به نام خدا

تمرین سری اول درس طراحی الگوریتم

(نیم‌سال اول ۱۴۰۱)

---

۱) تابع lg^*n برابر است با تعداد باری که از n لگاریتم (بر پایه ۲) بگیریم که حاصل ۱ یا کمتر از ۱ شود.

(مثلا $lg^*65536 = 4$) تابع lg^*n رشد بسیار ضعیفی دارد. در هر یک از موارد زیر درست یا نادرست بودن ترتیب رشد توابع را مشخص کنید.(در موارد که نادرست اعلام می‌کنید علت را نیز توضیح دهید و ترتیب درست را بنویسید)

$a)\ n^{\frac{1}{lgn}} < lg^2n < \lg(n!) < n^3 < \left(\frac{3}{2}\right)^n < 2^{2^n}$

$b)\ lg^*n < n^2 < \left(\sqrt{2}\right)^{lgn} < 2^{lg^*n} < n! < 2^{2^{n+1}}$

$c)\ lglgn < lg^*n < lgn < n^{lglgn}$

۲) مرتبه زمانی شبه کدهای زیر را به دست آورید

```
a) for i ← 1 to n
       for j ← n to i
           for k ← 1 to n^2
               sum ← sum + A
```

1

a) $n^{\frac{1}{\log n}} = n^{(\log n)^{-1}} = n^{\log_n 2} \xrightarrow{a^{\log_c b} = b^{\log_c a}} = 2^{\log_n n} = 2$

پس این یک نمودار خطی ثابت است که سرعت رشد کمتری از $\log^2 n$ دارد.

$\log^2 n < \log(n!) = \log(n \times (n-1) \times \cdots \times (1)) = \log n + \log(n-1) + \cdots + \log 1$ ← تقسیم هردو سمت بر $\log n$

$\Rightarrow \log n < 1 + \log_n(n-1) + \log_n(n-2) + \cdots + \log_n 1 \equiv \text{True}$

همچنین سرعت رشد n^3 بیشتر از تابع لگاریتمی خواهد بود، و کمتر از تابع نمایی $(\frac{3}{2})^n$ خواهد بود.

$(\frac{3}{2})^n < 2^{(2^n)} \rightarrow (2^{\log 3/2})^n < 2^{(2^n)} \rightarrow n \times \log 3/2 < \underbrace{2^n}_{\text{نمایی}} \equiv \text{True}$

b) $\log^* n < n^2 \equiv \text{True}$

$n^2 < \sqrt{2}^{\lg n} \rightarrow n^2 < \underbrace{2^{\frac{\lg n}{2}}}$ $\equiv \text{False}$

نمایی است اما سرعت رشد توان، بسیار رو به کاهش است.

$(\sqrt{2})^{\log n} < 2^{\log^* n} \Rightarrow 2^{\frac{1}{2}\log n} < 2^{\log^* n} \Rightarrow \frac{1}{2}\log n < \log^* n \equiv \text{False}$

می دانیم که تابع $\log^* n$ رشد بسیار کندی نسبت به $\log n$ دارد.

$n^2 < n! \Rightarrow n^2 < n \times (n-1)(n-2) \times \cdots \times 1 \Rightarrow \div n$

$\Rightarrow n < (n-1)(n-2) \times \cdots \times 1 \overset{n \geq 4}{\equiv} \text{True}$

$n! < \underbrace{2^{2^{n+1}}}_{\text{رشد بسیار سریع}}$ ← نمایی $\equiv \text{True}$

$\underbrace{\log^* n < 2^{\log^* n}}_{4 < 2^4 \equiv \text{True}} < \sqrt{2}^{\log n} < n^2 < n! < 2^{2^{n+1}}$

c) $\log(\log n) < \log^*(n) \equiv$ True

با کاهش سرعت رشد $\log n$
سرعت رشد به شدت کم می‌شود

$\log^*(n) < \log(n) \equiv$ True

وجود * سرعت رشد
را کاهش می‌دهد

$\log(n) < n^{\log(\log(n))} \equiv$ Tre

رفتار توانی دارد و سرعت رشد آن
بزرگ و بزرگ‌تر می‌شود.

2

a) $n \times \log(n) \times n^2 \times 1 \rightarrow O(n^3 \log n)$

b) $n \times \left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}\right)$

$\longrightarrow \sum_{r=1}^{n} \frac{1}{r} \approx \int_1^n \frac{dx}{x} = \log n$

$\longrightarrow n \log n$

```
b) for (i = 1; <= n; i++)
        for (j =1; j <= n; j= j+i)
        v ++
```

۳) در هر یک از رابطه‌های زیر به جای $\square$ کدام یک از نمادهای $o, O, \theta, \omega, \Omega$ را بگذاریم؟

$a)\ lglg^*n = \square\ lg^*lgn$

$b)\ \sum_{i=1}^{n} \sqrt{i} \ = \square\ n\sqrt{n}$

۴) درستی گزاره های زیر را مشخص کنید.

$a) \lg n \ \in \ O(n)$

$b)\ n \in \ O(n \lg n)$

$c) \mathrm{n} \lg n \ \in \ O(n^2)$

$d)\ 2^n \ \in \ \Omega(5^{\ln n})$

$e) \lg^3 n \ \in \ \mathrm{o}(n^{0.5})$

۵) روابط بازگشتی زیر را حل کنید. در تمامی موارد T(n) برای n<4 برابر مقدار ثابت ۱ است.

$a)\ T(n) = \ T(\sqrt{n}) + c$

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

$c)\ T(n) = \ 2T(\sqrt{n}) + \frac{\lg n}{\lg \lg n}$

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

$e)\ T(n) = \ T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n$

$f)\ T(n) = \ \sqrt{n}\, T(\sqrt{n}) + n$

۶) یک لیست شامل n عدد داریم. عدد x به ما داده می‌شود. الگوریتمی با مرتبه زمانی $\Theta(n \lg n)$ بیان کنید که مشخص کند آیا در این لیست دو عدد وجود دارند که مجموع آنها برابر x شود یا خیر.

3

b) $\sqrt{1}+\sqrt{2}+\sqrt{3}+\sqrt{4}+\dots+\sqrt{n}=X(n\sqrt{n})$

$\div\sqrt{n} \Rightarrow \underbrace{\sqrt{\frac{1}{n}}+\sqrt{\frac{2}{n}}+\sqrt{\frac{3}{n}}+\dots+\sqrt{\frac{n}{n}}}_{}=X(n)$

n عدد داریم

غیر از $\sqrt{\frac{n}{n}}$ همه از 1 کوچک ترند

$$\longrightarrow \sum_{i=1}^{n}\sqrt{i}=O(n\sqrt{n})$$

4

a) $\log n \in O(n) \equiv$ True

b) $n \in O(n\log n) \overset{n>2 \to \log n>1}{\equiv}$ True

c) $n\log n \in O(n^2) \xrightarrow{\div n} \log n \in O(n) \equiv$ True

d) $2^n \in \Omega(5^{\ln n}) = \Omega(2.5^{\ln n} \times 2^{\ln n}) \Leftrightarrow \frac{2^n}{2^{\ln n}} \in \Omega(2.5^{\ln n}) \Leftrightarrow 2^{n-\ln n} = 2^{\ln e^n - \ln n} = 2^{\ln \frac{e^n}{n}} \in \Omega(2.5^{\ln n})$

$\Leftrightarrow \ln\frac{e^n}{n} \in \Omega(\ln n) \Leftrightarrow \frac{e^n}{n} \in \Omega(n) \Leftrightarrow e^n \in \Omega(n^2) \equiv$ True

نمایی است و رشد سریع‌تری دارد

e)

$(\lg n)^3 \in O(n^{0.5}) \xleftrightarrow{\lg n = u \,\&\, 2^u = n} u^3 \in O(2^u)^{0.5} = O(2^{u/2})$

$\Leftrightarrow u^6 \in O(2^u) \equiv$ False

5

a)

$T(n) = T(\sqrt{n}) + c$

$u = \log n \Rightarrow 2^u = n$

$T(2^u) = T(2^{u/2}) + c$

$S(u) = T(2^u)$

$S(u) = S(u/2) + c \rightarrow S(u) = O(c \log u) = O(\log u)$

$\Rightarrow S(u) = O(\log u) \xrightarrow{S(u) = T(2^u)} T(2^u) = O(\lg u)$

$\xrightarrow{u = \log n \Rightarrow 2^u = n} T(n) = O(\lg \lg n)$

b)

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

$u = \lg n \rightarrow 2^u = n$

$\Rightarrow T(2^u) = 2T(2^{u/2}) + u$

$\Rightarrow T(2^u) = S(u)$

$\Rightarrow S(u) = 2S(u/2) + u = O(u \log u)$

$S(u) = O(u \log u) \Rightarrow T(2^u) = O(u \log u)$

$\Rightarrow T(n) = O(u \times \log u) = O(\lg n \times \lg \lg n)$

c) $T(n) = 2T(\sqrt{n}) + \frac{\lg n}{\lg\lg n}$

$n = 2^u \Rightarrow \lg n = u$

$\Rightarrow T(2^u) = 2T(2^{u/2}) + \frac{u}{\log u}$

$S(u) = T(2^u)$

$\Rightarrow S(u) = 2S(u/2) + \frac{u}{\log u} = 2\left(2S(u/4) + \frac{u}{2\log u/2}\right) + \frac{u}{\lg u}$

$= 4S(u/4) + \frac{u}{\lg u/2} + \frac{u}{\lg u} = 4\left(2S(u/8) + \frac{u}{4\lg u/4}\right) + \frac{u}{\lg u/2} + \frac{u}{\lg u}$

$= 8S(u/8) + \frac{u}{\lg u/4} + \frac{u}{\lg u/2} + \frac{u}{\lg u}$

$= uS(\frac{u}{u}) + \frac{u}{\lg 2} + \frac{u}{\lg 4} + \frac{u}{\lg 8} + \dots + \frac{u}{\lg u}$

$= u\underbrace{\left(1 + \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{\lg u}\right)}_{\text{behaves the same as harmonic series}} \quad k = \lg u$

$= O(u \lg k) \Rightarrow S(u) = O(u\lg\lg u) \Rightarrow T(2^u) = O(u\lg\lg u)$

$\Rightarrow T(n) = O(\lg n \times \lg\lg\lg n)$

d)

$T(n) = 3T(n/3) + \frac{n}{\lg n} = 3\left(3T(n/9) + \frac{n}{3\lg n/3}\right) + \frac{n}{\lg n}$

$= 3^k T(n/3^k) + \frac{n}{\lg \frac{n}{3^k}} + \frac{n}{\lg \frac{n}{3^{k-1}}} + \dots + \frac{n}{\lg \frac{n}{3^{k-k}}}$

$3^k \cong n$

$\lg_3 n = k$

$\xrightarrow{} n\left(1 + \frac{1}{\lg \frac{n}{3^{k-1}}} + \frac{1}{\lg \frac{n}{3^{k-2}}} + \dots + \frac{1}{\lg \frac{n}{3^{k-k}}}\right)$

$= n\left(1 + \frac{1}{\lg 3} + \frac{1}{\lg 9} + \dots + \frac{1}{\lg 3^k}\right) = n\left(1 + \frac{1}{\lg 3} + \frac{1}{2\lg 3} + \dots + \frac{1}{k\lg 3}\right)$

$= n \times \left(\frac{1}{\lg 3} \times \left(\lg 3 + 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{k}\right)\right)$

$$= n\left(\frac{1}{\lg 3} \times (\lg 3 + \overset{\lg_3 n}{\overbrace{k}})\right) = n\left(1 + \frac{\lg_3 n}{\lg 3}\right) = n\left(1 + \frac{\lg n}{\lg 3 \lg 3}\right)$$

$$= n + \frac{n \lg n}{\lg 3 \lg 3} = O(n + n \lg n)$$

e)

$$T(n) = T(n/2) + T(n/4) + T(n/8) + n$$

$$= \overbrace{(\ldots) + (\ldots) + (\ldots)}^{A_1} + \left(\frac{n}{2} + \frac{n}{4} + \frac{n}{8}\right) + n = A_1 + \frac{7}{8} n + n$$

$$A_1 = A_2 + \left(\frac{7}{8}\right)^2 n$$

$$\Rightarrow T(n) = \left(\frac{7}{8}\right)^0 n + \left(\frac{7}{8}\right)^1 n + \left(\frac{7}{8}\right)^2 n + \cdots \Rightarrow T(n) = \frac{1}{1 - \frac{7}{8}} n = 8n \Rightarrow T(n) = O(n)$$

f)

$$T(n) = \sqrt{n}\, T(\sqrt{n}) + n \xrightarrow{u = \lg n \ \& \ 2^u = n} T(2^u) = 2^{u/2} T(2^{u/2}) + 2^u \Rightarrow \frac{T(2^u)}{2^u} = \frac{2^{u/2} T(2^{u/2}) + 2^u}{2^u}$$

$$\Rightarrow \frac{T(2^u)}{2^u} = \frac{T(2^{u/2})}{2^{u/2}} + 1 \xrightarrow{S(u) = \frac{T(2^u)}{2^u}} S(u) = S(u/2) + 1$$

$$\longrightarrow \text{Master Theorem} \rightarrow \quad a = 1, \quad b = 2, \quad c = 1 \Rightarrow n^{\log_b a} = n^0 = 1 = c$$

$$\longrightarrow S(u) = O(c \lg u) = O(\lg u) \Rightarrow \frac{T(2^u)}{2^u} = O(\lg u)$$

$$\Rightarrow \frac{T(n)}{n} = O(\lg \lg n) \Rightarrow T(n) = O(n \lg \lg n)$$

۶) اگر آرایه مرتب نیست، آن را صعودی مرتب می‌کنیم. حال دو پوینتر می‌سازیم، یکی به اولین عدد و یکی به آخرین. در هر مرحله، اعدادی که پوینترها به آن‌ها اشاره می‌کنند را جمع می‌کنیم. اگر حاصل از n کمتر بود، پوینتر اول را یک واحد به جلو می‌بریم، اگر نه، پوینتر دوم را یکی به عقب می‌آوریم.

اگر هم که جمعشان برابر n بود، جفت مطلوب را یافته‌ایم.

اگر آن قدر ادامه دهیم که دو پوینتر به یک المان برسند، پس چنین دوعددی در این لیست نیستند.

۷) برای هرکدام از مسائل زیر الگوریتم مناسب را پیشنهاد داده و پیچیدگی آن را برای حالت کلی، بهترین و بدترین حالت به صورت جداگانه تحلیل کنید.

a) پیدا کردن یک عدد خاص از بین لیستی از اعداد
b) پیدا کردن kامین عدد بزرگ در یک لیست
c) پیدا کردن kامین عدد پرتکرار در یک لیست

تمرینات پیاده‌سازی

برای مشاهده سوالات پیاده سازی به سایت کوئرا مراجعه فرمایید.

لینک کلاس:

/https://quera.org/course/add_to_course/course/12639

رمز ورود: 0010

فرمت گزارش

* گزارش بایستی حاوی تمام نتایج بدست آمده از پیاده‌سازی‌ها در قالب فایل پی دی اف باشد. همچنین انتظار میرود که در این گزارش برای سوالات پیاده‌سازی، تحلیل خود را از نتایج به دست آمده ارائه دهید.

* فایل گزارش خود را تنها به شکل «StuNum_HWNum.pdf» نام گزاری کنید.(مانند 400123456_HW1.pdf)

تذکر

a) پیچیدگی زمانی الگوریتم های مربوط به این سوال، وابسته به ساختمان داده ای هستند که از آن برای ذخیره سازی اعدادمان استفاده کرده ایم.

اگر اعداد در یک آرایه ذخیره شده باشند، از ابتدای آرایه شروع به بررسی المان ها می کنیم، که در بدترین حالت، آخرین عددی که بررسی می کنیم، عدد مطلوب خواهد بود، که در این صورت $O(n)$ پیچیدگی آن خواهد بود. در بهترین حالت عدد مطلوب، اولین عضواست و در $O(1)$ آن را پیدا می کنیم. در average case هم در $O(n/2)$ به آن می رسیم.

ساختمان داده های مناسب تری برای این کار وجود دارد که به ما کمک می کند در زمان کمتری به پاسخ برسیم، برای مثال در یک BST در بدترین حالت در $O(n)$ و در حالت متوسط در $O(h)$ می توان به پاسخ رسید، که می دانیم در یک Balanced BST ارتفاع برابر $\log n$ خواهد بود.

b) برای یافتن k امین عدد بزرگ در یک لیست، یکی از راه حل ها این است که ابتدا آرایه را sort کنیم، سپس المان k ام (یا n-k ام) را برگردانیم. مرتب سازی را می توان در $O(n \log n)$ انجام داد و انتخاب یک عضو با اندیس هم در $O(1)$ انجام می شود.

راه حل دیگر این است که ابتدا آرایه را به یک Max Heap تبدیل می کنیم، که در $O(n)$ انجام می شود. سپس k دفعه max را از Heap انتخاب می کنیم (هرکدام $O(\log n)$) که در k امین انتخاب، عدد مطلوب بازگردانده می شود.

در نهایت، پیچیدگی زمانی $O(n+k\log n)$ خواهد بود.

یک راه حل دیگر این است که از Quick Sort استفاده می کنیم. در این sort، در هر مرحله یک pivot انتخاب

می‌کنیم که در پایان مرحله در جای درست خود قرار گرفته است. تا آن جایی به مرتب سازی ادامه می‌دهیم که یک عضو در اندیکس k ام قرار بگیرد.

C) یک دیکشنری می‌سازیم، سپس در $O(n)$ یک پیمایش روی المان ها می‌کنیم، و اعداد را به عنوان key و شمار هر یک را به عنوان value ذخیره می‌کنیم. سپس یک دور روی دیکشنری‌مان پیمایش می‌کنیم و در یک دیکشنری جدید کلیدها را تعداد تکرار هر عدد، و مقادیر (values) را آرایه‌ای از عددها با آن تعداد تکرار، قرار می‌دهیم، به ترتیب (sorted).

حال از ابتدا (یا از انتها، شروع به پیمایش می‌کنیم تا به k امین عضو پرتکرار برسیم. این الگوریتم در $O(n\log n)$ پاسخ می‌دهد.