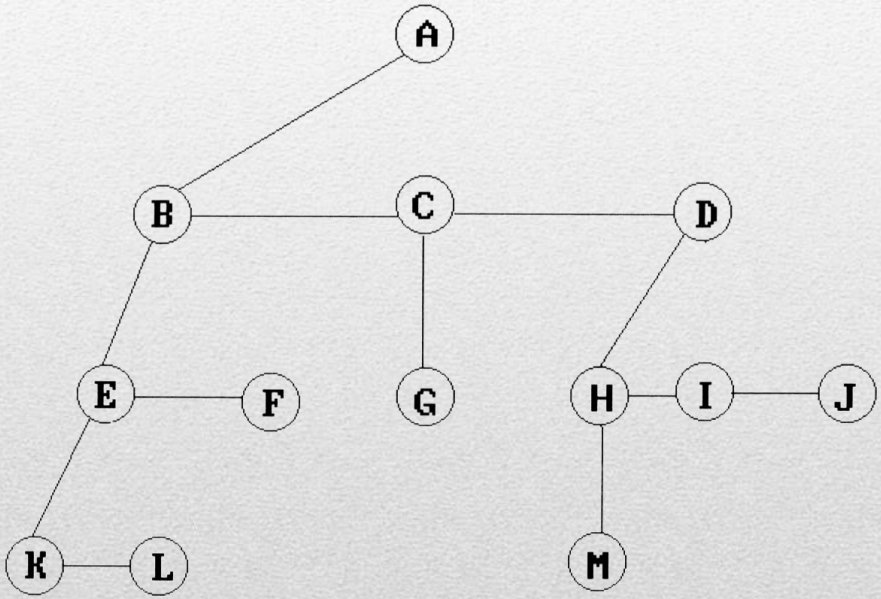
بازنمایی درخت ها

E-mail: Hadi.khademi@gmail.com

- نمایش بچه چپ همزاد راست

ساختار گره

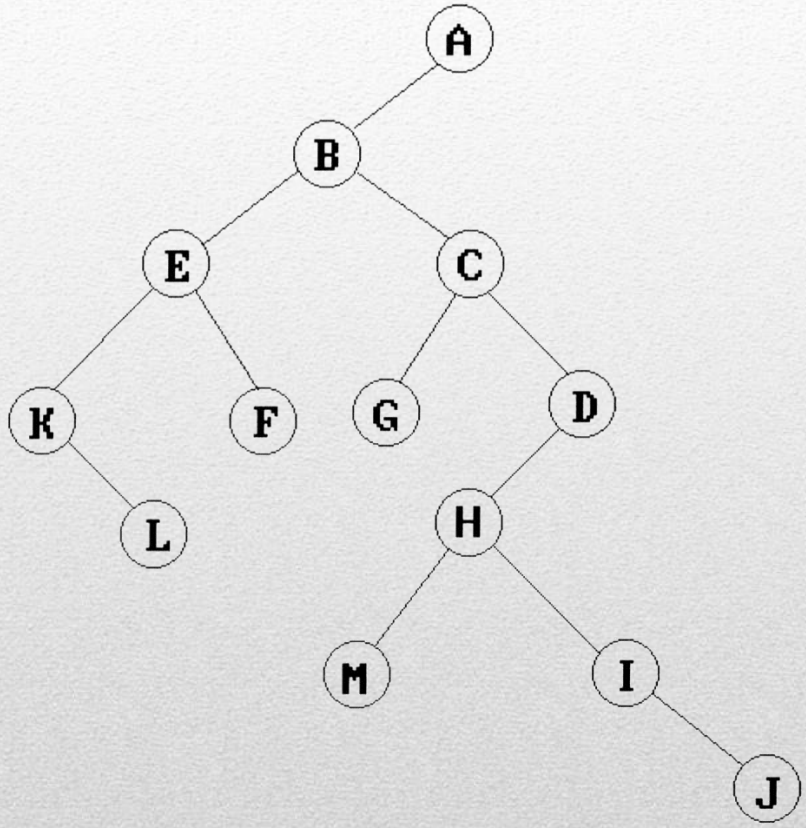
data	
left child	right sibling



بازنمایی درختها

E-mail: Hadi.khademi@gmail.com

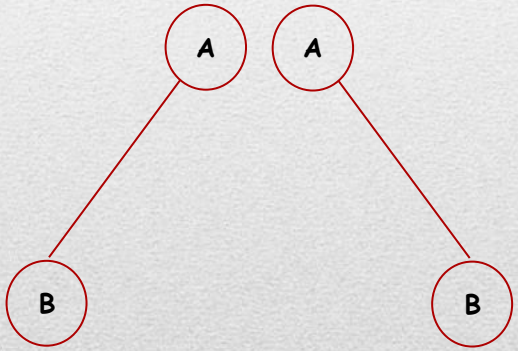
- نمایش درجه دو درخت (بچه چپ، بچه راست)



بازنمایی درختها

E-mail: Hadi.khademi@gmail.com

- 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.
- تعریف: درخت دودویی، مجموعه محدودی از گره ها است که یا خالی است (درخت دودویی خالی) یا حاوی گره خاصی به نام ریشه است و بقیه گره های آن، دو زیر درخت دودویی مجزا به نام های زیر درخت چپ و زیر درخت راست را تشکیل می دهند.
- بنابراین در درختهای دودویی بین زیر درخت چپ و راست تمایز قائل می شویم



- دو درخت دودویی بالا یکسان نیستند، زیرا گره A در یکی دارای فرزند چپ و در دیگری دارای فرزند راست است.

درختهای دودویی

E-mail: Hadi.khademi@gmail.com

- تفاوت های بین درخت دودویی و درخت عمومی:
  - درخت دودویی می تواند تهی باشد، ولی درخت عمومی نمی تواند تهی باشد.
  - در درخت دودویی، هر گره حداکثر دو فرزند دارد.
  - در درخت دودویی ترتیب فرزندان هر گره مهم است، در حالی که در درخت عمومی این طور نیست.

- هر درختی را می توان به فرم درخت دودویی در آورد.
  - با بازنمایی بچه چپ-همزاد راست

درختهای دودویی

E-mail: Hadi.khademi@gmail.com

- نوع داده تجریدی BinaryTree

structure *Binary_Tree* (abbreviated *BinTree*) is

objects: a finite set of nodes either empty or consisting of a root node, left *Binary_Tree*, and right *Binary_Tree*.

functions:

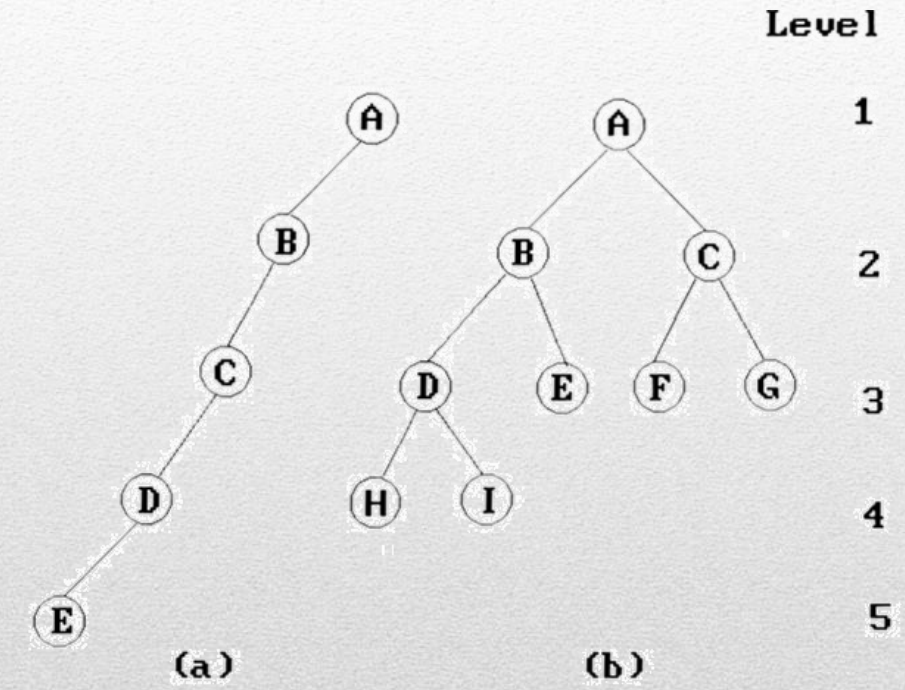
for all $bt, bt1, bt2 \in$ *BinTree*, *item* $\in$ *element*

BinTree Create()	::=	creates an empty binary tree
Boolean IsEmpty(*bt*)	::=	**if** (*bt* == empty binary tree) **return** *TRUE* **else return** *FALSE*
BinTree MakeBT(*bt*1, *item*, *bt*2)	::=	**return** a binary tree whose left subtree is *bt*1, whose right subtree is *bt*2, and whose root node contains the data *item*.
BinTree Lchild(*bt*)	::=	**if** (IsEmpty(*bt*)) **return** error **else return** the left subtree of *bt*.
element Data(*bt*)	::=	**if** (IsEmpty(*bt*)) **return** error **else return** the data in the root node of *bt*.
BinTree Rchild(*bt*)	::=	**if** (IsEmpty(*bt*)) **return** error **else return** the right subtree of *bt*.

درختهای دودویی

E-mail: Hadi.khademi@gmail.com

- درختهای دودویی مورب و کامل



درختهای دودویی

E-mail: Hadi.khademi@gmail.com

- حداکثر تعداد گره ها
  - حداکثر تعداد گره ها در سطح iام یک درخت دودویی 2^{i-1} است، $i \geq 1$.
  - حداکثر تعداد گره ها در یک درخت دودویی به عمق k ، 2^k-1 است ، $k \geq 1$

$$tn = 2^0 + 2^1 + 2^2 + \ldots + 2^{k-1} = \sum_{j=0}^{k-1} 2^j$$

$$\sum_{j=0}^{k-1} 2^j = 2^k - 1$$

خصوصیات درختهای دودویی

E-mail: Hadi.khademi@gmail.com

- رابطه بین تعداد گره های برگ و گره های درجه ۲
  - برای هر درخت دودویی غیر تهی مانند T ، اگر n_0 تعداد گره های پایانی و n_2 تعداد گره های درجه ۲ باشد ، آنگاه خواهیم داشت:

$$n_0=n_2+1$$

زیرا

$$n=n_0+n_1+n_2$$

- به غیر از ریشه به همه نودها یک شاخه از بالا متصل است
- هر نود بر اساس درجه تعدادی شاخه به پایین دارد

$$\left.\begin{array}{l} n=B+1 \\ B=n_1+2n_2 \end{array}\right\} \quad n=n_1+2n_2+1$$

$$0=n_2-n_0+1$$

خصوصیات درختهای دودویی

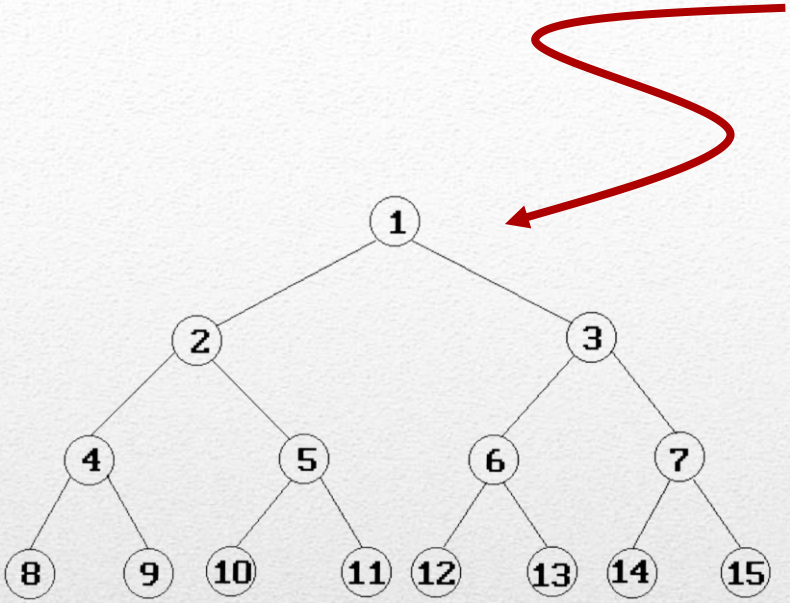
E-mail: Hadi.khademi@gmail.com

- **درخت دودویی پر به عمق k**

یک درخت دودویی به عمق k پر است مشروط بر اینکه حاوی 2^k-1 گره باشد که $k>=0$. توجه 2^k-1 حداکثر تعداد گره های یک درخت دودویی به عمق k است

- **درخت دودویی کامل به عمق k**

یک درخت دودویی با nگره و عمق k کامل است اگر و فقط اگر گره هایش مطابق با شماره گذاری گره های 1 تا n در یک درخت دودویی پر به عمق k باشد.

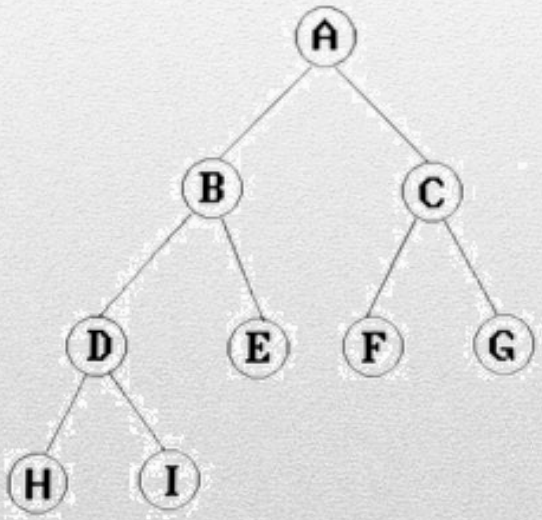


عمق یک درخت دودویی کامل با n گره برابر است با $\lceil \log_2(n+1) \rceil$

درختها

E-mail: Hadi.khademi@gmail.com

- نمایش به صورت ارایه
- از آنجایی که در یک درخت کامل گره ها از 1 تا n شماره گذاری شده اند ، از این رو از یک آرایه یک بعدی می توان برای ذخیره سازی گره ها استفاده کرد.



[1]	A
[2]	B
[3]	C
[4]	D
[5]	E
[6]	F
[7]	G
[8]	H
[9]	I

نمایش درختهای دودویی در حافظه

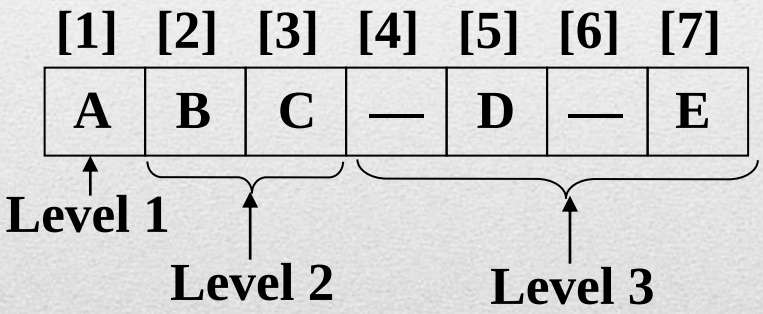
E-mail: Hadi.khademi@gmail.com

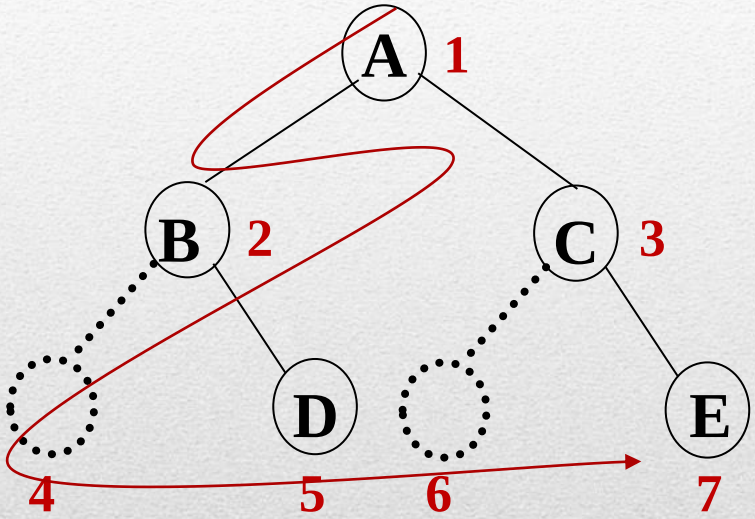
- **نمایش به صورت ارایه**
- اگر یک درخت دودویی کامل با n گره در یک آرایه ذخیره شود آنگاه برای هر گره با اندیس i و $1 \leq i \leq n$، داریم:
  - اگر $i \neq 1$ ، آنگاه پدر i در [i/2] است . اگر $i=1$ ، i ریشه است و پدری ندارد.
  - اگر $2i \leq n$ ، آنگاه فرزند چپ i در 2i است. اگر $2i > n$، آنگاه i فرزند چپ ندارد.
  - اگر $2i+1 \leq n$ ، آنگاه فرزند راست i در 2i+1 است. اگر $2i+1 > n$ ، آنگاه i فرزند راست ندارد

نمایش درختهای دودویی در حافظه

E-mail: Hadi.khademi@gmail.com

- نمایش به صورت ارایه
- هر درخت غیر کامل را نیز می توان در یک ارایه ذخیره کرد

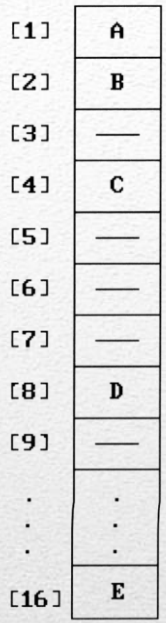


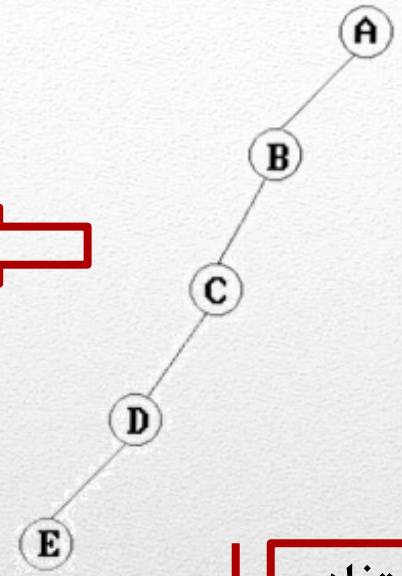


نمایش درختهای دودویی در حافظه

E-mail: Hadi.khademi@gmail.com

- نمایش به صورت ارایه
- با به هدر رفتن حافظه همراه است





در بدترین حالت ، یک درخت مورب به عمق k ، به 2^k-1 حافظه نیاز دارد که از این مقدار، فقط k محل مورد استفاده قرار می گیرد.

اگر از ارایه c++ برای نمایش درخت استفاده شود مکان صفرم ارایه خالی رها می شود.

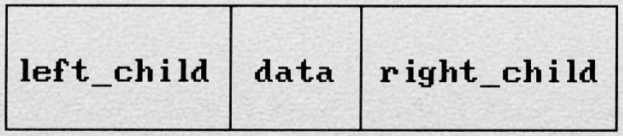
- اضافه/حذف کردن گره ها از وسط یک درخت مستلزم جابجایی بسیاری از گره ها است.

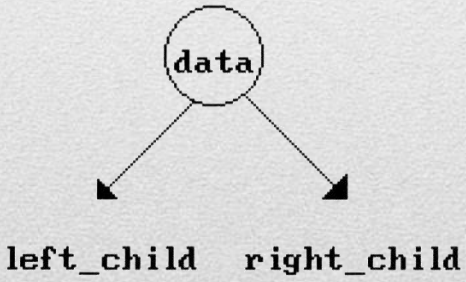
نمایش درختهای دودویی در حافظه

E-mail: Hadi.khademi@gmail.com

- **نمایش به صورت پیوندی**
- نارسایی های موجود در نمایش ترتیبی می تواند با به کارگیری نمایش پیوندی به آسانی حل شود.

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

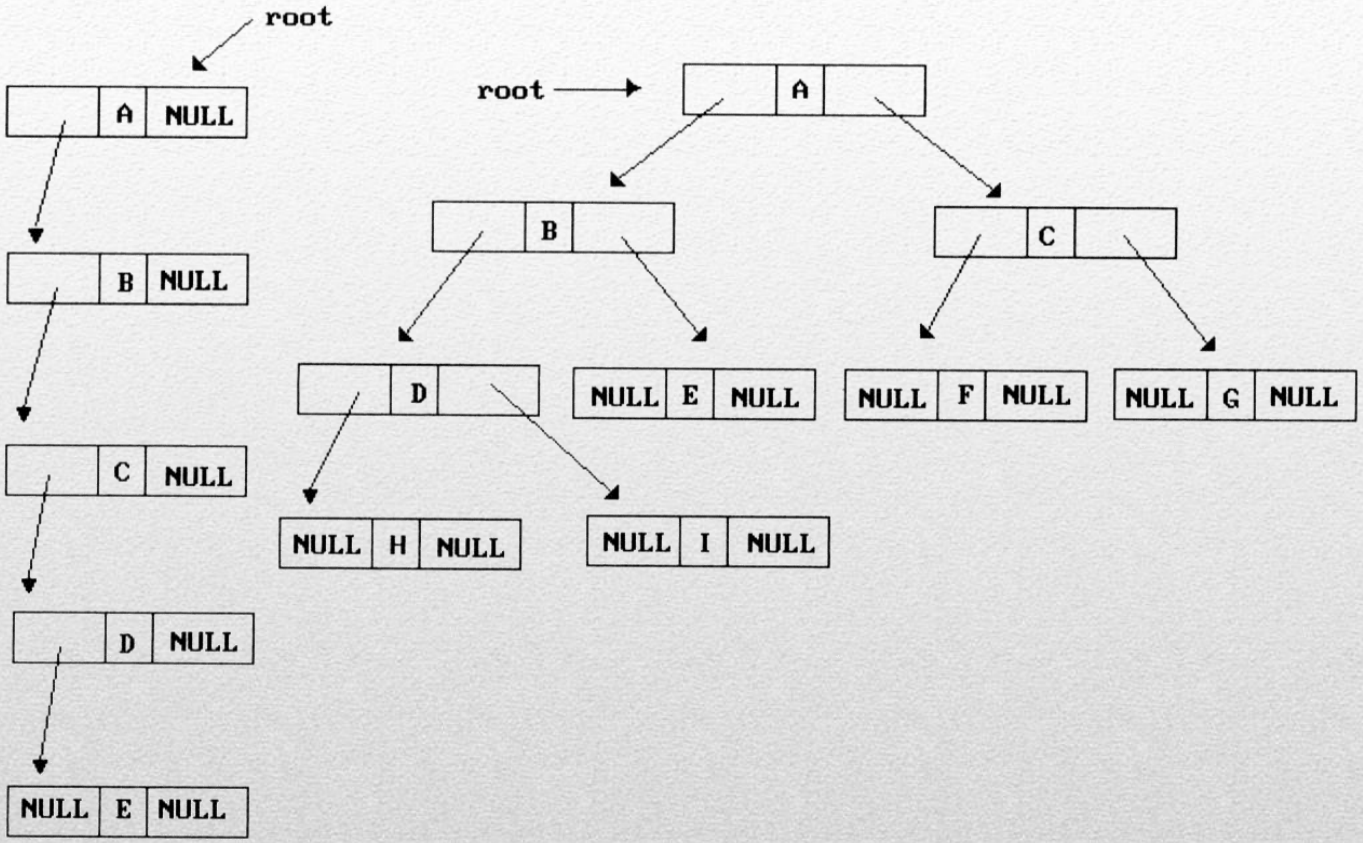




نمایش درختهای دودویی در حافظه

E-mail: Hadi.khademi@gmail.com

- نمایش به صورت پیوندی



نمایش درختهای دودویی در حافظه

E-mail: Hadi.khademi@gmail.com

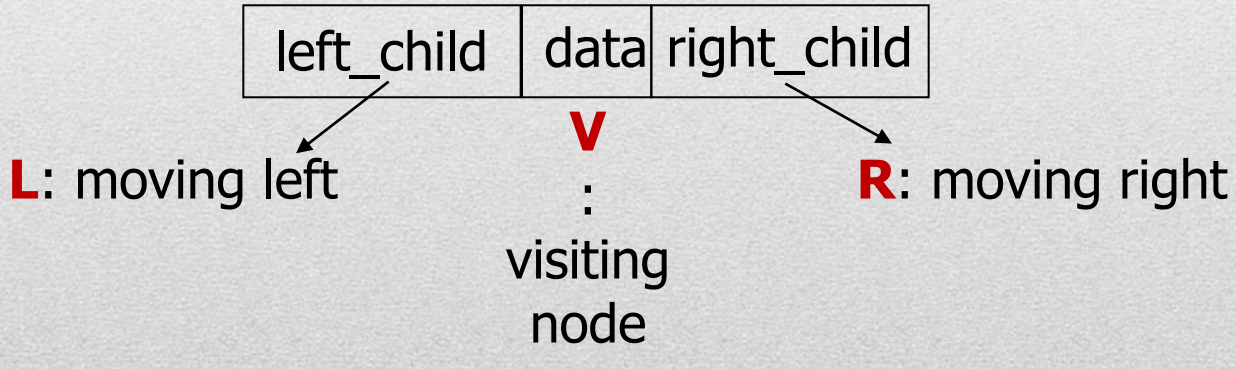
- چگونه می توان هر گره درخت را فقط یک بار ملاقات کرد؟

برای پیمایش درخت ۶ ترکیب ممکن زیر وجود دارد که در آن L، V و R به ترتیب حرکت به چپ ، ملاقات کردن یک گره (برای مثال ، چاپ فیلد داده آن گره) و حرکت به راست می باشد

- LVR, LRV, VLR, VRL, RVL, RLV

اگر قرارداد کنیم که ابتدا پیمایش را از چپ و بعد راست انجام می دهیم ۳ نوع پیمایش باقی می ماند

- LVR (inorder), LRV (postorder), VLR (preorder)



پیمایش درخت دودویی

E-mail: Hadi.khademi@gmail.com

- در پیمایش میان ترتیب inorder یک گره موقعی ملاقات و چاپ می شود که زیردرخت چپ آن قبلا ملاقات شده باشند.

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

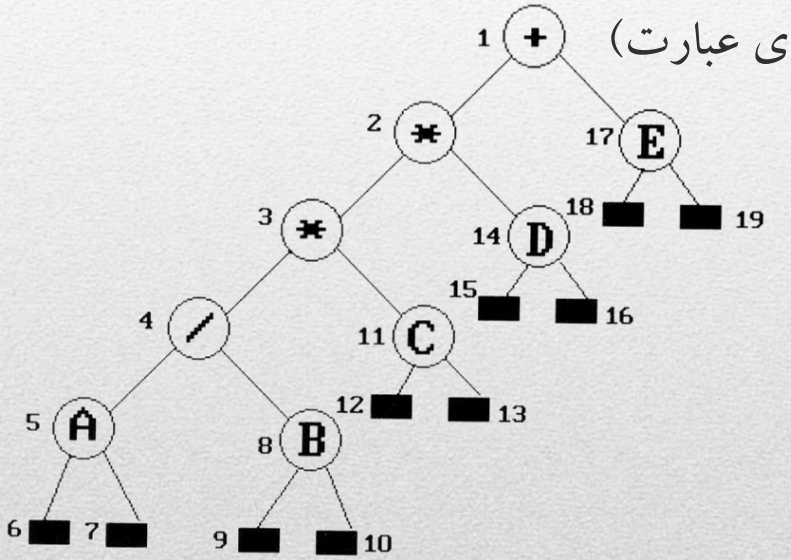
- در پیمایش پیش ترتیب preorder ، یک گره قبل از پیمایش زیردرختان چپ و راست ، ملاقات می گردد.

LVR (inorder), LRV (postorder), VLR (preorder)

پیمایش درخت دودویی

E-mail: Hadi.khademi@gmail.com

- درخت دودویی با عبارت های محاسباتی
  - پیمایش میان ترتیب (صورت میانوندی عبارت)
    A / B * C * D + E
  - پیمایش پیش ترتیب (صورت پیشوندی عبارت)
    + * * / A B C D E
  - پیمایش پس ترتیب (صورت پسوندی عبارت)
    A B / C * D * E +
  - پیمایش به ترتیب سطح
    + * E * D / C A B



پیمایش درخت دودویی

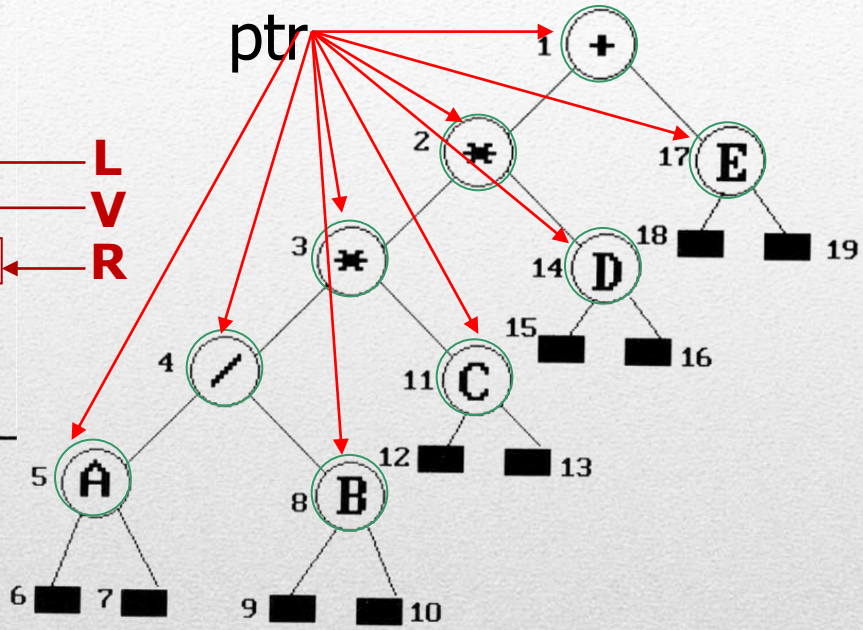
E-mail: Hadi.khademi@gmail.com

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

L

V

R

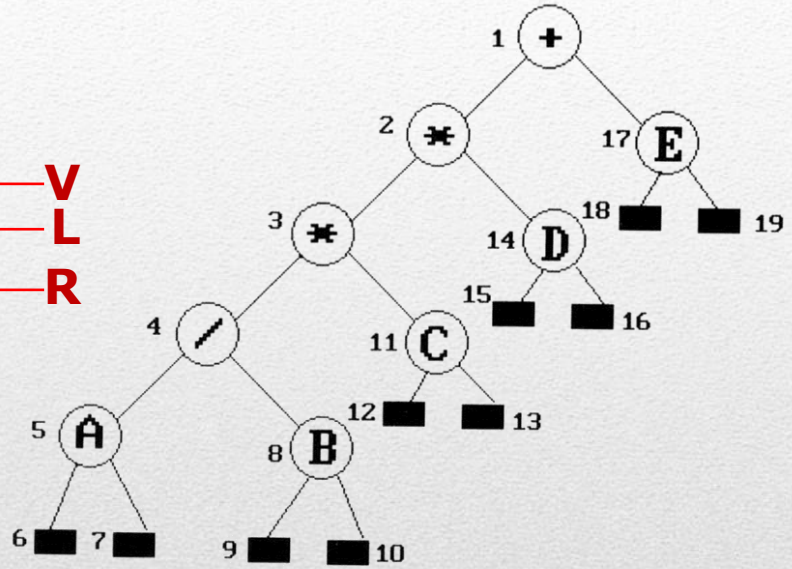


output: A / B* C* D+ E

پیمایش درخت دودویی (میان ترتیب بازگشتی)

E-mail: Hadi.khademi@gmail.com

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

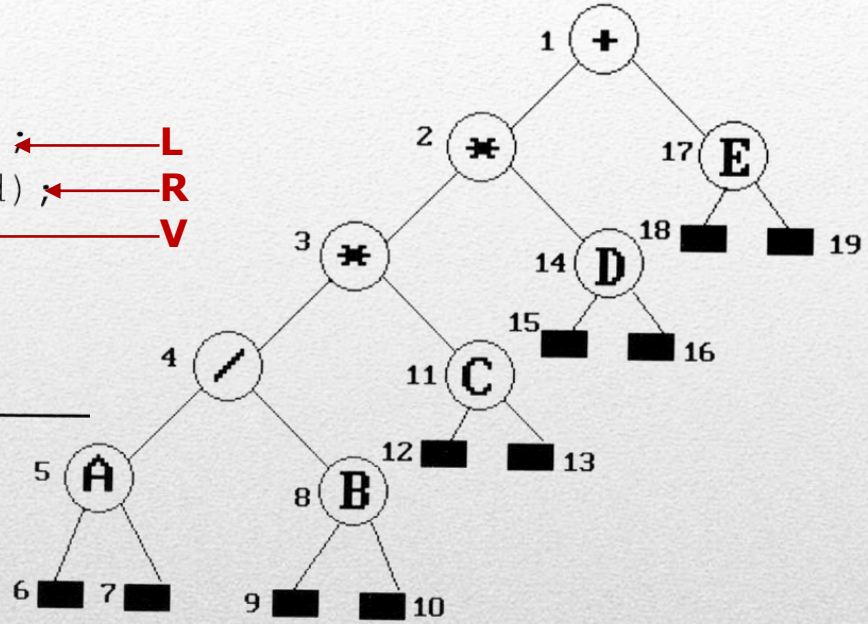


output: + * * / A B C D E

پیمایش درخت دودویی (پیش ترتیب بازگشتی)

E-mail: Hadi.khademi@gmail.com

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



output: A B / C *D *E +

پیمایش درخت دودویی (پس ترتیب بازگشتی)

E-mail: Hadi.khademi@gmail.com

- **پیمایش میان ترتیب**
  - در پیمایش میان ترتیب حرکت به سمت پایین درخت و نیز به سمت چپ انجام می شود و این عمل تا آخرین گره صورت می گیرد.
  - آنگاه می توان گره را ملاقات کرد، یک گره به سمت راست حرکت کرد و به همین ترتیب به کار خود ادامه داد.
  - اگر نتوانیم به سمت راست حرکت کنیم یک گره یا بیشتر به عقب بر می گردیم

پیمایش درخت دودویی (میان ترتیب به روش تکرار)

E-mail: Hadi.khademi@gmail.com

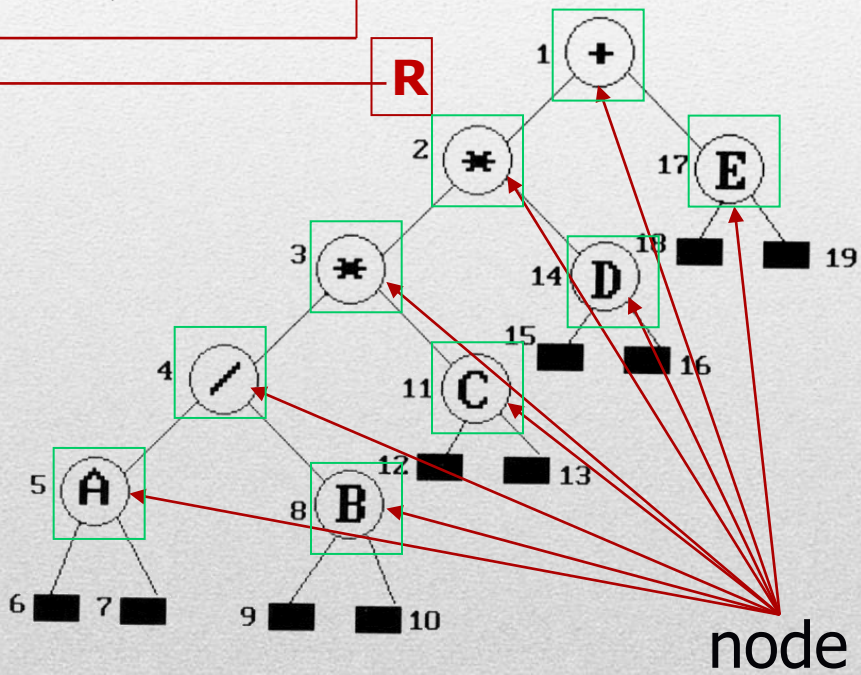
```
void iter_inorder(tree_pointer node)
{
   int top = -1; /* initialize stack */
   tree_pointer stack[MAX_STACK_SIZE];
   for (;;) {
      for(; node; node = node->left_child)
         add(&top, node); /* add to stack */
      node = delete(&top); /* delete from stack */
      if (!node) break; /*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

5	8	11	14	17
A	B	C	D	E

L

V

R



output:A /B *C *D +E

پیمایش درخت دودویی (میان ترتیب به روش تکرار)

E-mail: Hadi.khademi@gmail.com

- **تحلیل زمان و حافظه**
  - فرض کنید تعداد گره های درخت n باشد.
  - پیچیدگی زمانی: O(n)

    هر گره درخت فقط و فقط یکبار در پشته قرار داده شده و از آن حذف می شود.
  - پیچیدگی مکانی: O(n)

    حافظه مورد نیاز برای پشته برابر با عمق درخت است که در بدترین حالت (درخت مورب) n است

پیمایش درخت دودویی (میان ترتیب به روش تکرار)

E-mail: Hadi.khademi@gmail.com

- **پیمایش به ترتیب سطح یک درخت دودویی**
  - روش
    - ابتدا ریشه، انگاه بچه چپ ریشه و بعد از آن بچه راست ریشه را ملاقات می کنیم
    - این روش را با ملاقات گره ها در هر سطح جدید از سمت چپ ترین گره تا سمت راست ترین گره ادامه می دهیم.
  - این پیمایش برای پیاده سازی به صف نیاز دارد.

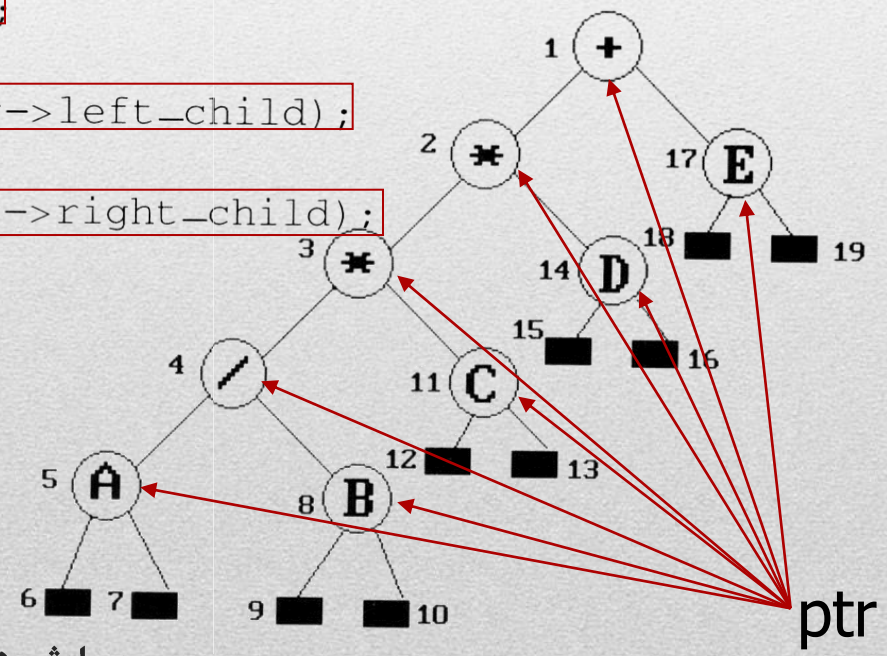
پیمایش درخت دودویی

E-mail: Hadi.khademi@gmail.com

```
void level_order(tree_pointer ptr)
/* level order tree traversal */
{
   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("%d",ptr->data);
         if(ptr->left_child)
            addq(front,&rear,ptr->left_child);
         if (ptr->right_child)
            addq(front,&rear,ptr->right_child);
      }
      else break;
   }
}
```

FIFO

2	17	3	14	4	11	5	8
*	E	*	D	/	C	A	B



output: +* E* D/ C A B

پیمایش درخت دودویی (به ترتیب سطح با کمک یک صف)

E-mail: Hadi.khademi@gmail.com

- **کپی درخت های دودویی**
  - می توانیم الگوریتم پیمایش پس ترتیب درخت دودویی را اندکی تغییر دهیم تا برای کپی درخت به کار رود.

```
tree_pointer copy(tree_pointer original)
/* this function returns a tree_pointer to an exact copy
of the original tree */
{
   tree_pointer temp;
   if (original) {
      temp = (tree_pointer) malloc(sizeof(node));
      if (IS_FULL(temp)) {
         fprintf(stderr, "The memory is full\n");
         exit(1);
      }
      temp->left_child = copy(original->left_child);
      temp->right_child = copy(original->right_child);
      temp->data = original->data;
      return temp;
   }
   return NULL;
}
```

شبیه پیمایش پس ترتیب

عملیات دیگر روی درخت های دودویی

E-mail: Hadi.khademi@gmail.com

- **آزمایش برابری درخت های دودویی**
  - دو درخت دودویی را برابر (هم ارز) گویند اگر ساختار (شکل) یکسان داشته و اطلاعات داخل گره های متناظرشان نیز یکسان باشد.

V L R

```
int equal(tree_pointer first, tree_pointer second)
{
/* function returns FALSE if the binary trees first and
second are not equal, Otherwise it returns TRUE */
   return ((!first && !second) || (first && second &&
            (first->data == second->data) &&
            equal(first->left_child,second->left_child) &&
            equal(first->right_child, second->right_child))
}
```

شکل و ساختار شبیه پیمایش پیش ترتیب

عملیات دیگر روی درخت های دودویی

E-mail: Hadi.khademi@gmail.com

- **مساله صدق پذیری (درستی یک عبارت)**
- **در متغیرها تنها می توان یکی از دو مقدار true یا false را ذخیره کرد.**
- **عملگرها : $\wedge$(and), $\vee$(or), $\neg$(not)**
- **عبارات حساب گزاره ای (Propositional Calculus)**
  - هر متغیر یک عبارت است.
  - اگر x و y دو عبارت باشند آنگاه $\neg x, x \wedge y, x \vee y$ نیز عبارت هستند.
  - از پرانتز می توان برای تغییر ترتیب طبیعی ارزیابی عبارات ($\neg > \wedge > \vee$) استفاده کرد.
- **مثال: $x_1 \vee (x_2 \wedge \neg x_3)$**

عملیات دیگر روی درخت های دودویی

E-mail: Hadi.khademi@gmail.com

- **مساله صدق پذیری:**
  - ایا جایگزینی مقادیری وجود دارد که باعث شود مقدار عبارت true شود؟
- **پاسخ برای مثال : $x_1 \vee (x_2 \wedge \neg x_3)$**
  - x_1 and x_3 =false, x_2 = true
  - $false \vee (true \wedge \neg false) = false \vee true = true$

- **برای n متغیر 2^n ترکیب ممکن وجود دارد.**
  - زمان اجرای صدق پذیری: $g2^n$
  - که g زمان جایگزینی مقادیر در متغیرها و ارزیابی عبارت است.

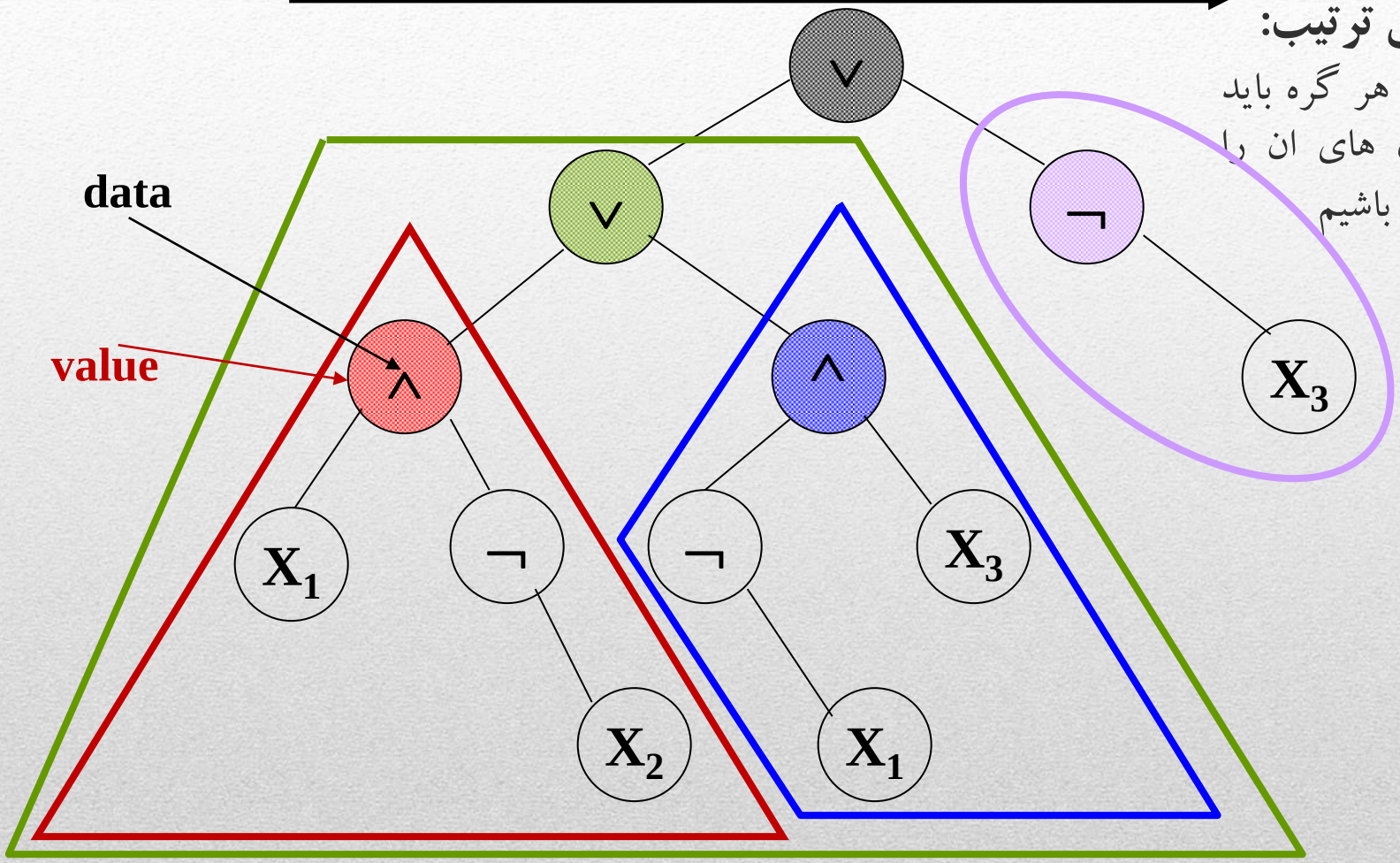
عملیات دیگر روی درخت های دودویی (مساله صدق پذیری)

E-mail: Hadi.khademi@gmail.com

$$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee \neg x_3$$

پیمایش پس ترتیب:

برای ارزیابی هر گره باید قبلا ارگومان های ان را محاسبه کرده باشیم



عملیات دیگر روی درخت های دودویی (مساله صدق پذیری)

E-mail: Hadi.khademi@gmail.com

- **ساختار گره**
  - data={ not, and, or, true, false} نوع گره را مشخص می کند.
  - value مقدار ارزیابی شده تا این گره
  - اشاره گر بچه چپ و راست مانند سایر درخت ها

left_child	*data*	*value*	*right_child*

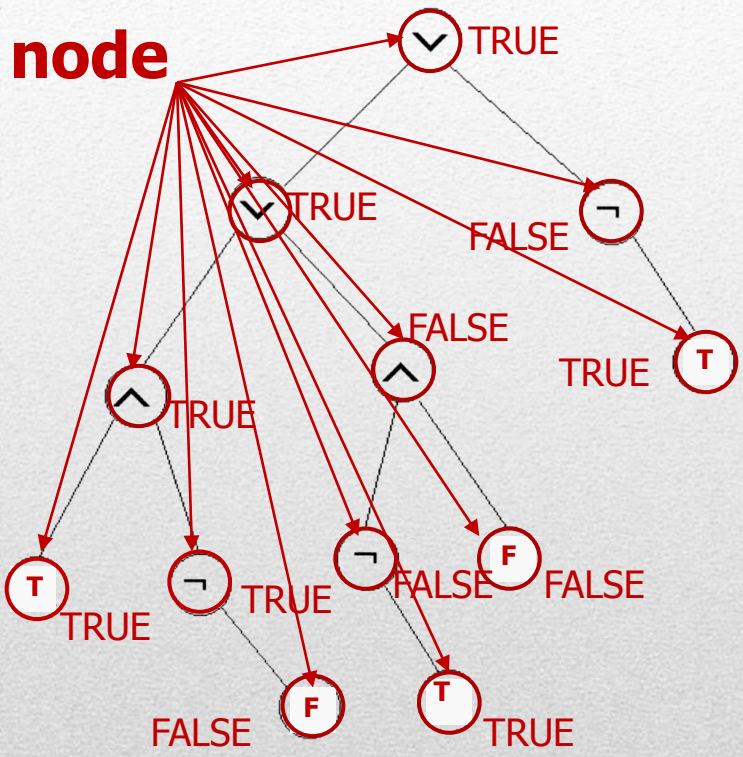
```
typedef enum {not,and,or,true,false} logical;
typedef struct node *tree_pointer;
typedef struct node {
        tree_pointer left_child;
        logical      data;
        short int    value;
        tree_pointer right_child;
        } ;
```

- فرض : در گره های پایانی مقدار data حاوی مقدا رمتغیر ارائه شده در این گره است

عملیات دیگر روی درخت های دودویی (مساله صدق پذیری)

E-mail: Hadi.khademi@gmail.com

ارزیابی یک فرمول




```
void post_order_eval(tree_pointer node)
{
/* modified post order traversal to evaluate a
propositional calculus tree */
   if (node) {
      post_order_eval(node->left_child);
      post_order_eval(node->right_child);
      switch(node->data) {
         case not:   node->value =
                     !node->right_child->value;
                     break;
         case and:   node->value =
                     node->right_child->value &&
                     node->left_child->value;
                     break;
         case or:    node->value =
                     node->right_child->value ||
                     node->left_child->value;
                     break;
         case true:  node->value = TRUE;
                     break;
         case false: node->value = FALSE;
      }
   }
}
```

L
R
V

عملیات دیگر روی درخت های دودویی (مساله صدق پذیری)

E-mail: Hadi.khademi@gmail.com

- الگوریتمی به نام swaptree بنویسید که یک درخت دودویی را دریافت کند و جای بچه های چپ و راست هر گره را عوض کند.
- الگوریتمی برای شمارش تعداد گره های برگ /گره های غیر برگ /کل گره ها در یک درخت دودویی بنویسید.
- الگوریتمی برای حذف تمام گره ها در یک درخت دودویی بنویسید.

تمرین

E-mail: Hadi.khademi@gmail.com

- **نخ کشی کردن**
  - عیب درختهای دودویی بالا: تعداد زیادی از اشاره گرها NULL است
    - تعداد گره n
    - تعداد اشاره گر غیر صفر n-1
    - تعداد کل اشاره گر ها 2n
    - تعداد اشاره گر صفر n+1
  - راهکار : به جای اشاره گرهای صفر از این اشاره گرها برای اشاره به گره های دیگر درخت استفاده کنیم. به چنین اشاره گرهایی نخ گویند.

درختهای دودویی نخ کشی شده

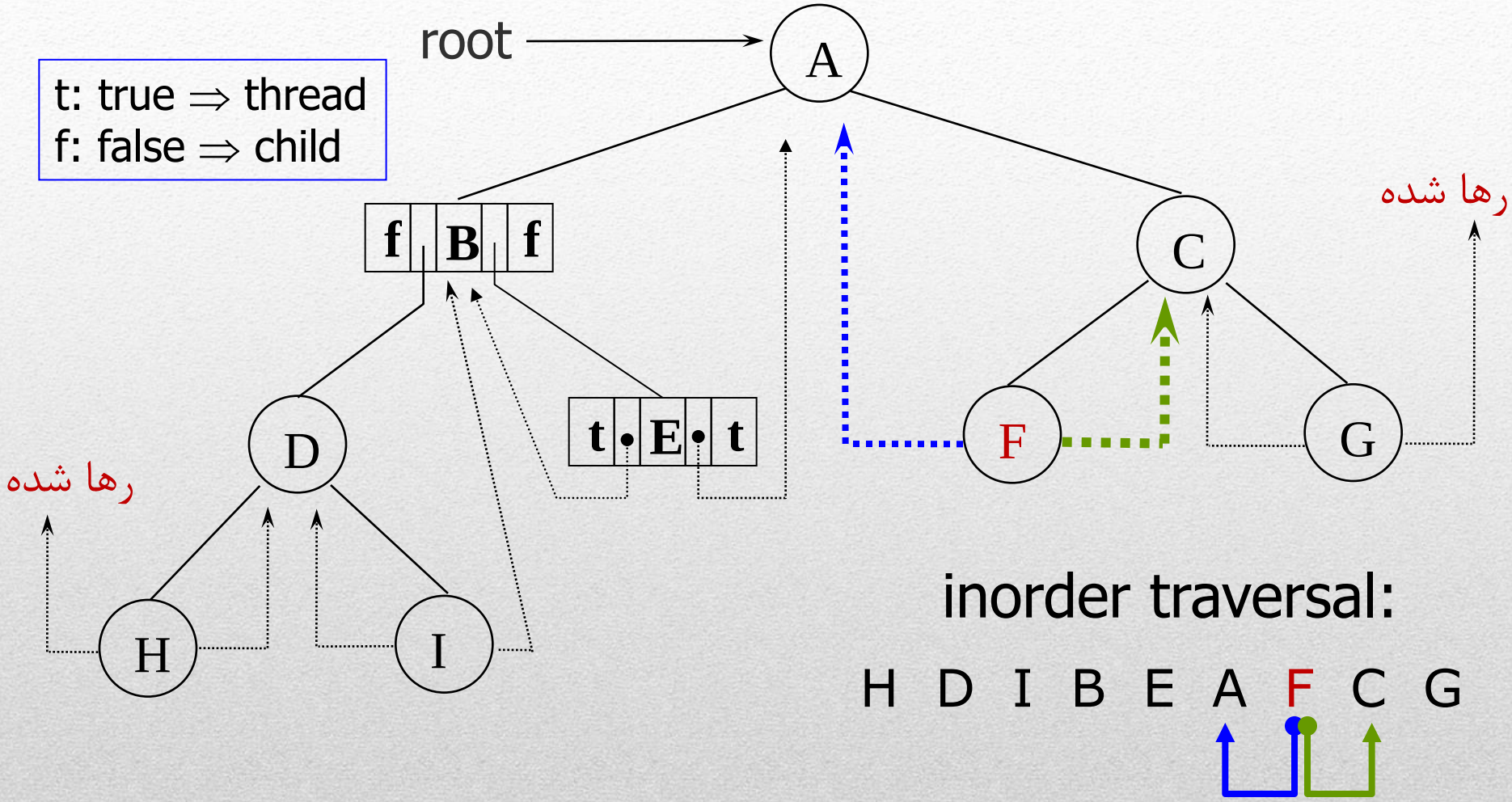
E-mail: Hadi.khademi@gmail.com

- **درختهای نخ کشی شده با استفاده از قوانین زیر ساخته می شوند:**
  - اگر **ptr->left_child** صفر باشد تبدیل به اشاره گر به گره ای می شود که در پیمایش میان ترتیب بلافاصله قبل از ptr ملاقات می شود.
  - اگر **ptr->right_child** صفر باشد تبدیل به اشاره گر به گره ای می شود که در پیمایش میان ترتیب بلافاصله بعد از ptr ملاقات می شود.

درختهای دودویی نخ کشی شده

E-mail: Hadi.khademi@gmail.com

- یک درخت دودویی نخ کشی شده



درختهای دودویی نخ کشی شده

E-mail: Hadi.khademi@gmail.com

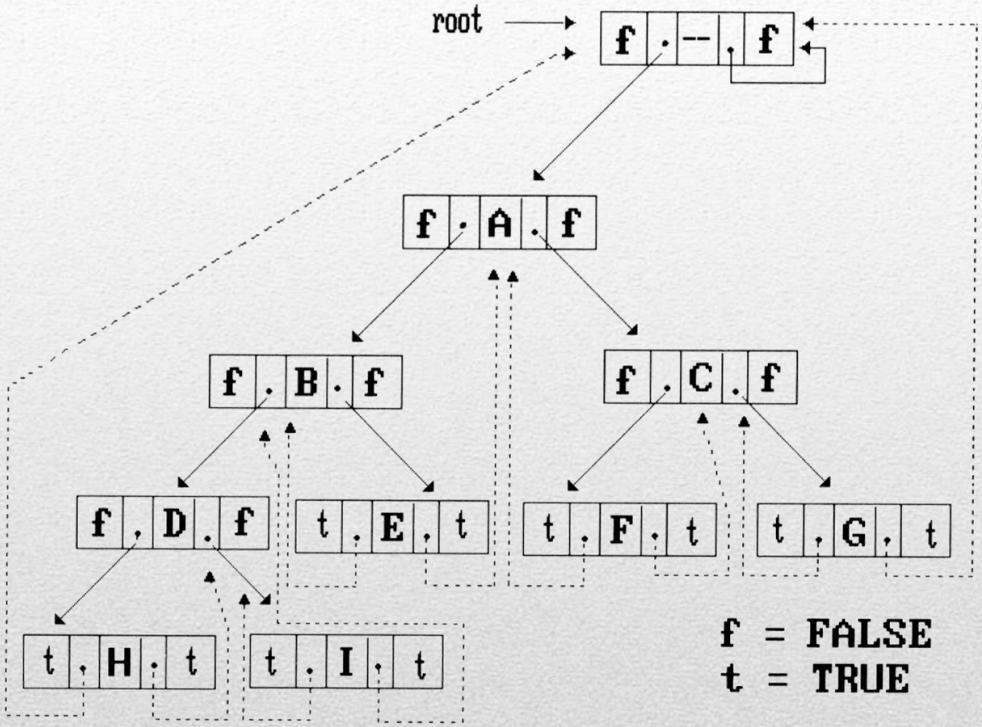
- **اضافه کردن دو فیلد boolean به نامهای left-thread و right-thread**
  - اگر ptr->left-thread=TRUE
  - انگاه ptr->left-child حاوی نخ کشی است در غیر اینصورت حاوی یک اشاره گر به بچه چپ است.
  - همین ترتیب در مورد بچه راست هم صادق است

```
typedef struct threaded_tree *threaded_pointer;
typedef struct threaded_tree {
        short int left_thread;
        threaded_pointer left_child;
        char data;
        threaded_pointer right_child;
        short int right_thread;
        };
```

درختهای دودویی نخ کشی شده

E-mail: Hadi.khademi@gmail.com

- اگر نخواهیم که اشاره گر چپ Hو اشاره گر راست G رها شده باقی بمانند یک گره سر برای درخت دودویی در نظر می گیریم



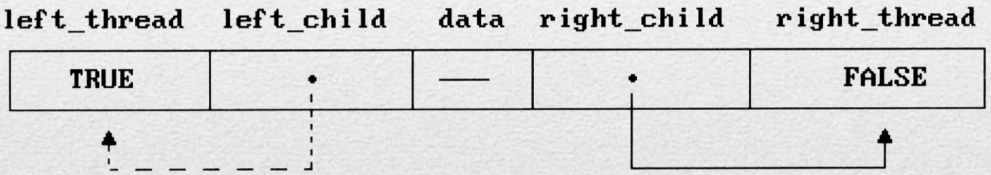
درختهای دودویی نخ کشی شده

E-mail: Hadi.khademi@gmail.com

- **در گره سر :**

اگر درخت تهی نباشد left_child به درخت اشاره می کند و Left_thread، false می شود.

چنانچه درخت تهی باشد left_child به گره سر درخت اشاره می کند و Left_thread، true می شود.



درختهای دودویی نخ کشی شده

E-mail: Hadi.khademi@gmail.com

- **پیمایش میان ترتیب درخت دودویی نخ کشی شده**

با استفاده از نخ کشیها می توان پیمایش میان ترتیب را بدون استفاده از پشته انجام داد.

۱- اگر ptr->right_thread = TRUE باشد آنگاه عضو بعد از ptr در روش میان ترتیب با توجه به تعریف نخ کشیها ptr->right_child است.

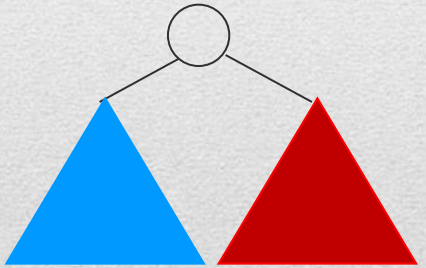
۲- در غیر اینصورت عضو بعد از ptr در روش میان ترتیب با دنبال کردن مسیری از اشاره گرهای بچه چپ از بچه راست ptr به دست می آید تا به گره ای با left_thread = TRUE برسیم.

درختهای دودویی نخ کشی شده

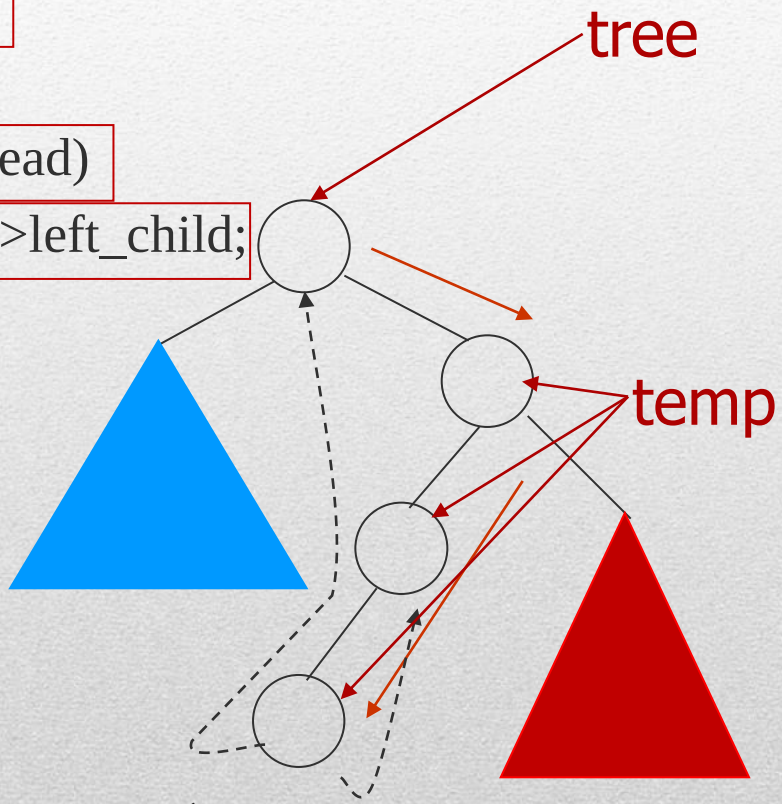
E-mail: Hadi.khademi@gmail.com

- پیدا کردن عضو بعدی (Successor) در یک درخت دودویی نخ کشی شده

```
threaded_pointer insucc(threaded_pointer tree){
        threaded_pointer temp;
        temp = tree->right_child;
        if (!tree->right_thread)
                while (!temp->left_thread)
                        temp = temp->left_child;
        return temp;
}
```



Inorder



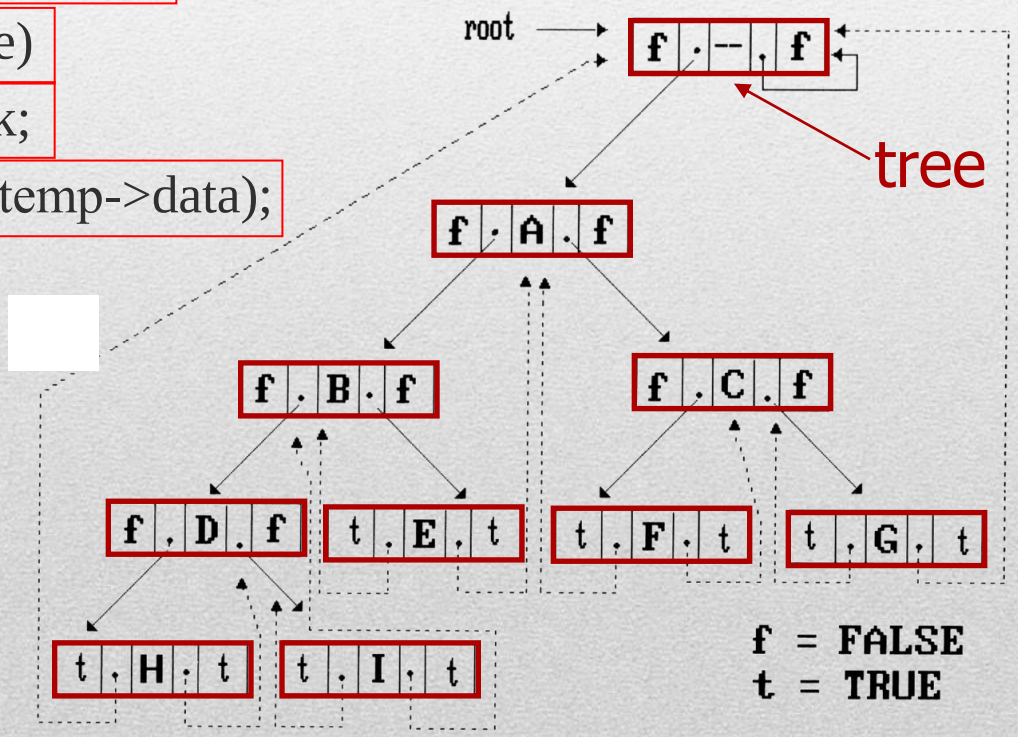
درختهای دودویی نخ کشی شده

E-mail: Hadi.khademi@gmail.com

- پیمایش میان ترتیب درخت دودویی نخ کشی شده

```
void tinorder(threaded_pointer tree){
/* traverse the threaded binary tree inorder */
        threaded_pointer temp = tree;
        for (;;) {
                temp = insucc(temp);
                if (temp==tree)
                        break;
                printf("%3c",temp->data);
        }
}
```

output: H D I B E A F C G



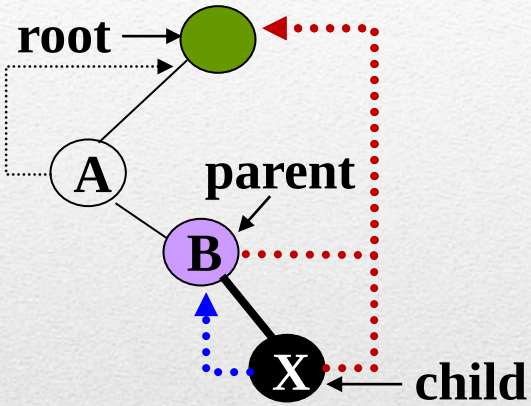
Time Complexity: O(*n*)

درختهای دودویی نخ کشی شده

E-mail: Hadi.khademi@gmail.com

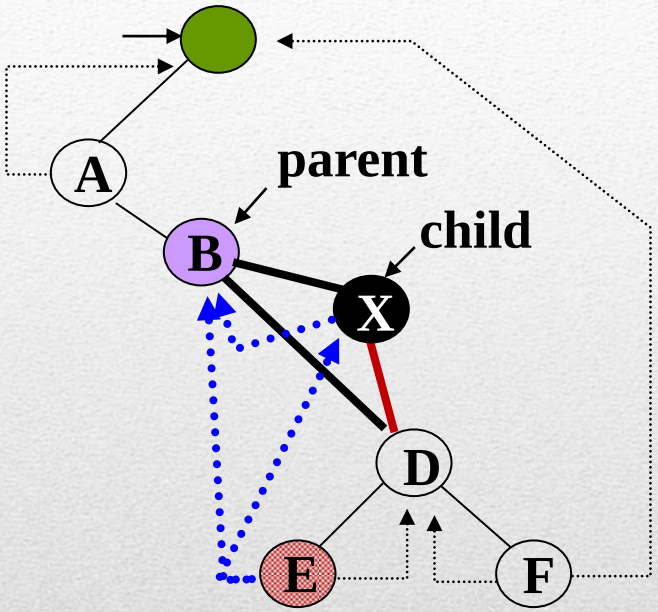
- **اضافه کردن گره به درخت دودویی نخ کشی شده**

اضافه کردن یک گره به عنوان بچه راست یک گره والد



حالت اول:

زیر درخت راست B خالی باشد



حالت دوم:

زیر درخت راست Bخالی نباشد

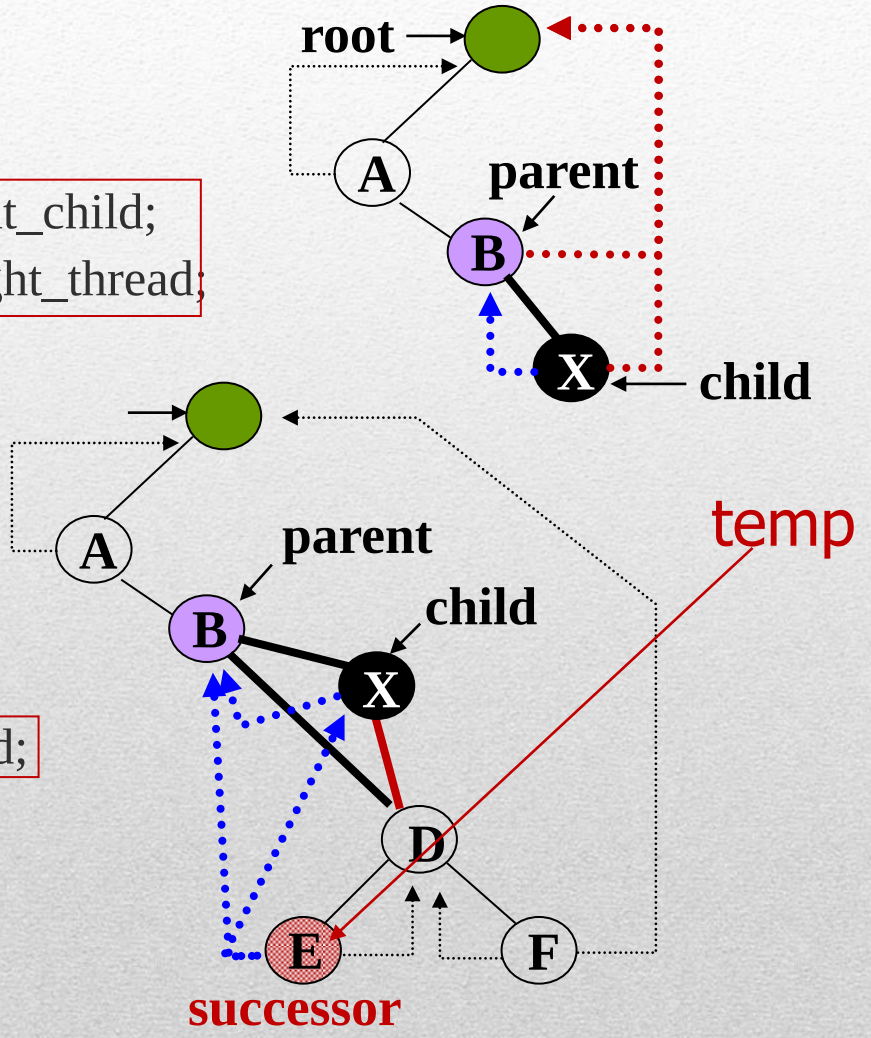
- **در حالت دوم** عنصر بعدی X در پیمایش میان ترتیب را پیدا کرده و عنصر قبل از آن را X قرار می دهیم

درختهای دودویی نخ کشی شده

E-mail: Hadi.khademi@gmail.com

- اضافه کردن گره به صورت بچه راست والد در یک درخت دودویی نخ کشی شده

```
void insert_right(thread_pointer parent, threaded_pointer child){
/* insert child as the right child of
parent in a threaded binary tree */
        threaded_pointer temp;
        child->right_child = parent->right_child;
        child->right_thread = parent->right_thread;
        child->left_child = parent;
        child->left_thread = TRUE;
        parent->right_child = child;
        parent->right_thread = FALSE;
        If(!child->right_thread){
                temp = insucc(child);
                temp->left_child = child;
        }
}
```



حالت دوم

درختهای دودویی نخ کشی شده

E-mail: Hadi.khademi@gmail.com

- **عضوی که باید حذف شود عضو با بالاترین یا پایین ترین اولویت است**
- **مثال : تخصیص منابع**

هر وقت که کامپیوتری آزاد شد

کاربری که متقاضی زمان کمتری است انتخاب شود : (صف اولویت مینیموم)

در هر عمل حذف عضوی که کوچکترین مقدار کلید را دارد حذف می شود

کاربری که پول بیشتری پرداخته است انتخاب شود: (صف اولویت ماکزیموم)

در هر عمل حذف عضوی که بزگترین مقدار کلید را دارد حذف می شود

صف های اولویت

E-mail: Hadi.khademi@gmail.com

- **صف اولویت ماکزیمم (صف مینیموم به صورت مشابه قابل تعریف)**
- **ساده ترین پیاده سازی : لیست خطی مرتب نشده**
- **نمایش ترتیبی**

اضافه کردن عضو به آسانی در انتهای صف O(1)

حذف یک عضو نیازمند جستجوی عضو با بزرگترین کلید O(n)

- **نمایش پیوندی**

اضافه کردن عضو به آسانی در ابتدای زنجیر O(1)

حذف یک عضو نیازمند جستجوی عضو با بزرگترین کلید O(n)

صف های اولویت

E-mail: Hadi.khademi@gmail.com

- **صف اولویت ماکزیمم (صف مینیموم به صورت مشابه قابل تعریف)**
- **ساده ترین پیاده سازی : لیست خطی مرتب نشده**

- **نمایش ترتیبی Unordered array**

اضافه کردن عضو به آسانی در انتهای صف O(1)

حذف یک عضو نیازمند جستجوی عضو با بزرگترین کلید O(n)

- **نمایش پیوندی Unordered linked list**

اضافه کردن عضو به آسانی در ابتدای زنجیر O(1)

حذف یک عضو نیازمند جستجوی عضو با بزرگترین کلید O(n)

صف های اولویت

E-mail: Hadi.khademi@gmail.com

- صف اولویت ماکزیمم (صف مینیموم به صورت مشابه قابل تعریف)
- پیاده سازی با لیست خطی مرتب
- نمایش ترتیبی Sorted Array

عضوها به صورت غیر نزولی

اضافه کردن عضو	O(n)
حذف یک عضو	O(1)

در صورت استفاده ار heap ماکزیمم زمان حذف و اضافه O(log n) می شود

- نمایش پیوندی Sorted linked list

عضوها به صورت غیر صعودی

اضافه کردن عضو	O(n)
حذف یک عضو	O(1)

صف های اولویت

E-mail: Hadi.khademi@gmail.com

- صف اولویت ماکزیمم (صف مینیموم به صورت مشابه قابل تعریف)

Representation	Insertion	Deletion
Unordered array	$\Theta(1)$	$\Theta(n)$
Unordered linked list	$\Theta(1)$	$\Theta(n)$
Sorted array	$O(n)$	$\Theta(1)$
Sorted linked list	$O(n)$	$\Theta(1)$
Max heap	$O(\log_2 n)$	$O(\log_2 n)$

صف های اولویت

E-mail: Hadi.khademi@gmail.com

- تعریف

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

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

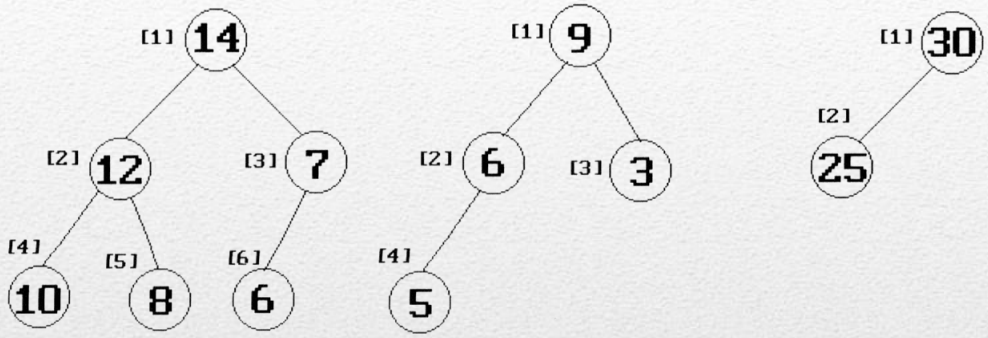
min tree درختی است که مقدار کلید هر گره آن بیشتر از مقادیر کلیدهای فرزندانش نباشد.

min heap یک درخت دودویی کامل است که همزمان min tree نیز می باشد.

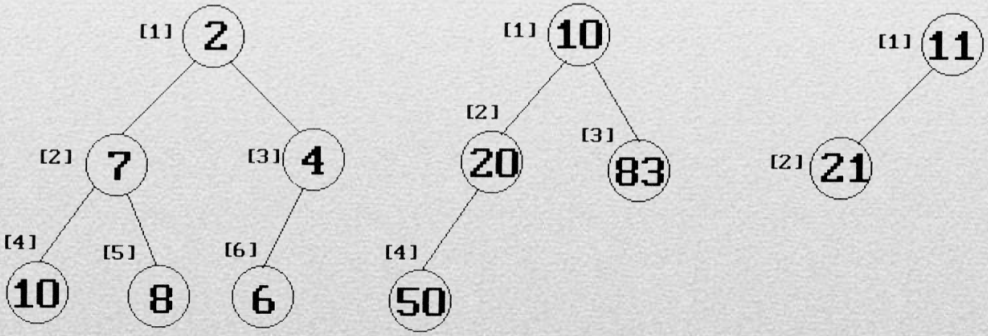
Heap

E-mail: Hadi.khademi@gmail.com

- مقدار کلید در گره ریشه در max heap (min heap) بزرگترین (کوچکترین) کلید در درخت است.
- مثال



max heap



min heap

Heap

E-mail: Hadi.khademi@gmail.com

- **عملیات اصلی روی یک Heap**
  - ایجاد یک Heap خالی
  - اضافه کردن یک عنصر جدید به Heap
  - حذف بزرگترین عنصر از Heap

Heap

- نوع داده مجرد Heap

structure *MaxHeap* is

objects: a complete binary tree of $n > 0$ elements organized so that the value in each node is at least as large as those in its children

functions:

for all *heap* ∈ *MaxHeap*, *item* ∈ *Element*, *n*, *max_size* ∈ integer

MaxHeap Create(*max_size*)	::=	create an empty heap that can hold a maximum of *max_size* elements.
Boolean HeapFull(*heap*, *n*)	::	**if** (*n* == *max_size*) **return** *TRUE* **else return** *FALSE*
MaxHeap Insert(*heap*, *item*, *n*)	::=	**if** (!HeapFull(*heap*, *n*)) insert *item* into *heap* and return the resulting heap **else return** error.
Boolean HeapEmpty(*heap*, *n*)	::	**if** ($n > 0$) **return** *TRUE* **else return** *FALSE*
Element Delete(*heap*, *n*)	::=	**if** (!HeapEmpty(*heap*, *n*)) **return** one instance of the largest element in the heap and remove it from the heap **else return** error.

Heap

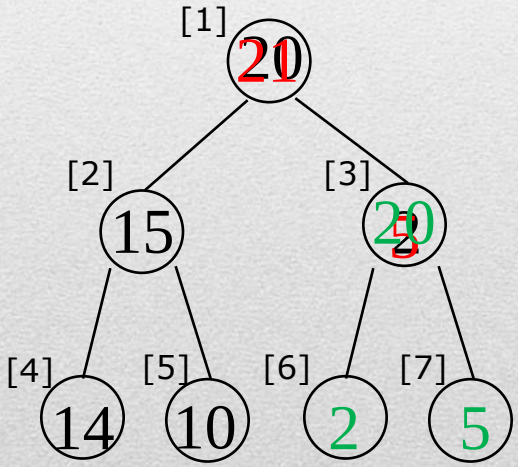
E-mail: Hadi.khademi@gmail.com

پیچیدگی اضافه کردن عنصر $O(\log_2 n)$

insert 21

*n= 6

i= 3




```
void insert_max_heap(element item, int *n)
{
/*insert item into a max heap of current size *n */
   int i;
   if (HEAP_FULL(*n)){
      fprintf(stderr, "The heap is full. \n");
      exit(1);
   }
   i = ++(*n);
   while ((i != 1) && (item.key > heap[i/2].key)) {
      heap[i] = heap[i/2];
      i /= 2;
   }
   heap[i] = item;
}
```

parent link

item up heap

اضافه کردن یک عنصر در Max Heap

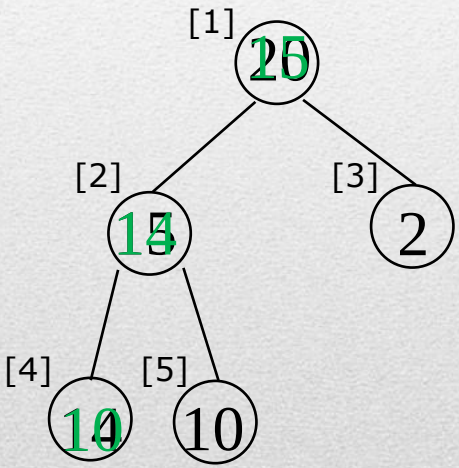
E-mail: Hadi.khademi@gmail.com

پیچیدگی حذف کردن عنصر $O(\log_2 n)$

parent = 1 4

child = 2 8

*n= 5 4




```
element delete_max_heap(int *n)
{
/* delete element with the highest key from the heap */
   int parent, child;
   element item, temp;
   if (HEAP_EMPTY(*n)) {
      fprintf(stderr, "The heap is empty\n");
      exit(1);
   }
   /* save value of the element with the highest key */
   item = heap[1];
   /* use last element in heap to adjust heap */
   temp = heap[(*n)--];
   parent = 1;
   child = 2;
   while (child <= *n) {
      /* find the larger child of the current parent */
      if (child < *n) && (heap[child].key
   < heap[child+1].key)
         child++;
      if (temp.key >= heap[child].key) break;
      /* move to the next lower level */
      heap[parent] = heap[child];
      parent = child;
      child *= 2;
   }
   heap[parent] = temp;
   return item;
}
```

item.key = 20

temp.key = 10

حذف کردن یک عنصر از Max Heap

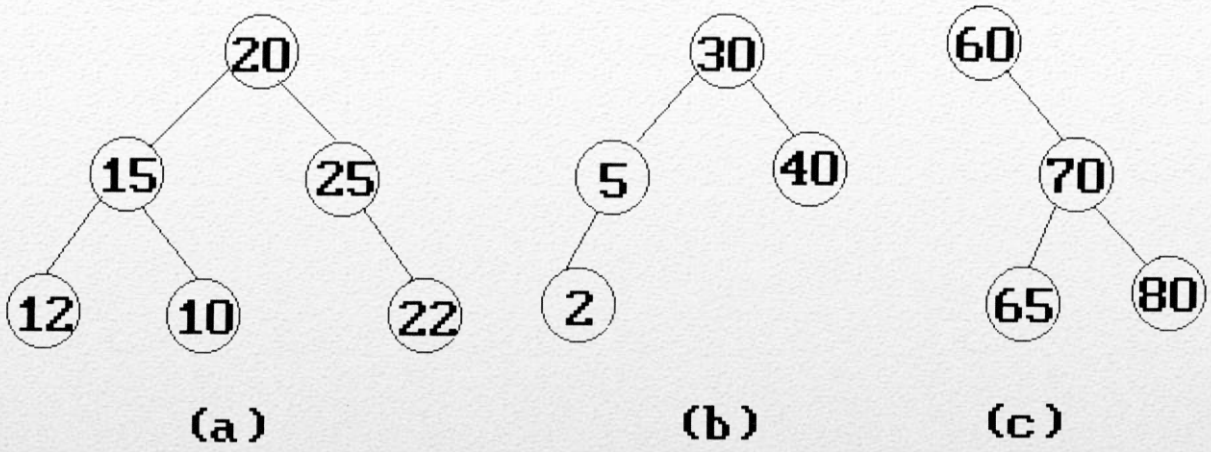
E-mail: Hadi.khademi@gmail.com

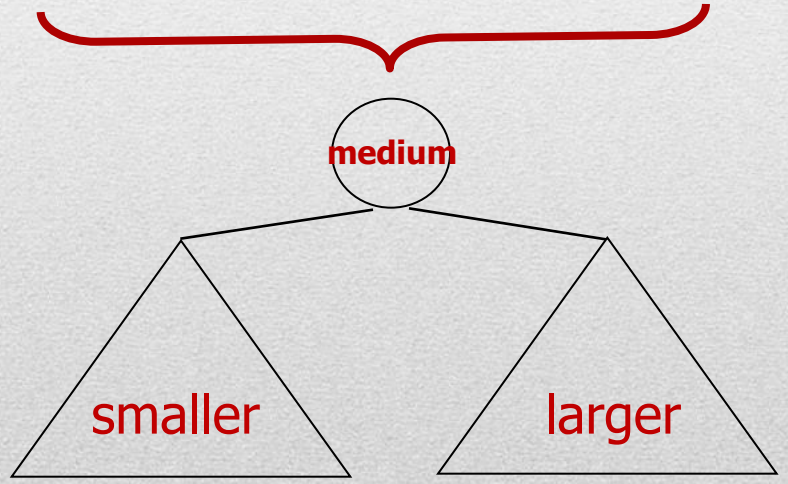
- **دلیل نیاز به درختهای جستجوی دودویی**
  - Heap برای کاربردهایی که نیاز به حذف یک عنصر دلخواه دارند اصلا مناسب نیست.
  - حذف کوچکترین (بزرگترین) عنصر $O(\log_2 n)$
  - حذف یک عنصر دلخواه $O(n)$
  - جستجوی یک عنصر دلخواه $O(n)$
- **تعریف درخت جستجوی دودویی**
  - هر عنصر دارای یک کلید است.
  - کلیدهای زیردرخت غیرتهی چپ (راست) کوچکتر (بزرگتر) از کلید واقع در ریشه هستند.
  - زیردرختهای چپ و راست نیز خود درختهای جستجوی دودویی هستند.

درختهای جستجوی دودویی

E-mail: Hadi.khademi@gmail.com

- مثال: b و c درخت جستجوی دودویی هستند





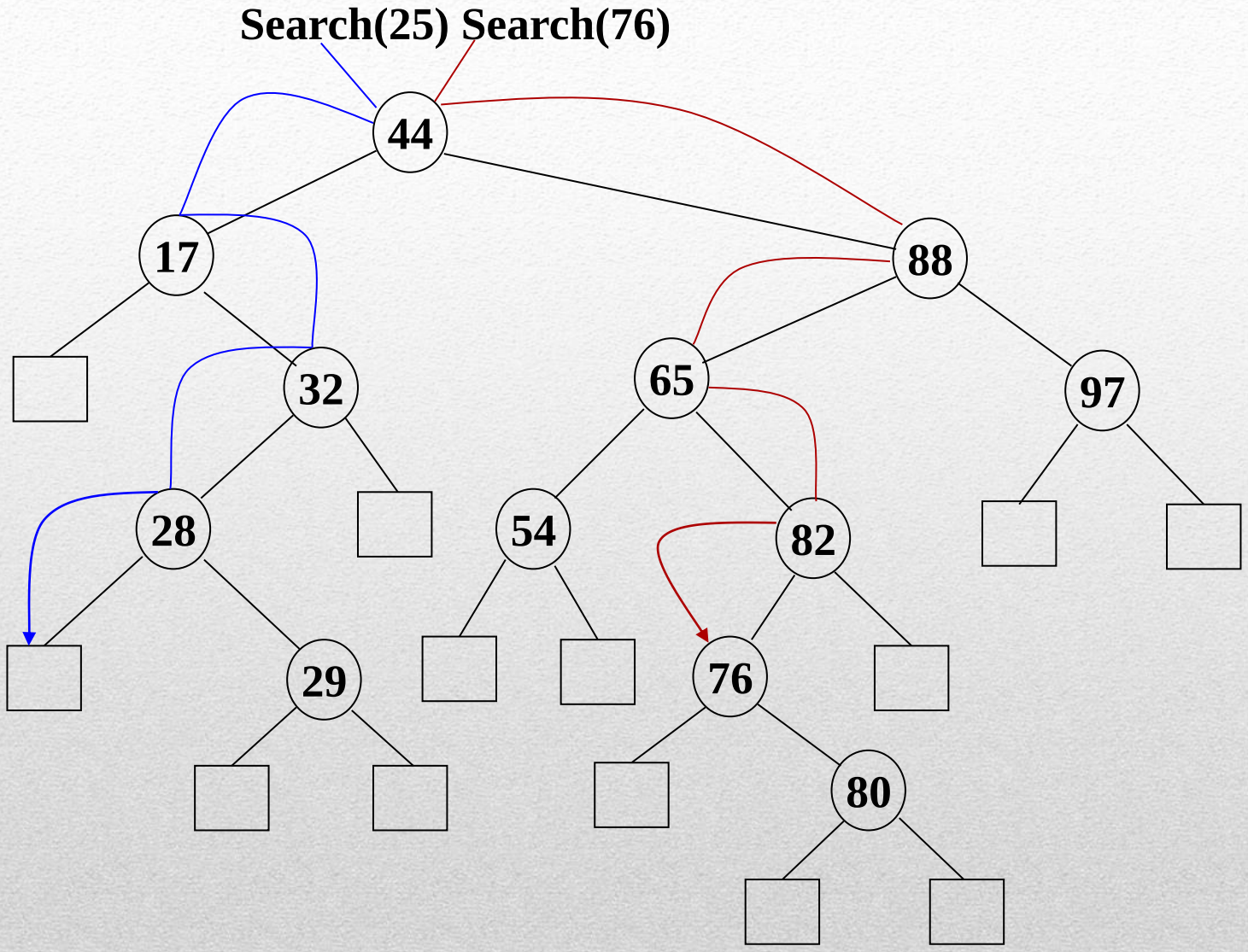
درختهای جستجوی دودویی

E-mail: Hadi.khademi@gmail.com

- فرض کنید می خواهیم مکان عنصری با کلید X را پیدا کنیم. کار خود را 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 X را با مقدار کلید ریشه مقایسه می کنیم :
- اگر X برابر با این کلید باشد آنگاه جستجو به طور موفقیت آمیز به پایان می رسد.
- اگر X کوچکتر از مقدار کلید ریشه باشد، هیچ عنصری در زیردرخت راست نمی تواند که مقدار کلید X را دارا باشد و فقط زیر درخت چپ باید جستجو شود.
- اگر X بزرگتر از مقدار کلید ریشه باشد، فقط لازم است زیردرخت راست را جستجو می کنیم.

جستجوی یک عضو در درخت جستجوی دودویی

E-mail: Hadi.khademi@gmail.com



جستجوی یک عضو در درخت جستجوی دودویی

E-mail: Hadi.khademi@gmail.com

جستجوی بازگشتی

```
tree_pointer search(tree_pointer root, int key)
{
/* return a pointer to the node that contains key.  If
there is no such node, return NULL. */
   if (!root) return NULL;
   if (key == root->data) return root;
   if (key < root->data)
      return search(root->left_child, key);
   return search(root->right_child,key);
}
```

جستجوی تکراری

```
tree_pointer search2(tree_pointer tree, int key)
{
/* return a pointer to the node that contains key.  If
there is no such node, return NULL. */
   while (tree) {
      if (key == tree->data) return tree;
      if (key < tree->data)
         tree = tree->left_child;
      else
         tree = tree->right_child;
      }
   return NULL;
}
```

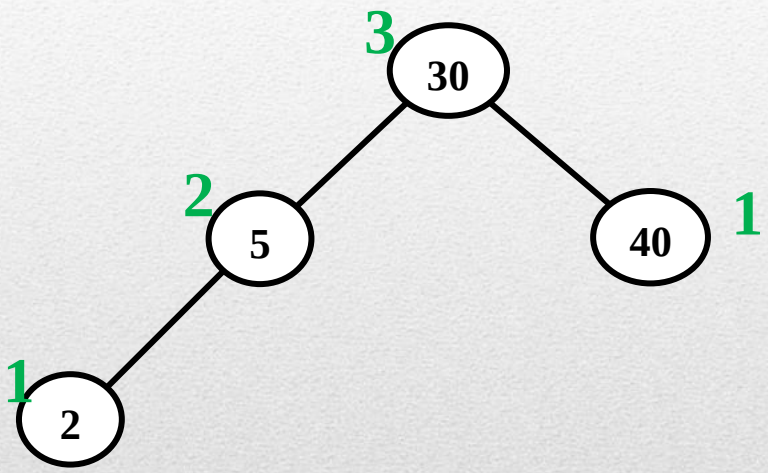
O(h)

اگر h ارتفاع یا عمق یک درخت جستجوی دودویی باشد، عمل جستجو در مدت O(h) انجام می شود.

جستجوی یک عضو در درخت جستجوی دودویی

E-mail: Hadi.khademi@gmail.com

- هر گره دارای یک فیلد اضافی به نام leftsize است که یکی بیشتر از تعداد عضوهای زیر درخت چپ گره است.



جستجوی درخت - جستجوی دودویی (از طریق اندیس)

E-mail: Hadi.khademi@gmail.com

```
tree-pointer  Indexsearch(tree-pointer  tree, int k)
//Search the binary search tree for the kth smallest element
{
        tree –pointer  t = tree;
        while ( t)
        {
                if ( k== t->leftsize) return t;
                if  ( k<  t-> leftsize) t= t-> leftchild;
                else {
                        k - = t -> leftsize;
                        t    = t -> rightchild;
                }
        }
        return 0;
}
```

اگر h ارتفاع یا عمق یک درخت جستجوی دودویی باشد، عمل جستجو در مدت O(h) انجام می شود.

جستجوی درخت – جستجوی دودویی (از طریق اندیس)

E-mail: Hadi.khademi@gmail.com

- برای درج عنصر جدید X در درخت ، ابتدا باید مشخص کنیم که آیا با کلید عناصر موجود در درخت 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 اضافه می کنیم.

اضافه کردن یک عضو در درخت جستجوی دودویی

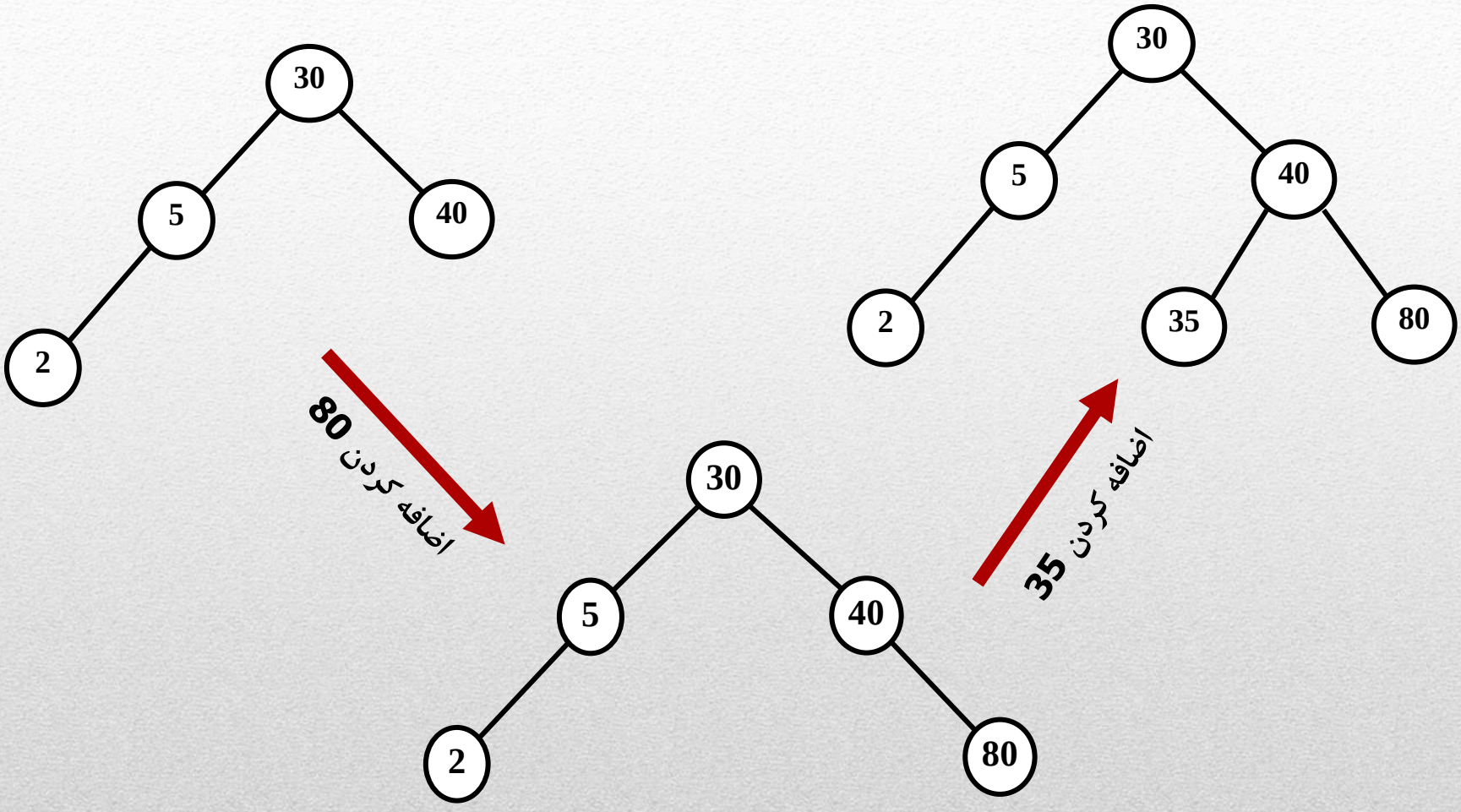
E-mail: Hadi.khademi@gmail.com

```
int Insert_node(tree-pointer  tree, int x)
//Insert x into the binary search tree
{
// search for x, q is the parent of p
tree –pointer  p = tree, q=0;
while ( p) {
    q=p;
    if (x==p-> data) return false; //x is already in tree
    if (x <p->data) p=p->leftchild;
    else p=p->rightchild;
}
//perform insertion
p= (tree-pointer) malloc (sizeof (node));
p->leftchild=p->rightchild=0; p->data=x;
if  (!tree) tree=p;
else if (x< q-> data) q->leftchild =p;
else q-> rightchild=p;
return true;
}
```

اگر h ارتفاع یا عمق یک درخت جستجوی دودویی باشد، عمل اضافه کردن در مدت O(h) انجام می شود.

اضافه کردن یک عضو در درخت جستجوی دودویی

E-mail: Hadi.khademi@gmail.com



اضافه کردن یک عضو در درخت جستجوی دودویی

E-mail: Hadi.khademi@gmail.com

- **۳ حالت باید در نظر گرفته شود:**

حالت ۱: حذف عضو برگ

حالت ۲: حذف عضو غیر برگ که فقط یک بچه دارد

حالت ۳: حذف عنصر غیر برگ که دو بچه دارد. بزرگترین عضو در زیر درخت چپ یا کوچکترین عضو در زیر درخت راست (نزدیکترین عددها به عدد مذکور) جایگزین آن می شوند. حال خود عنصر جایگزین شونده باید از محل قبلی خود حذف شود. عنصر جایگزین شونده همواره درجه حداکثر یک دارد (حالت ۱ یا ۲) پس حذف آن ساده است.

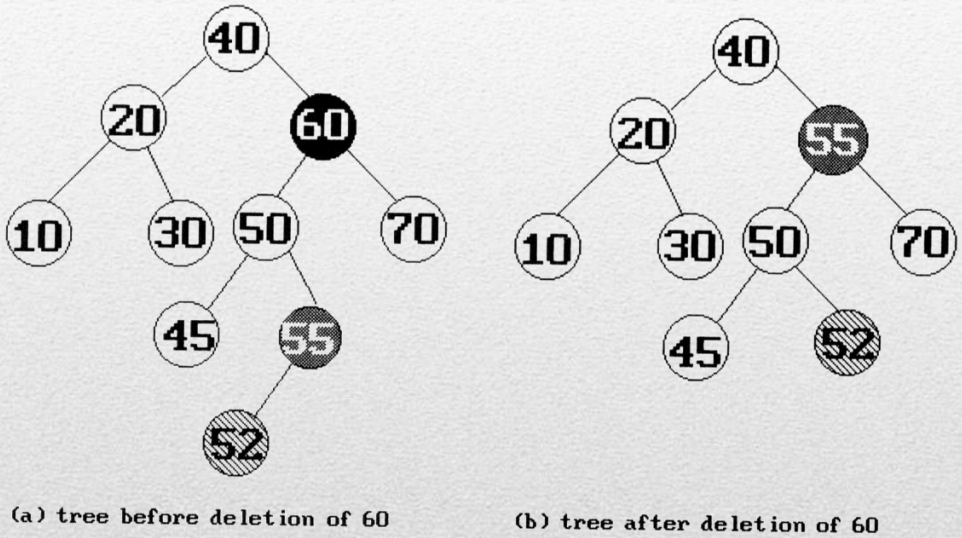
حذف یک عضو از درخت جستجوی دودویی

E-mail: Hadi.khademi@gmail.com

- **حالت ۳:**

عنصر جایگزین شونده عنصر قبلی یا بعدی در پیمایش میانوندی است.

عنصر قبلی یا بعدی در پیمایش میانوندی نزدیکترین عنصرها به عنصر حذف شونده هستند.



اگر h ارتفاع یا عمق یک درخت جستجوی دودویی باشد، عمل حذف کردن در مدت O(h) انجام می شود.

حذف یک عضو از درخت جستجوی دودویی

E-mail: Hadi.khademi@gmail.com

- **ارتفاع / عمق درخت جستجوی دودویی**

ارتفاع یک درخت جستجوی دودویی n عضوی می تواند به بزرگی n باشد.

وقتی که عمیات اضافه و حذف کردن به صورت تصادفی انجام شود ارتفاع درخت جستجوی دودویی به طور متوسط $O(\log_2 n)$ است.

درختهای جستجویی که در بدترین حالت عمق $O(\log_2 n)$ دارند درختهای جستجوی متوازن balanced search trees نامیده می شوند.

درختان جستجوی متوازنی وجود دارند که عمل جستجو، درج و حذف در آنها در زمان $O(h)$ انجام می شود از جمله درختان red_black، 2-3، AVL

درخت های جستجوی دودویی

E-mail: Hadi.khademi@gmail.com

- **فرض کنید دارای k دنباله مرتب شده از عناصر هستیم که باید در یک دنباله واحد ادغام شوند. هر دنباله شامل تعدادی رکورد به ترتیب غیرنزولی از یک فیلد مشخص به نام key می باشد.**
- **کوچکترین کلید باید از بین k کلید موجود پیدا شود:**
- **راهکار**

ساده ترین روش نیازمند k-1 مقایسه برای تعیین و انتقال رکورد بعدی به خروجی می باشد.

به ازای k>2 ، می توانیم با استفاده از ایده درخت انتخاب، تعداد مقایسه های لازم جهت تعیین کوچکترین عنصر را کاهش دهیم.

درخت های انتخاب

E-mail: Hadi.khademi@gmail.com

- **درخت انتخاب**

  درخت برنده

  درخت بازنده

- **درخت برنده**

یک درخت برنده، یک درخت دودویی است که در آن هر گره کوچکتر از دو فرزند خود می باشد

گره ریشه نشان دهنده کوچکترین گره در درخت می باشد.

مسابقه بین همزادها

برنده در مسابقه رکورد با کلید کوچکتر است

ساده ترین روش نیازمند k-1 مقایسه برای تعیین و انتقال رکورد بعدی به خروجی می باشد.
به ازای k>2 ، می توانیم با استفاده از ایده درخت انتخاب، تعداد مقایسه های لازم جهت تعیین کوچکترین عنصر را کاهش دهیم.

درخت های انتخاب

E-mail: Hadi.khademi@gmail.com

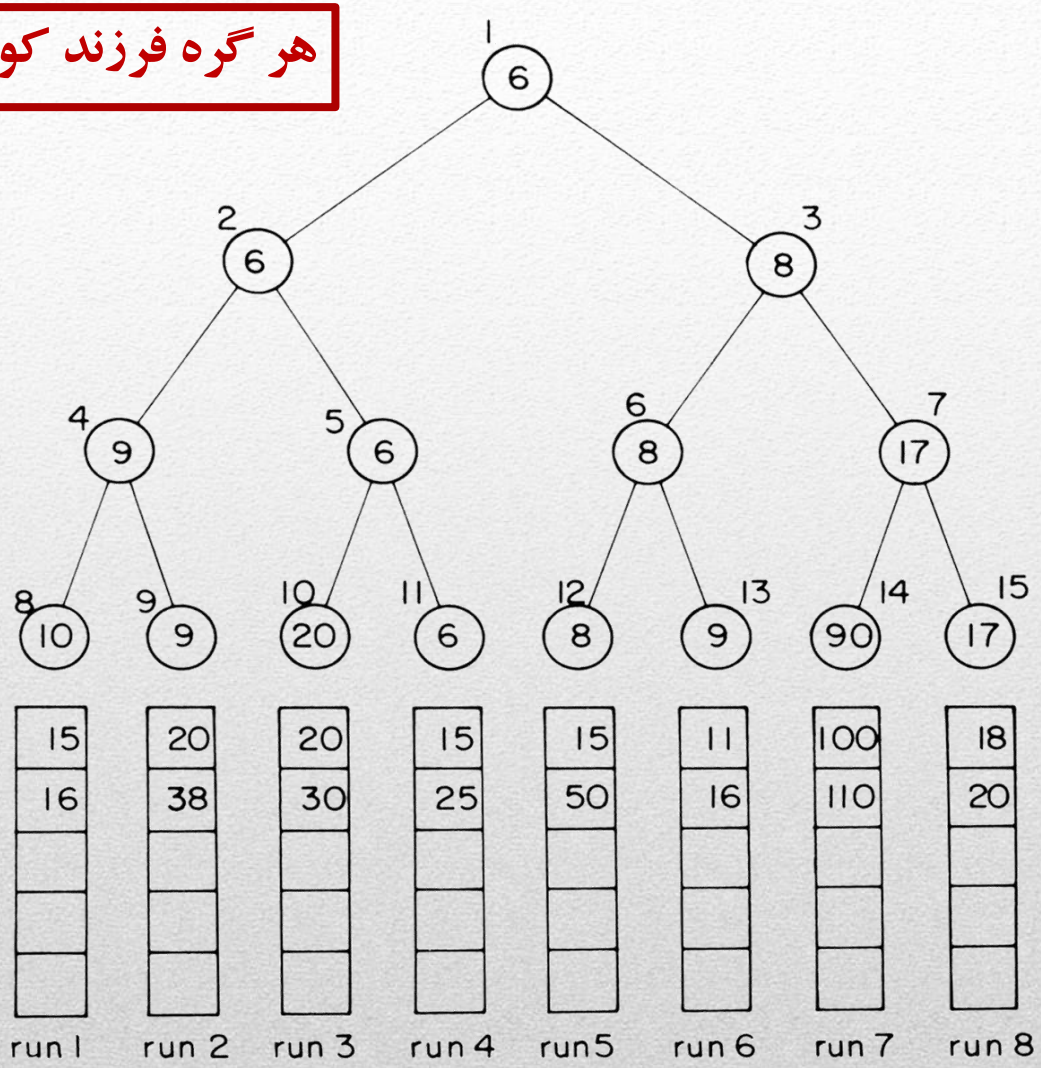
هر گره فرزند کوچکتر خود را برمی گرداند

بازنمایی ترتیبی

(درخت دودویی کامل)

درخت برنده

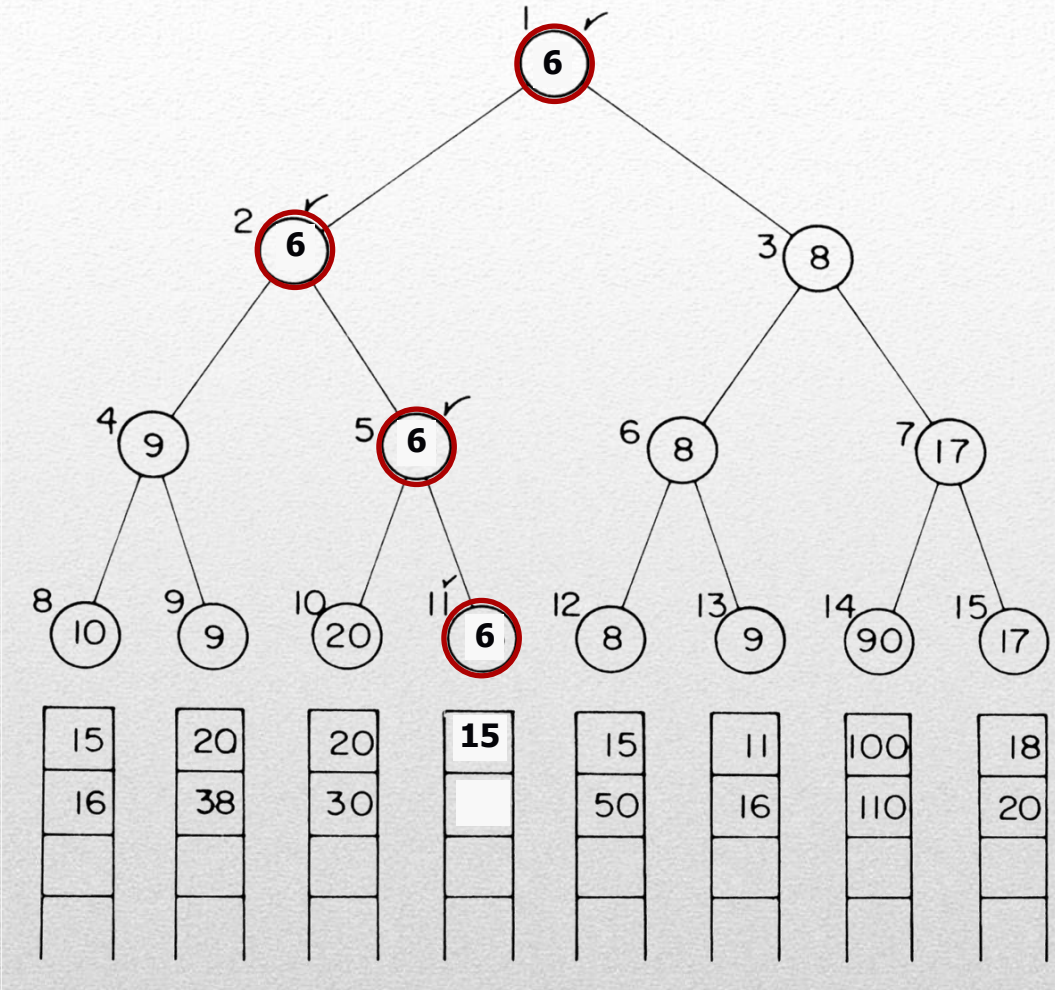
دنباله مرتب



درخت های انتخاب

E-mail: Hadi.khademi@gmail.com

درخت برنده پس از آنکه یک رکورد در خروجی چاپ و درخت از نو ساخته شد.



درخت های انتخاب

E-mail: Hadi.khademi@gmail.com

- **تجزیه و تحلیل ادغام اجراها با استفاده از درخت برنده**

تعداد سطوح	$\lceil \log_2 K + 1 \rceil$
زمان ساختن دوباره درخت	$O(\log_2 K)$
زمان ادغام تمام n رکورد	$O(n\log_2 K)$
زمان لازم برای ساختن درخت انتخاب در بار اول	$O(K)$
زمان کل مورد نیاز	$O(n\log_2 K)$

درخت های انتخاب

E-mail: Hadi.khademi@gmail.com

- **درخت بازنده ها**

یک درخت انتخاب که هر گره غیر برگ آن دارای یک اشاره گر به بازنده است درخت بازنده ها نامیده می شود.

به جای درخت برنده ها می توان درخت بازنده ها را ایجاد کرد. درخت بازنده ها را می توان از روی درخت برنده ها ایجاد کرد.

دارای یک گره اضافی یعنی گره 0 است تا برنده نهایی مسابقه را نشان دهد

- بعد از چاپ برنده نهایی در خروجی برای تجدید ساختار رکوردهایی که باید در مسابقه با آنها بازی شود:

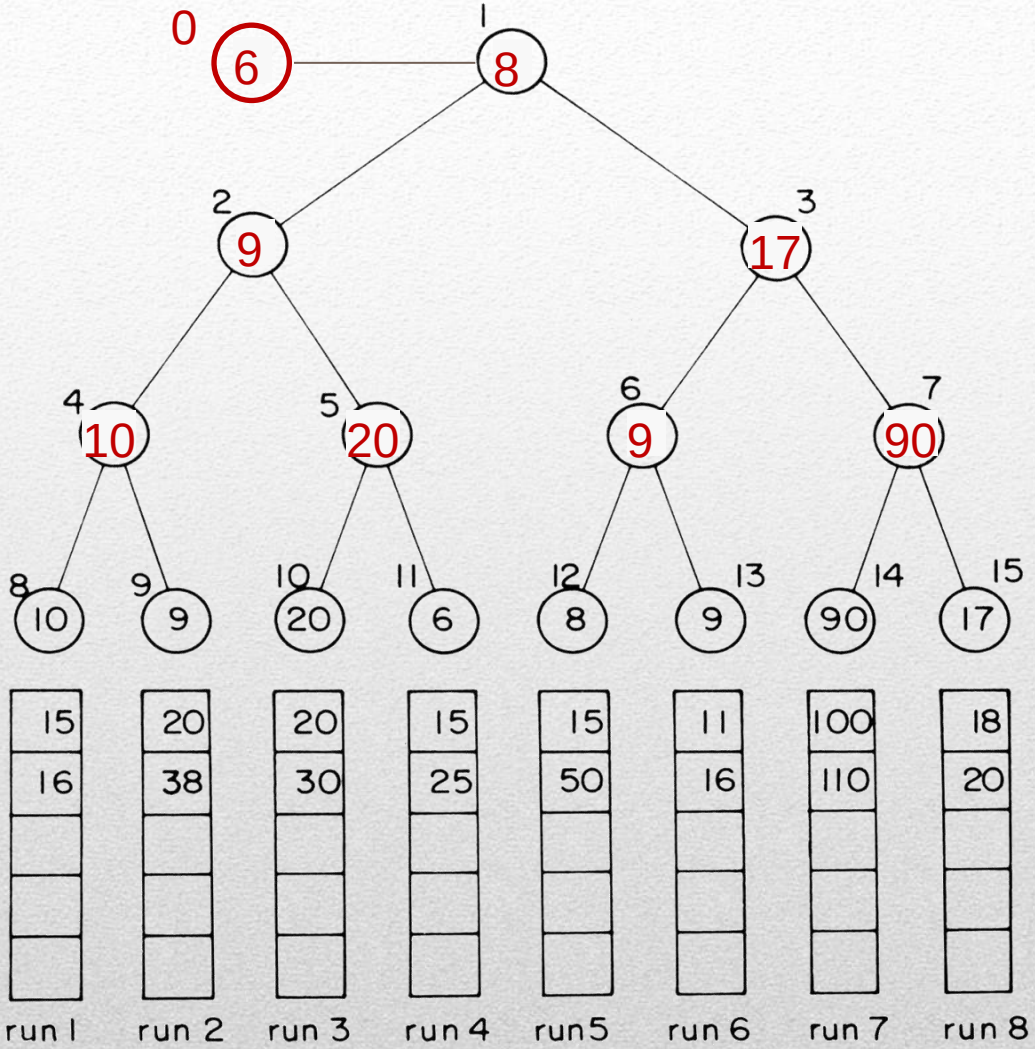
در درخت برنده ها: گره های همزاد در طول مسیر تا ریشه

در درخت بازنده ها: گره های پدر در طول مسیر تا ریشه

درخت های انتخاب

E-mail: Hadi.khademi@gmail.com

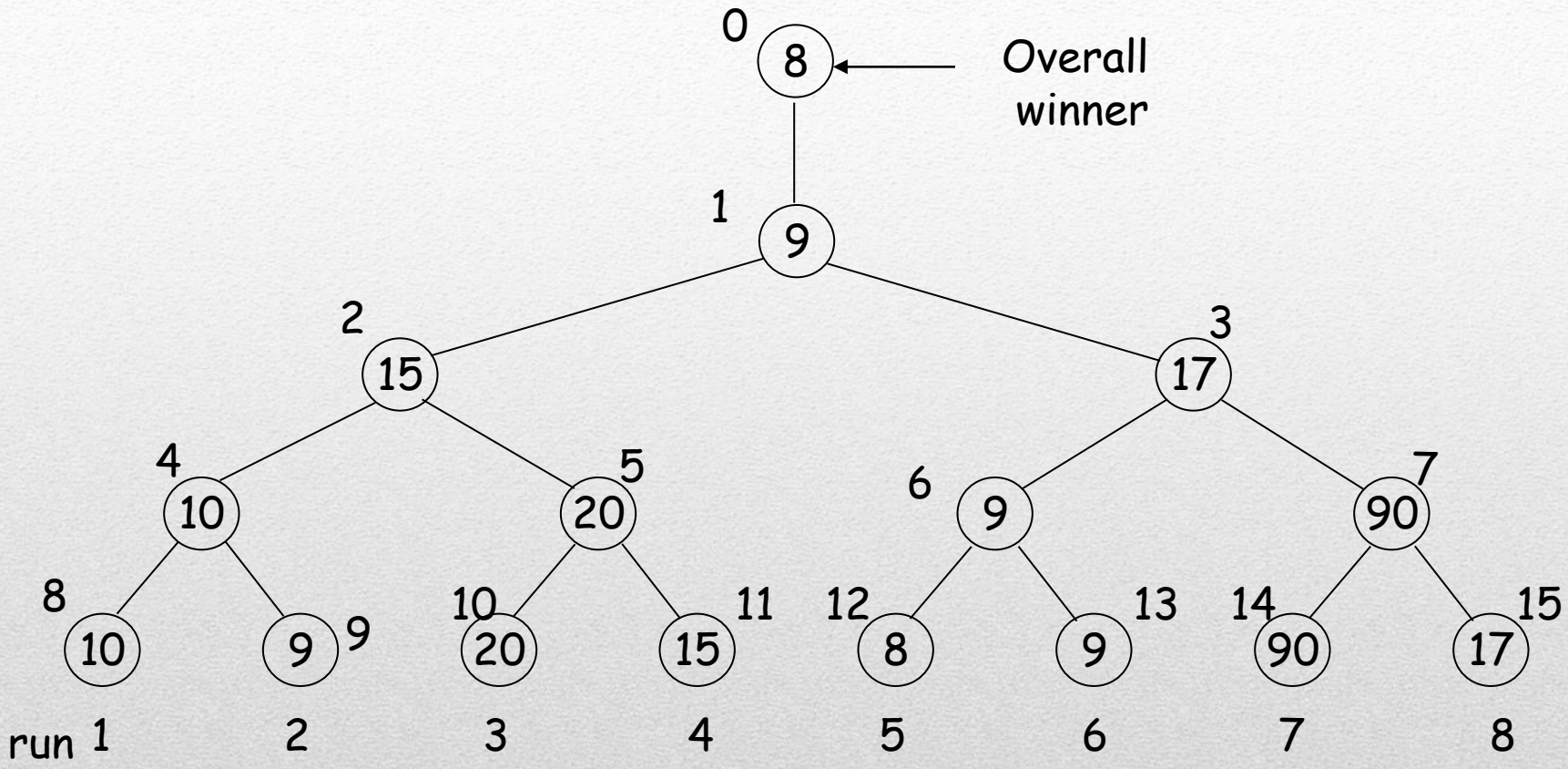
درخت بازنده ها



درخت های انتخاب

E-mail: Hadi.khademi@gmail.com

- درخت بازنده بعد از چاپ ٦

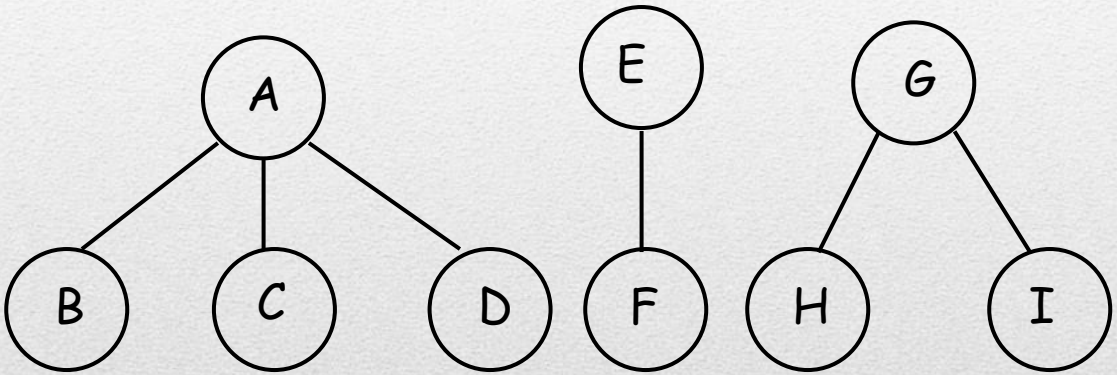


درخت های انتخاب

E-mail: Hadi.khademi@gmail.com

- **تعریف: جنگل مجموعه $n \geq 0$ درخت مجزا می باشد**

اگر ریشه یک درخت را حذف کنیم آنگاه یک جنگل به دست می آید. به عنوان مثال با حذف ریشه از هر درخت دودویی آنگاه یک جنگل با دو درخت به دست می آید



جنگلی با ۳ درخت

جنگل ها

E-mail: Hadi.khademi@gmail.com

- **تبدیل جنگل به درخت دودویی**

جنگلی از درخت ها $T_1, \ldots, T_n$

درخت دودویی متناظر $B(T_1, \ldots, T_n)$

- **الگوریتم: درخت دودویی متناظر با این جنگل برابر است با**

۱- تهی است اگر $n = 0$

۲- ریشه آن برابر ریشه T_1 و زیر درخت چپ آن برابر با $B(T_{11}, \ldots, T_{1m})$ است که $T_{11}, \ldots, T_{1m}$ زیر درختهای ریشه T_1 هستندو زیر درخت راست آن $B(T_2, \ldots, T_n)$ است

جنگل ها

E-mail: Hadi.khademi@gmail.com

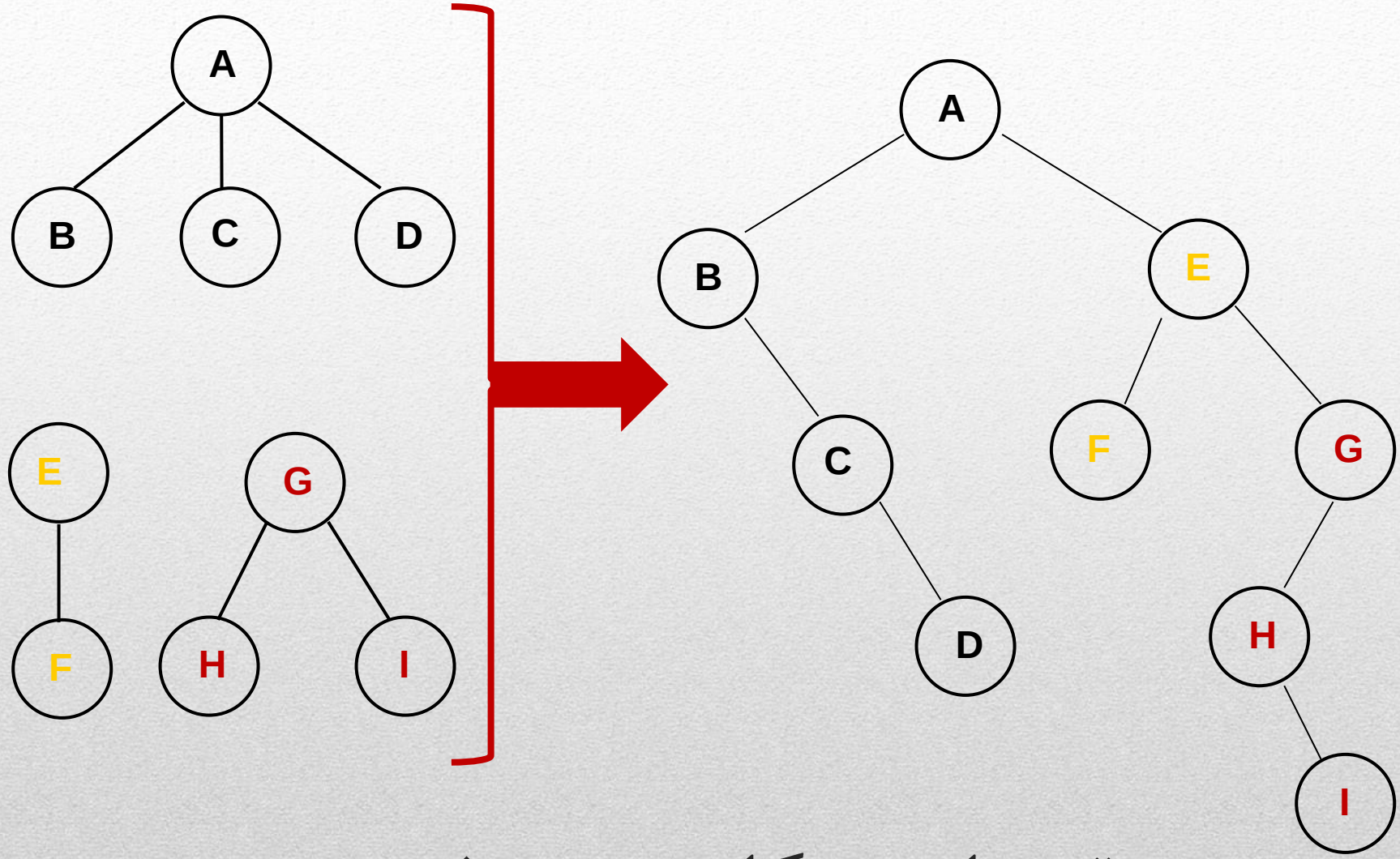
- **برای تبدیل یک جنگل به درخت دودویی**

ابتدا هر درخت جنگل را به درخت دودویی تبدیل می کنیم

درخت های دودویی را از طریق اشاره گر راست گره های ریشه به هم وصل می کنیم

تبدیل جنگل به درخت دودویی

E-mail: Hadi.khademi@gmail.com



تبدیل جنگل به درخت دودویی

E-mail: Hadi.khademi@gmail.com

- **پیمایش پیش ترتیب (preorder) جنگل**
  - اگر F تهی باشد ، آنگاه برگردید.
  - ریشه درخت اول F را ملاقات کنید.
  - زیردرختهای اولین درخت را به صورت پیش ترتیب درخت پیمایش کنید.
  - سایر درختان F را به صورت پیش ترتیب درخت پیمایش کنید.

- پیمایش پیش ترتیب مربوط به درخت دودویی معادل (T) هم ارز با ملاقات گره های F در پیمایش پیش ترتیب جنگل می باشد.

پیمایش جنگل ها

E-mail: Hadi.khademi@gmail.com

- **پیمایش میان ترتیب (inorder) جنگل**
  - اگر F تهی باشد، آنگاه برگردید.
  - زیردرختهای اولین درخت را به صورت میان ترتیب درخت پیمایش کنید.
  - ریشه درخت اول F را ملاقات کنید.
  - سایر درختان F را به صورت میان ترتیب درخت پیمایش کنید.

- پیمایش میان ترتیب مربوط به درخت دودویی معادل (T) هم ارز با ملاقات گره های F در پیمایش میان ترتیب جنگل می باشد.

پیمایش جنگل ها

E-mail: Hadi.khademi@gmail.com

- **پیمایش پس ترتیب (postorder) جنگل**
  - اگر F تهی باشد، آنگاه برگردید.
  - زیردرختهای اولین درخت را به صورت پس ترتیب درخت پیمایش کنید.
  - سایر درختان F را به صورت پس ترتیب درخت پیمایش کنید.
  - ریشه درخت اول F را ملاقات کنید.

- پیمایش پس ترتیب مربوط به درخت دودویی معادل (T) لزوما با ملاقات گره های F در پیمایش پس ترتیب جنگل نتیجه یکسانی تولید نمی کند.

پیمایش جنگل ها

E-mail: Hadi.khademi@gmail.com

- **پیمایش به ترتیب سطح جنگل**
  - گره ها به ترتیب سطح ملاقات می شوند.
  - ملاقات از ریشه ی هر درخت جنگل شروع می شود.
  - در هر سطح گره ها از چپ به راست ملاقات می شوند.
  - پیمایش به ترتیب سطح یک جنگل و درخت دودویی معادل آن لزوما نتیجه یکسانی تولید نمی کند.

پیمایش جنگل ها

E-mail: Hadi.khademi@gmail.com