

فصل ۱

طراحی الگوریتم

همچنین کد عملیات‌های Rotation و Combine وقتی p فرزند چپ r است به صورت زیر می‌باشد:

//Rotation when p is the left child of r and q is the middle child of r .

$p \rightarrow \text{dataL} = r \rightarrow \text{dataL};$

$p \rightarrow \text{MiddleChild} = q \rightarrow \text{LeftChild};$

$r \rightarrow \text{dataL} = q \rightarrow \text{dataL};$

$q \rightarrow \text{dataL} = q \rightarrow \text{dataR};$

$q \rightarrow \text{LeftChild} = q \rightarrow \text{MiddleChild};$

$q \rightarrow \text{MiddleChild} = q \rightarrow \text{RightChild};$

$q \rightarrow \text{dataR.Key} = \text{MAXKEY};$

.....

//Combine when p is the left child of r and q is the right sibling of p .

$p \rightarrow \text{dataL} = r \rightarrow \text{dataL};$

$p \rightarrow \text{dataR} = q \rightarrow \text{dataL};$

$p \rightarrow \text{MiddleChild} = q \rightarrow \text{LeftChild};$

```

q → RightChild = q → MiddleChild;
if(r → dataR.Key = MAXKEY)// r was a 2-node
    r → dataL.Key = MAXKEY;
else {
    r → dataL = r → dataR;
    r → dataR.Key = MAXKEY ;
    r → MiddleChild = r → RightChild;
}

```

۱.۰.۰ تجزیه و تحلیل عملکرد حذف از یک درخت ۲-۳

مشخص است که یک عملکرد ترکیب یا چرخش به تنهایی در زمانی برابر با $O(1)$ انجام می‌گیرد. اگر یم چرخش انجام شود، عمل حذف کامل می‌گردد. اگر یک ترکیب صورت گیرد، p در درخت ۲-۳ به یک سطح بالاتر منتقل می‌شود. بنابراین تعداد ترکیباتی که در خلال یک حذف می‌توانند صورت گیرند از ارتفاع درخت ۲-۳ تجاوز نخواهد کرد و در نتیجه عمل حذف از یک درخت ۲-۳ با n عنصر به زمانی برابر با $O(\log n)$ نیاز دارد.

۱.۰ Red-Black سیاه قرمز - درخت قرمز - سیاه

یک درخت قرمز - سیاه به صورت زیر می‌باشد:^۱

۱.۱.۰ خواص درخت قرمز - سیاه

- هر گره یا قرمز و یا سیاه است.
- هر برگ nil سیاه است.
- دو فرزند یک گره قرمز، سیاه هستند. (در یک گره قرمز، فرزند قرمز نمی‌تواند باشد)

^۱مطالب این قسمت برگرفته از جزوه دکتر محمد قدسی می‌باشد.

- هر مسیر ساده از یک گره به برگ فرزند (نه لزوماً فرزند مستقیم) شامل تعداد یکسانی گره سیاه است.
- ریشه درخت سیاه است. (این شرط از شروط اساسی نمی باشد)

مانند مثال زیر:

۲.۱.۰ تعاریف و قضایای ابتدایی

$\text{Black-Height}(x)$: بنابر تعریف $\text{Black-Height}(x)$ یا $bh(X)$ برابر است با تعداد گره‌های سیاه از x تا یک برگ فرزند^۲.

$\text{uncle}(x)$:

```
if parent[x]=right[parent[parent[x]]] then
    uncle[x]:=left[parent[parent[x]]]
else uncle[x]:=right[parent[parent[x]]]
```

قضیه ۱: حداکثر ارتفاع یک درخت RB که دارای n گره داخلی است، برابر $2 \log_2^{(n+1)}$ است.

اثبات: برای اثبات قضیه فوق ابتدا نشان می‌دهیم که یک زیردرخت به ریشه دلخواه x حداقل دارای $2^{bh(x)} - 1$ گره داخلی است. برای اثبات از استقرا بر روی ارتفاع گره داخلی x در درخت استفاده می‌شود.

پایه استقراء:

اگر ارتفاع x صفر باشد، x حتماً برگ و در نتیجه nil است. بنابراین زیردرخت به ریشه x حداقل $2^{bh(x)} - 1$ یعنی $0 = 1 - 1$ گره داخلی دارد.

گام استقرائی:

گره داخلی x را در نظر بگیرید. $bh(x)$ عددی مثبت می‌باشد و x دارای دو فرزند می‌باشد. هر کدام از فرزندان بر حسب آنکه قرمز باشند یا سیاه، Black-Height آنها برابر $bh(x)$ یا $bh(x) - 1$ خواهد بود. به علت آنکه هر فرزند x ارتفاعی کمتر از خود x دارد می‌توانیم با استفاده از فرض استقرا نتیجه بگیریم که هر زیردرخت به ریشه یک فرزند x حداقل

^۲ رنگ خود x بی‌تأثیر است و آنرا نمی‌شماریم

۱ - $2^{bh(x)-1}$ گره داخلی دارد. بنابراین زیردرخت به ریشه x حداقل $2^{bh(x)} - 1 = 2^{bh(x)-1} - 1 + 2^{bh(x)-1} - 1 + 1$ گره داخلی خواهد داشت.

از طرف دیگر می‌دانیم که در درخت به ارتفاع h حداقل نیمی از گره‌ها (بدون در نظر داشتن ریشه) بر روی هر مسیر ساده از ریشه به برگ، سیاه هستند. زیرا در غیر این صورت خاصیت ۳ از خواص درخت نقض خواهد شد. در نتیجه Black-Height ریشه حداقل $\frac{h}{2}$ خواهد بود.

$$n \geq 2^{\frac{h}{2}} - 1 \Rightarrow 2^{\frac{h}{2}} \leq n + 1 \Rightarrow h \leq 2 \log_2^{(n+1)}$$

نتیجه: ارتفاع درخت، $hsj. O(\log n)$ RB.

۳.۱.۰ دوران

برای بازیابی خواص درخت RB بعد از عملیات درج و حذف در بعضی شرایط مجبور خواهیم شد که رنگ بعضی از گره‌ها را عوض کنیم یا تغییراتی در ساختار اشاره‌گرها اعمال نماییم. به همین منظور دو عمل دوران (راستگرد و چپگرد) تعریف می‌نماییم. با توجه به شکل زیر واضح است که این دو عمل هیچ اشکالی در خواص یک درخت دودویی ایجاد نخواهد کرد.

نکته: هنگامی که یک دوران چپگرد انجام می‌شود فرض می‌نماییم که فرزند راست گره مورد نظر nil نباشد. واضح است که دروان از $O(1)$ باشد زیرا تنها اشاره‌گرها در اثر یک دوران تغییر می‌نمایند و سایر اجزا بدون تغییر می‌ماند.

Left-Rotate(T,x)

1. $y \leftarrow \text{right}[x]$ // SET y
2. $\text{right}[x] \leftarrow \text{left}[y]$ // Turn y's left subtree into x's right subtree
3. if $\text{left}[y] \neq \text{NIL}$ then
4. $p[\text{left}[y]] \leftarrow x$
5. $p[y] \leftarrow p[x]$ // Link x's parent to y
6. if $p[x] = \text{Nil}$ then
7. $\text{root}[T] \leftarrow y$

8. else if $x = \text{left}[p[x]]$ then
9. $\text{left}[p[x]] \leftarrow y$
10. else $\text{right}[p[x]] \leftarrow y$
11. $\text{left}[y] \leftarrow x$ // Put x on y 's left
12. $p[x] \leftarrow y$

درج ۴.۱۰