دانشکده علوم ریاضی و آمار

مدرس: دکتر مجتبی رفیعی نیمسال اول ۱۴۰۰-۱۴۰۱

ساختمان داده‌ها و الگوریتم‌ها

جلسه ۱۹

نگارنده: شقایق اسماعیلیان

۹ آبان ۱۴۰۰

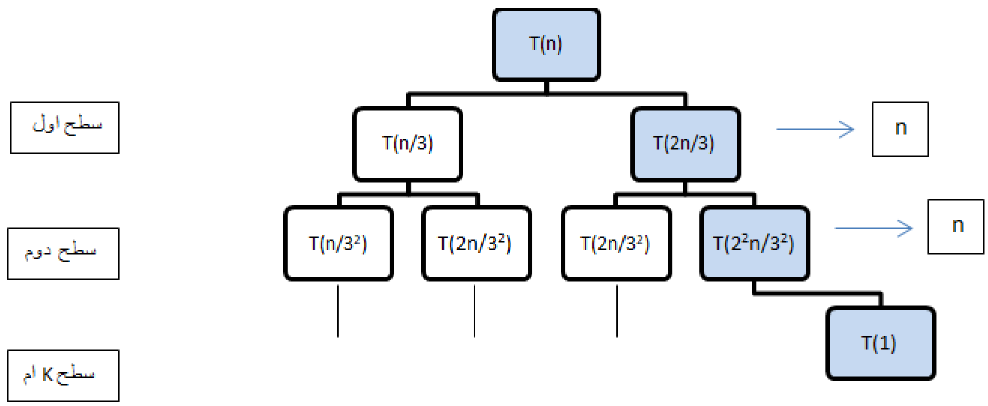
فهرست مطالب

۱ استفاده از دیدگاه مجانبی

همواره نیاز نیست رابطه بازگشتی را به صورت دقیق حل کنیم، بلکه می‌توانیم در مواردی یک حد بالای معقول برای آن پیدا کنیم. به عنوان مثال، رابطه زیر را در نظر بگیرید:

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + n$$

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



تا کجا باید پیش بریم (مسیر طولانی تر):

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

تعداد نودها در سطح k:

$$2^k = 2^{log^2_{\frac{3}{2}}} = n^{log^2_{\frac{3}{2}}}$$

حد بالا: $T(n) = O(nlog^n) = O(nlog^n_{\frac{3}{2}})$
حد پائین هم می‌توان گرفت: شاخه با رشد کمتر:

$$T(n) = \Omega(nlog^n) = \Omega(nlog^n_3)$$

نکته: نشان دهید رابطه $log^n_b = O(log^n)$ برقرار است.

$$\exists c, n > 0 \; \forall n \geq n_0 0 \leq log^n_b \leq clog^n$$

می‌دانیم که $log^a_b = \frac{log^a_c}{log^b_c}$، بنابراین برای $0 \leq log^n_b \leq clog^n$ داریم:

$$0 \leq \frac{log^n}{log^b} \leq clog^n$$

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 روش حدس و استقرا، حل رابطه بازگشتی را انجام دهیم.

۲ روش ابتکاری

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

تغییر متغیر: $2^m = n$ بنابراین $m = log^n_2$ و داریم:

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

اگر $T(2^m) = S(m)$ خواهیم داشت: $S(m) = 2S(\frac{m}{2}) + m$ که فرم آشنایی است و قبلا آن را حل کردیم (روش حدس و استقرا و روش جایگزینی)،

$$S(m) = \theta(mlog^m)$$

از آنجائیکه $S(m) = T(2^m)$ داریم:

$$T(2^m) = \theta(mlog^m)$$

و از آنجائیکه $m = log_2^n$ بود، داریم:

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

۳ قضیه اصلی (Master Theorem)

در معرفی رویکرد 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:

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

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

۱.۳ 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 بزرگتر می‌رویم. $T(n) = aT(m = n/b) + f(n)$ جایی که $m > n$ است. در نتیجه، عملا روال بازگشتی هیچ‌وقت تمام نمی‌شود، از اینرو $0 < b < 1$ بی معنا است.

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

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

$$\rightarrow T(n) - aT(n) = f(n) \rightarrow (1-a)T(n) = f(n) \rightarrow T(n) = f(n)/1-a$$

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

* **فرض کنید** $0 < a < 1$ **باشد:** اگر چنین باشد، با بهره گیری از حل رابطه بازگشتی با استفاده از روش جایگذاری که قبلا محاسبه کردیم، داریم:

$$T(n) = aT(n/b) + f(n) = a^{log_b^n}.T(1) + \sum_{i=0}^{(log_b^a)-1} a^i f(n/b^i)$$

$$A^* = a^{log_b^n}.T(1)$$

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

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

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

همچنین از آنجاییکه در تحلیل پیچیدگی الگوریتم‌ها، به طور معمول $f(n)$ یک تابع صعودی است، داریم:

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

جمع بندی: با توجه به بررسی های انجام شده در بالا، می‌توان قضیه اساسی را برای مقادیر $a \geq 1$ و $b > 1$ به صورت خوش فرمی که در ادامه آمده است، تعریف کرد.
قضیه: اگر رابطه بازگشتی به فرم زیر داشته باشیم:

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

آنگاه:

- **حالت ۱:** اگر $f(n) = O(n^{c-\mathcal{E}})$ باشد، جائیکه $\mathcal{E} > 0$ آنگاه $T(n) = \theta(n^c)$.
- **حالت ۲:** اگر $f(n) = \theta(n^c log_n^c)$ باشد، جائیکه k یک عدد صحیح مثبت باشد، آنگاه $T(n) = \theta(n^c log_n^{k+1})$.
- **حالت ۳:** اگر $f(n) = \Omega(n^{c+\mathcal{E}})$ باشد، جائیکه $\mathcal{E} > 0$ است و همچنین $af(n/b) \leq hf(n)$ باشد برای یک $0 < h < 1$ و هر n به اندازه کافی بزرگ، آنگاه $T(n) = \theta(f(n))$.

در هر سه حالت ذکر شده در بالا $c = log_b^a$ می‌باشد (با توجه به روش های قبلی ذکر شده برای حل روابط بازگشتی، علت این مقدار برای c واضح است).
توجه: برای مشخص کردن حالت های ۱،۲،۳، به نحوی بحث روی n^c است:

- یک $\mathcal{E}$ کمتر،
- یک log_n^k ضربدر n^c (تقریبا در مرتبه n^c)،
- یک $\mathcal{E}$ بیشتر.

سوال: آیا حالت‌های بیان شده در قضیه بالا، همه توابع $f(n)$ صعودی را پوشش می‌دهد؟ پاسخ منفی است، در ادامه با ذکر مثال هایی به این موضوع می‌پردازیم.