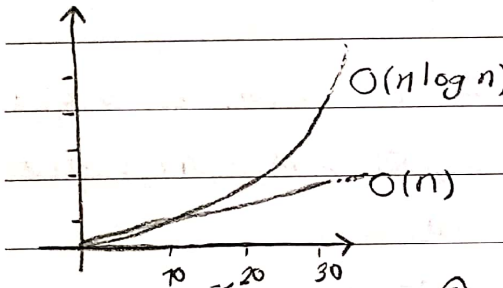
"به نام خدا"

Subject:
Year. 00 Month. 3 Day. 12

امتحان میانترم تحلیل و طراحی الگوریتم‌ها — مصطفی نصلی تهری

الف) درست

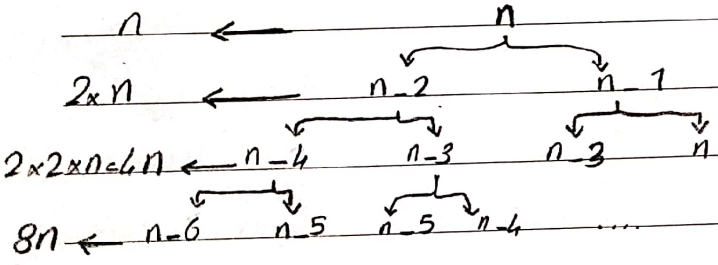
1- تابع $(n\log n)$ فاصله‌ای که $(\log n)$ کوچکتر از یک باشد از (n) کوچکتر و در بقیه نقاط که فضای بزرگی است از تابع (n) بزرگ‌تر است ← با در نظر گرفتن مجانبی بودن این دو مرتبه زمانی می‌توان نتیجه گرفت که $(n\log n)$ در آن نقاط به صورت مجانبی از (n) بزرگ‌تر است، همچنین با توجه به نمودارهای این دو مرتبه زمانی این موضوع را می‌توان نتیجه گرفت.



ب) درست، با توجه به دلایلی که می‌توان برای حل مسائل از DP استفاده کرد که عبارتند از ① بازگشتی بودن ② وجود تکرار در زیرمسائل ③ بهینه بودن، اگر شرط بهینه بودن را نیز دارا باشد پس همیشه می‌توان از DP استفاده کرد، در غیر این صورت نادرست و نیاز به بهینه بودن زیرمسئله‌ها داریم.

2- الگوریتم سری فیبوناچی الگوریتمی است که می‌توان هم به صورت بازگشتی هم به صورت محاسباتی آن را پیاده‌سازی کرد، در این سوال با توجه به بازگشتی بودن نوع پیاده‌سازی می‌توان فهمید که در هر مرحله درخت به دو زیرشاخه تقسیم می‌شود، اگر هما نند توانع کلاس درخت آن را رسم کنیم، داریم:

$$(I)\quad fib(n) = fib(n-1) + fib(n-2)$$



کلی توان با استفاده از رابطه ① و درخت رسم شده پیچیدگی زمانی آن را به صورت زیر نشان داد.

$$T(n) = \underbrace{T(n-1)}_{} + \underbrace{T(n-2)}_{} + ① \rightarrow \Rightarrow$$

در این رابطه می‌توان فهمید که برای محاسبه هر عدد به دو عدد قبلی نیاز داریم که هر دوی آن دو عدد به دو عدد قبلی خود نیاز هنوز هستند، پس درختی با 2 شاخه برای هر n بدست می‌آید که می‌توان رابطه آن را به صورت زیر نوشت:

$$T(n) = \overbrace{n}^{(2^0)n} + (2^1)\times n + (2^2)n + (2^3)n + \dots = (2^{1+2+\dots})n = 2^n \Rightarrow O(2^n)$$

در بدترین حالت که برای محاسبه هر فود درخت با n عدد قبلی را دوباره محاسبه کنیم و نیاز به همه شاخه‌های درخت باشد، مرتبه زمانی $O(2^n)$ می‌رسیم!!

اما در بهترین حالت، هنگامی که عناصر محاسبه شده را در حافظه نگهداری کنیم و هر زیرشاخه درخت را تنها یکبار محاسبه کنیم به پیچیدگی زمانی $O(2^{n/2}+2)$ خواهیم رسید که در گزینه‌ها بیشتر از $2^{n/2}$ داریم.

Year. Month. Day.

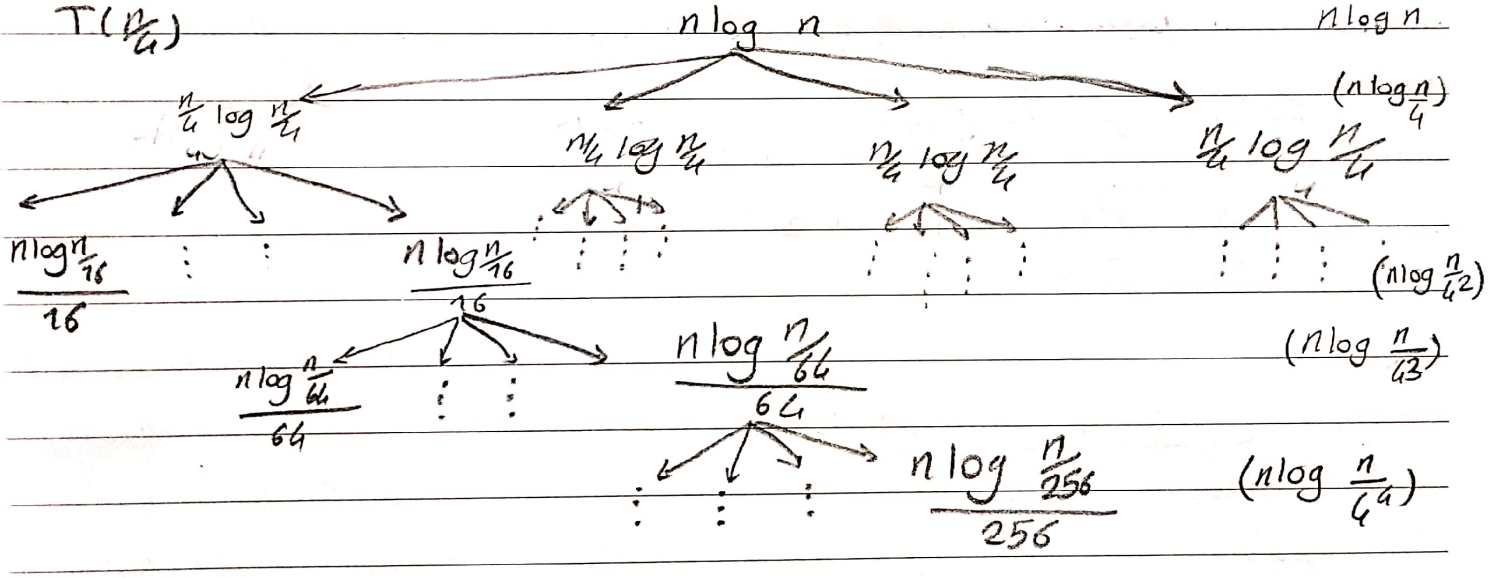
ج. باتوجه به این که دو حلقه for داریم و تابع swap درون حلقه درونی قرارمارد، به صورت خیلی ساده می توان گفت پیچیدگی زمانی این تابع در بدترین حالت $O(n^2 \log n)$ است.
اما با ریز شدن در تابع درمی یابیم که حلقه درونی به حلقه بیرونی وابسته شده است و همیشه یک واحد از i بیشتر است و البته اینبار تا خرد n ادامه دارد، پس اینبار درمی یابیم که در هر مرتبه یکسری اعداد کوچکتر از n حذف می شوند، پس به پیچیدگی $\Theta(n^2 \log n)$ نزدیکتر می شویم،
اما باز هم اگر بخواهیم دقیق تر محاسبه کنیم، با استفاده از سیگما داریم:

$$\left(\sum_{i=1}^{n} i\right)\left(\sum_{j=i+1}^{n}\right)(\log n) = \text{So} \rightarrow \left(\frac{n(n+1)}{2}\right)(\log n) = \left(\frac{n^2+n}{2}\right)(\log n) \Rightarrow$$

$$\sim \Theta(n^2 \log n)$$

ز. الف) ابتدا درخت بازگشت آن را رسم می کنیم.

$T(n) = T\left(\frac{n}{4}\right) + n \log n$ $\quad a=4$ $\quad b=4$



که این درخت همینان با همین روند ادامه دارد، همانطور که مشاهده می شود پیچیدگی آن از رده $n \log n$ و ترکیب آن با ضریب n است، پس می توان نوشت:

$$T(n) = n \sum_{i=0}^{\log_4 n} \log n \;\; ? \;\; n \sum_{i=0}^{\log_4 n} \log \frac{n}{4}$$

$$\rightarrow \in \Theta(n \log n)$$

ب) خیر زیرا با توجه به $n^{\log_4 4} = n^1 = n < n \log n$ اگر هر مقدار ε به n اضافه شود، رابطه $n \log n$ از بین می رود و این پیچیدگی زمانی را نمی توان از طریق قضیه اصلی حل نمود.

$$n^{\varepsilon+1} \not< n \log n$$

Year. Month. Day.

$$f(n) = \begin{cases} 4f(\frac{n}{2}) + \Theta(n^3) & n > 1 \\ 1 & n = 1 \end{cases}$$

$a = 4$
$b = 2$
$f(n) = n^3$

5

با استفاده از قضیه اصلی داریم:

$n^{\log_2^4} = n^2 \rightarrow n^2 \text{ ؟ } f(n) \rightarrow n^2 < n^3 \rightarrow$

رابطه سوم در این رابطه بازگشتی برقرار است یعنی $n^{\log_b^a} < f(n)$ پس داریم $\Theta(f(n))$ $\Theta(n^3)$

6. الف) ساختار بهینه در ضرب زنجیره‌ای ماتریس‌ها با استفاده از Dynamic Programming در راه‌حل بازگشتی آن بدست می‌آید. بدین صورت که برای ضرب بهینه n ماتریس، با استفاده از فرمول

$$m_{ij} = \min_{K} (m_{iK} + m_{K+1,j} + P_{i-1} P_K P_j)$$

minimum تعداد ضرب‌ها را بدست می‌آوریم که برای بدست آوردن هر چه زودتر جواب، باطول کمتر را باید داشته باشیم.

ب) مقادیر پایه یا مقادیر اولیه هستند که تعیین می‌کنند بازگشت، تابع در کجا خاتمه پیدا می‌کند و جای زیر مسئله جدید، مقدار برمی‌گرداند، معمولاً اندیس‌های هر ماتریس، همان مقادیری هستند که ابعاد ماتریس را تعیین کرده و پس از محاسبه تغییر می‌کنند، به عبارت دیگر مقادیر اولیه در این m_{ii} هستند که به کمک آن‌ها مقادیر دیگر را محاسبه می‌کنیم، این مقادیر در ماتریس $m_{11}, m_{22}, \ldots, m_{nn}$ هستند.

ج) فرمول کلی محاسبه:

$$m_{ij} = \min(m_{iK} + m_{K+1,j} + P_{i-1} P_K P_j)$$

$$\begin{cases} m_{11} = m_{22} = m_{33} = m_{44} = 0 \end{cases}$$

$$A_{10\times20} \times B_{20\times50} \times C_{50\times1} \times D_{1\times100}$$

$$\begin{cases} m_{12} = \min_1 (\overset{0}{m_{11}} + \overset{0}{m_{22}} + \overset{10{,}000}{10\times20\times50}) = 10{,}000 \\ m_{23} = \min_1 (\overset{0}{m_{22}} + \overset{0}{m_{33}} + \overset{1000}{20\times50\times1}) = 1{,}000 \\ m_{34} = \min_1 (\overset{0}{m_{33}} + \overset{0}{m_{44}} + \overset{5{,}000}{50\times1\times100}) = 5{,}000 \end{cases}$$

$$m = \begin{bmatrix} 0 & 10000 & 1200 & 2200 \\ & 0 & 1000 & 3000 \\ & & 0 & 5000 \\ & & & 0 \end{bmatrix} \Rightarrow$$

$$m_{13} = \min \begin{pmatrix} \overset{0}{m_{11}} + \overset{1000}{m_{23}} + \overset{200}{10\times20\times1} \\ \overset{10000}{m_{12}} + \overset{0}{m_{33}} + \overset{500}{10\times50\times1} \end{pmatrix} = \min \begin{pmatrix} 1200 \\ 10500 \end{pmatrix} = 1200$$

$$P = \begin{bmatrix} 1 & 1 & 3 \\ & 2 & 3 \\ & & 3 \end{bmatrix}$$

$$m_{24} = \min \begin{pmatrix} \overset{0}{m_{22}} + \overset{5000}{m_{34}} + \overset{100000}{20\times50\times100} \\ \underset{1000}{m_{23}} + \underset{0}{m_{44}} + \underset{2000}{20\times1\times100} \end{pmatrix} = \min \begin{pmatrix} 10\,5000 \\ 3000 \end{pmatrix} = 3000$$

$$m_{14} = \min \begin{pmatrix} m_{11} + m_{24} + 10\times20\times100 \\ m_{12} + m_{34} + 10\times50\times100 \\ m_{13} + m_{44} + 10\times1\times100 \end{pmatrix} = \min \begin{pmatrix} 23000 \\ 65000 \\ 2200 \end{pmatrix}$$

Year. Month. Day.

از ماتریس M نتیجه می‌گیریم که مینیمم تعداد ضرب ها در این مثال 2,200 عدد ضرب است.

$(A_1 (A_2 \times A_3)) A_4$

برای پرانتز گذاری بین، ضرب ماتریس ها بدین گونه می شود،

برای امتحان راه حل می توان از حل دیگر استفاده کرد:

$A_{10\times20}$ $B_{20\times50}$ $C_{50\times1}$ $D_{1\times100}$

1000 200 1000

$20\times50\times1 + 10\times20\times1 + 10\times100\times1$

2200 ✓

$A_{10\times20}$ ($E_{20\times1}$) $D_{1\times100}$

(($F_{10\times1}$)) $D_{1\times100}$

(($G_{10\times100}$)))

$\Rightarrow (A_1 \times (A_2 \times A_3)) \times A_4)$

که می توان دریافت که جواب بدست آمده در هر دو روش یکسان و صحیح می باشد.