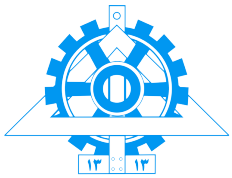
باسمه تعالی

دانشگاه تهران - دانشکده مهندسی برق و کامپیوتر



ساختمان‌داده‌ها و الگوریتم‌ها

تمرین سوم - درخت

محمدصادق ابوفاضلی، حامد میرامیرخانی

تاریخ تحویل: ۱۴۰۲/۹/۴

۱. مسیر درختی — ۱۰ نمره

یک درخت دودویی کامل با n گره و به ارتفاع log(n) داریم. هر گره از این درخت به گره های فرزند و گره پدر خود دسترسی دارد. میخواهیم مسیر ساده‌ای بین دو راس u و v پیدا کنیم (کوتاهترین مسیر بین این دو راس).

الف) الگوریتم بهینه‌ای برای این کار ارائه دهید.

ب) مرتبه زمانی الگوریتم خود را بدست آورید.

پاسخ :

الف)

برای پیدا کردن مسیر بین u و v ، ابتدا پایین ترین جد مشترک این دو گره را بدست می آوریم.

سپس مسیر بین این دو گره به این صورت است: از u تا پایین ترین جد مشترک بعلاوه پایین ترین جد مشترک تا v

برای پیدا کردن پایین ترین جد مشترک، مراحل زیر را طی میکنیم:

۱- پیدا کردن ارتفاع دو گره:

برای این کار کافی است بشماریم چند بار باید از آن گره پدر گرفته شود تا به ریشه برسیم.

۲- هم ارتفاع کردن دو گره:

اگر یکی از گره ها دارای ارتفاع بیشتری است، از گره دارای ارتفاع بیشتر، به اندازه اختلاف ارتفاع دو گره، پدر میگیریم.

۳-حرکت به سمت پایین‌ترین جد مشترک:

حال که به دو گره هم ارتفاع رسیده ایم، چک میکنیم آیا آن دو گره برابرند یا خیر، و اگر برابر نبودند، آنقدر از هر دو پدر میگیریم تا در یک گره برابر شوند. آن گره، پایین ترین جد مشترک u و v است (میدانیم این اتفاق می افتد زیرا دو راس هم ارتفاع اند و حداکثر در ریشه به یکدیگر میرسند و برابر میشوند)

در طول انجام مراحل ۲ و ۳، مسیر حرکت از هر دو گره را ذخیره میکنیم تا در نهایت با در کنار هم قرار دادن آنها به مسیر بین u و v برسیم.

ب)

پیدا کردن ارتفاع دو گره از مرتبه ارتفاع درخت است پس در O(log(n)) انجام میشود.

هم ارتفاع کردن دو گره و همچنین رسیدن به پایین ترین جد مشترک نیز از مرتبه ارتفاع درخت، یعنی O(log(n)) است.

در نتیجه مرتبه زمانی الگوریتم : O(log(n))

۲. درخت AVL
۱۵ نمره

در رابطه با درخت AVL به سوالات زیر پاسخ دهید:

الف) حداقل تعداد رئوس یک درخت AVL را برحسب ارتفاع آن بدست بیاورید و ادعای خود را ثابت کنید.

ب) نشان دهید پیچیدگی زمانی عمل insert در یک درخت AVL که دارای n گره است، از مرتبه ((O(log(n است.

پاسخ :

الف)

از خاصیت درخت AVL استفاده میکنیم که حداکثر اختلاف ارتفاع زیر درخت چپ و راست یک واحد است. بنابراین، حداقل ارتفاع ممکن برای یک درخت AVL به صورت h-۱ و h-۲ است. بنابراین، تعداد کمینه رئوس برای درخت با ارتفاع h برابر است با جمع تعداد کمینه رئوس برای زیردرخت چپ، زیر درخت راست و ریشه. که به صورت زیر است:

$$min(h) = min(h-۱) + min(h-۲) + ۱$$

با شروع از شرایط اولیه یعنی $min(۱) = ۱$ و $min(۲) = ۲$ ، می‌توانیم مقادیر $min(h)$ را برای ارتفاع‌های بیشتر به دست آوریم.
با تغییر متغیر و تابع به صورت $f(h) = min(h) + ۱$ داریم:

$$f(h) = f(h-۱) + f(h-۲)$$

$$f(۱) = ۲, f(۳) = ۳$$

که در واقع به دنباله فیبوناچی رسیدیم. بنابراین:

$$min(h) = \frac{\varphi^{h+۲} - (-\varphi)^{h+۲}}{\sqrt{۵}} - ۱$$

ب)

میدانیم عملیات insert در درخت AVL همانند insert در یک درخت جستجوی دودویی (BST) است بعلاوه چند عمل Rotation که بعد از آن انجام می‌شود . در زمان Rotation تنها چند پوینتر در O(۱) جابجا میشوند پس میتوان از زمان آن صرف نظر کرد.

از آنجایی که پیچیدگی زمانی insert در درخت AVL همانند یک BST است، پس در O(h) انجام میشود. و از آنجایی که درخت AVL یک Balanced-Tree است، میدانیم که در آن h=log(n) می‌باشد پس در نهایت برای پیچیدگی زمانی insert در درخت AVL به مرتبه ((O(log(n می‌رسیم.

۳. درخت هافمن
۱۵ نمره

جدول زیر را در نظر بگیرید:

الف) با توجه به جدول تکرار حروف، درخت هافمن را رسم کنید و کد بهینه هافمن را برای حروف بدست آورید.

ب) با توجه به درخت رسم شده، برای تعداد تکرار حرف D یک بازه ارائه دهید که اگر در این بازه باشد، شکل درخت و کد هافمن حروف تغییر نکند. (برای جواب خود استدلال بیاورید)

ج) تعداد تکرار حرف C را حداقل چند واحد باید افزایش دهید تا یک واحد به ارتفاع درخت هافمن فعلی اضافه شود؟

حرف	تعداد تکرار
A	۱۸
B	۵
C	۴۳
D	۷
E	۴
F	۳۹
G	۲
H	۱۰

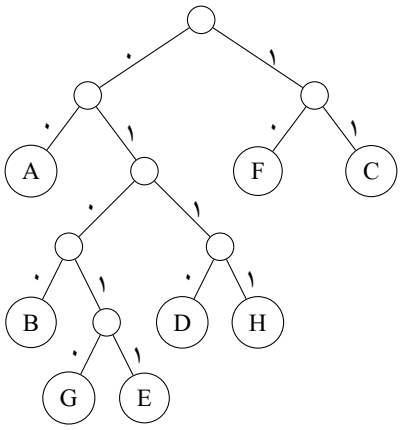
پاسخ:

الف)

ابتدا دو حرف با کمترین تعداد تکرار یعنی (G) و (E) ادغام می‌شوند که گره حاصل از ادغام آنها به مجموع تکرار ۶ می‌رسد. سپس گره حاصل، با حرف (B) ادغام شده و مجموع تکرار در آن به ۱۱ می‌رسد. دو حرف (D) و (H) هر دو دارای تکرار کمتر از ۱۱ هستند پس در این مرحله با هم ادغام شده و گره‌ای با مجموع تکرار حروف ۱۷ را می‌سازند. گره حاصل از ادغام (G, E, B) و گره حاصل از ادغام (D, H) به ترتیب دارای مجموع تکرار ۱۱ و ۱۷ هستند که از تعداد تکرار حرف (A) کمتر است. پس در این مرحله این دو گره ادغام شده و گره‌ای با مجموع تکرار ۲۸ حاصل می‌شود. سپس این گره با حرف (A) ادغام شده و مجموع تعداد تکرار حروف در گره بدست آمده به ۴۶ می‌رسد.

در مرحله بعدی دو حرف (F) و (C) که دارای تعداد تکرار کمتر از ۴۶ هستند ادغام شده و گره‌ای با مجموع تکرار ۸۲ می‌سازند. در نهایت این گره بدست آمده با گره حاصل از ادغام (G, E, B, D, H, A) ادغام می‌شود و درخت نهایی تشکیل می‌شود.

درخت نهایی:



کد هافمن حروف:

$$A = ۰۰$$

$$B = ۰۱۰۰$$

$$C = ۱۱$$

$$D = ۰۱۱۰$$

$$E = ۰۱۰۱۱$$

$$F = ۱۰$$

$$G = ۰۱۰۱۰$$

ب)

اگر (D) مقداری کمتر از ۶ داشت (مقادیر ۵ و کمتر از ۵)، بجای اینکه (B) با گره حاصل از ادغام (G, E) ادغام شود، با (D) ادغام می‌شد و شکل درخت و در نتیجه کد هافمن حروف تغییر می‌کرد.

از طرف دیگر، اگر (D) مقداری بالای ۸ داشت (مقادیر ۹ و بیشتر از ۹)، ادغام آن با (H) دارای مجموع تکرار بیشتر از ۱۸ می‌شد و گره حاصل از ادغام (G, E, B) بجای اینکه با گره حاصل از ادغام (D, H) ادغام شود، با گره (A) ادغام می‌شد.

در نتیجه تعداد تکرار D باید بین ۶ تا ۸ باشد.

$$۶ \leqslant D \leqslant ۸$$

ج)

گره حاصل از ادغام (G, E, B, D, H, A) را گره (K) می‌نامیم. این گره دارای مجموع تکرار حروف ۴۶ است که از تعداد تکرار حرف $(F = ۳۹)$ و $(C = ۴۳)$ بیشتر است که باعث می‌شود ابتدا (F) با (C) ادغام شود و سپس گره حاصل از ادغام آنها با گره (K) ادغام شود. برای اینکه یک واحد به ارتفاع درخت هافمن اضافه شود، باید کاری کنیم که یک حرف از بین (F) یا (C) با گره (K) ادغام شود و سپس گره حاصل با حرف باقی مانده ادغام شود. اینگونه ارتفاع درخت یک واحد بیشتر می‌شود. از طرفی باید این کار را صرفا با تغییر تکرار حرف (C) انجام دهیم، پس باید تعداد تکرار این حرف را تا حدی زیاد کنیم که از مجموع تعداد تکرار حروف گره (K) بیشتر شود. در این صورت است که گره (K) ابتدا با گره (F) ادغام شده و سپس گره بدست آمده با گره (C) ادغام می‌شود.

برای اینکه تعداد تکرار (C) که ۴۳ است، از مجموع تعداد تکرار حروف گره (K) که برابر با ۴۶ است بیشتر شود، باید حداقل ۴ واحد آن را افزایش دهیم. پس جواب این بخش ۴ است.

۴. میانه‌ی لحظه‌ای — ۱۵ نمره

می‌خواهیم دنباله‌ای از اعداد را در حافظه‌ای نگه داریم. در ابتدا حافظه خالی است. در هر مرحله یک عدد از ورودی خوانده می‌شود و ما می‌خواهیم پس از insert کردن هر عدد در حافظه، میانه تمام اعداد موجود در حافظه (شامل عدد insert شده) را با صرف کمترین هزینه برگردانیم. از چه داده ساختاری (داده ساختارهایی) استفاده کنیم؟ الگوریتمی ارائه دهید که این کار را انجام دهد. مرتبه زمانی الگوریتم خود را بدست آورید.

پاسخ :

از داده ساختار Min-Heap و Max-Heap استفاده میکنیم.

یک متغیر به نام mid برای نگهداری میانه تعریف می‌کنیم.

از یک Min-Heap برای نگهداری اعداد بزرگتر از میانه استفاده میکنیم.

و از یک Max-Heap برای نگهداری اعداد کوچکتر از میانه استفاده میکنیم.

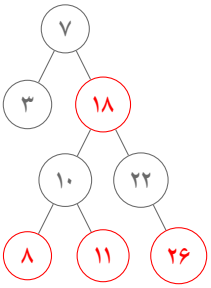
در ابتدا اولین عدد را در mid ذخیره کرده و همان را بعنوان میانه خروجی میدهیم.

در مراحل بعدی، عدد وارد شده را با mid مقایسه کرده و اگر از mid بزرگتر بود آن را در Min-Heap و اگر کوچکتر بود آن را در Max-Heap درج می‌کنیم. سپس تعداد اعضای این دو Heap را مقایسه کرده و اگر تعداد آنها متوازن نبود، mid را در Heap با تعداد کمتر push کرده، سپس از Heap با اعضای بیشتر pop کرده و آن را در mid قرار میدهیم تا توازن حذف شود. در نهایت mid را خروجی میدهیم.

این الگوریتم بدلیل استفاده از Heap در مرتبه زمانی O(log(n)) انجام پذیر است.

۵. درخت قرمز-سیاه
۱۰ نمره

درخت Red-Black زیر را در نظر بگیرید.

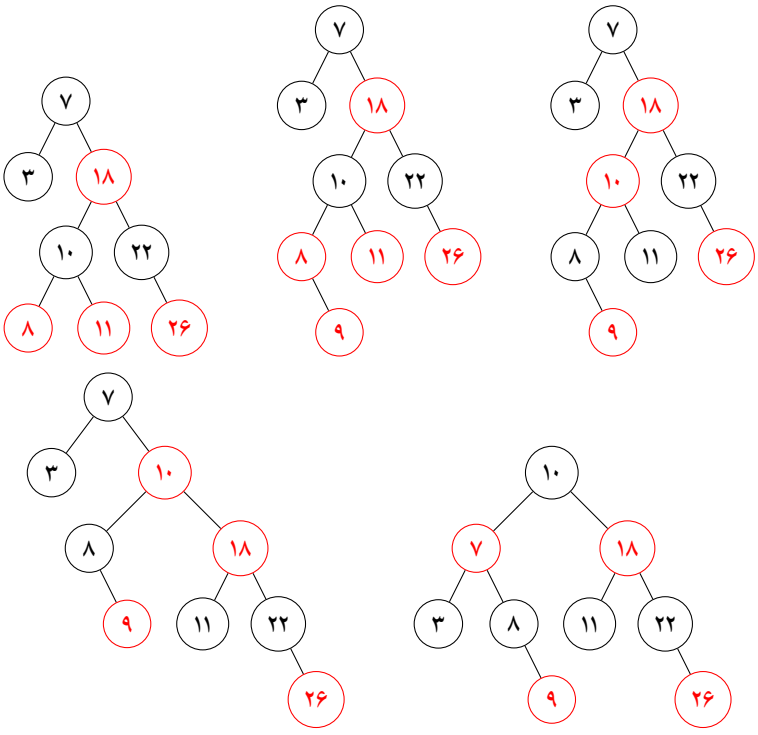


الف) راس ۹ را به درخت اضافه کنید.

ب) ثابت کنید در درخت Red-Black به ازای هر راس v مقدار ارتفاع آن راس حداکثر می‌تواند ۲ برابر فاصله ی آن راس تا نزدیکترین برگ خود باشد.

پاسخ :

الف) مراحل اضافه شدن گره ۹ مطابق زیر است:



ب) میدانیم $bh(v) \geqslant \frac{h(v)}{۲}$ و چون در مسیر v تا برگ هیچ دو راس قرمز متوالی نداریم، پس حتماً حداقل نصف رئوس، سیاه هستند. همچنین اگر $d(v)$ طول مسیر v تا نزدیکترین برگ را در نظر بگیریم، میدانیم $bh(v) \leqslant d(v)$ زیرا در این مسیر حداکثر $d(v)$ تا راس سیاه داریم. بنابراین:

$$\frac{h(v)}{۲} \leqslant bh(v) \leqslant d(v) \Rightarrow h(v) \leqslant ۲ \times d(v)$$

۶. کاغذ نازک ۲۰ نمره

در تمرین درس ساختمان داده، صادق جواب یک سوال که ترسیم یک درخت دودویی با شرایطی خاص است را بلد نیست. بنابراین به سمت برگه تمرین حامد می‌رود تا این سوال را از روی او بنویسد. حامد که می‌خواهد صادق را اذیت کند، برگه خود را از روی میز برمی‌دارد تا صادق نتواند روی برگه را ببیند. اما از آنجایی که برگه تمرین حامد نازک بوده، صادق تصویر درختی را از پشت برگه حامد تشخیص می‌دهد. درختی که صادق از پشت برگه حامد دیده، همراه با مقادیر روی هر گره بوده است.

الف) الگوریتمی با پیچیدگی زمانی بهینه ارائه دهید که درخت دیده شده توسط صادق را ورودی گرفته و بدون دانستن صورت سوال، آن را به درخت مطلوب سوال تبدیل کند. شبه‌کد الگوریتم خود را بنویسید.

ب) در سوال بعدی از صادق خواسته شده تا سنگین‌ترین مسیر بین دو راس در آن درخت را پیدا کند. وزن مسیر بین دو راس درخت را برابر با مجموع وزن رئوس روی آن مسیر تعریف می‌کنیم. الگوریتمی ارائه دهید که در زمان خطی سنگین‌ترین مسیر را پیدا کند. (وزن رئوس می‌تواند عددی منفی باشد)

پاسخ :

الف)

تصویر درختی که صادق دیده است، در واقع آینه شده‌ی درخت اصلی مطلوب سوال بوده است. پس برای حل این سوال کافیست الگوریتمی ارائه دهیم که درختی دودویی را آینه کند:

برای این کار باید فرزند چپ و راست تمام گره‌ها با هم جابجا شوند. ایده این است که درخت را به صورت بازگشتی پیمایش کنیم و پس از پیمایش زیر درختان، زیر درخت راست و چپ را جابجا کنیم:

```
mirror(node) {
    if (node == NULL)
        return
    else {
        mirror(node.left)
        mirror(node.right)

        temp = node.left
        node.left = node.right
        node.right = temp
    }
}
```

ب)

ابتدا مقادیر S ، T و R را برای هر راس v مطابق زیر تعریف میکنیم:

T(v) = سنگین ترین مسیری که از v به سمت پایین گراف حرکت میکند (درجه v در هر مسیر حداکثر ۱ باشد)

S(v) = وزن مسیر T(v)

R(v) = سنگین ترین مسیر در زیردرخت با ریشه v

مراحل الگوریتم:

از پایین ترین طبقه درخت شروع به محاسبه S ، T و R میکنیم. S ، T و R هر راس به کمک S و T فرزندانش (که در صورت وجود از قبل محاسبه شده اند) به دست می آیند.

T هر راس یا برابر با خود آن راس و یا برابر با مسیری است که از اتصال آن راس با T یکی از فرزندانش بدست می آید به این صورت که اگر S هر دو فرزندش منفی باشد برابر خود آن راس و در غیر این صورت از اتصال این راس و T فرزندی که که S آن بیشتر است بدست می آید.

S هر راس با توجه به T آن، یا برابر با وزن خود آن راس و یا برابر با مجموع وزن آن راس با S یکی از فرزندانش میباشد.

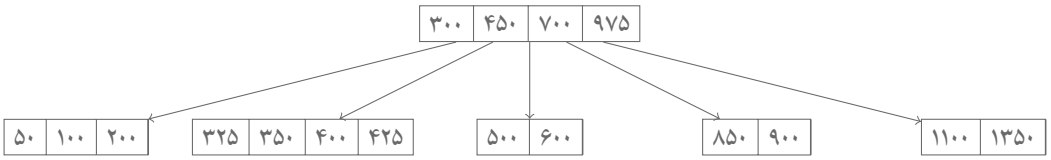
R هر راس یا برابر با خود آن راس و یا برابر مسیری است که از اتصال R با T یک یا هر دو فرزندش بدست می آید. به این صورت عمل می‌کنیم که ابتدا R را برابر خود راس در نظر می‌گیریم و سپس هر فرزندی که S آن مثبت بود آن فرزند را به مسیر R اضافه میکنیم.

در آخر نیز سنگین ترین مسیر برابر با سنگین ترین R میباشد. چرا که اگر بالاترین راس سنگین ترین مسیر را در نظر بگیریم وزن R آن راس بیشتر مساوی وزن این مسیر (سنگین‌ترین مسیر) است و با توجه به اینکه سنگین ترین مسیر یکتاست R آن راس باید برابر با همین مسیر باشد.

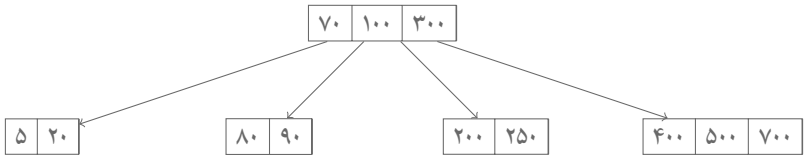
۷. درخت B ۱۵ نمره

برای تمام B-Tree های زیر، کمینه تعداد کلید هر گره را ۲ و بیشترین تعداد کلید ممکن کلید برای هر گره را ۵ در نظر بگیرید.

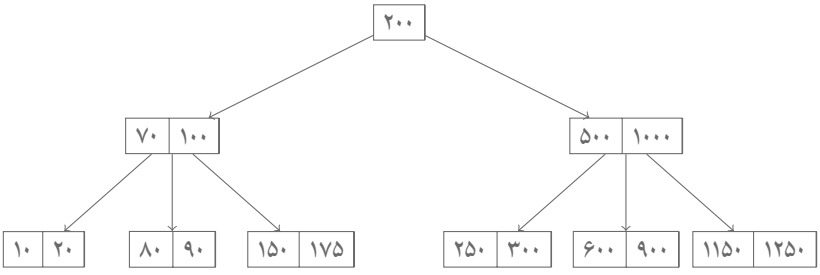
الف) کلید ۳۷۵ را به درخت زیر اضافه کنید، مراحل درج و متوازن سازی را به طور کامل توضیح دهید.



ب) کلید ۲۵۰ را از درخت زیر حذف کنید، مراحل حذف و متوازن سازی را بطور کامل توضیح دهید.



ج) کلید ۱۰۰ را از درخت زیر حذف کنید، مراحل حذف و متوازن سازی را به طور کامل توضیح دهید.



پاسخ:

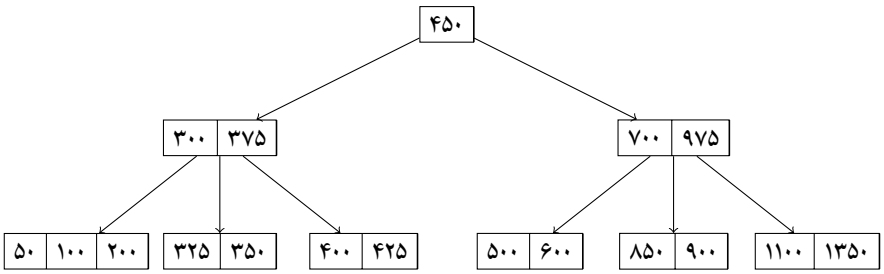
الف)

کلید ۳۷۵ درون گره برگی که دارای کلیدهای ۳۲۵، ۳۵۰، ۴۰۰ و ۴۲۵ است درج می‌شود (بین کلید ۳۵۰ و ۴۰۰).

در صورتی که محدودیت بیشینه تعداد کلید هر گره ۴ باشد، حل این سوال به صورت زیر انجام می‌شود:

کلید ۳۷۵ درون گره برگی که دارای کلیدهای ۳۲۵، ۳۵۰، ۴۰۰ و ۴۲۵ است درج می‌شود (بین کلید ۳۵۰ و ۴۰۰) ولی از آنجایی که گره‌ها نمی‌توانند بیش از ۴ کلید را نگه دارند، بنابراین این گره شکسته می‌شود و کلید میانی آن یعنی ۳۷۵ به سطح بالاتر یعنی گره پدر می‌رود. با این کار گره پدر نیز دارای بیش از ۴ کلید می‌شود. پس این گره را نیز می‌شکنیم و کلید میانی آن یعنی ۴۵۰ را به سطح بالاتر میبریم. از آنجایی که گره‌ای در سطح بالاتر وجود ندارد، ۴۵۰ به تنهایی بعنوان ریشه جدید درخت قرار میگیرد. (در B-Tree ریشه درخت استثناء است و می‌تواند کمتر از مقدار کمینه کلید داشته باشد)

شکل نهایی:



ب)

وقتی کلید ۲۵۰ را از برگ مربوطه حذف می‌کنیم، گره برگ دیگر دارای کمینه تعداد کلید مجاز برای گره‌ها نیست. پس باید این موضوع را حل کنیم.

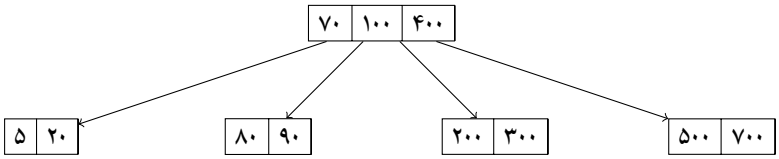
در نگاه اول به دنبال این می‌گردیم که آیا میتوان از یکی از برادرانش یک کلید گرفت و به این گره اضافه کرد یا خیر؟

به سراغ برادر سمت چپ می‌رویم و مشاهده میکنیم که این کار امکانپذیر نیست زیرا برادر سمت چپ دارای کمینه تعداد کلید مجاز برای گره است و نمیتواند کلید بیشتری از دست دهد.

حال به سراغ برادر سمت راست می‌رویم. میتوانیم یک کلید از این گره بگیریم زیرا ۳ کلید دارد که یکی از کمینه تعداد کلید مجاز برای گره‌ها بیشتر است. پس تصمیم میگیریم چپ‌ترین کلید برادر سمت راست به گره مورد نظرمان انتقال دهیم، یعنی کلید ۴۰۰ را.

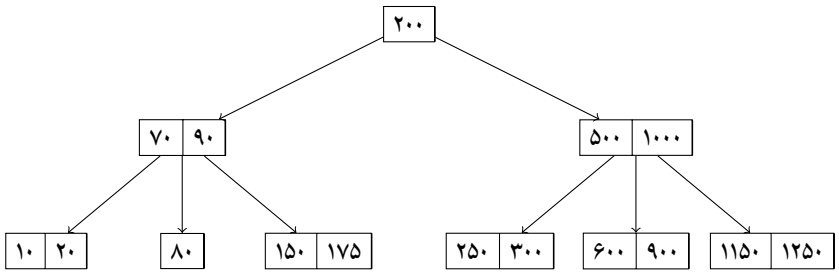
اما ۴۰۰ نمیتواند مستقیما انتقال یابد، زیرا ترتیب حاکم بر درخت B را بهم میریزد (۴۰۰ از ۳۰۰ بزرگتر است پس نمیتواند در زیردرخت سمت چپ آن حضور یابد). پس ۴۰۰ را به گره پدر انتقال میدهیم و از گره پدر کلید ۳۰۰ را به گره فرزندی که عمل حذف روی آن صورت گرفته منتقل می‌کنیم.

شکل نهایی:

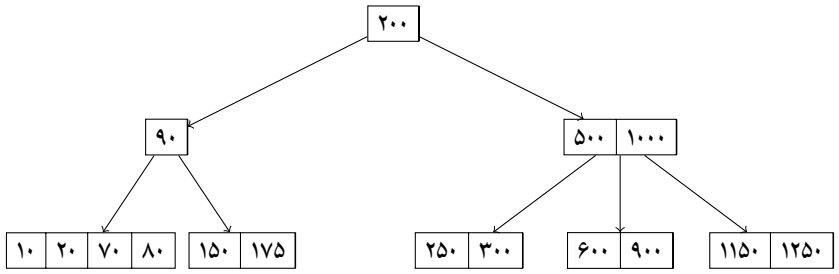


ج)

کلید ۱۰۰ در یک نود میانی واقع شده است پس برای حذف آن ابتدا جای ۹۰ و ۱۰۰ را عوض کرده و سپس ۱۰۰ را که حالا در برگ قرار گرفته حذف میکنیم. در واقع بزرگترین کلید زیردرخت چپ را پاک میکنیم و آنرا بجای ۱۰۰ مینویسیم.



این مساله باعث میشود که برگ فقط دارای یک کلید (۸۰) باشد که از حداقل تعداد کلید برای یک نود کمتر است. پس باید از یکی از برادرانش یک کلید بگیرد اما اینکار ممکن نیست زیرا هم برادر چپ و هم برادر راست دارای حداقل تعداد کلید برای یک نود هستند و نمیتوان کلیدی از آنها جدا کرد. پس برای حل مشکل مجبوریم نود دارای کلید ۸۰ را با یکی از برادرانش merge کنیم. آنرا با نود دارای کلیدهای ۱۰ و ۲۰ merge میکنیم و همچنین باید کلید ۷۰ را از نود پدر به سطح پایینتر بیاوریم. زیرا ۷۰ این دو برادر را از یکدیگر جدا میکرد اما الان این دو merge شده اند. به شکل زیر میرسیم:



بعلت انتقال کلید ۷۰ به سطح پایین، گره پدر یک کلیده شد. پس سعی میکنیم این مشکل را حل کنیم. این نود، برادر سمت چپ ندارد و از برادر سمت راست هم نمیتوان کلید جدا کرد و به آن اضافه کرد زیرا برادر سمت راست دارای حداقل تعداد کلید است. پس مانند فرآیند بالا، این دو نود را merge میکنیم و به شکل نهایی میرسیم: