دانشگاه هرمزگان
دانشکده فنی و مهندسی
گروه مهندسی برق و کامپیوتر

# داده‌ساختارها و الگوریتم‌ها

## جلسه نوزدهم: درخت دودویی و درخت دودویی جست‌وجو

دکتر شهرام گلزاری

پاییز ۱۴۰۵

# اهداف جلسه

در پایان این جلسه انتظار داریم که:

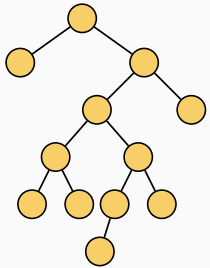
- درخت دودویی را تعریف کنیم و پیاده‌سازی آن را با اشاره‌گرهای *left* و *right* و *parent* توضیح دهیم.
- تعداد درخت‌های دودویی متفاوت با $n$ گره را با رابطه بازگشتی و عدد کاتالان به دست آوریم.
- کمترین ارتفاع درخت دودویی جست‌وجو را محاسبه کنیم.
- ویژگی اصلی درخت جستجوی دودویی را بیان کنیم .
- رویه‌های جستجو (Tree-Search)، کمینه (Tree-Minimum)، بیشینه (Tree-Maximum)، جانشین (Tree-Successor) و پیشگام را روی یک درخت جستجوی دودویی ردیابی کنیم.
- نحوه درج یک گره جدید در درخت جستجوی دودویی (Tree-Insert) را دنبال کنیم.

# درخت دودویی

# درخت دودویی

«درخت دودویی حالت خاصی از درخت‌های مرتب و کلی است که کاربردهای فراوان دارد. هر گره در درخت دودویی دو فرزند چپ و راست دارد که ممکن است یک یا هر دو فرزندش تهی باشند. درخت عبارت (بخش ۴-۴-۷) و درخت تصمیم (بخش ۶-۱) نمونه‌هایی از کاربردهای مهم درخت دودویی هستند. بسیاری از درخت‌ها که گره‌های آن‌ها حداکثر ۲ فرزند دارند هم دودویی فرض می‌شوند. شکل ۴-۳۰ یک درخت دودویی را نشان می‌دهد.»

داده‌ساختارها و مبانی الگوریتم‌ها، محمد قدسی، صفحه ۱۹۲



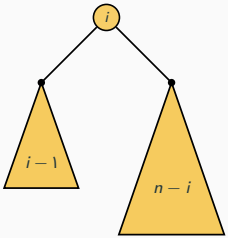
**شکل ۴-۳۰** یک درخت دودویی.

# تعداد درخت‌های دودویی

# تعداد درخت‌های دودویی

«می‌خواهیم تعداد درخت‌های دودویی متفاوت را که با $n$ گره می‌توان ساخت به‌دست آوریم. فرض کنید که گره‌های هر کدام از این درخت‌ها را به‌روش میان‌ترتیب از ۱ تا $n$ شماره‌گذاری کرده‌ایم. با این شماره‌گذاری، فرض کنید که گره شماره‌ی $i$ ($1 \leq i \leq n$) ریشه‌ی درخت باشد. در آن صورت زیردرخت چپ ریشه حتماً باید $i - 1$ گره و زیردرخت راست آن حتماً باید $n - i$ گره داشته باشد (شکل ۴-۳۱ را ببینید). حال اگر $T(n)$ تعداد درخت‌های دودویی با $n$ گره باشد، تعداد زیردرخت‌های چپ و راست ریشه به‌ترتیب برابر $T(i-1)$ و $T(n-i)$ خواهند بود. در آن‌صورت اگر گره‌ی $i$ ریشه باشد، حاصل‌ضرب $T(i-1)\,T(n-i)$ برابر تعداد کل درخت‌هاست. برای در نظر گرفتن همه‌ی حالت‌های ریشه، باید این مقدار را برای همه‌ی مقادیر $i$ جمع بزنیم که رابطه‌ی بازگشتی ۴-۴ به‌دست می‌آید.»

داده‌ساختارها و مبانی الگوریتم‌ها، محمد قدسی، صفحه ۱۹۲-۱۹۳




**شکل ۴-۳۱** شمارش تعداد درخت‌های دودویی با $n$ گره.

# رابطه بازگشتی تعداد درخت‌های دودویی

$$T(n) = \begin{cases} T(۰) = T(۱) = ۱ \\ T(n) = \sum_{i=۱}^{n} T(i-۱)\,T(n-i), \quad i > ۱ \end{cases}$$

«جواب این رابطه‌ی بازگشتی

$$T(n) = \frac{۱}{n+۱}\binom{۲n}{n}$$

است که همان «**عدد $n$ام کاتالان**» است.»

داده‌ساختارها و مبانی الگوریتم‌ها، محمد قدسی، صفحه ۱۹۴

# پیاده‌سازی و پیمایش

# پیاده‌سازی درخت‌های دودویی

«در پیاده‌سازی درخت دودویی از اشاره‌گر استفاده می‌کنیم. هر گره دو مؤلفه‌ی *left* و *right* دارد که به‌ترتیب به فرزندهای چپ و راست آن گره اشاره می‌کنند، و ممکن است برخی از آن‌ها تهی باشند. مانند قبل، معمولاً هر گره مؤلفه‌ی *parent* هم دارد که به پدر آن گره اشاره می‌کند. بدیهی است که پدر ریشه تهی است. با این پیاده‌سازی پیمایش پیش‌ترتیب یک درخت دودویی $T$ به ریشه‌ی $r$ به‌صورت زیر خواهد بود:»

داده‌ساختارها و مبانی الگوریتم‌ها، محمد قدسی، صفحه ۱۹۴

```
PREORDER(T, r)
1  if r == NIL
2     then return
3  MEET element(T, r)
4  PREORDER(T, left[r])
5  PREORDER(T, right[r])
```

پیمایش پیش‌ترتیب درخت $T$ به ریشه‌ی $r$.

# درخت جستجوی دودویی

# درخت جستجوی دودویی چیست؟

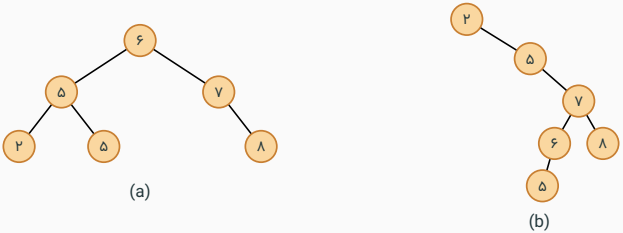
«یک درخت جستجوی دودویی، همان‌طور که از نامش پیداست، در یک درخت دودویی سازمان‌دهی می‌شود، همان‌طور که در شکل ۱.۱۲ نشان داده شده است. شما می‌توانید چنین درختی را با یک داده‌ساختار پیوندی، مانند بخش ۳.۱۰، نشان دهید. علاوه بر یک کلید (key) و داده‌های جانبی (satellite data)، هر شیء گره شامل ویژگی‌های *left*، *right* و *p* است که به‌ترتیب به گره‌های متناظر با فرزند چپ، فرزند راست و پدر آن اشاره می‌کنند. اگر یک فرزند یا پدر وجود نداشته باشد، ویژگی مربوطه حاوی مقدار nil خواهد بود. خود درخت دارای یک ویژگی *root* است که به گره ریشه اشاره می‌کند، یا اگر درخت تهی باشد مقدار آن nil است. گره ریشه $T.root$ تنها گره در درخت $T$ است که پدر آن nil است. کلیدها در یک درخت جستجوی دودویی همواره به‌گونه‌ای ذخیره می‌شوند که **ویژگی درخت جستجوی دودویی** را برآورده کنند:»

*CLRS, 4th ed.*، صفحه ۳۱۳ (ترجمه ماشینی)

# ویژگی درخت جستجوی دودویی

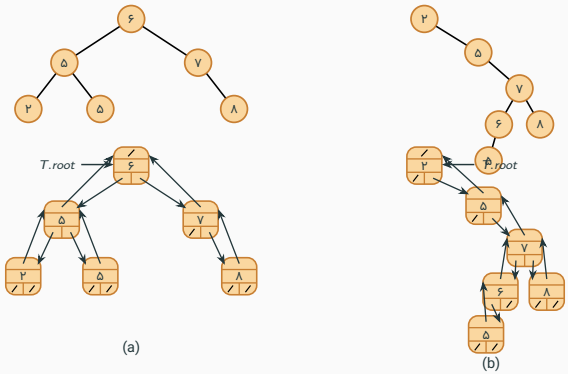
«برای هر گره $x$، کلیدهای زیردرخت چپ $x$ حداکثر $x.key$ هستند و کلیدهای زیردرخت راست $x$ حداقل $x.key$ هستند. درخت‌های جستجوی دودوییِ متفاوت می‌توانند مجموعه مقادیر یکسانی را نشان دهند. زمان اجرای بدترین حالت برای بیشتر عملیاتِ درختِ جستجو متناسب با ارتفاع درخت است.»

*CLRS, 4th ed.*، صفحه ۳۱۳ (ترجمه ماشینی)




**شکل ۱.۱۲ (نمای مفهومی):** **(a)** درخت جستجوی دودویی ۶ گره‌ای با ارتفاع ۲. **(b)** درخت کم‌بازده‌تر با ارتفاع ۴ شامل همان کلیدها.

# شکل ۱.۱۲: نمایش مفهومی و پیوندی درخت جستجوی دودویی




**شکل ۱.۱۲:** نمایش مفهومی (بالا) و نمایش پیوندی شامل ویژگی‌های *left*، *right* و *p* در هر گره (پایین).

*CLRS, 4th ed.*، صفحه ۳۱۳ (ترجمه ماشینی)

# ارتفاع و تعداد درخت‌های جست‌وجو

# ارتفاع درخت دودویی جست‌وجو

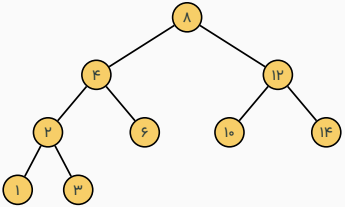
«تعداد د.د.ج‌های متفاوت با $n$ عنصر و با ارتفاع $n-1$ برابر $2^{n-1}$ است. دلیل این نکته آن است که هر یک از $n-1$ عنصر غیر برگ تنها یک فرزند دارد و هر کدام فقط می‌تواند یا فرزند چپ باشد یا راست. طبق اصل ضرب، تعداد کل حالات برابر $2^{n-1}$ می‌شود.»

داده‌ساختارها و مبانی الگوریتم‌ها، محمد قدسی، صفحه ۲۱۶

# کمترین ارتفاع درخت دودویی جست‌وجو

«کم‌ترین ارتفاع برای یک د.د.ج با $n$ عنصر برابر $\lfloor \lg n \rfloor$ است که یک نمونه‌ی آن برای ۸ عنصر مورد نظر در شکل ۴-۳۷ دیده می‌شود. اثبات این فرمول با استقرا و یا با روش‌های شمارشی دیگر ساده است.»

داده‌ساختارها و مبانی الگوریتم‌ها، محمد قدسی، صفحه ۲۱۶




**شکل ۴-۳۷** نمونه‌ای از درخت با کمترین ارتفاع برای ۸ عنصر.

# تعداد درخت‌های جست‌وجوی دودویی

«تعداد د.د.ج‌هایی که با $a_1 < a_2 < \cdots < a_n$ می‌توان ساخت برابر است با

$$\frac{1}{n+1}\binom{2n}{n}$$

هر درخت دودویی با $n$ عنصر را می‌توان به‌صورت میان‌ترتیب پیمایش کرد و عناصر $a_1$ تا $a_n$ را به تک‌تک گره‌های آن تخصیص داد و یک د.د.ج ساخت. بنابراین مطابق رابطه‌ی بازگشتی بیان‌شده در بخش ۴-۴-۶، تعداد د.د.ج‌هایی که ساخته می‌شوند همان عدد $n$ام کاتالان است.»

داده‌ساختارها و مبانی الگوریتم‌ها، محمد قدسی، صفحه ۲۱۷

# پرس‌وجو در درخت جستجوی دودویی

# جستجو در درخت جستجوی دودویی

«برای جستجوی یک گره با کلیدی مشخص در یک درخت جستجوی دودویی، رویه TREE-SEARCH را فراخوانی می‌کنیم. با داشتن اشاره‌گر $x$ به ریشه‌ی یک زیردرخت و یک کلید $k$، TREE-SEARCH$(x, k)$ اشاره‌گری به یک گره با کلید $k$ برمی‌گرداند در صورتی که چنین گرهی در زیردرخت وجود داشته باشد؛ در غیر این صورت nil برمی‌گرداند. برای جستجوی کلید $k$ در کل درخت جستجوی دودویی $T$، رویه TREE-SEARCH$(T.root, k)$ را فراخوانی می‌کنیم.»

*CLRS, 4th ed.*، صفحه ۳۱۶ (ترجمه ماشینی)

TREE-SEARCH$(x, k)$

```
1 if x == NIL or k == x.key
2     return x
3 if k < x.key
4     return TREE-SEARCH(x.left, k)
5 else return TREE-SEARCH(x.right, k)
```

# جستجوی ترتیبی

«از آنجا که رویه TREE-SEARCH در زیردرخت چپ یا زیردرخت راست به‌صورت بازگشتی فراخوانی می‌شود، اما نه در هر دو، می‌توانیم الگوریتم را بازنویسی کنیم تا بازگشت را در یک حلقه **while** «باز» (unroll) کنیم. در بیشتر رایانه‌ها، رویه ITERATIVE-TREE-SEARCH در صفحه مقابل کارآمدتر است.»

*CLRS, 4th ed.*، صفحه ۳۱۷ (ترجمه ماشینی)

ITERATIVE-TREE-SEARCH$(x, k)$

```
1  while x ≠ NIL and k ≠ x.key
2      if k < x.key
3          x = x.left
4      else x = x.right
5  return x
```

# کمینه و بیشینه

«برای یافتن عنصری در یک درخت جستجوی دودویی که کلید آن کمینه است، کافیست اشاره‌گرهای فرزند چپ را از ریشه دنبال کنید تا زمانی که به یک nil برسید، همان‌طور که در شکل ۲.۱۲(b) نشان داده شده است. رویه TREE-MINIMUM اشاره‌گری به کمترین عنصر در زیردرخت به ریشه‌ی یک گره داده شده $x$ برمی‌گرداند، که فرض می‌کنیم غیر nil است.»

*CLRS, 4th ed.*، صفحه ۳۱۷ (ترجمه ماشینی)

```
TREE-MINIMUM(x)
1 while x.left ≠ NIL
2     x = x.left
3 return x
```

```
TREE-MAXIMUM(x)
1 while x.right ≠ NIL
2     x = x.right
3 return x
```

«ویژگی درخت جستجوی دودویی تضمین می‌کند که TREE-MINIMUM صحیح است. ... هر دو TREE-MINIMUM و TREE-MAXIMUM در زمان $O(h)$ روی درختی به ارتفاع $h$ اجرا می‌شوند زیرا، مانند TREE-SEARCH، دنباله گره‌هایی که با آن‌ها برخورد می‌شود یک مسیر ساده رو به پایین از ریشه تشکیل می‌دهد.»

*CLRS, 4th ed.*، صفحه ۳۱۷-۳۱۸ (ترجمه ماشینی)

# جانشین و پیشگام

«با داشتن یک گره در درخت جستجوی دودویی، چگونه می‌توان جانشین آن را در ترتیب مرتب‌سازی‌شده‌ی تعیین‌شده توسط پیمایش میان‌ترتیب درخت پیدا کرد؟ اگر همه کلیدها متمایز باشند، جانشین گره $x$ گرهی با کوچکترین کلیدِ بزرگتر از $x.key$ است. صرف‌نظر از اینکه کلیدها متمایز باشند یا نه، ما جانشین (successor) یک گره را به‌عنوان گره بعدی که در پیمایش میان‌ترتیب درخت بازدید می‌شود، تعریف می‌کنیم. ساختار یک درخت جستجوی دودویی به ما اجازه می‌دهد جانشین یک گره را بدون مقایسه کلیدها تعیین کنیم. رویه Tree-Successor در صورت وجود، جانشین گره $x$ در درخت جستجوی دودویی را برمی‌گرداند، یا اگر $x$ آخرین گرهی باشد که در پیمایش میان‌ترتیب بازدید می‌شود، nil برمی‌گرداند.»

*CLRS, 4th ed.*، صفحه ۳۱۸ (ترجمه ماشینی)

«زمان اجرای Tree-Successor روی درختی با ارتفاع $h$ برابر $O(h)$ است، زیرا الگوریتم یا یک مسیر ساده را به سمت بالای درخت دنبال می‌کند یا یک مسیر ساده را به سمت پایین درخت. رویه Tree-Predecessor که متقارن با Tree-Successor است، نیز در زمان $O(h)$ اجرا می‌شود.»

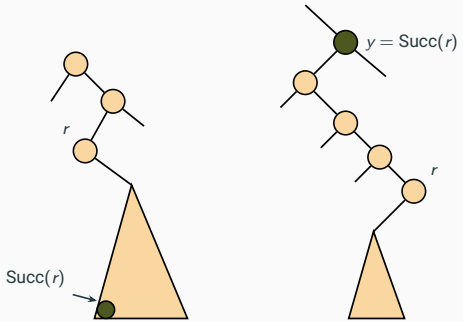
*CLRS, 4th ed.*، صفحه ۳۱۹ (ترجمه ماشینی)

# رویه جانشین در درخت جستجوی دودویی

```
TREE-SUCCESSOR(x)
1  if x.right ≠ NIL
2     return TREE-MINIMUM(x.right)    // leftmost node in right subtree
3  else  // find the lowest ancestor of x whose left child is an ancestor of x
4     y = x.p
5     while y ≠ NIL and x == y.right
6        x = y
7        y = y.p
8     return y
```

*CLRS, 4th ed.*، صفحه ۳۱۹ (ترجمه ماشینی)

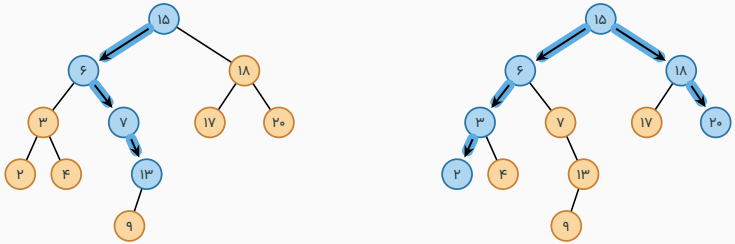
# یافتن عنصر بعدی (جانشین) در د.د.ج




**شکل ۴-۳۸** یافتن عنصر بعدی $r$ در د.د.ج. راه‌حل برای دو حالتی که $r$ فرزند راست داشته یا نداشته باشد، مشخص شده است.

داده‌ساختارها و مبانی الگوریتم‌ها، محمد قدسی، صفحه ۲۲۰

# شکل ۲.۱۲: پرس‌وجوها روی یک درخت جستجوی دودویی (بخش اول)

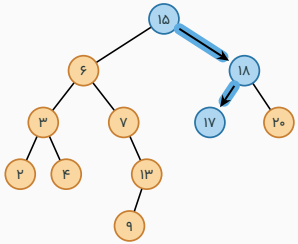



**(a)** جستجو برای کلید ۱۳ در درخت، مسیر ۱۵ → ۶ → ۷ → ۱۳ را از ریشه دنبال می‌کند.

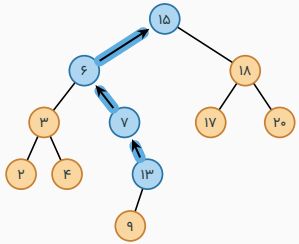
**(b)** کمترین کلید در درخت ۲ است که با دنبال کردن اشاره‌گرهای *left* از ریشه یافت می‌شود. بیشترین کلید ۲۰ است که با دنبال کردن اشاره‌گرهای *right* از ریشه پیدا می‌شود.

*CLRS, 4th ed.*، صفحه ۳۱۷ (ترجمه ماشینی)

# شکل ۲.۱۲: پرس‌وجوها روی یک درخت جستجوی دودویی (بخش دوم)




(c)




(d)

**(c)** جانشین گره با کلید ۱۵، گره با کلید ۱۷ است، زیرا کمترین کلید در زیردرخت راست ۱۵ است.

**(d)** گره با کلید ۱۳ هیچ زیردرخت راستی ندارد، و بنابراین جانشین آن پایین‌ترین جدی است که فرزند چپ آن نیز یک جد است. در این حالت، گره با کلید ۱۵ جانشین آن است.

*CLRS, 4th ed.*، صفحه ۳۱۷ (ترجمه ماشینی)

# درج در درخت جستجوی دودویی

# درج در درخت جستجوی دودویی

«عملیات درج و حذف باعث می‌شوند مجموعه پویای نمایش‌داده‌شده توسط یک درخت جستجوی دودویی تغییر کند. داده‌ساختار باید به‌گونه‌ای اصلاح شود که این تغییر را منعکس کند، اما به‌طریقی که ویژگی درخت جستجوی دودویی همچنان حفظ شود. خواهیم دید که اصلاح درخت برای درج یک عنصر جدید نسبتاً ساده است، اما حذف یک گره از یک درخت جستجوی دودویی پیچیده‌تر است. رویه TREE-INSERT یک گره جدید را در یک درخت جستجوی دودویی درج می‌کند. این رویه یک درخت جستجوی دودویی $T$ و یک گره $z$ را می‌گیرد که برای آن $z.key$ از قبل پر شده، $z.left = \text{nil}$ و $z.right = \text{nil}$ است. این رویه $T$ و برخی از ویژگی‌های $z$ را اصلاح می‌کند تا $z$ را در موقعیت مناسبی در درخت درج کند.»

*CLRS, 4th ed.*، صفحه ۳۲۱ (ترجمه ماشینی)

# رویه درج در درخت جستجوی دودویی

TREE-INSERT($T, z$)

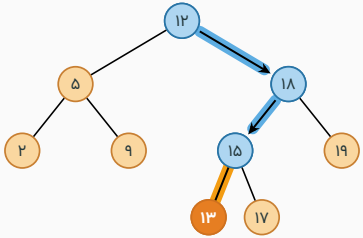
```
1  x = T.root                  // node being compared with z
2  y = NIL                     // y will be parent of z
3  while x ≠ NIL               // descend until reaching a leaf
4     y = x
5     if z.key < x.key
6        x = x.left
7     else x = x.right
8  z.p = y                     // found the location—insert z with parent y
9  if y == NIL
10    T.root = z               // tree T was empty
11 elseif z.key < y.key
12    y.left = z
13    else y.right = z
```

*CLRS, 4th ed.*، صفحه ۳۲۱ (ترجمه ماشینی)

# درج یک گره

«درج یک گره با کلید ۱۳ در یک درخت جستجوی دودویی. مسیر ساده از ریشه به سمت پایین تا موقعیتی که گره در آن درج می‌شود با رنگ آبی نشان داده شده است. گره جدید و پیوند آن به پدرش با رنگ نارنجی برجسته شده‌اند.»

*CLRS, 4th ed.*، صفحه ۳۲۲ (ترجمه ماشینی)



# منابع

# منابع


- محمد قدسی، داده‌ساختارها و مبانی الگوریتم‌ها، انتشارات فاطمی، چاپ ۱۳۹۷.
- T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, 4th ed., MIT Press, 2022.