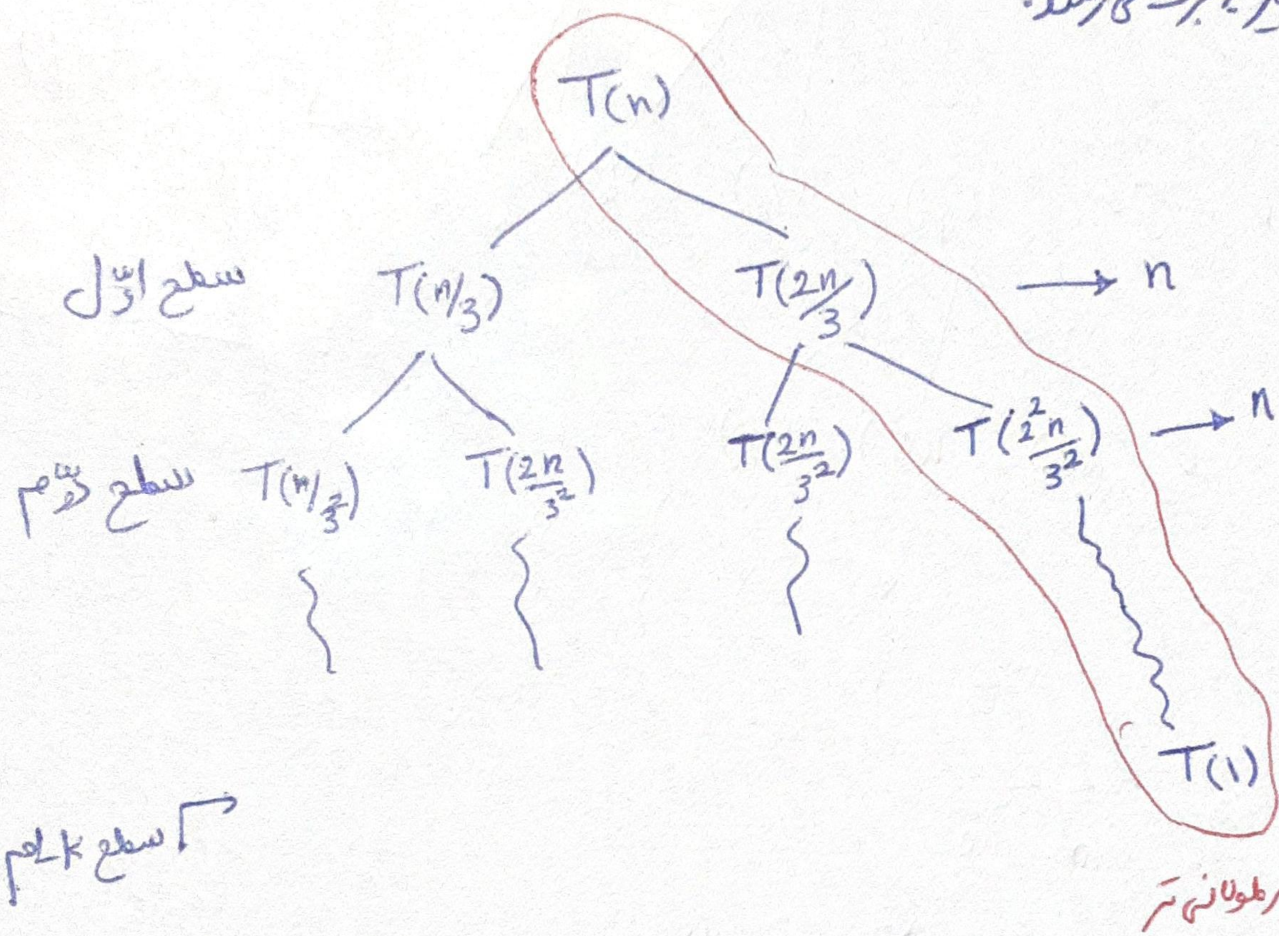
مثال ۳: استفاده از دیدگاه جانبی ← نیاز نیست رابطه بازگشتی را به صورت دقیق حل کنیم،
← قرار است یک حد بالای معقول بگیریم.

$$T(n) = T(n/3) + T(2n/3) + n$$

درخت که نامتوازن است و برخی مسیرها سریع‌تر به برگ می‌رسند.



* تا کجا باید پیش بریم ←

$$\left(\frac{2}{3}\right)^k n = 1 \Rightarrow \frac{n}{\left(\frac{3}{2}\right)^k} = 1 \Rightarrow n = \left(\frac{3}{2}\right)^k$$

$$\log_{\frac{3}{2}} n = k$$

* تعداد نودها در سطح k ←

$$2^k = 2^{\log_{\frac{3}{2}} n} = n^{\log_{\frac{3}{2}} 2}$$

حد بالا ←

$$T(n) = O(n \log n) = O(n \log_{\frac{3}{2}} n)$$

حد پایین هم می‌توان گرفت سمت چپ‌ها رشد کمتر

$$\rightarrow T(n) = \Omega(n \log n) = \Omega(n \log_3 n)$$

نکته: نشان دهید رابطه $\log_b n = O(\log n)$ برقرار است.

$$\exists c, n_0 > 0 \quad s.t. \quad \forall n \geq n_0 \quad 0 \leq \log_b n \leq c \log n \qquad (*)$$

می‌دانیم که $\log_b^a = \frac{\log_c a}{\log_c b}$، بنابراین برای (*) داریم:

$$0 \leq \frac{\log n}{\log b} \leq c \log n$$

و کافی است که $c = \log b$ فرض کنیم.

نکته: درخت بازگشت، روش خوبی برای اثبات نیست، بلکه روش خوبی برای حدس زدن است و برای اثبات دقیق می‌بایست با استفاده از استقرا، حل رابطه بازگشتی را انجام دهیم.

مثال ۴: روش ابتکاری:

$$T(n) = 2T(\sqrt{n}) + \log n$$

تغییر متغیر ← $2^m = n$ بنابراین $m = \log_2 n$ و داریم:

$$T(2^m) = 2T(2^{\frac{m}{2}}) + m$$

اگر $S(m) = T(2^m)$ خواهیم داشت:

$$S(m) = 2S(\tfrac{m}{2}) + m$$

که فرم آشنایی است و قبلاً آن را حل کردیم (روش حدس و استقرا و روش جایگزینی)

$$S(m) = \Theta(m \log m)$$

از آنجائیکه $S(m)=T(2^m)$ داریم :

$$T(2^m)=\Theta(m\log m)$$

و از آنجائیکه $m=\log_2 n$ بوده، داریم :

$$T(2^m)=T(n)=\Theta(\log n \log(\log n))$$

قضیه اصلی (Master Theorem):

در طراحی روش تقسیم و غلبه دیدیم که پیچیدگی زمانی چنین الگوریتم‌هایی در فرم زیر قابل بیان است:

$$T(n) = aT(n/b) + D(n) + C(n)$$

واضح است که برای حل چنین رابطه بازگشتی، می‌توان فرم کلی زیر را برای عبارت بالا بازنویسی کرد:

$$T(n) = aT(n/b) + f(n) \quad (*)$$

در این بخش با معرفی قضیه اصلی سعی داریم تا به صورت کارایی برای حالت‌های خاصی از $f(n)$، رابطه $(*)$ را حل کنیم.

بازگشتی

بحث روی مقادیر a و b در رابطه بازگشتی $(*)$:

* فرض کنید $0 < b < 1$ باشد: اگر چنین باشد، هر بار برای حل $T(n)$، سراغ یک n بزرگتری رویم:

$$T(n) = aT\left(m = \frac{n}{b}\right) + f(n)$$

جائیکه $m > n$ است.

در نتیجه، عملاً روال بازگشتی صحیح وقت تمام نمی‌شود. از اینرو $0 < b < 1$ بی‌معناست.

* فرض کنید $b=1$ باشد: اگر چنین باشد، رابطه بازگشتی $T(n)$ به صورت بدیهی مایل کاهنده است:

$$T(n) = aT(n) + f(n)$$

$$\rightarrow T(n) - aT(n) = f(n) \rightarrow (1-a)T(n) = f(n) \rightarrow T(n) = \frac{f(n)}{1-a}$$

نکته: از آنجایی که $T(n)$ باید مثبت باشد و $f(n)$ یک تابع صعودی است پس $0<a<1$ باید باشد.

* فرض کنید $0<a<1$ باشد: اگر چنین باشد، با بهره گیری از قضیه حل رابطه بازگشتی با استفاده از روش جانشانی که قبلاً مورد بحث کردیم، داریم:

$$T(n) = aT(n/b) + f(n) = \underbrace{a^{\log_b n} \cdot T(1)}_{A^*} + \sum_{i=0}^{(\log_b a)-1} a^i f\left(\frac{n}{b^i}\right)$$

از آنجایی که $0<a<1$ مد نظر است مقدار $A^* < T(1)$ است و بنابراین داریم:

$$T(n) \leq \underbrace{T(1) + \sum_{i=0}^{(\log_b a)-1} a^i f\left(\frac{n}{b^i}\right)}_{B^*}$$

همچنین از آنجایی که در تحلیل پیچیدگی الگوریتم ها، $f(n)$ به طور معمول یک تابع صعودی است، داریم:

$$T(n) \leq B^* \leq T(1) + f(n) \sum_{i=0}^{(\log_b a)-1} a^i$$

$$\leq T(1) + f(n) * \frac{1}{1-a} = O(f(n))$$

جمع بندی: با توجه به بررسی های انجام شده در بالا، می توان قضیه اساسی را برای مقادیر $a \geq 1$, $b > 1$ به صورت خوش فرمی که در ادامه آمده است، تعریف کرد.

قضیه: اگر رابطه بازگشتی به فرم زیر داشته باشیم:

$$T(n) = aT(n/b) + f(n), \text{ where } b>1 \text{ and } a \geq 1$$

آنگاه:

- حالت 1: اگر $f(n)=O(n^{c-\varepsilon})$ باشد، جایکه $\varepsilon>0$ آنگاه $T(n)=\theta(n^c)$.

- حالت 2: اگر $f(n)=\theta(n^c \log^k n)$ باشد، جایکه k یک عدد صحیح مثبت باشد آنگاه $T(n)=\theta(n^c \log^{k+1} n)$.

- حالت 3: اگر $f(n)=\Omega(n^{c+\varepsilon})$ باشد جایکه $\varepsilon>0$ است و همچنین $af(n/b) \leq hf(n)$ باشد برای یک $h<1$ و هر n به اندازه کافی بزرگ، آنگاه $T(n)=\theta(f(n))$.

در هر سه حالت ذکر شده در بالا $c=\log_b^a$ می باشد (با توجه به روش های مثالی ذکر شده برای حل روابط بازگشتی، علت این مقدار برای c واضح است).

نکته 1: برای مشخص کردن حالت های 1، 2 و 3، به نحوی بحث روی n^c است:

* یک ε کمتر،
* یک $\log_n^k$ ضربدر در n^c (تقریباً در مرتبه n^c)
* یک ε بیشتر.

سوال: آیا حالت های بیان شده در قضیه بالا، همهٔ توابع $f(n)$ صعودی را پوشش می دهند؟

پاسخ منفی است، در ادامه با ذکر مثال هایی بر این موضوع می پردازیم.