# الگوریتم‌های پیشرفته

## دکتر معینی

## تمرین سوم


پارسا محمدپور

حسام مومیوند فرد

محمد رهبری مقدم

سعید تریک


پاییز ۱۴۰۳

۱- کلیدهای زیر را به ترتیب در یک درخت AVL که در ابتدا خالی است وارد کنید و سپس درخت نهایی حاصل را رسم کنید.

۳۰, ۴۰, ۲۴, ۵۸, ۴۸, ۲۶, ۱۱, ۱۳

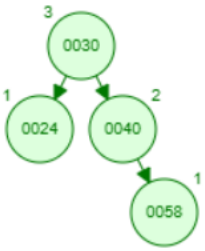
برای این کار از یک نمایشگر آنلاین درخت avl[1] استفاده می‌کنیم.[۲] درخت حاصل پس از وارد کردن اعداد به ترتیب داده‌شده، برابر زیر است:

بعد از وارد کردن عدد ۳۰:


بعد از وارد کردن عدد ۴۰:


بعد از وارد کردن عدد ۲۴:

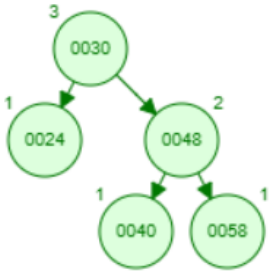

بعد از وارد کردن عدد ۵۸:



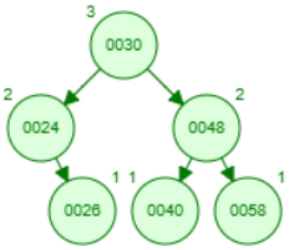

[1] avl tree visualization

[2] آدرس وب‌سایت

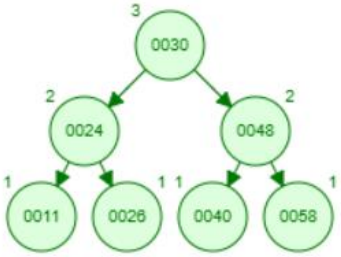
بعد از وارد کردن عدد ۴۸:



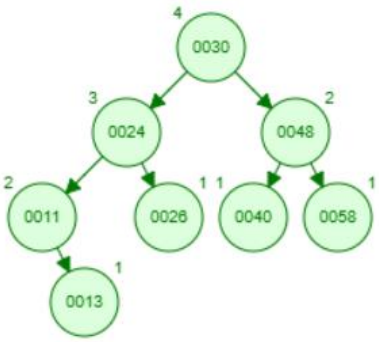

بعد از وارد کردن عدد ۲۶:




بعد از وارد کردن عدد ۱۱:




بعد از وارد کردن عدد ۱۳:



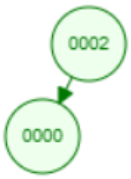
۲- کلیدهای زیر را به ترتیب در یک درخت Splay که در ابتدا خالی است وارد کنید و سپس درخت نهایی حاصل را رسم کنید.

۰, ۲, ۴, ۶, ۸, ۱۰, ۱۲, ۱۴, ۱۶, ۱۸

برای این کار از یک نمایشگر آنلاین درخت splay[3] استفاده می‌کنیم.[۴] درخت حاصل پس از وارد کردن اعداد به ترتیب داده‌شده، برابر زیر است:

بعد از وارد کردن عدد ۰:


بعد از وارد کردن عدد ۲:



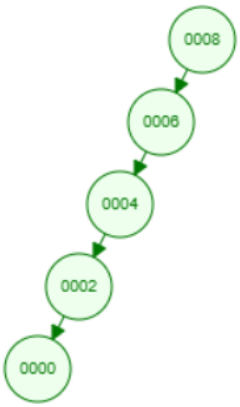

بعد از وارد کردن عدد ۴:


بعد از وارد کردن عدد ۶:

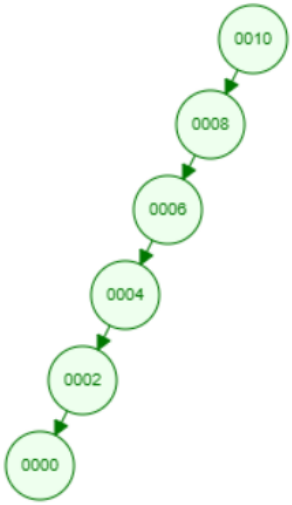

[3] splay tree visualization

[4] آدرس وب‌سایت

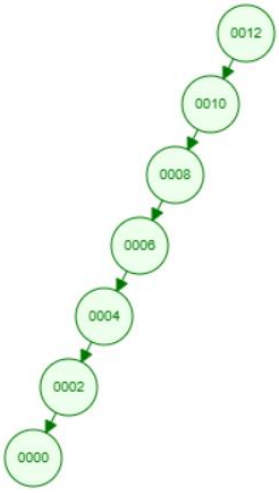
بعد از وارد کردن عدد ۸:



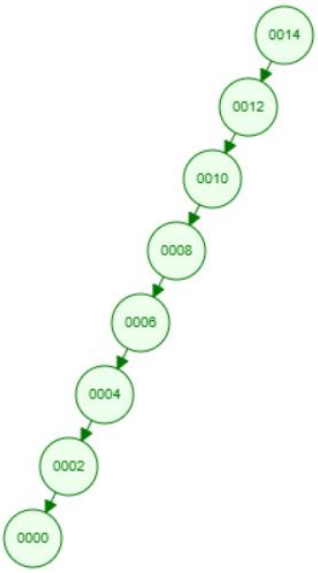

بعد از وارد کردن عدد ۱۰:



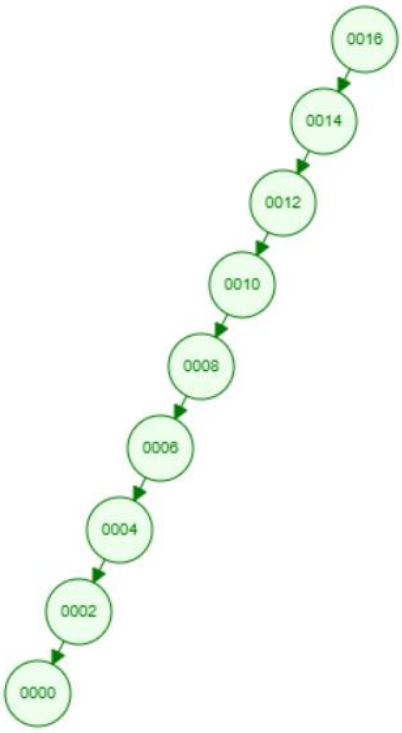

بعد از وارد کردن عدد ۱۲:



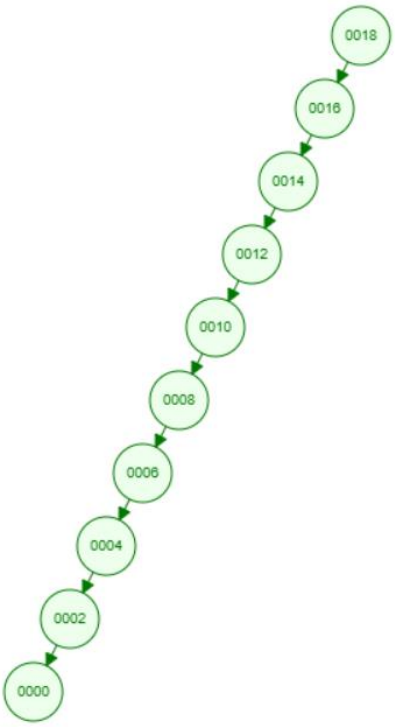
بعد از وارد کردن عدد ۱۴:




بعد از وارد کردن عدد ۱۶:



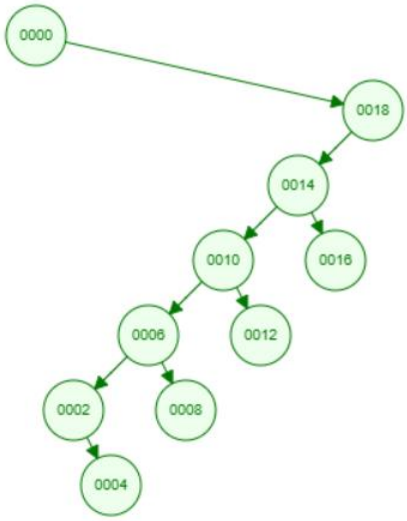
بعد از وارد کردن عدد ۱۸:



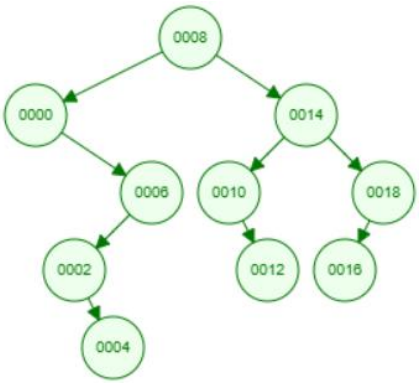

در اینجا سوال تمام شده است، اما برای اینکه درخت splay را مورد بررسی قرار دهیم، فرض می‌کنیم در همین حالت یک بار به ترتیب بر روی صفر و سپس هشت و بار دیگر به ترتیب بر روی هشت و سپس صفر عمل جستجو انجام شود. درخت به شکل زیر در ای‌آید:

- جستجو بر روی صفر و سپس هشت:

درخست پس از جستجو بر روی صفر:

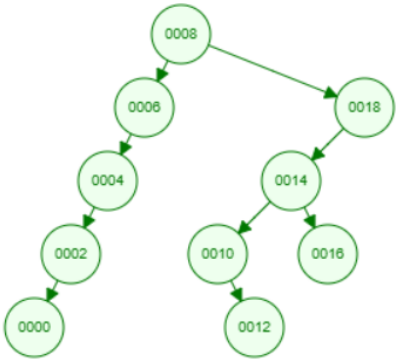


درخت پس از جستجو بر روی هشت:

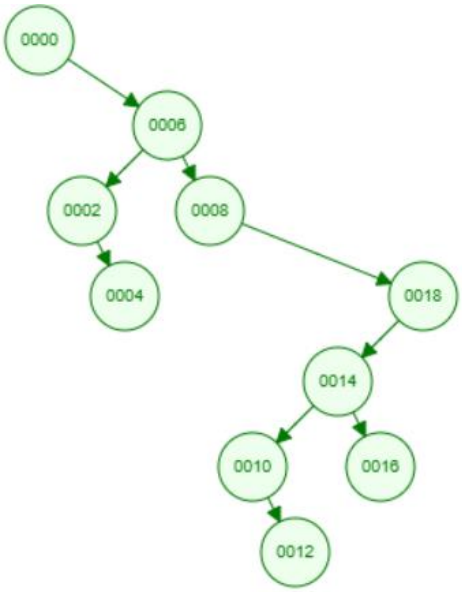



- جستجو بر روی هشت و سپس صفر:

درخت پس از جستجو بر روی هشت:




درخست پس از جستجو بر روی صفر:



۳- حداقل تعداد گره های داخلی در یک درخت AVL با ارتفاع ۷ چقدر است؟

می‌دانیم درخت AVL، درختی جستجوی دودویی[5] است که در هر گره از آن، تفاوت ارتفاع دو زیر درخت راست و چپ آن، حداکثر یکی با یکدیگر تفاوت دارند. پس با این حساب اگر بخواهیم به کمترین تعداد گره در این درخت دست پیدا کنیم، در هر گره، اختلاف ارتفاع زیر درخت‌های سمت راست و چپ‌شان را باید برابر با یک در نظر بگیریم. یعنی برای مثال اگر اتفاع یک گره‌ای برابر با پنج است، باید ارتفاع زیر درخت‌های آن چهار و سه باشند. (طبیعتا ارتفاع خود گره یکی بیشترین از ماکزیمم ارتفاع دو زیر درخت راست و چپ آن می‌شود) حالا اگر زیر درخت‌های سمت راست و چپ آن هم بخواهند کمترین مقدار ممکن گره را داشته باشند، باید به همین شکل جلو بروند. پس اگر بخواهیم به یک فرمول جامع در رابطه با آن برسیم، و اگر تعداد مینیمم گره‌های هر درخت با ارتفاع h را با N(h) نشان بدهیم، پس تعداد گره‌های مورد نیاز برای ساختن یک درخت[6] با ارتفاع h، به فرمول زیر می‌رسیم:

$$N(h) = N(h-1) + N(h-2) + 1$$

که در این فرمول، عملا توضیح همان فرمول بازگشتی‌ای است که در بالا ارائه دادیم و آن بعلاوه یک نیز برای خود گره‌ای است که درحال بررسی زیردرخت‌های چپ و راست آن بودیم. همچنین میدانیم که برای ارتفاع یک، حداقل دو گره نیاز داریم و برای ارتفاع صفر نیز یک گره نیاز داریم. پس این فرمول بازگشتی را برای ارتفاع هفت حل می‌کنیم. برای این کار از کد زیر استفاده می‌کنیم. (میشد از روش‌های دیگر مانند حل معادله همگن و ... هم استفاده کرد که به معدله $x^2 - x - 1 = 0$ می‌رسیدیم و ریشه‌های آن برابر $\phi = \frac{1 \pm \sqrt{5}}{2}$ می‌باشد)

```
def fun(x) -> int:
    if x == 0: return 1
    if x == 1: return 2
    return fun(x-1) + fun(x-2) + 1
fun(7)
```

```
54
```

پس مینیمم تعداد گره‌های مورد نیاز برای ساخت درخت AVL برابر ۵۴ تا گره است.

[5] Binary search tree

[6] ارتفاع درخت برابر با ارتفاع ریشه است

۴- حداقل تعداد گره های داخلی در یک درخت قرمز-سیاه[7] با ارتفاع ۸ چقدر است؟

تعداد مینیمم گره‌های مورد نیاز در درخت قرمز-سیاه، برای داشتن ارتفاع h برابر $n = 2^{\lceil \frac{h}{2} \rceil} - 1$ است. دو تا راه برای آن وجود دارد. هر دو راه را به اختصار توضیح می‌دهیم:

۱. با خواندن منابع مختلف[8]، میدانیم که ماکزیمم ارتفاع یک درخت قرمز-سیاه با n گره برابر از فرمول زیر پیروی می‌کند:

$$h \leq 2\ \log_2(n+1)$$

حالا با استفاده از این فرمول، می‌توانیم مقدار حدی n را بر حسب h پیدا کنیم. پس داریم:

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

$$\xrightarrow{n \text{ عدد صحیح بودن}} 2^{\lceil \frac{h}{2} \rceil} - 1 \leq n$$

۲. یک راه دیگر نیز رسیدن به این مقدار به صورت مستقیم است.[9] برای این کار T را یک درخت قرمز-سیاه در نظر بگیرید که ارتفاع آن برابر با h می‌باشد. سپس با حذف کردن تمام گره‌های برگ آن (که با توجه به تعریف درخت قرمز-سیاه میدانیم که سیاه هستند) یک درخت دیگر مانند T' می‌سازیم. با توجه به اینکه برگ‌هایی که از درخت T حذف کردیم، رنگ‌شان سیاه بود، پس ارتفاع درخت T'، یکی کمتر می‌شود. (یعنی ارتفاع T' برابر با h-1 می‌شود) همچنین با توجه به تعریف درخت‌های قرمز-سیاه میدانم که فرزند هیچ گره قرمزی، قرمز نخواهد بود، بنابراین در تمام مسیرهای بین ریشه تا هر نواده برگی از آن (همه مسیرهای بین ریشه تا برگ‌های مختلف) از درخت، حداقل $\lceil \frac{h}{2} \rceil - 1$ تا گره سیاه وجود دارد. (در غیر اینصورت باید حتما یک گره قرمز، فرزند یک گره قرمز دیگر باشد که با فرض اینکه در درخت قرمز-سیاه، هیچ گره قرمزی فرزند گره قرمز دیگری نیست به تناقض میخوریم) پس با توجه به اینکه در هر مسیر بین ریشه تا هر برگی حداقل $\lceil \frac{h}{2} \rceil - 1$ تا گره سیاه وجود دارد، پس باید یک درخت دودویی کامل با ارتفاع حداقل $\lceil \frac{h}{2} \rceil - 1$ داشته باشیم. پس باید درخت T' ما یک درخت دودویی کامل با ارتفاع $\lceil \frac{h}{2} \rceil - 1$ باشد پس باید درخت T ما حداقل تعداد گره‌هایی حداقل به اندازه یک درخت دودویی با تعداد گره $2^{\lceil \frac{h}{2} \rceil} - 1$ باشد.

پس حداقل تعداد گره‌های مورد نیاز برای ساخت درخت قرمز-سیاه با ارتفاع هشت، برابر **۱۵ تا گره** است.

---

[7] Red-Black tree

[8] کتاب

[9] لینک ویدئو یوتیوب

۵- حداقل تعداد گره های داخلی در یک درخت Splay با ارتفاع ۹ چقدر است؟

حداقل تعداد گره‌های داخلی برای ساخت درخت Splay با ارتفاع نه، برابر با نه گره است. درخت Splay، یک درخت دودویی جستجو است که در آن این مزیت وجود دارد که دسترسی به عنصرهایی که اخیرا مورد دسترسی قرارگرفته‌اند سریع‌تر است. یعنی در این درخت (همانطور که در سوال دو به آن اشاره شد) ما یک درخت دودویی جستجو میسازیم و تنها تفاوت این است که بعد از جستجوی یک عنصر، در آن درخت تغییراتی صورت می‌دهیم. اما اگر قرار فقط ساختن درخت Splay باشد، همانطور که در سوال دو دیدم، می‌تواند ارتفاع برابر با تعداد گره‌ها شود. پس مینیمم تعداد گره مورد نیاز برای ساختن درخت Splay با ارتفاع دلخواه، برابر با همان ارتفاع می‌باشد. پس مینیمم تعداد گره مورد نیاز برای ساختن دخت Splay با ارتفاع نه، برابر با **نه تا گره** است.

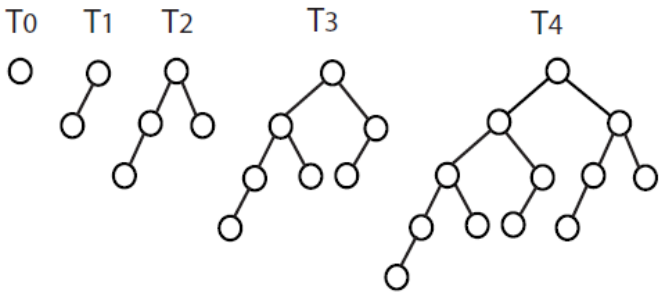
۶- با استفاده از روش استقرا نشان دهید که حداقل تعداد گره‌های موجود در درخت AVL با ارتفاع h، با استفاده از رابطه زیر تعریف می‌شود:

$$\forall h \geq 1: n_h = F_{h+2} - 1$$

که در آن $F_k$ عدد فیبوناچی از مرتبه $k$ است.

اثبات مشابه با مطلب بیان شده در سوال سه است. برای استفاده از استقرا ابتدا باید فرض استقرا و سپس مرحله استقرا را مشخص کنیم.

- **فرض استقرا:** برای ارتفاع صفر، میدانیم این کار با یک گره هم قابل انجام است. همچنین برای ارتفاع یک نیز میدانیم که این کار، با دو گره قابل انجام است. همچنین میدانیم برای ارتفاع دو، این کار با چهار گره قابل انجام است. (درخت‌های مانند شکل زیر هستند که به آن‌ها درخت‌های خلوت[10] نیز می‌گویند[11])




- گام استقرا: در اینجا فرض می‌کنیم که این کار برای ارتفاع‌های *k-1* و *k-2* قابل انجام است. حالا با استفاده از این موضوع نشان می‌دهیم که این کار برای ارتفاع $k$ هم قابل انجام است. اثبات این کار از روی رابطه مشخص است. برای این کار ابتدا میایم وگره ریشه را قرار می‌دهیم و میدانیم که ارتفاع آن باید برابر با $k$ شود. سپس زیردرخت مربوط به هر کدام از فرزندانن آن را مشخص می‌کنیم. (میدانیم که ارتفاع ریشه یکی بیشتر از ماکزیمم ارتفاع بین زیردرخت‌های سمت راست و چپ است) با توجه به اینکه میدانیم حداکثر اختلاف ارتفاع بین دو زیر درخت یک گره در درخت avl، برابر با یکی است و ارتفاع گره یکی بیشتر از ماکزیمم مقدار بین این دو ارتفاع است، پس ارتفاع زیر درخت‌های آن را برای استفاده از کمترین گره، متفاوت می‌گذاریم به طوریکه یکی بیشتر از آن یکی باشد (باید تنها یک واحد از دیگری بیشتر باشد وگرنه شرط avl بودنت درخت نقض می‌شود). پس باید ارتفاع زیردرخت‌های ریشه برابر *k-1* و *k-2* باشد. سپس این دو زیردرخت را با استفاده از گره ریشه به هم میچسبانیم و کمترین تعداد گره‌های مورد نیاز برای این کار را هم میدانیم چون فرض کردین تعداد مینیمم گره مورد نیاز برای ساختن درخت‌هایی با ارتفاع *k-1* و *k-2* را داریم. پس درخت حاصل ارتفاع آن یکی بیشتر از ارتفاع زیردرخت‌هایش است که با توجه به اینکه ارتفاعه زیردرخت‌هایش برابر *k-1* و *k-2* می‌باشد، پس ارتفاعه درخت حاصل برابر $\max\{k-1, k-2\} + 1 = k - 1 + 1 =$ $k$ می‌شود. از طرفی تعداد گره‌های آن نیز برابر با تعداد گره‌های مورد نیاز برای ساختن زیردرخت‌هایش بعلاوه یک است. (خود ریشه که زیردرخت‌ها را به هم می‌چسباند است) پس داریم:

$$n_k = n_{k-1} - n_{k-2} + 1$$

حالا با توجه به تعریف تابع فیبوناچی که می‌دانیم برابر $F_n = F_{n-1} + F_{n-2}$ است، این دو عبارت خیلی شبیه یکدیگر هستند. اگر جملات صفرم و اول فیبوناچی را به ترتیب برابر با یک و یک در نظر بگیریم، پس با کنار هم نوشتن این دو دنباله خواهیم داشت:

[10] sparse
[11] لینک عکس

| شماره سری $(n)$ | ۰ | ۱ | ۲ | ۳ | ۴ | ۵ | ۶ | ۷ |
|---|---|---|---|---|---|---|---|---|
| فیبوناچی - $F_n$ | ۱ | ۱ | ۲ | ۳ | ۵ | ۸ | ۱۳ | ۲۱ |
| مینیمم تعداد گره $N_n$ | ۱ | ۲ | ۴ | ۷ | ۱۲ | ۲۰ | ۳۳ | ۵۴ |

با توجه به اعداد دنباله فیبوناچی و اعداد مربوط به کمترین تعداد گره با ارتفاع $h$، بدیهی است که تعداد می‌توان فرمول بالا را به صورت زیر نوشت:

$$n_h = F_{h+2} - 1$$

۷- نشان دهید در درخت قرمز- سیاه، نیازی به استفده از متغیر Boolean برای مشخص کردن رنگ گره ها(قرمز یا سیاه) نیست . یک روش برای پیاده سازی درخت قرمز-سیاه، بدون افزودن فضای اضافی برای ذخیره رنگ ها توصیف کنید.

برای اینکار، چندین روش وجود. چندتا از آن‌ها را به اختصار توضیح می‌دهیم:

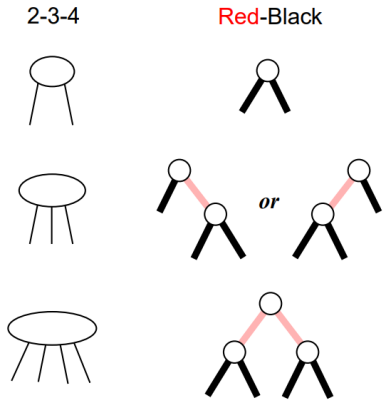
- می‌توانیم به جای استفاده از بیت رنگ، آن را به صورت دیگری بیان کنیم. به این شکل که به ازای اشاره‌گر[۱۲] به هر گره قرمز، مقدار کم‌ارزش‌ترین بیت[۱۳] آن را یکی زیاد کنیم. در این حالت هر سری با توجه به اشاره‌گر موجود به هر گره، می‌توانیم از روی کم‌ارزش‌ترین بیت آن ببینیم که مقدار رنگ آن قرمز بوده است یا سیاه بوده است.

- یک راه دیگر نیز این است که بیایم و اگر مقدار گره‌های ما، در محدوده (رِنج) خاصی قرار دارند، در صورت قرمز بودن رنگ هر گره، مقدار آن را به نوعی تغییر دهیم که هم بتوانیم مقدار آن گره را بعدا به درستی تشخیص دهیم و اینکه متوجه شویم رنگ آن گره، قرمز بوده است.

برای مثال فرض کنید که ورودی‌های ما، اعداد بزرگتر از صفر هستند، بیایم و در صورت قرمز بودن رنگ هر گره، منفی همان مقدار را در آن گره قرار دهیم. در این صورت وقتی به گره‌ای با مقدار منفی برخورد کنیم، میفهمیم که رنگ آن گره قبلا قرمز بوده است و همچنین مقدار آن گره را هم می‌دانیم که برابر با منفی مقدار فعلی موجود در آن است.

این راه، مستلزم به دانستن محدوده ورودی‌ها و داشتن یک نگاشت برای تبدیل آن‌ها می‌باشد که در خیلی از موارد ممکن نیست یا داده‌های ما اصلا به صورت عددی نیستند یا مثلا اعداد اعشاری شامل اعداد منفی هستند که خب در این صورت کار خیلی سخت می‌شود. (کار پیدا کردن نگاشتی برای این اعداد) ولی خب برای بعضی از حالات به راحتی جواب می‌دهد و خوب عمل می‌کند.

- حالت دیگر نیز این است که بیایم و درخت قرمز سیاه را به درخت دو-سه-چهار[۱۴] تبدیل کنیم. به طوریکه اگر گره‌ای شامل دو فرزند سیاه است، یکسان باقی بماند و تنها همین دو تا گره فرزند را داشته باشد(همین گره دو بماند)؛ اما اگر گره‌ای شامل فرزند قرمز است، مقدار آن را هم نگه‌دارد و همچنین دو فرزند گره قرمز را هم به عنوان فرزند خودش نگه‌دارد (تبدیل به گره سه شود) و در آخر اگر هم گره‌ای شامل دو گره قرمز باشد، تبدیل به گره چهار شود به طوریکه مقار آن‌ها را نگه دارد همچنین فرزندان این گره‌های قرمز را هم به عنوان فرزند خودش نگه دارد (تبدیل به گره چهار شود). شکل زیر این تبدیلات را نشان می‌دهد:




یال‌های قرمز به این معنی هستند که رنگ گره پایین‌تر، قرمز است. این یک نوع نمایش دیگر درخت‌های قرمز-سیاه است.[۱۵]

[12] pointer
[13] Least significant bit
[14] 2-3-4 tree
[15] لینک منبع

همچنین میدانیم که ممکن نیست دیگر فرزندان گره‌های قرمز هم قرمز باشد، پس می‌توانیم به راحتی این کار را انجام دهیم و این باعث می‌شود که هیچ دو گره قرمزی فرزند یکدیگر نباشند برای همین دیگر هیچ گره‌ای با رنگ قرمز باقی نمی‌ماند و برای همین به مشکل نمی‌خوریم که بعد از ازبین بردن گره قرمز و تبدیل گره پدر آن به گره سه یا چهار، گره بعدی هم قرمز باشد.
پس با این حساب تمام گره‌های قرمز از بین می‌روند؛ اما درخت دودویی، تبدیل به درخت دو-سه-چهار می‌شود.