به نام خدا

تمرین چهارم طراحی الگوریتم‌ها

شهرزاد جوادی کوشش - 99243027

سوال اول

a) True

$f_1(n) \le c_1 g_1(n)$

$f_2(n) \le c_2 g_2(n)$

دو طرف دو رابطه‌ی بالا را با هم جمع می‌کنیم:

$f_1(n) + f_2(n) \le c(g_1(n) + g_2(n))$

از اصل زیر استفاده می‌کنیم:

$a + b \le 2 \times max(a, b)$

پس نتیجه می‌گیریم:

$f_1(n) + f_2(n) \le c(max(g_1(n), g_2(n))$

$f_1(n) + f_2(n) \in O(max(g_1(n), g_2(n))$

b) False

مثال نقض:

$f(n) = n, g(n) = n^2 \rightarrow f(n) = O(g(n))$

$2^{f(n)} = 2^n, 2^{g(n)} = 2^{n^2}$

اما سرعت رشد 2^{n^2} بسیار بیشتر از 2^n است.

$2^{f(n)} \notin O(2^{g(n)})$

c) False

تابع $\lceil logn \rceil !$ ابتدا سقف حاصل لگاریتم را محاسبه می‌کند و بعد از آن فاکتوریل می‌گیرد. یعنی اگر n بسیار زیاد باشد، حاصل از چندجمله‌ای بیشتر است. اما تابع $log(n!)$ ابتدا فاکتوریل n را حساب می‌کند و سپس از آن لگاریتم می‌گیرد که دارای کران $nlogn$ است.

d) False

$f(n) = n^5,\ g(n) = n^3 \rightarrow f(n) \notin O(g(n),\ g(n) \in O(f(n)$

سوال دوم

a)

$T(n) - 3T(n-1) - 4T(n-2) = 0$

$a_0 = 1$

$a_1 = -3$

$a_2 = -4$

$P(x) = x^2 - 3x - 4 = 0 \rightarrow r_1 = -1,\ r_2 = 4$

$T(n) = c_1(-1^n) + c_2(4^n)$

$T(0) = c_1 + c_2 = 0$

$T(1) = -c_1 + 4c_2 = 5c_2 = 2 \rightarrow c_2 = 0.4,\ c_1 = -0.4$

$T(n) = -0.4(-1^n) + 0.4(4^n)$

b)

استفاده از تغییر متغیر و تغییر فانکشن

$n = 2^m,\ m = log_2^n$

$T(2^m) = 2T(2^{m-1}) + m,\ S(m) = T(2^m)$

$S(m) = 2S(m-1) + m$

$P(x) = x^3 - 4x^2 + 5x - 2 = (x-1)^2(x-2) \rightarrow r_1 = 2, m_1 = 1 / r_2 = 1, m_2 =$

$S(m) = c_1 2^m + c_2 m + c_3$

$T(2^m) = c_1 2^m + c_2 m + c_3$

$T(n) = c_1 n + c_2 log_2^n + c_3$

استفاده از جایگزینی:

$T(1) = c_1, c_2 = 1, c_3 = 0$

$T(n) = T(1)n + log_2^n, \in \theta(n)$

c)

کل معادله را بر n^2 تقسیم می‌کنیم و تغییر متغیر $n = 2^{2^m}$ و تغییر فانکشن $S(n) = \frac{T(n)}{n^2}$ را اعمال می‌کنیم.

$n = 2^{2^m} \rightarrow m = log_2^{log_2^n}$

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

$S(n) = S(\sqrt{n}) + loglogn$

$S(2^{2^m}) = S(2^{2^{m-1}}) + m$

$X(m) = c_1 2^m + c_2 m + c_3$

$S(2^{2^m}) = c_1 2^m + c_2 m + c_3$

$S(n) = c_1((log_2^{log_2^n})^2 + c_2(log_2^{log_2^n}) + c_3$

d)

e)

از دو طرف معادله log می‌گیریم:

$log(T(n-1)) = \frac{1}{2}(log(T(n-3) - log(T(n-2))$

تغییر متغیر:

$log(T(n)) = \frac{1}{2}(log(T(n-2) - log(T(n-1))$

$S(n) = \frac{1}{2}(S(n-2) - S(n-1))$

$P(x) = (x+1)(2x-1) = 0 \rightarrow r_1 = 0.5, r_2 =-1$

$log(T(n)) = c_1 r_1^n + c_2 r_2^n = c_1(0.5)^n + c_2(-1)^n$

$T(n) = 2^{c_1(0.5)^n + c_2(-1)^n}$

$T(0) = 2^{c_1+c_2} = 4 \rightarrow c_1 + c_2 = 2$

$T(1) = 2^{0.5c_1-c_2} = \sqrt{2} \rightarrow 0.5c_1 - c_2 = 0.5$

$c_1 = \frac{5}{3}, c_2 = \frac{1}{3}$

$T(n) = 2^{\frac{5}{3}(0.5)^n + \frac{1}{3}(-1)^n}$

سوال سوم

a)

برای حل این سوال از قضیه‌ی اصلی استفاده می‌کنیم.

معادله‌ی کلی:

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

درخت این معادله log_b^n عمق دارد و عمق i شامل a^i گره است. بنابراین تعداد برگ‌ها در درخت با معادله‌ی $n^{log_b^a} = a^{log_b^n}$ محاسبه می‌شوند و زمان اجرای الگوریتم $O(n^{log_b^a})$ است. اما $f(n)$ شرایطی به مسئله اضافه می‌کند.

$a = 2, b = 2, f(n) = n^3$

چون $f(n)$ از O و θی $n^{log_2^2}$ نیست، پس حالت سوم معادله برقرار است که:

$T(n) = \theta(f(n)) = \theta(n^3)$

b)

طبق قضیه‌ی اصلی:

$a = 3, b = 4, f(n) = nlogn$

بدون در نظر گرفتن $f(n)$، زمان اجرای الگوریتم برابر است با $O(n^{log_4^3})$.

چون $f(n)$ از O و θی $n^{log_4^3}$ نیست، پس حالت سوم معادله برقرار است که:

$T(n) = \theta(f(n)) = \theta(nlogn)$

c)

چون سرعت رشد $\frac{n}{2}$ بسیار بیشتر از $\sqrt{n}$ است از $\sqrt{n}$ صرف نظر می‌کنیم:

$T(n) = 2T(\frac{n}{2}) + T(\frac{n}{2}) + \theta(1)$

طبق قضیه‌ی Akra-Bazzi داریم:

$a_1 = 2$

$b_1 = \frac{1}{2}$

$a_2 = 1$

$b_2 = \frac{1}{2}$

$g(n) = \theta(1)$

$a_1 \times b_1^{\ p} + a_2 \times b_2^{\ p} = 1 \rightarrow 2(0.5)^p + 1(0.5)^p = 1$

$(0.5)^p = \frac{1}{3} \rightarrow p = log_2^3$

$$T(n) \in \theta(n^{log_2^3}(1 + \int_1^n \frac{g(u)}{u^{p+1}} du)) \rightarrow \in \theta(n^{log_2^3})$$

سوال چهارم

الف)

حلقه‌ی خارجی n بار و حلقه‌ی داخلی $\frac{n}{i}$ بار اجرا می‌شود.

مجموع تعداد دفعات اجرا:

$$\sum_{i=1}^{n} \frac{n}{i} \in O(nlogn)$$

ب)

حلقه‌ی خارجی هربار i را نصف می‌کند و حلقه‌ی داخلی هر بار j را دو برابر می‌کند. پس تعداد دفعات اجرای حلقه‌ی درونی بستگی به این دارد که چند بار j می‌تواند دو برابر شود، قبل از اینکه مقدار آن از i بیشتر شود.

تعداد iteration حلقه‌ی بیرونی = k

$$\frac{n}{2^k} = 1$$

$$k = log_2^n \rightarrow \in O(logn)$$

ج)

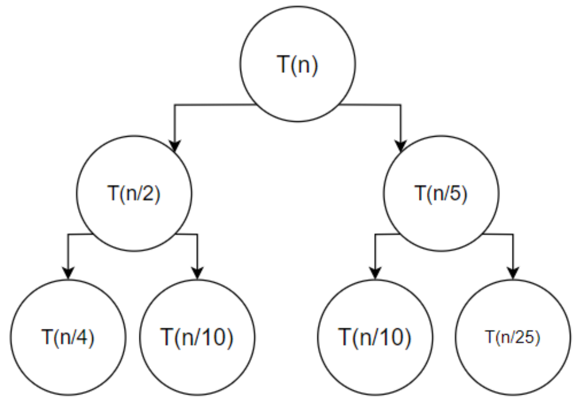
شرط چک کردن $n \leq 1 \in O(1)$

اگر وارد این شرط نشویم، تابع به صورت بازگشتی خودش را فراخوانی می‌کند و تا جایی ادامه دارد که n از 1 کوچک‌تر یا با آن مساوی باشد. عمق درخت بازگشتی بستگی به تعداد دفعاتی که می‌شود n را بر d تقسیم کرد، تا وقتی که کمتر یا مساوی یک شود تعیین بستگی دارد (log_d^n). چون هر بار فراخوانی بازگشتی، n را به d تقسیم می‌کنیم.

مجموع تعداد دفعات اجرا $\in O(log_d^n)$

سوال پنجم

الف)

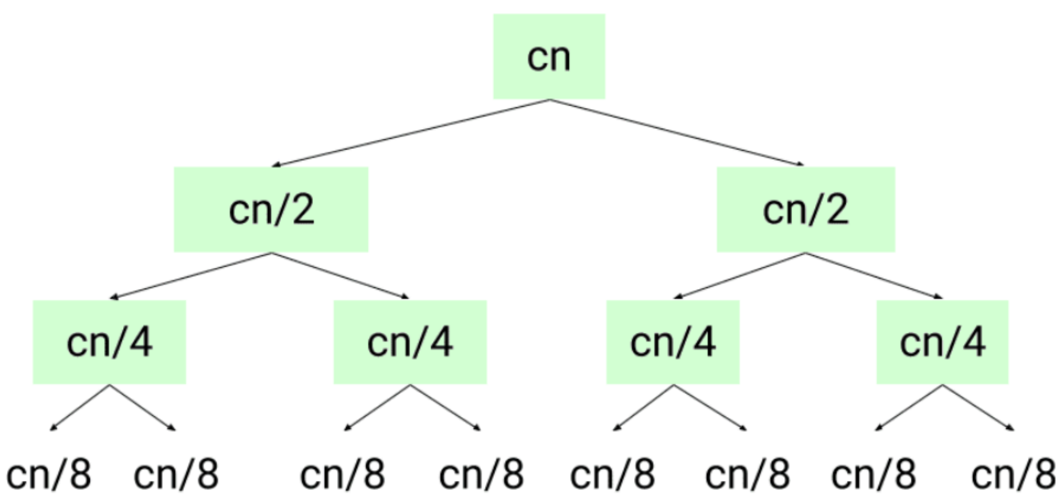


سقف هزینه در عمق i درخت: $nlogn$

$Total\ cost \in O(nlog_2^n)$

ب)

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



$Total\ cost \in O(cn(logn)\ +\ 1)$

سوال ششم

$$nT(n) = 2\sum_{i=0}^{n-2} T(i) + 2T(n-1) + \theta(n^2)$$

$$(n-1)T(n-1) = 2\sum_{i=0}^{n-2} T(i) + \theta((n-1)^2)$$

از دو رابطه‌ی بالا نتیجه می‌گیریم:

$$T(n) = \frac{n+1}{n}T(n-1) + \theta(1)$$

$$T(n) = \frac{n+1}{n-1}T(n-2) + \left[1 + \frac{n+1}{n}\right]\theta(1)$$

$$T(n) = \frac{n+1}{n-2}T(n-3) + \left[1 + \frac{n+1}{n} + \frac{n+1}{n-1}\right]\theta(1)$$

$$T(n) = \frac{n+1}{n-3}T(n-4) + \left[\frac{n+1}{n+1} + \frac{n+1}{n} + \frac{n+1}{n-1} + \frac{n+1}{n-2}\right]\theta(1)$$

$$T(n) = (n+1)T(0) + (n+1)\sum_{i=1}^{n+1}\frac{1}{i}$$

$$T(n) = (n+1)T(0) + (n+1)log(n+1) \in \theta(nlogn)$$