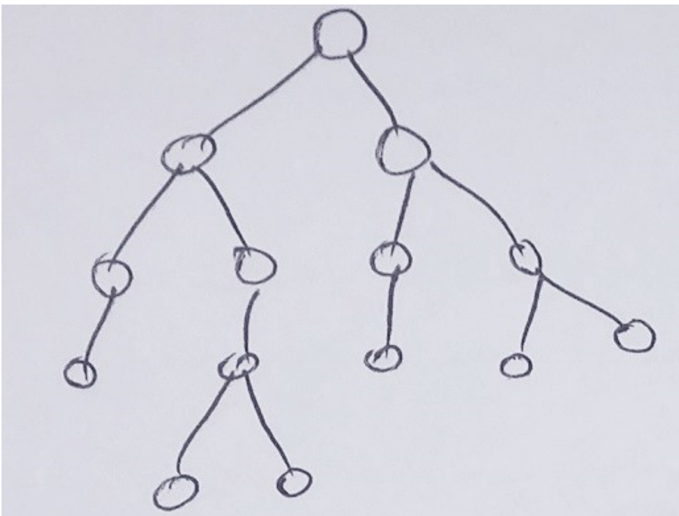
درس ساختمان داده‌ها
دانشکده ریاضی. دانشگاه صنعتی خواجه نصیرالدین طوسی — بهار ۹۹
تدریس توسط: حسین جوهری

ساختار داده درخت (قسمت دوم)

۱ درخت باینری

در صورتیکه هر راس درخت ریشه دار مرتب حداکثر ۲ فرزند داشته باشد به آن درخت باینری (دودویی) گفته می‌شود. شکل زیر یک نمونه از درختهای باینری را نشان می‌دهد.



درختهای باینری کاربردهای فراوانی در طراحی ساختار داده‌ها و الگوریتمها دارند. برای مثال یک عبارت ریاضی (حاوی اعداد و عملگرها) را می‌توان بصورت یک درخت باینری نشان داد (شکل زیر).

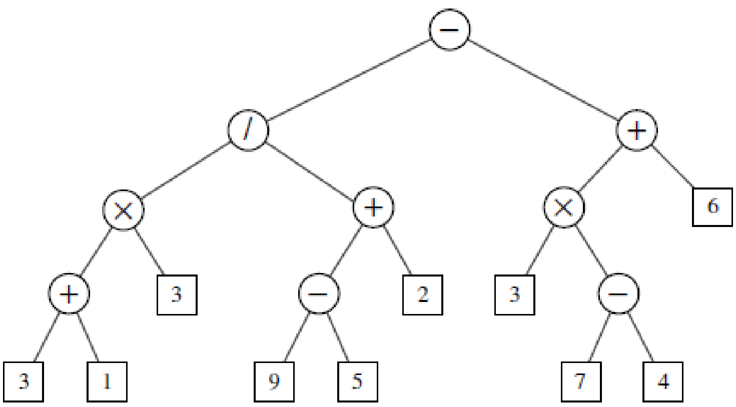


Figure 8.8: A binary tree representing an arithmetic expression. This tree represents the expression $((((3+1)\times 3)/((9-5)+2))-((3\times(7-4))+6))$. The value associated with the internal node labeled "/" is 2.

۱.۱ ویژگیهای آماری درختهای باینری

- n تعداد راسها
- h ارتفاع درخت
- n_E تعداد برگها. (برگ راسی است که فرزند ندارد.) دقت کنید در تئوری گراف، برگ راسی تعریف می‌شود که درجه یک دارد (تنها یک همسایه دارد.) تعریف برگ در اینجا (برای درخت ریشه‌دار مرتب است) و لذا قدری متفاوت است.
- n_I تعداد غیربرگها

برای مثال برای درخت باینری شکل صفحه قبل (شکل بالا) داریم:

$$n = 14, \qquad h = 4, \qquad n_E = 6, \qquad n_I = 8$$

برای هر درخت باینری نامساویهای زیر برقرار است (اثبات بر عهده دانشجو)

- کران بالا و پایین برای تعداد راسها بر اساس ارتفاع

$$h + 1 \ \leq n \leq \ 2^{h+1} - 1$$

- کران بالا و پایین برای ارتفاع بر اساس تعداد راسها

$$\log(n+1) - 1 \leq h \leq n - 1$$

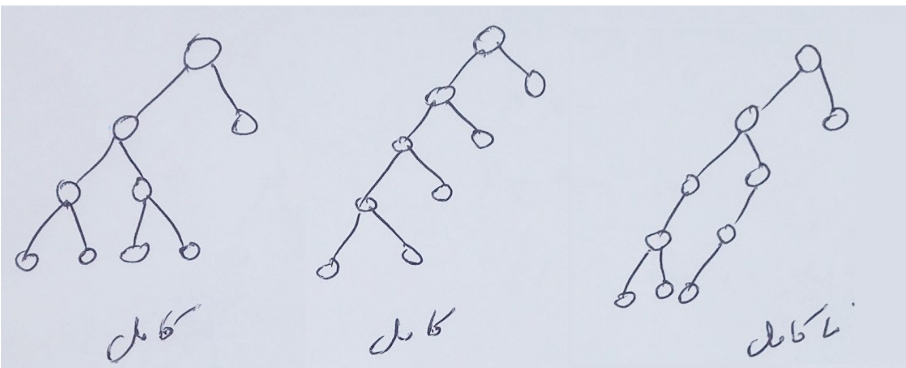
- کران بالا و پایین برای تعداد برگها بر اساس ارتفاع

$$1 \leq n_E \leq 2^h$$

- کران بالا و پایین برای تعداد غیربرگها بر اساس ارتفاع

$$h \leq n_I \leq 2^h - 1$$

درخت باینری را کامل یا پر full گویند اگر و فقط اگر هر راس یا فرزند نداشته یا اینکه دو فرزند داشته باشد.



برای درختهای باینری کامل نامساویهای زیر برقرار است (اثبات بر عهده دانشجو)

- کران بالا و پایین برای تعداد راسها بر اساس ارتفاع

$$2h+1 \ \leq n \leq \ 2^{h+1}-1$$

- کران بالا و پایین برای تعداد برگها بر اساس ارتفاع

$$h+1 \leq n_E \leq 2^h$$

- $$n_E = n_I + 1$$

۲ ساختار داده انتزاعی درخت باینری

ساختار داده انتزاعی درخت باینری T بطور ویژه اعمال اصلی زیر را نیز (علاوه بر اعمال اصلی برای درختها ذکر شد) پشتیبانی می‌کند.

- `T.left(p)`

  فرزند سمت چپ p را برمی‌گرداند. در صورت عدم وجود مقدار None را برمی‌گرداند.

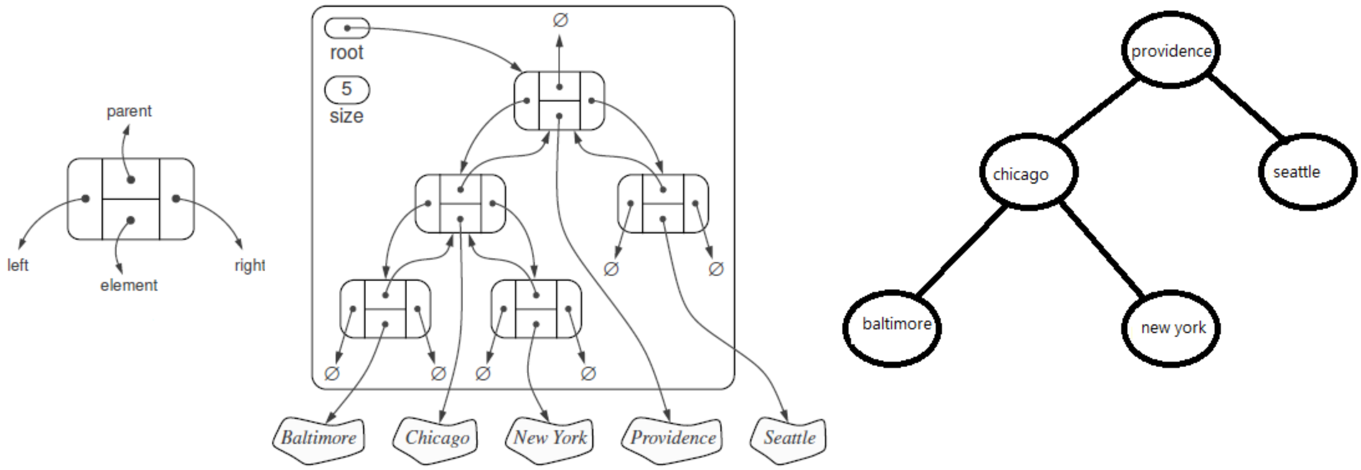
- `T.right(p)`

  فرزند سمت راست p را برمی‌گرداند. در صورت عدم وجود مقدار None را برمی‌گرداند.

- `T.sibling(p)`

۳ پیاده‌سازی درختهای باینری

درختهای باینری را می‌توان بصورت پیوندی (مشابه ساختار لیست پیوندی) پیاده‌سازی کرد. هر راس درخت باینری چهار مولفه خواهد داشت. left فرزند سمت چپ، right فرزند سمت راست، parent پدر، و elemenet عنصر داده‌ای. در شکل زیر این شیوه پیاده‌سازی برای یک نمونه درخت نشان داده شده است. علاوه بر خود درخت، یک اشاره‌گر هم به نام root ذخیره می‌شود که به ریشه درخت اشاره می‌کند.



با در نظر گرفتن پیاده‌سازی بالا، زمان اجرای اعمال اصلی مربوط به درخت باینری در جدول زیر آمده است. برای تعریف بعضی از اعمال مثل delete و attach (سطر آخر جدول) کتاب مرجع را ببینید. برای مثال متد delete راس p با حداکثر یک فرزند را از درخت حذف کرده و فرزند p را به پدر p وصل می‌کند.

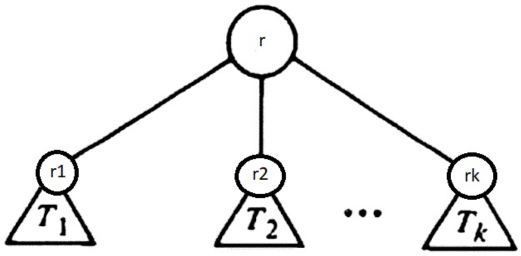
Operation	**Running Time**
len, is_empty	$O(1)$
root, parent, left, right, sibling, children, num_children	$O(1)$
is_root, is_leaf	$O(1)$
depth(p)	$O(d_p+1)$
height	$O(n)$
add_root, add_left, add_right, replace, delete, attach	$O(1)$

۴ پیمایش درختها

هدف از پیمایش یک درخت ملاقات همه راسهای آن درخت است بطوری که هر راس یک بار ملاقات شود. سه روش معمول برای پیمایش درختها وجود دارد. فرض کنید می‌خواهیم درخت T با ریشه r که k زیر درخت دارد را پیمایش کنیم.

- پیمایش پیش ترتیب Preorder : اول ریشه r ملاقات می‌شود و سپس زیردرخت فرزندان به ترتیب از چپ به راست پیمایش می‌شوند.

$$Preorder(T) = r, Preorder(T_1), Preorder(T_2), \ldots, Preorder(T_k)$$



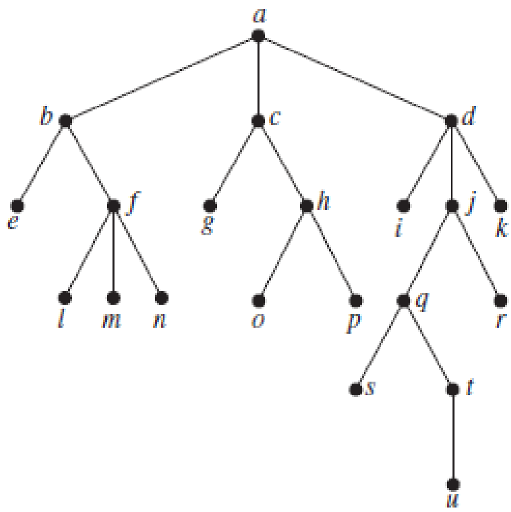
- پیمایش میان ترتیب Inorder : ابتدا زیردرخت سمت چپ پیمایش می‌شود، سپس ریشه ملاقات می‌شود و سپس زیردرخت بقیه فرزندان به ترتیب از چپ به راست پیمایش می‌شوند.

$$Inorder(T) = Inorder(T_1), r, Inorder(T_2), \ldots, Inorder(T_k)$$

- پیمایش پس ترتیب Postorder : اول زیر درخت فرزندان به ترتیب از چپ به راست پیمایش می‌شوند و سپس ریشه r ملاقات می‌شود.

$$Postorder(T) = Postorder(T_1), Postorder(T_2), \ldots, Postorder(T_k), r$$

یک مثال. فرض کنید می‌خواهیم درخت زیر را با روشهای بالا پیمایش کنیم.



$$Preoder(T) = a, b, e, f, l, m, n, c, g, h, o, p, d, i, j, q, s, t, u, r, k$$
$$Inorder(T) = e, b, l, f, m, n, a, g, c, o, h, p, i, d, s, q, u, t, j, r, k$$
$$Postoder(T) = e, l, m, n, f, b, g, o, p, h, c, i, s, u, t, q, r, j, k, d, a$$