دانشکده علوم ریاضی و آمار

مدرس: دکتر مجتبی رفیعی نیمسال اول ۱۴۰۰-۱۴۰۱

ساختمان داده‌ها و الگوریتم‌ها

جلسه ۲۹: ساختمان داده و الگوریتم‌ها

نگارنده: مریم دهقان

۳ آذر ۱۴۰۰

فهرست مطالب

۱ ساخت درخت دودویی یکتا به کمک پیمایش‌های عمقی ۱
۲ پیاده‌سازی درخت k تایی ۲
۳ تبدیل یک درخت k تایی به درخت دودویی معادل ۳
۱.۳ رابطه بین پیمایش درخت k تایی و درخت دودویی معادل ۴

۱ ساخت درخت دودویی یکتا به کمک پیمایش‌های عمقی

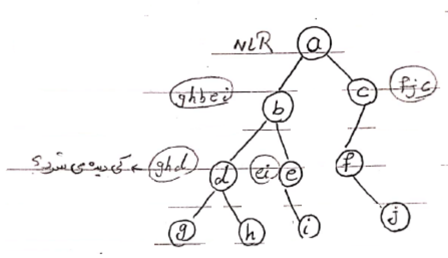
نکته تکمیلی ۱: با داشتن پیمایش‌های $Inorder$ و $Preorder$ با یک درخت دودویی می‌توان درخت یکتا را رسم کرد.
نکته تکمیلی ۲: با داشتن پیمایش‌های $Inorder$ و $Postorder$ برای یک درخت دودویی می‌توان درخت یکتا را رسم کرد.

علت داشتن درخت یکتا با پیمایش اخیر آن است که پیمایش $Inorder$ باعث می‌شود تفکیک بین گره‌های فرزند نیز لحاظ شود در حالیکه دیگر پیمایش‌ها تنها بر مکان ملاقات ریشه نسبت به سایر فرزندان تاکید داشت.

مثال ۱: برای درخت دودویی

$LNR \qquad Inorder(T) = g\ d\ h\ b\ e\ i\ a\ f\ j\ c$

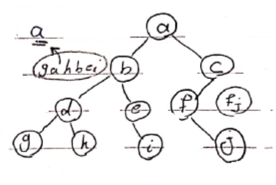
$NLR \qquad Preorder(T) = a\ b\ d\ g\ h\ e\ i\ c\ f\ j$ ریشه a است.



مثال ۲: برای درخت دودویی

$Inorder(T) = g\ d\ h\ b\ e\ i\ a\ f\ j\ c \qquad LNR$

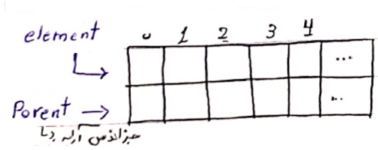
$Postorder(T) = g\ h\ d\ i\ e\ b\ j\ f\ c\ a \qquad NLR$ ریشه a است.



جمع‌بندی: درخت حاصل از مثال ۱ و ۲ به صورت یکسانی ترسیم شده.

۲ پیاده‌سازی درخت k تایی

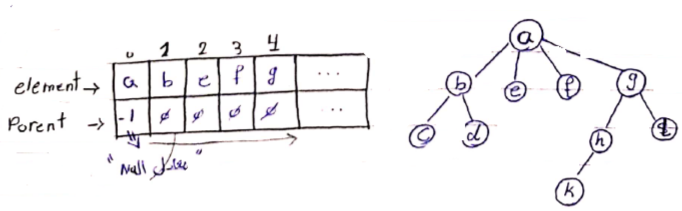
راه‌حل ۱: می‌توان از یک آرایه دوبعدی به صورت زیر استفاده کرد.



به تعداد k تا عنصر در نظر بگیریم عنصر با اندیس صفر ریشه قرارگرفته است.

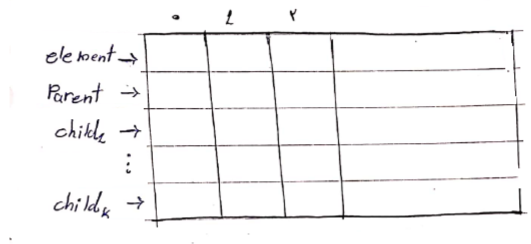
درخت ما برچسب‌دار مرتب

مثال:

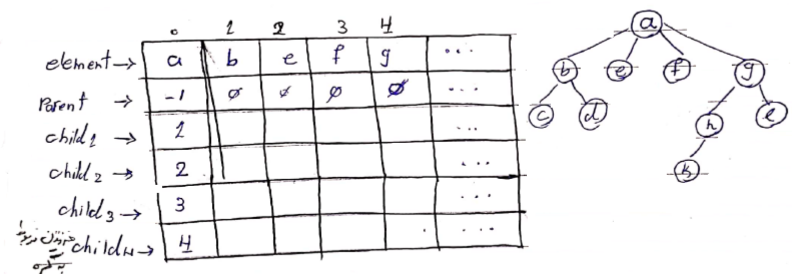


نکته: ترتیب فرزندان از اندیس کوچکتر به بزرگتر لحاظ شده است. بنابراین از روی نمایش بالا می‌توان یک درخت ترسیم کرد.

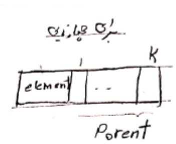
مشکلات راه‌حل ۱: ۱- دسترسی به فرزندان به سادگی امکان‌پذیر نیست و علنا باید عنصرهای آرایه را پیدایش کنیم.
۲- در صورت حذف و اضافه عنصرها در آرایه، ترتیب زیر درخت‌ها باید مدیریت شود که کار ساده‌ای نیست.
مزیت راه‌حل ۱: به ازای تعداد گره‌های درخت فضا گرفته می‌شود و هم درختی از حافظه نداریم.
راه‌حل ۲: در نظر گرفتن k=۲ آرایه برای یک درخت :تاییk



مثال:

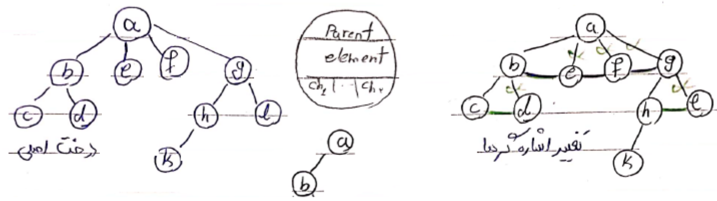


مزایای راه‌حل ۲: ۱- نگرانی در رابطه با ترتیب فرزندان نیست و در نتیجه مدیریت این ترتیب در مقایسه با راه‌حل ۱ به صورت خودکار انجام می‌شود. ۲- دسترسی به فرزندان هر گره به سادگی و در زمان O(۱) قابل انجام است.
مشکل راه‌حل ۲: پتانسیل هدر رفت حافظه در صورتی که نرده‌های یک درخت تاییk کمتر از k باشد بالا است. هدر رفت زیادی دارد.
نکته تکمیلی: پیاده‌سازی‌های فوق می‌تواند با استفاده از لیست پیوندی انجام شود که برتری آن نسبت به آرایه آن است که محدودیت هول آرایه را در صورت رشد درخت نداریم.

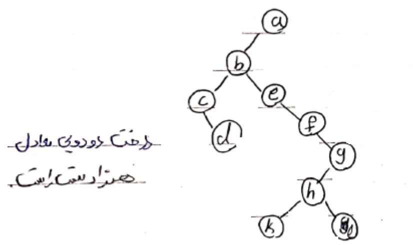


۳ تبدیل یک درخت k تایی به درخت دودویی معادل

راهکار استفاده از k=۲ آرایه برای یک درخت k تایی. همان‌طور که دیدیم پتانسیل اتلاف حافظه دارد. با این‌حال این اتلاف برای درخت دودویی به مراتب کمتر است یک راهکار برای جلوگیری ااز این اتلاف حافظه در درخت k تایی تبدیل آن به درخت دودویی معادل است.
در این تبدیل برای هر گره، اشاره‌گرهای پدر و چپ‌ترین فرزند ثابت باقی می‌ماند و اشاره‌گر سمت راست گره به نزدیکترین همزاد سمت راست (در صورت وجود) اشاره کرد.



میگه $Parent$و$element$را نگه‌دار. یکی تغییرهایی داخل $child$ ها بده $child_2$ را نکه‌دار اما بقیه ۱-k فرزند را یه جورایی فرزند سمت راست خودشان کن. لینکه رو قطع کنرو اینایی که پدر مشترک دارند با هم وصل میشن.



۱.۳ رابطه بین پیمایش درخت k تایی و درخت دودویی معادل

تمرین: رابطه‌های زیر را برای درخت k تایی (T) و درخت دودویی معادل آن (T’) بررسی کنید:
۱- آیا$Preorder(T) = Preorder(T')$ است؟
۲- آیا$Inorder(T) = Inorder(T')$ است؟
پاسخ: برای حالت کلی اگر بخواهیم نشان دهیم که در پیمایش یکسان نیستند فقط کافی است که یک مثال نقض پیدا کنیم، مثلا برای حالت ۲ و ۳ بالا مثال قبل یکسان نبودند پیدایش‌ها را به وضوح نشان می‌دهد.

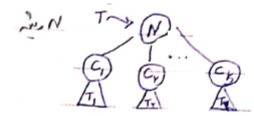
$$\begin{cases} Postorder(T') = d\ c\ k\ l\ h\ g\ f\ e\ b\ a \\ Postorder(T) = c\ d\ b\ e\ f\ k\ h\ l\ g\ a \qquad \Longrightarrow Postorder(T) \neq Postorder(T') \end{cases}$$
$$\begin{cases} Inorder(T') = c\ d\ b\ e\ f\ k\ h\ l\ g\ a \\ Inorder(T) = c\ d\ b\ a\ e\ f\ k\ h\ g\ l \qquad \Longrightarrow Inorder(T) \neq Inorder(T') \end{cases}$$

با این حال پیمایش پیش‌ترتیب مثال قبل برای درخت T و درخت معادل آن یعنی T’ برابر است با:

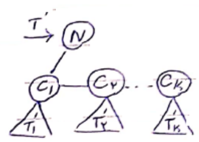
$$\begin{cases} Preorder(T') = a\ b\ c\ d\ e\ f\ g\ h\ k\ l \\ Pteorder(T) = a\ b\ c\ d\ e\ f\ y\ h\ k\ l \qquad \Longrightarrow Preorder(T) = Preorder(T') \end{cases}$$

آیا می‌توانیم ادعا کنیم برای هر درخت برقرار است؟ خیر
برای اثبات حالت کلی برای پیمایش پیش‌ترتیب می‌بایست از تعریف‌های بازگشتی به صورت زیر بهره گرفت.
پیش‌ترتیب آنگاه شهود اثبات

$$Preorder(T) = N\ C_1\ T_1\ C_2\ T_2\ ...\ C_k\ T_k \qquad (۱)$$

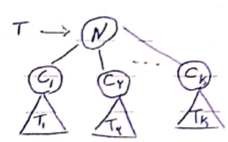


$$Preorder(T') = N\ C_1\ T'_1\ C_2\ T'_2\ ...\ C_k\ T'_k \qquad (۲)$$

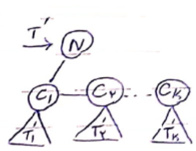


واضح است که جایگاه نودهای ملاقات شده در پیمایش پیش‌ترتیب T وT’ یکسان است. می‌توان از ایده بالا برای رد یکسان بودن پیدایش میان‌ترتیب و پس‌ترتیب درخت T وT’ نیز استفاده کرد.

$$Postorder(T) = T_1\ C_1\ T_2\ C_2\ ...\ T_k\ C_k\ N \qquad (۳)$$

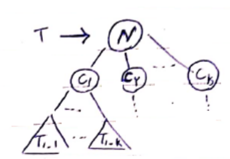


$$Postorder(T') = T'_1\ T'_2\ T'_3\ ...\ T'_k\ C_k\ ...\ C_2\ C_1\ N \qquad (۴)$$



$$Inorder(T) = T_{l-1}\ C_1\ T_{l-2}\ ...\ T_{l-k}\ N$$ (۵)

$$T_{2-1}\ C_2\ T_{2-2}\ ...\ T_{2-k} ...\ T_{k-1}\ C_k\ T_{k-2} ...\ T_{k-k}$$ (۶)



$$Inorder(T') = [C_1]\ C_1\ [C_2]\ C_2\ ...\ [C_k]\ C_k\ N$$ (۷)