دانشگاه هرمزگان
دانشکده فنی و مهندسی
گروه مهندسی برق و کامپیوتر

داده‌ساختارها و الگوریتم‌ها

جلسه بیستم: اعمال در درخت دودویی جستجو

دکتر شهرام گلزاری

پاییز ۱۴۰۵

اهداف جلسه

در پایان این جلسه انتظار داریم که:

- 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.
- راهبرد کلی حذف یک گره از درخت جستجوی دودویی و سه حالت آن را بیان کنیم.
- زیرروال TRANSPLANT و رویه TREE-DELETE را روی یک درخت جستجوی دودویی اجرا کنیم.
- پیچیدگی زمانی اعمال درج و حذف را تحلیل کنیم.

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.»

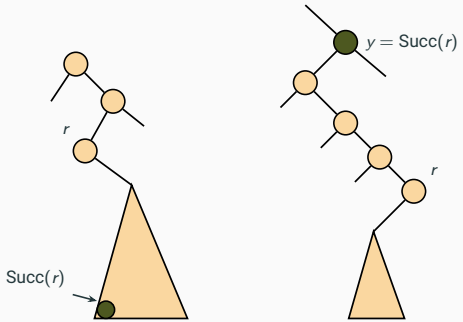
CLRS, 4th ed.، صفحه ۳۱۹ (ترجمه ماشینی)

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)

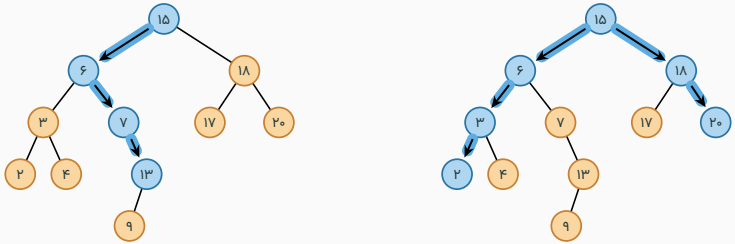
یافتن عنصر بعدی (جانشین) در د.د.ج



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)

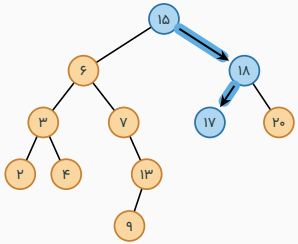


(a) جستجو برای کلید ۱۳ در درخت، مسیر ۱۵ → ۶ → ۷ → ۱۳ را از ریشه دنبال می‌کند.

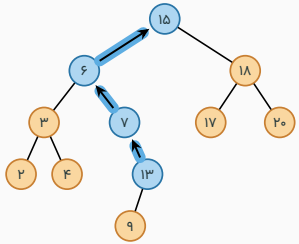
(b) کمترین کلید در درخت ۲ است که با دنبال کردن اشاره‌گرهای *left* از ریشه یافت می‌شود. بیشترین کلید ۲۰ است که با دنبال کردن اشاره‌گرهای *right* از ریشه پیدا می‌شود.

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)

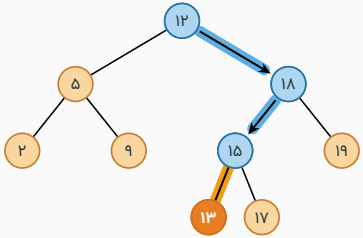
```
1  x = T.root                  // node being compared with z
2  y = NIL                     // y will be parent of z
3  while x ≠ NIL               // descend until reaching a leaf
4     y = x
5     if z.key < x.key
6        x = x.left
7     else x = x.right
8  z.p = y                     // found 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)



حذف در درخت جستجوی دودویی

حذف (Deletion)

«راهبرد کلی برای حذف یک گره z از یک درخت جستجوی دودویی T شامل سه حالت پایه است و همان‌طور که خواهیم دید، یکی از این حالت‌ها کمی ظریف و چالش‌برانگیز است.»

- **حالت اول:** اگر z هیچ فرزندی نداشته باشد، کافی است با تغییر اشاره‌گر والد آن و جایگزین کردن z با nil به عنوان فرزندش، آن را به سادگی حذف کنیم.
- **حالت دوم:** اگر z تنها یک فرزند داشته باشد، با تغییر والد z و جایگزین کردن z با فرزندش، آن فرزند را ارتقا می‌دهیم تا جایگاه z را در درخت بگیرد.

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

حذف (Deletion)

حالت سوم: اگر z دارای دو فرزند باشد:

- جانشین z یعنی y را پیدا می‌کنیم — که الزاماً در زیردرخت راست z قرار دارد — و y را جابه‌جا می‌کنیم تا جایگاه z را در درخت بگیرد.
- باقی‌مانده زیردرخت راستِ اصلیِ z به زیردرخت راستِ جدیدِ y تبدیل می‌شود و زیردرخت چپ z نیز زیردرخت چپِ جدیدِ y خواهد شد.
- از آنجا که y جانشین z است، نمی‌تواند فرزند چپ داشته باشد، و فرزند راست اصلی y به موقعیت قبلی y منتقل می‌شود و بقیه زیردرخت راست اصلی y نیز به طور خودکار به دنبال آن قرار می‌گیرد.

این حالت چالش‌برانگیز است زیرا همان‌طور که خواهیم دید، اینکه آیا y فرزند راستِ مستقیمِ z هست یا خیر، اهمیت دارد.

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

رویه TRANSPLANT - معرفی و شبه‌کد

«به عنوان بخشی از فرآیند حذف یک گره، لازم است زیردرخت‌ها در درون درخت جستجوی دودویی جابه‌جا شوند. زیرروال TRANSPLANT یک زیردرخت را به عنوان فرزندی از والدش با زیردرخت دیگری جایگزین می‌کند. هنگامی که TRANSPLANT زیردرخت به ریشه گره u را با زیردرخت به ریشه گره v جایگزین می‌کند، والد گره u به والد گره v تبدیل می‌شود و در نهایت والد u، گره v را به عنوان فرزند متناظر خود خواهد داشت. زیرروال TRANSPLANT اجازه می‌دهد که v به جای اشاره‌گر به یک گره، برابر nil باشد.»

```
TRANSPLANT(T, u, v)
1  if u.p == NIL
2      T.root = v
3  elseif u == u.p.left
4      u.p.left = v
5  else u.p.right = v
6  if v ≠ NIL
7      v.p = u.p
```

رویه TRANSPLANT - توضیح خط به خط

«نحوه عملکرد TRANSPLANT به این صورت است:»

- **سطرهای ۱ تا ۲:** حالتی را مدیریت می‌کنند که در آن u ریشه درخت T است.
- **سطرهای ۳ تا ۴:** در غیر این صورت، u یا فرزند چپ والد خود است یا فرزند راست آن. این سطرها در صورتی که u فرزند چپ باشد، به‌روزرسانی $u.p.left$ را انجام می‌دهند.
- **سطر ۵:** در صورتی که u فرزند راست باشد، $u.p.right$ را به‌روزرسانی می‌کند.
- **سطرهای ۶ تا ۷:** از آنجا که ممکن است v برابر nil باشد، این سطرها تنها در صورتی $v.p$ را به‌روزرسانی می‌کنند که v غیرِ nil باشد.

نکته: روال TRANSPLANT تلاشی برای به‌روزرسانی $v.left$ و $v.right$ نمی‌کند. انجام دادن یا ندادن این کار بر عهده فراخواننده‌ی TRANSPLANT است.

CLRS, 4th ed.، صفحه ۳۲۴ (ترجمه ماشینی)

رویه TREE-DELETE - معرفی و شبه‌کد

«روال TREE-DELETE از TRANSPLANT برای حذف گره z از درخت جستجوی دودویی T استفاده می‌کند. این روال چهار حالت را به شرح زیر اجرا می‌کند. سطرهای ۱ تا ۲ حالتی را مدیریت می‌کنند که در آن گره z هیچ فرزند چپی ندارد (شکل ۴.۱۲،(a)) و سطرهای ۳ تا ۴ حالتی را مدیریت می‌کنند که در آن z دارای فرزند چپ است اما فرزند راست ندارد (شکل ۴.۱۲«.(b))

TREE-DELETE(T, z)

```
1  if z.left == NIL
2     TRANSPLANT(T, z, z.right)          // replace z by its right child
3  elseif z.right == NIL
4     TRANSPLANT(T, z, z.left)           // replace z by its left child
5  else y = TREE-MINIMUM(z.right)        // y is z's successor
6     if y ≠ z.right                     // is y farther down the tree?
7        TRANSPLANT(T, y, y.right)       // replace y by its right child
8        y.right = z.right               // z's right child becomes
9        y.right.p = y                   //                    y's right child
10    TRANSPLANT(T, z, y)                // replace z by its successor y
11    y.left = z.left                    // and give z's left child to y,
12    y.left.p = y                       //                    which had no left child
```

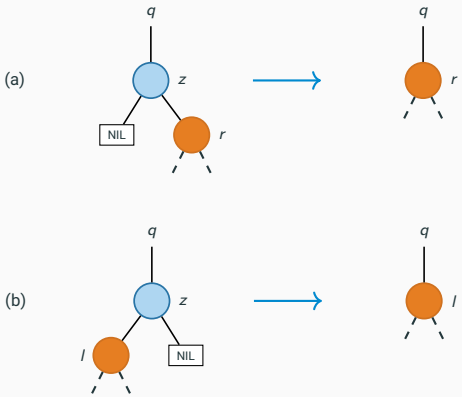
رویه TREE-DELETE

«سطرهای ۵ تا ۱۲ به دو حالت باقی‌مانده می‌پردازند که در آن‌ها z دارای دو فرزند است.»

- **سطر ۵:** گره y را که جانشین z است پیدا می‌کند. از آنجا که زیردرخت راست z ناتهی است، جانشین آن باید گره‌ای با کوچک‌ترین کلید در آن زیردرخت باشد؛ به همین دلیل فراخوانی $\text{TREE-MINIMUM}(z.right)$ انجام می‌شود.
- اگر y فرزند راست z باشد (شکل ۴.۱۲:(c)) سطرهای ۱۰ تا ۱۲ گره z را به عنوان فرزندی از والدش با y جایگزین می‌کنند و فرزند چپ y را با فرزند چپ z جایگزین می‌نمایند. گره y فرزند راست خود را حفظ می‌کند.
- اگر y فرزند راست z نباشد (شکل ۴.۱۲:(d)) سطرهای ۷ تا ۹ گره y را به عنوان فرزندی از والدش با فرزند راست y جایگزین می‌کنند و فرزند راست z را به فرزند راست y تبدیل می‌نمایند. در نهایت، سطرهای ۱۰ تا ۱۲ گره z را با y جایگزین کرده و فرزند چپ y را با فرزند چپ z جایگزین می‌کنند.

CLRS, 4th ed.، صفحه ۳۲۵ (ترجمه ماشینی)

شکل ۴.۱۲ (a) و (b)

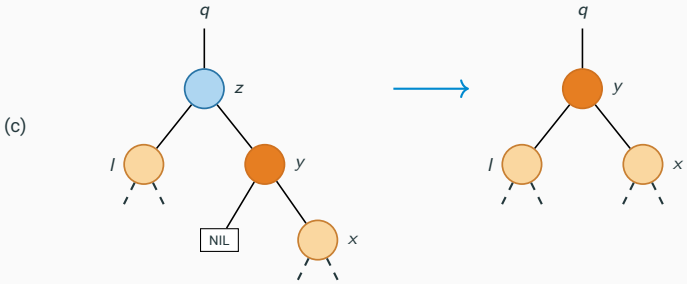


(a) گره z فرزند چپ ندارد. گره z را با فرزند راست آن یعنی r جایگزین می‌کنیم که ممکن است nil باشد یا نباشد.

(b) گره z دارای یک فرزند چپ l است اما فرزند راست ندارد. گره z را با l جایگزین می‌کنیم.

CLRS, 4th ed.، صفحه ۳۲۳ (ترجمه ماشینی)

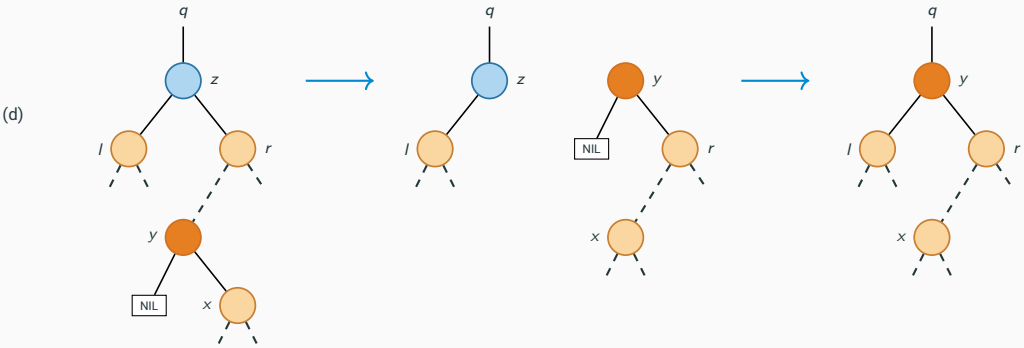
شکل ۴.۱۲ (c)



(c) گره z دارای دو فرزند است. فرزند چپ آن گره l، فرزند راست آن جانشینش یعنی y (که هیچ فرزند چپی ندارد)، و فرزند راست y نیز گره x است. گره z را با y جایگزین می‌کنیم، فرزند چپ y را به‌روزرسانی می‌کنیم تا برابر l شود، اما x را به عنوان فرزند راست y به حال خود باقی می‌گذاریم.

CLRS, 4th ed.، صفحه ۳۲۳ (ترجمه ماشینی)

شکل ۴.۱۲ (d)



ابتدا گره y (جانشین z) را با فرزند راست خودش یعنی x جایگزین می‌کنیم و y را به عنوان والد r تنظیم می‌کنیم. سپس y را به عنوان فرزند q و والد l قرار می‌دهیم.

CLRS, 4th ed.، صفحه ۳۲۳ (ترجمه ماشینی)

«هر سطر از TREE-DELETE، از جمله فراخوانی‌های TRANSPLANT، به جز فراخوانی TREE-MINIMUM در سطر ۵، زمان ثابتی طول می‌کشد. بنابراین، روال TREE-DELETE روی یک درخت به ارتفاع h در زمان $O(h)$ اجرا می‌شود.»

قضیه ۳.۱۲

عملیات مجموعه پویای INSERT (درج) و DELETE (حذف) را می‌توان به گونه‌ای پیاده‌سازی کرد که هر یک روی یک درخت جستجوی دودویی به ارتفاع h در زمان $O(h)$ اجرا شوند.

CLRS, 4th ed.، صفحه ۳۲۵ (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.