دانشکده‌ی علوم ریاضی

ساختمان داده‌ها ۱۵ بهمن ۹۱

جلسه‌ی ۲۰: درخت‌های جست‌وجو، ۲ـ۳ و قرمز‌ـسیاه

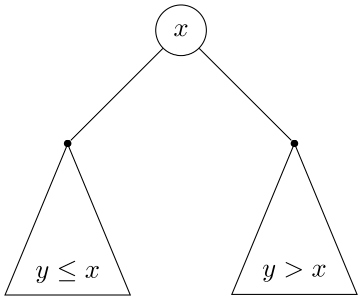
مدرّس: دکتر شهرام خزائی *نگارنده: مصطفی کریمی*

۱ درخت دودویی جست‌وجو

۱.۱ مقدمه‌ی بر درخت دودویی جست‌وجو

در جلسه قبل با درخت دودویی جست‌وجو آشنا شدیم که دارای ویژگی زیر است:

- گره y در زیر درخت سمت چپx قرار دارد، اگر $y.key \leq x.key$
- گره y در زیر درخت سمت راستx قرار دارد، اگر $y.key > x.key$



و همانطور که مشاهده کردیم در درخت دودویی جست‌وجو همه‌ی اعمال درج، حذف، گره پسین، گره پیشین، پیدا کردن کمینه و بیشینه در $\mathrm{O}(h)$ یعنی مرتبه ارتفاع درخت انجام می‌شود. ولی مطلوب ما $\mathrm{O}(\log n)$ است. به خاطر اینکه ممکن است درخت متعادل نباشد و ارتفاع آن از مرتبه $\mathrm{O}(\log n)$ نباشد از درخت‌های دودویی جدیدی استفاده می‌کنیم که درخت‌های متعادلی هستند. در ادامه با درخت‌های متعادل دودویی از قبیل درخت ۲ـ۳ و درخت قرمز‌ـسیاه آشنا می‌شویم.

۱.۲ مرتب کردن توسط درخت دودویی جست‌وجو

برای مرتب کردن یک آرایه کافی است در ابتدا درخت دودویی جست‌وجو مرتبط با آن را بسازیم و به وسیله پیمایش میان ترتیبی، که در جلسه قبل معرفی کردیم، آرایه را مرتب کنیم. الگوریتم مربوط به مرتب کردن در ادامه آمده است:

Algorithm 1 Algorithm: BST-SORT

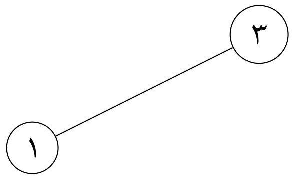
```
function BST-Sort(array A, length n)
    T.root ← NIL
    for i = 1 to n do
        make a node x with key A[i] and NIL children
        BST-Insert(T.root, x)
    InorderTraversal(T.root)
```

مثال ۱ *برای مرتب کردن آرایه [۳ ۱ ۸ ۲ ۶ ۷ ۵]، در ابتدا درخت جست‌وجو مرتبط با آن را با درج عناصر به ترتیب در یک درخت تهی می‌سازیم سپس پیمایش میان ترتیبی انجام می‌دهیم.*

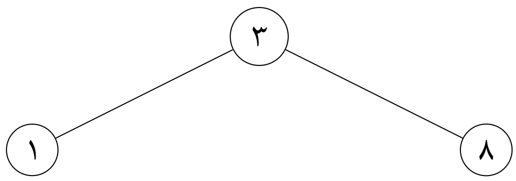
مراحل درج کردن به صورت زیر است:

در ابتدا عدد ۳ را درج می‌کنیم:

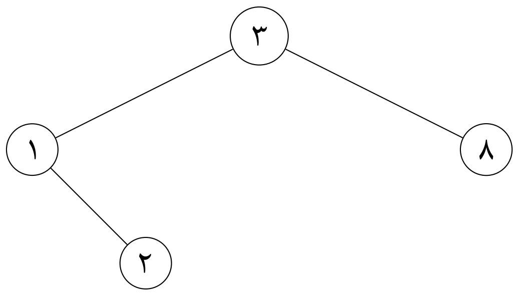
سپس عدد ۱ را درج می‌کنیم:



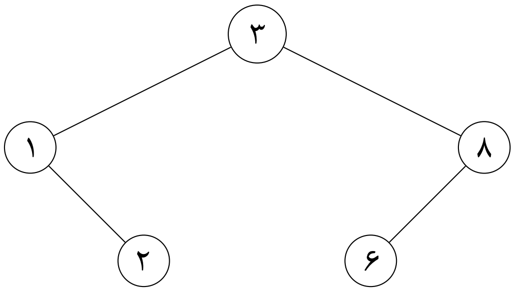
حال عدد ۸ را درج می‌کنیم:



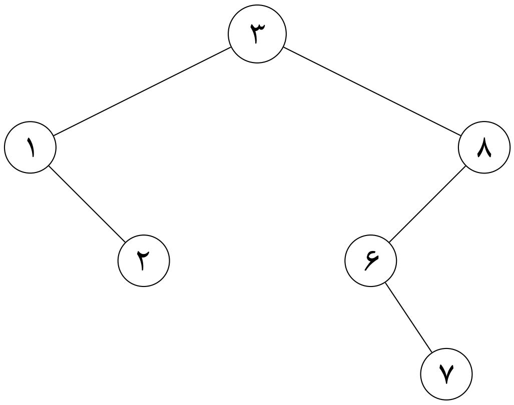
سپس عدد ۲ را درج می‌کنیم:



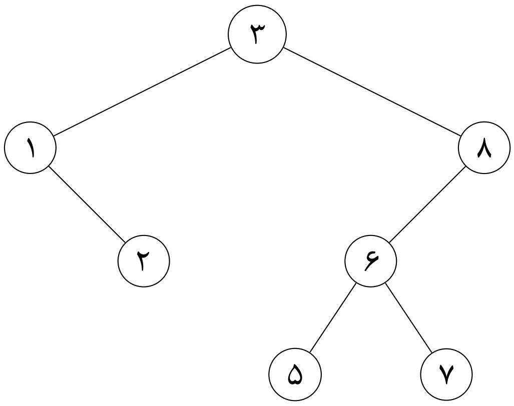
حال عدد بعدی یعنی ۶ را درج می‌کنیم:



سپس عدد ۷ را درج می‌کنیم:



و در نهایت عدد ۵ را درج می‌کنیم. درخت جست‌وجو نهایی به صورت زیر خواهد شد:



و با انجام پیمایش میان ترتیبی آرایه زیر حاصل می‌شود:

1	2	3	5	6	7	8

۳.۱ ارتباط بین مرتب‌سازی با درخت دودویی جست‌وجو و مرتب‌سازی سریع

بین مرتب‌سازی با درخت دوددویی جست‌وجو و مرتب‌سازی سریع[۱] تناظر یک به یکی وجود دارد. به طور مثال، در مثال قبل قرار دادن عدد ۳ در ریشه معادل با اعمال اولین عمل بخش‌بندی[۲] در مرتب سازی سریع است. حال با استفاده

[۱] Quicksort
[۲] Partition

از این تناظر یک به یک و الهام از تصادفی کردن روش مرتب‌سازی سریع، روش مرتب‌سازی توسط درخت دودویی جست‌وجو را تصادفی می‌کنیم. کافی است که تصادفی بودن را در آرایه اعمال کنیم. الگوریتم به صورت زیر خواهد شد:

Algorithm 2 Algorithm: RANDOM-BST-SORT

```
function RANDOM-BST-SORT(array A, length n)
    Randomly Permute array A
    T.root ← NIL
    for i = 1 to n do
        make a node x with key A[i] and NIL children
        BST-INSERT(T.root, x)
    INORDERTRAVERSAL(T.root)
```

از آنجایی که متوسط زمان اجرای مرتب‌سازی سریع از مرتبه $n \log n$ است و همانطورکه گفته شده، تناظر یک به یک بین زمان اجرای مرتب‌سازی سریع و الگوریتم RANDOM-BST-SORT وجود دارد. از این تناظر می‌توان نتیجه گرفت که متوسط میانگین ارتفاع همه گره‌های موجود در درخت دودویی تشکیل شده از مرتبه $\log n$ است. با این وجود می‌توان اثبات کرد که متوسط ارتفاع یک درخت دودویی جست‌وجو که به صورت تصادفی تشکیل شده نیز از مرتبه $\log n$ است. (در ضمیمه این ادعا اثبات شده است.)

۲ درخت‌های دودویی جست‌وجوی تصادفی

۱.۲ مقدمه

همانطور که در بخش‌های قبل مشاهده کردیم، مرتبه زمانی اجرای عملیات مختلف در درخت دودویی جست‌وجو از مرتبه $O(h)$ است. در صورتی‌که مطلوب ما $O(\log n)$ است. برای دستیابی به این هدف می‌توانیم از ایده تصادفی کردن درخت دودویی جست‌وجو استفاده کنیم. روش دیگری که رایج‌تر است، استفاده مستقیم از درخت‌های است که از لحاظ ارتفاع متعادل باشند. در بخش‌های بعد به طور کامل با درخت های متعادل مانند درخت ۳ـ۲ و یا درخت قرمز سیاه آشنا می‌شویم. هدف ما وارد کردن عناصر یک آرایه داده شده به طول n و با عناصر متمایز در یک درخت دودویی جست‌وجو به صورت "تصادفی" است. در این بخش به بررسی دو نوع روش مختلف تصادفی کردن درخت دودویی جست‌وجو می‌پردازیم.

۲.۲ روش اول: انتخاب کاملاً تصادفی یک یک درخت دودویی و پر کردن آن با عناصر آرایه

در این روش ابتدا از بین همه درخت‌های دودویی ممکن یکی را به طور کاملاً تصادفی (با توزیع یکنواخت) انتخاب می‌کنیم. توجه کنید که یک درخت دودویی مفروض با n رأس را به طور یکتا می‌توان با مقادیر آرایه داده شده پر نمود به طوری که حاصل درخت دودویی جست‌وجو باشد. بنابراین تعداد درخت‌های دودویی جست‌وجو که متناظر با یک آرایه با n عنصر متمایز، برابر با تعداد درخت‌های دودویی با n رأس است. همچنین تعداد درخت‌های دودویی با n رأس با استفاده از رابطه بازگشتی زیر قابل محاسبه است که جواب آن دنباله اعداد کاتالان است:

$$C_n = \sum_{i=۰}^{n-۱} C_i C_{n-۱-i} \ , C_۰ = ۱$$

$$C_n = \frac{۱}{n+۱}\binom{۲n}{n} \sim \frac{۴^n}{n^{۳/۲}\sqrt{\pi}}$$

در این روش تصادفی، متوسط ارتفاع درخت تصادفی از مرتبه $\mathrm{O}\left(\sqrt{n}\right)$ خواهد شد. و این مرتبه برای ما مطلوب نیست.

۳.۲ روش دوم: تشکیل درخت با درج عناصر آرایه به ترتیب کاملاً تصادفی در آن

در این روش، آرایه ورودی را تصادفی خواهیم کرد. یعنی، در ابتدا از بین جایگشت‌های مختلف آرایه ورودی یکی را به صورت کاملاً تصادفی انتخاب می‌کنیم. سپس با درج کردن متوالی عناصر آرایه تصادفی شده، درخت تصادفی مربوط به آن را می‌سازیم. تعداد درخت‌های (نه لزوماً متمایز) تصادفی حاصل در این روش برابر تعداد جایگشت‌های n تایی خواهد شد. با توجه به اینکه $n! > C_n$ (برای $n \geq ۳$) توزیع حاصل دیگر کاملاً یکنواخت روی همه درخت‌های دودویی ممکن نخواهد بود.

خوشبختانه در این روش تصادفی، متوسط ارتفاع درخت تصادفی حاصل از مرتبه $\mathrm{O}\left(\log n\right)$ خواهد شد. یعنی، اگر ترتیب درج کردن عناصر یک آرایه در یک درخت دودویی جست‌وجو به صورت تصادفی باشد، بهتر از این است که از ابتدا خود درخت را به صورت کاملاً تصادفی انتخاب کنیم.

۳ درخت ۳ـ۲

۱.۳ مقدمه

درخت‌های ۳ـ۲ اولین نوع درخت‌هایی است که برای متعادل کردن ارتفاع درخت‌ها استفاده شده است. یکی از اولین روش‌ها، غیر از تصادفی کردن درخت‌های دودویی جست‌وجو است. در ادامه به تعریف و بررسی ویژگی‌های این درخت می‌پردازیم.

با این وجود اگر در یک درخت دودویی جست‌وجوی متعادل اعمال درج و حذف را به صورت متوالی انجام دهیم درخت دوباره نامتعادل می‌شود و ارتفاع آن از مرتبه $\mathrm{O}\left(\sqrt{n}\right)$ خواهد شد. این نشان می‌دهد که اعمال درج و حذفی که قبلاً ارائه شدند مناسب نیستند و باید به دنبال روش برای متعادل نگه داشتن درخت باشیم. در ادامه با درخت‌های جدیدی آشنا می‌شویم که از لحاظ ارتفاع متعادل هستند.

۲.۳ تعریف درخت ۳ـ۲

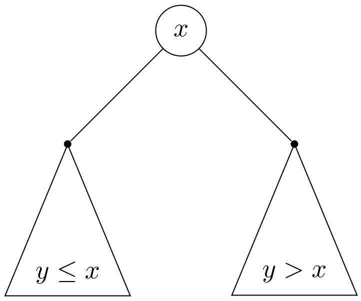
درخت ۳ـ۲، درختی است که ویژگی‌های زیر را داشته باشد:

- دو نوع مختلف گره دارد:
  - گره دارای ۲ فرزند: شامل یک عنصر باشد.
  - گره دارای ۳ فرزند: شامل دو عنصر باشد.
- ویژگی درخت جست‌وجو را دارا است

در شکل زیر ویژگی درخت جست‌وجو بودن درخت ۳ـ۲ را برای گره ۲ فرزند x مشاهده می‌کنید:

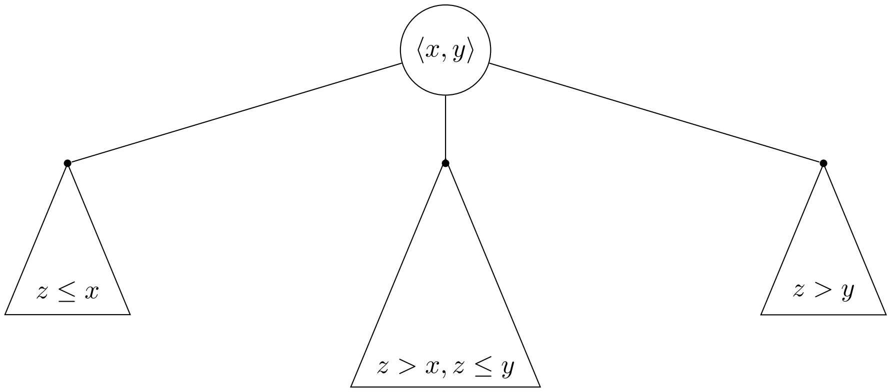
- گره y در زیر درخت سمت چپ x قرار دارد، اگر $y.key \leq x.key$

- گره y در زیر درخت سمت راست x قرار دارد، اگر $y.key > x.key$



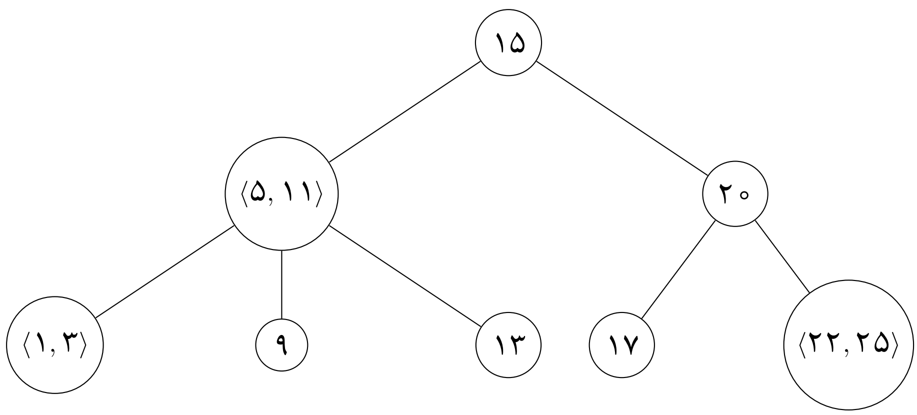
در شکل زیر ویژگی درخت جست‌وجو بودن درخت ۳ـ۲ را برای گره ۳ فرزند $\langle x, y\rangle$ مشاهده می‌کنید:

- گره z در زیر درخت سمت چپ قرار دارد، اگر $z.key \leq x.key$
- گره z در زیر درخت وسط قرار دارد، اگر $x.key < z.key \leq y.key$
- گره z در زیر درخت سمت راست قرار دارد، اگر $z.key > y.key$



همانطور که در شکل‌های بالا مشاهده می‌کنیم، شکل اول از بالا همان حالت عادی ویژگی جست‌وجو است که برای گره‌های دو فرزند درخت ۳ـ۲ استفاده می‌شود. شکل دوم از بالا مربوط به گره‌های ۳ فرزند است که دارای ۲ عنصر $\langle x, y\rangle$ هستند که $x.key \leq y.key$ است. گره‌هایی که مقدار کلید آنها از کلید x کوچکتر هستند در سمت چپ و گره‌هایی که مقدار کلید آنها بین مقدار کلید x و y قرار دارند در وسط و گره‌هایی که مقدار کلید آنها از مقدار کلید y بزرگتر هستند در سمت راست قرار می‌گیرند.

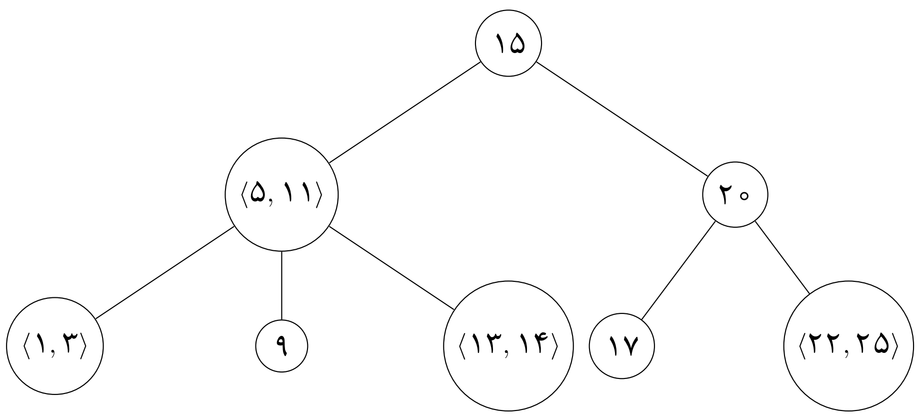
مثال ۲ *در شکل زیر، که به عنوان شکل پایه در چند مثال آینده خواهد آمد، یک درخت ۳ـ۲ را مشاهده می‌کنید. می‌توان متعادل بودن این درخت را از لحاظ ارتفاع مشاهده کرد.*



۳.۳ درج کردن در درخت ۲ـ۳

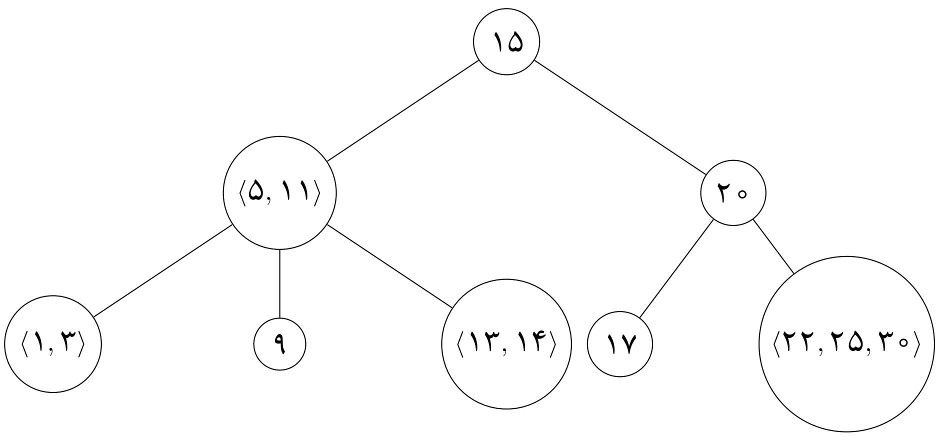
در ابتدا با مثال‌هایی از درج کردن آشنا می‌شویم سپس حالت کلی درج کردن را بررسی می‌کنیم:

مثال ۳ *در مثال ۲ اگر بخواهیم عدد ۱۴ را درج کنیم به درخت زیر می‌رسیم:*

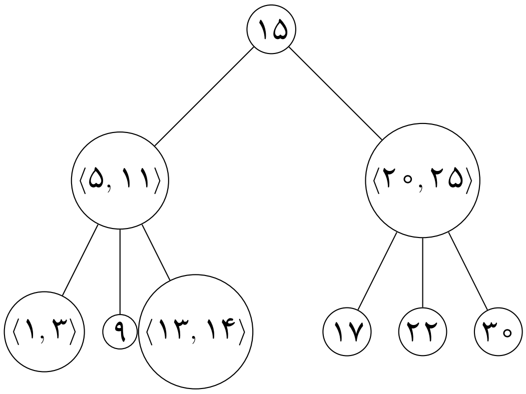


در شکل بالا مشاهد می‌کنیم که ۱۳ و ۱۴ با هم یک گره ۳ فرزند را تشکیل می‌دهند.

مثال ۴ *در مثال ۲ اگر بخواهیم عدد ۳۰ را درج کنیم به درخت زیر می‌رسیم:*

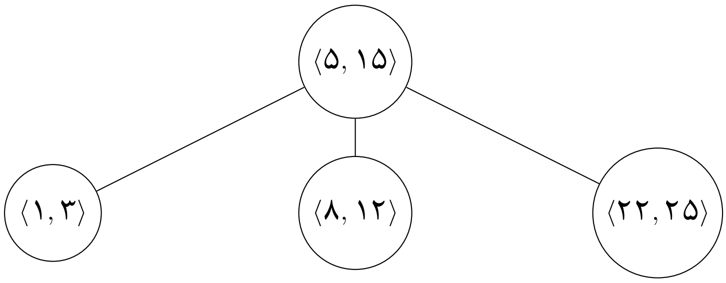


همانطور که مشاهده می‌کنیم به طور موقت یک گره شامل ۳ عنصر است. برای اینکه به شکل درخت ۳ـ۲ تبدیل شود، کافی است کلید وسط را به بالا انتقال دهیم. در نهایت درخت ۳ـ۲ زیر به وجود می‌آید.

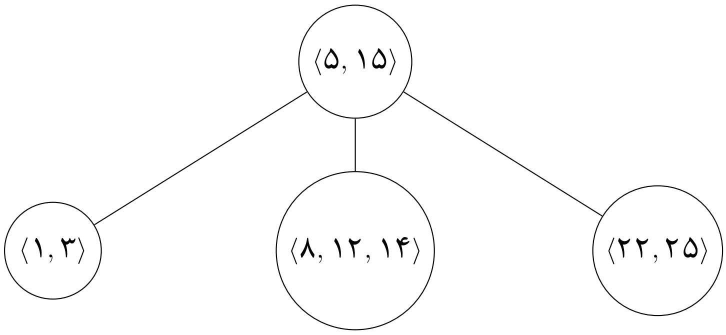


همانطور که در شکل بالا مشاهده می‌کنید کلید وسط را به گره بالای انتقال می‌دهیم در نتیجه گره بالایی دو کلید می‌شود.

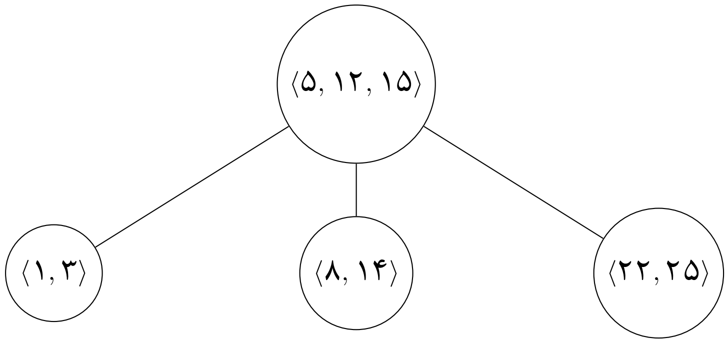
مثال ۵ *در درخت ۳ـ۲ زیر می‌خواهیم عدد ۱۴ را درج کنیم:*



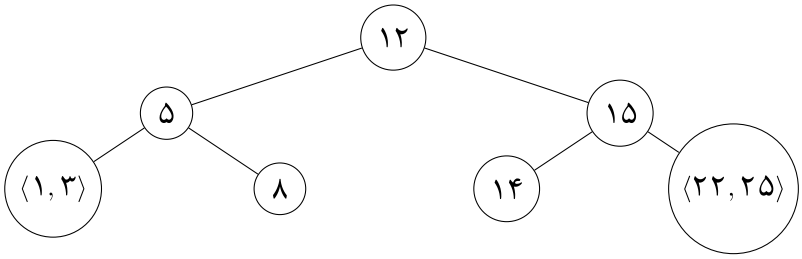
با درج کردن عدد ۱۴ در ابتدا شکل زیر بدست می‌آید:



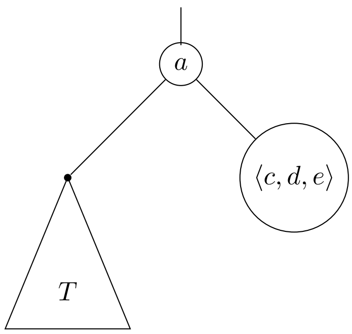
حال مانند مثال قبل کلید ۱۲ را به گره بالایی انتقال می‌دهیم و درخت به صورت زیر می‌شود:



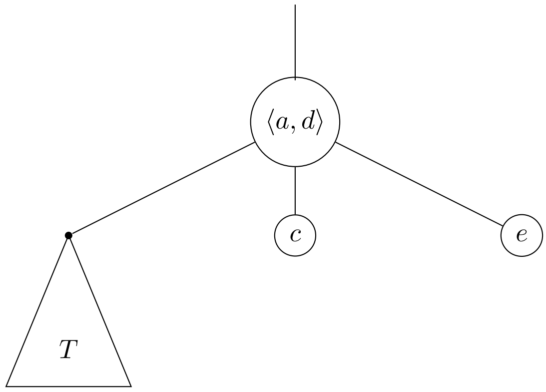
در شکل بالا دوباره یک گره با ۳ کلید داریم دوباره باید کلید وسط را به بالا انتقال دهیم ولی این بار چون گره‌ی بالاتر نداریم یک ریشه جدید تشکیل می‌دهیم. درخت درنهایت به صورت زیر اصلاح خواهد شد:



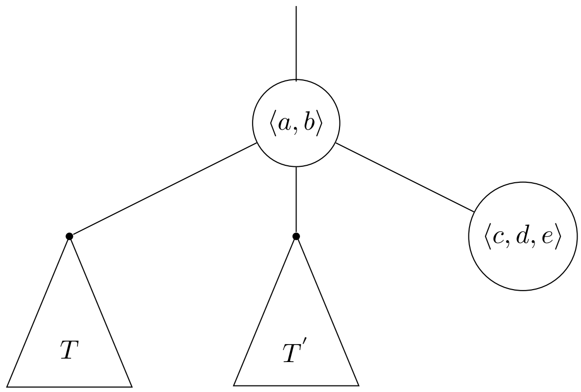
همانطور که در شکل بالا مشاهده می‌کنیم ارتفاع درخت یکی زیاد شده است.
در مثال‌های بالا با حالات مختلف درج کردن آشنا شدیم. حال مسئله درج کردن را به صورت کلی مورد بررسی قرار می‌دهیم. در ابتدا هر کلید را به صورت معمولی درج می‌کنیم. اگر برگ ۲ عنصر داشته باشد که نیازی نیست تغییری بدهیم ولی اگر ۳ عنصر داشته باشد ۲ حالت ممکن است به وجود بیاید که به صورت زیر است:
حالت ۱) یک گره ۳ تا عنصر داشته باشد و گره بالایی آن تک عنصر داشته باشد:



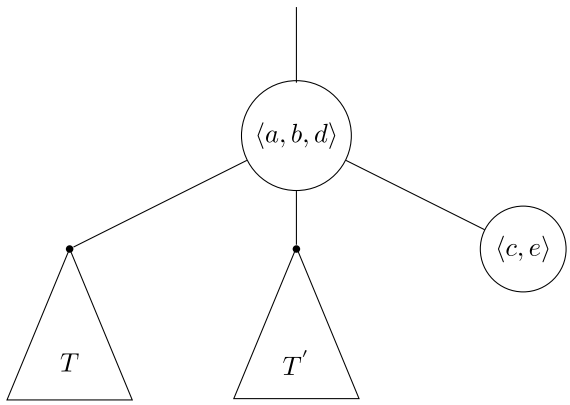
در این صورت کلید وسطی را به بالا انتقال می‌دهیم و گره بالایی ۲ کلید خواهد شد. درخت به صورت زیر خواهد شد:



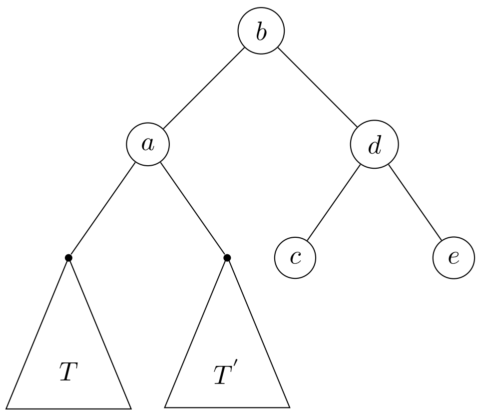
حالت ۲) یک گره ۳ تا کلید داشته باشد و گره بالایی نیز ۲ کلید داشته باشد:



درابتدا کلید وسط را به بالا انتقال می‌دهیم و مشاهده می‌کنیم که این بار گره بالای دارای ۳ کلید خواهد شد.



حال دوباره به صورت بازگشتی، گره ۳ عنصر حاصل را اصلاح می‌کنیم. اگر گره ۳ عنصر به عنوان ریشه قرار گیرد به صورت زیر خواهد شد.



تا وقتی که گره ۳ کلید داریم، کلید وسط را به بالا انتقال می‌دهیم تا گره ۳ کلید دیگر نداشته باشیم یا اگر مجبور شدیم یک ریشه جدید با کلید وسطی به وجود می‌آوریم.

۴ درخت قرمز-سیاه

۱.۴ مقدمه

در این بخش با درخت‌های قرمز-سیاه آشنا می‌شویم که نوع دیگری از درختهای متعادل کننده است. در نهایت ارتباط بین درخت‌های ۳-۲ و درخت‌های قرمز-سیاه را بررسی می‌کنیم و نوع خاصی از درخت‌های قرمز-سیاه به نام درخت‌های قرمز-سیاه متمایل به چپ را معرفی می‌کنیم.

۲.۴ معرفی درخت قرمز سیاه

درخت قرمز سیاه، درختی است که ویژگی‌های زیر را داشته باشد:

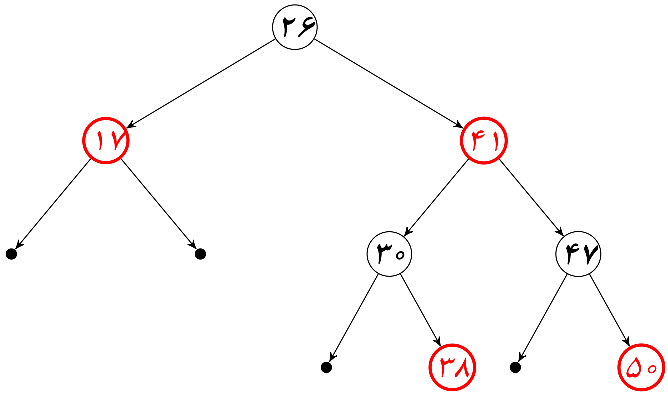
- یک درخت دودویی جست‌وجو است
- برای گره‌ها پارامتر جدیدی به نام رنگ معرفی می‌شود:
  - گره می‌تواند رنگ قرمز داشته باشد.
  - گره می‌تواند رنگ سیاه داشته باشد.
- ریشه همیشه رنگ سیاه دارد.
- برگ‌ها (که همیشه گره تهی هستند) رنگ سیاه دارند.
- اگر گره‌ی رنگ قرمز داشته باشد، پدرش رنگ سیاه دارد.
- همه‌ی مسیرها از هر گره به همه نوادگانش به تعداد یکسان گره سیاه دارد.

در درخت قرمز-سیاه، برای هر گره ۲ نوع ارتفاع تعریف می‌شود:

- ارتفاع ساده گرهx: طولانی‌ترین مسیر از گره x تا نوادگان برگی‌اش که با h نشان می‌دهند.

• ارتفاع سیاه گره x: ارتفاع سیاه برای گره x برابر با تعداد گره‌های سیاه از گره x به نوادگان برگی‌اش با احتساب رنگ برگ و عدم احتساب رنگ خود h است. این ارتفاع را با bh نشان می‌دهند.

مثال ۶ درخت زیر یک درخت *قرمز–سیاه است:*



همانطور که در درخت بالا مشاهده می‌کنیم تمام ویژگی‌های مربوط به درخت قرمز–سیاه لحاظ شده است.

۳.۴ متعادل بودن درخت قرمز–سیاه

قضیه ۱ *در درخت قرمز–سیاه با n راس روابط زیر برقرار است:*

$$h \leq 2\log(n+1) \tag{1}$$

$$bh \leq \log(n+1) \tag{2}$$

$$h \leq 2bh \tag{3}$$

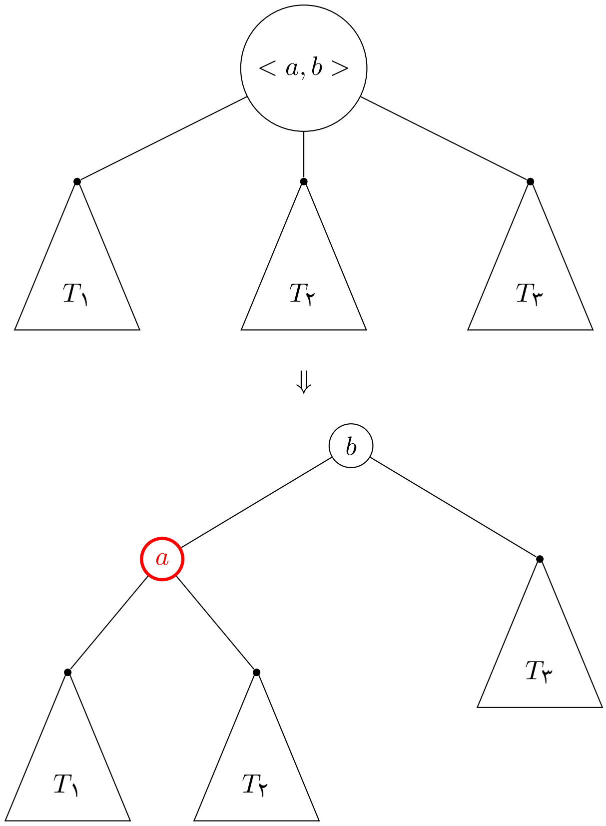
روابط بالا با استقرا قابل اثبات است. اثبات روابط بالا به خواننده واگذار می‌شود.

طبق قضیه بالا ارتفاع درخت قرمز سیاه از مرتبه $\log n$ است در نتیجه تمام اعمال اصلی در مرتبه $\log n$، که مطلوب ما است، انجام می‌شود.

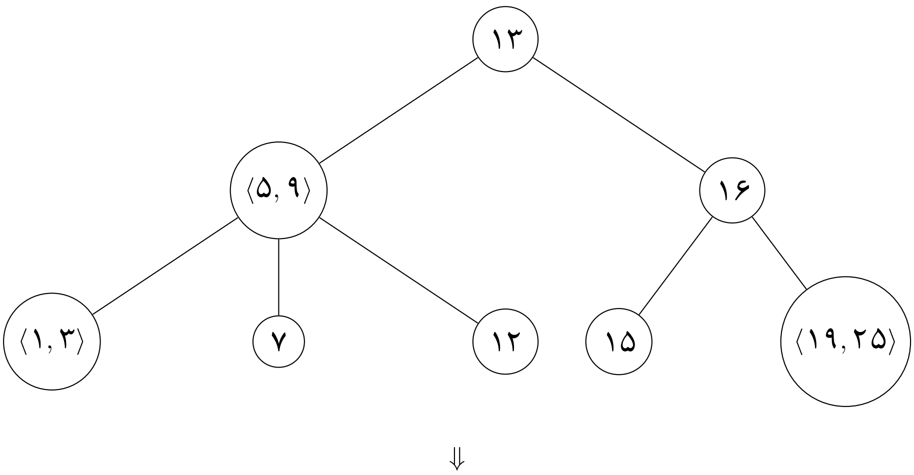
۴.۴ ارتباط بین درخت قرمز–سیاه و درخت ۳–۲

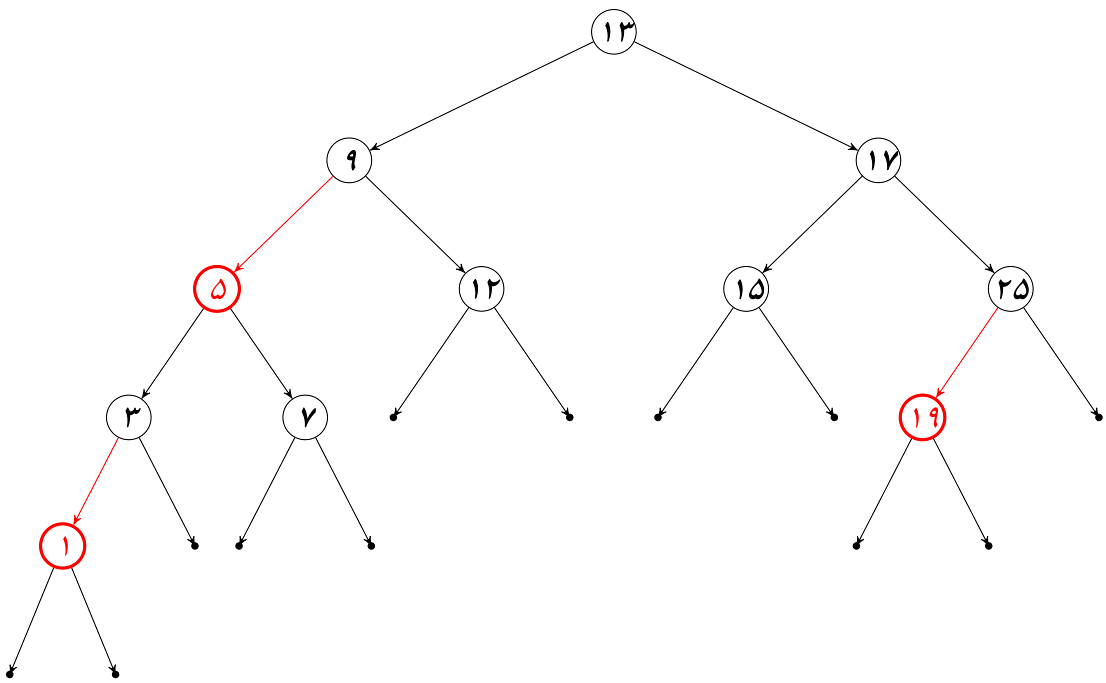
دراین قسمت نحوه تبدیل درخت‌های ۳–۲ را به درخت قرمز–سیاه را بررسی می‌کنیم.

به طور کلی برای تبدیل درخت ۳–۲ به درخت قرمز–سیاه کافی است گره‌ها با ۲ کلید را به صورت زیر تبدیل کرد:



مثال ۷ در این مثال درخت ۳ـ۲ زیر را به درخت قرمز-سیاه تبدیل می‌کنیم:





برای درخت‌های قرمز سیاه می‌توان یال‌ها را نیز رنگی در نظر گرفت. یال‌های قرمز مربوط به یالی است که از پدر سیاه به فرزند قرمز وصل می‌شود(رنگ‌ها برای یال‌ها فقط نوعی تصور است وگرنه هیچ جا ذخیره نمی‌شوند).

بین درخت ۳_۲ و درخت قرمز سیاه جدید یک تناظر یک به یکی وجود دارد. چون درخت‌های ۳_۲ از لحاظ ارتفاع متعادل هستند در نتیجه درخت‌های قرمز_سیاه جدید از لحاظ ارتفاع سیاه متعادل هستند.

درخت‌های قرمز_سیاه جدید تبدیل شده از درخت ۳_۲ حالت خاصی از درخت‌های قرمز_سیاه هستند که به آنها درخت‌های قرمز_سیاه متمایل به چپ می‌گویند.

تعریف ۱ درخت قرمز_سیاه متمایل به چپ: *نوع خاصی از درخت‌های قرمز_سیاه است که گره قرمز فقط می‌تواند فرزند سمت چپ باشد به خاطر همین به آنها درخت قرمز_سیاه متمایل به چپ می‌گویند.*

۵ ضمیمه

۱.۵ میانگین و متوسط ارتفاع درخت های دودویی جست‌وجو تصادفی (روش دوم تصادفی کردن)

همانطور که گفته شد میانگین ارتفاع گره‌های درخت تصادفی از مرتبه $\log n$ است که به صورت دقیق در زیر نمایش داده شده است:

$$\begin{aligned} Average(node\ path) &= \frac{1}{n}\mathbb{E}[\sum_{i=1}^{n}[Comparison\ to\ insert\ node\ i\ ]] \\ &= \frac{1}{n}\,\mathrm{O}\,(n\log n) \\ &= \mathrm{O}\,(\log n) \end{aligned} \tag{4}$$

در ادامه اثبات می‌کنم که متوسط ارتفاع درخت‌های تصادفی نیز از مرتبه $\log n$ است ولی در این اثبات از یک لم استفاده می‌کنم که به این درس مربوط نیست و بدون اثبات از آن استفاده می‌کنم.

تعریف ۲ (تابع محدب) تابع $f\colon \mathbb{R} \to \mathbb{R}$ محدب است اگر به ازای هر $\alpha \geq 0$ و β که $\alpha + \beta = 1$ است و برای هر x و y حقیقی رابطه زیر برقرار می‌باشد:

$$f(\alpha x+\beta y) \leq \alpha f(x)+\beta f(y) \tag{5}$$

لم ۲ (جانسون) اگر f تابعی محدب باشد و X متغیرتصادفی باشد آنگاه رابطه زیر برقرار است:

$$f(\mathbb{E}[X]) = \mathbb{E}[f(X)] \tag{6}$$

قضیه ۳ (متوسط ارتفاع درخت دودویی تصادفی) متوسط ارتفاع درخت جست‌وجو دودویی تصادفی از مرتبه $\mathrm{O}(\log n)$ است

برهان. متغیر تصادفی X_n ارتفاع درخت تصادفی با n گره است و $Y_n = 2^{X_n}$ ارتفاع نمایی است. اگر ریشه درخت رتبه k داشته باشدآنگاه:

$$X_n = 1 + \max(X_{k-1}, X_{n-k}) \tag{7}$$

چون زیر درخت‌های چپ و راست نیز به صورت تصادفی تشکیل شده‌اند و همچنین رابطه زیر برقرار است:

$$Y_n = 2 \times \max(Y_{k-1}, Y_{n-k}) \tag{8}$$

متغیر تصادفی Z_{nk} را به صورت زیر تعریف می‌کنیم:

$$Z_{nk} = \begin{cases} 1 \ \textit{if the root has rank } k \\ 0 \ \textit{otherwise} \end{cases} \tag{9}$$

درنتیجه رابطه زیر برقرار است:

$$\Pr(Z_{nk} = 1) = \mathbb{E}[Z_{nk}] = \frac{1}{n} \tag{10}$$

و چون Z_{nk} فقط در یک نقطه مقدار دارد و در بقیه نقاط صفر است رابطه زیر برقرار است:

$$Y_n = \sum_{k=1}^{n} Z_{nk}(2 \times \max(Y_{k-1}, Y_{n-k})) \tag{11}$$

حال از دو طرف امید ریاضی می‌گیریم:

$$
\begin{aligned}
\mathbb{E}[Y_n] =& \mathbb{E}[\sum_{k=1}^{n} Z_{nk}(2 \times \max(Y_{k-1}, Y_{n-k}))] \\
&= \sum_{k=1}^{n} \mathbb{E}[Z_{nk}(2 \times \max(Y_{k-1}, Y_{n-k}))] \\
&= 2\sum_{k=1}^{n} \mathbb{E}[Z_{nk}]\mathbb{E}[\max(Y_{k-1}, Y_{n-k})] \\
&\leq \frac{2}{n}\sum_{k=1}^{n} \mathbb{E}[\max(Y_{k-1}, Y_{n-k})] \\
&= \frac{4}{n}\sum_{k=0}^{n-1} \mathbb{E}[Y_k]
\end{aligned} \tag{12}
$$

حال با استفاده از جای‌گذاری می‌توان نشان داد که $\mathbb{E}[Y_n]$ از مرتبه $n^۳$ است:

$$
\begin{aligned}
\mathbb{E}[Y_n] =& \frac{4}{n}\sum_{k=0}^{n-1} \mathbb{E}[Y_k] \\
&\leq \frac{4}{n}\sum_{k=0}^{n-1} ck^3 \\
&\leq \frac{4c}{n}\int_{k=0}^{n-1} k^3 \\
&= \frac{4c}{n}(\frac{n^4}{4}) \\
&= cn^3
\end{aligned} \tag{13}
$$

تا الان نشان دادیم که که $\mathbb{E}[Y_n]$ از مرتبه $n^۳$ است حال از آنجایی که ۲^x یک تابع محدب است به رابطه زیر خواهیم رسید:

$$
\begin{aligned}
2^{\mathbb{E}[Y_n]} &\leq \mathbb{E}[2^{X_n}] \\
&= \mathbb{E}[Y_n] \\
&= cn^3
\end{aligned} \tag{14}
$$

با گرفتن لگاریتم از دو طرف:

$$
\mathbb{E}[X_n] \leq 3\log n + \mathrm{O}(1) \tag{15}
$$

درنتیجه قضیه اثبات شد و حکم برقرار است. یعنی، متوسط ارتفاع درخت جست‌وجو دودویی تصادفی از مرتبه $\mathrm{O}(\log n)$ است.

■