بسمه تعالی

طراحی الگوریتم ها

پاسخنامه تمرین شماره یک

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

پاسخنامه سوالات تشریحی:

۱- هر کدام از معیارهای Big O, Big Omega , Theta در چه حالتهایی برای مقایسه دو یا چند الگوریتم مناسب می باشند؟ (۱۰ امتیاز)

۱- Big O :

معنای $g(n) = O(f(n))$ یعنی رشد الگوریتم را نشان می دهد. یعنی نرخ رشد الگوریتم کمتر یا مساوی $f(n)$ است. برای نشان دادن حد بالای الگوریتم از Big O استفاده می شود. (راجع به حد پایین حرفی نمی زند)، به طور کلی در مقایسه با theta و Ω، worst case را در نظر می گیرد و معیاری برای طولانی ترین زمان ممکن برای اجرای الگوریتم است.

$$g(n) = O(f(n)) \Leftrightarrow \exists\, n_0, c > 0 \quad s.t. \quad \forall n \geq n_0 \;,\; g(n) \leq c.f(n)$$

۲- Big Omega : $g(n) = \Omega(f(n))$ یعنی نرخ رشد الگوریتم بزرگتر یا مساوی $f(n)$ است و حداقل (مینیموم) رشد الگوریتم را نشان می دهد. برای نشان دادن حد پایین الگوریتم از Big Omega استفاده می شود (راجع به حد بالا حرفی نمی زند) به طور کلی در مقایسه با Big O و theta، best case را در نظر می گیرد. نسبت به زمان مورد نیاز Big O (برای تکمیل الگوریتم)، مقدار کمتری زمان می برد. برای نمایش حد پایین بیشتر مشکل کاربرد دارد. (پیچیدگی مسئله همواره از $f(n)$ بیشتر یا مساوی آن است.)

$$g(n) = \Omega(f(n)) \Leftrightarrow \exists\, c, n_0 > 0 \quad s.t. \quad \forall n \geq n_0 \;,\; g(n) \geq c.f(n)$$

۳- theta :

مقدار نرخ رشد الگوریتم را نشان می دهد (مشابه $=$) یعنی نرخ رشد $g(n) = \Theta(f(n))$ الگوریتم مساوی $f(n)$ است. برای نشان دادن هر دو حد پایین و بالای الگوریتم استفاده می شود (هر دو را که با هم برابرند نشان می دهد) رفتار دقیق رشد الگوریتم را مشخص می کند به طور کلی در مقایسه با Big O و Ω، Average case و زمان کلی (به طور میانگین) را در نظر می گیرد. به طور کلی زمان متوسط در مقایسه با Big O, Ω برای اتمام الگوریتم صرف می کند.

$$g(n) = \Theta(f(n)) \Leftrightarrow \exists\, c_1, c_2, n_0 > 0 \quad s.t. \quad \forall n \geq n_0 \;,$$
$$c_1.f(n) \leq g(n) \leq c_2.f(n)$$

۲- به نظر شما علاوه بر معیارهای گفته شده در سوال ۱، چه معیارهای دیگری نیز می توان تعریف کرد که پیچیدگی الگوریتم ها را با یکدیگر مقایسه نمایند؟ پیچیدگی مکانی(حافظه) در چه نوع مسائلی اهمیت دارد؟ (۱۰ امتیاز)

پیچیدگی مکانی، میزان قابلیت موازی سازی، توزیع داده های ورودی برای یک کاربرد خاص و ...

بسمه تعالی

طراحی الگوریتم ها

پاسخنامه تمرین شماره یک

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

اهمیت حافظه برای الگوریتم هایی که درجا نمی باشند، یا از روشهای خلاصه سازی و پیش پردازش یا پردازش بخشی از جریان داده ها استفاده می کنند بسیار بیشتر است.

۳- برای حل یک مساله دو الگوریتم طراحی کرده ایم. زمان لازم برای اجرای الگوریتم اول بر روی یک کامپیوتر خاص متناسب با $4n^4 + 2n^2$ عمل جمع و برای الگوریتم دوم متناسب با $8n^2$ عمل ضرب است. اگر هر عمل ضرب ۵ برابر یک عمل جمع هزینه داشته باشد و الگوریتم اول مساله ای به اندازه ۱۰۰ را در واحد زمان حل کند، الگوریتم دوم در واحد زمان مساله ای به چه اندازه را حل خواهد کرد؟ کدام الگوریتم برای حل این مساله مناسب تر است؟ دلایل خود را بیان کنید. (۱۰ امتیاز)

4*(100^4)+2*(100^2)=5*8*(n^2) → n=3162.36

علی رغم اینکه الگوریتمی که از ضرب استفاده می کند برای هر عمل ضرب هزینه بالاتری نسبت به عمل جمع دارد، اما در یک واحد زمان قادر به حل مساله ای با اندازه های n بزرگتری است، در صورتیکه n را تا حد زیادی افزایش دهیم (n>5) الگوریتم دوم سریعتر خواهد بود.

۴- برای هر کدام از مسائل زیر الگوریتم مناسب را پیشنهاد داده و پیچیدگی آن را برای سه حالت: حالت کلی، بهترین و بدترین حالت به صورت جداگانه تحلیل کنید.

مرتب بودن یا نبودن آرایه اولیه، همچنین وجود یا عدم وجود اعداد تکراری تاثیر زیادی در انتخاب روش حل مناسب خواهد داشت. مساله ها را با فرض های مختلف حل می کنیم.

a. پیدا کردن یک عدد خاص از بین لیستی از اعداد (۱۰ امتیاز)

i. با فرض مرتب بودن از جستجوی باینری استفاده می کنیم.

۱. در بهترین حالت در اولین مقایسه عدد پیدا می شود.
۲. در بدترین حالت عدد در آرایه موجود نیست و بعد از Log n بار مقایسه متوجه می شویم.
۳. به صورت کلی اگر احتمال هر حالتی را برابر درنظر بگیریم، ممکن است در اولین مقایسه به جواب دست یابیم یا با تعداد ۱ تا Log n بار مقایسه به جواب دست یابیم. با درنظر گرفتن دامنه های مختلف برای عددی که در جستجوی آن هستیم تحلیل های مختلفی می توان ارائه کرد. به عنوان نمونه اگر احتمال پیدا شدن یا نشدن عدد را برابر درنظر بگیریم و همچنین در صورت پیدا شدن عدد احتمال مساوی برای اینکه با ۱ تا Log n مقایسه پیدا شود را لحاظ کنیم به صورت میانگین با (3/4)*Log n مقایسه خواهیم توانست به مساله پاسخ دهیم.

i. با فرض مرتب نبودن از جستجوی خطی استفاده می کنیم.

۱. بهترین حالت یک مقایسه
۲. بدترین حالت n مقایسه
۳. به صورت کلی یا همان میانگین، n/2 مقایسه لازم داریم.

b. پیدا کردن kامین عدد بزرگ در یک لیست(۱۰ امتیاز)

بسمه تعالی

طراحی الگوریتم ها

پاسخنامه تمرین شماره یک

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

در این مساله نیز تنوع زیادی از روشها با توجه به فرضیات اولیه وجود دارد. بدترین و بهترین حالت ها نیز همچنین علاوه بر موارد مطرح شده در قسمت قبل تحت تاثیر مقدار k خواهند بود. به عنوان مثال الگوریتمی را درنظر بگیرید که ابتدا آرایه را با پیچیدگی O(n Log n) مرتب کرده و سپس به صورت خطی به دنبال kامین عدد بزرگ می گردد. این الگوریتم در تمامی حالات هزینه اولیه‌ای برابر با n Log n داشته و در بهترین حالت ۱ جستجو ، در بدترین حالت n جستجو و به صورت میانگین n/2 جستجو برای یافتن پاسخ نیاز داریم.

c. پیدا کردن اولین عدد پر تکرار در یک لیست(۱۰ امتیاز)

الگوریتم نمونه: ابتدا آرایه را مرتب کرده و سپس با یک بار پیمایش تمامی اعضای آرایه عضو پر تکرار را می یابیم. در هر صورت باید هزینه مرتب سازی و یک بار پیمایش کامل پرداخت شود.

d. پیدا کردن kامین عدد پر تکرار در یک لیست(۱۰ امتیاز)

الگوریتم نمونه: مشابه روش قبل آرایه مرتب شده و آرایه جدید شامل مقادیر و تعداد تکرار آنها ثبت می گردد. این هزینه ها الزامی است. سپس بر روی آرایه جدید از الگوریتم قسمت b استفاده می کنیم.

* برای تمامی پاسخ هایی که فرضیات مختلفی را درنظر گرفته اند به صورت مجزا بررسی و نمره دهی شده است. الگوریتم های معرفی شده در این پاسخنامه صرفا جنبه معرفی داشته و پاسخ های صحیح دیگری نیز وجود دارد.

۵- درستی گزاره های زیر را مشخص کنید: (۲۰ امتیاز)

a. $\lg n \in O(n)$
b. $n \in O(n \lg n)$
c. $n \lg n \in O(n^2)$
d. $2^n \in \Omega(5^{\ln n})$

a) $c=1,\ n_0=1 \Rightarrow \forall n \geq n_0 \Rightarrow \lg(n) \leq cn$ ✓

b) $c=1,\ n_0=2 \Rightarrow \forall n \geq n_0 \Rightarrow 1 \leq \lg(n) \Rightarrow n \leq cn\lg(n)$ ✓

c) $c=1, n_0=1 \Rightarrow \forall n \geq n_0 \Rightarrow \lg(n) \leq cn \Rightarrow n\lg(n) \leq cn^2$ ✓

d) $c=1,\ n_0=7 \Rightarrow \forall n \geq n_0 \Rightarrow 2^n \geq cn^{\lg 5} \Rightarrow 2^n \geq c\,5^{\lg n}$ ✓

بسمه تعالی

طراحی الگوریتم ها

پاسخنامه تمرین شماره یک

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

۶- روابط بازگشتی زیر را حل کنید. در تمامی موارد T(n) برای n<4 برابر مقدار ثابت 1 است: (۶۰ امتیاز) (هر آیتم ۱۰ امتیاز)

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

a. $T(n) = T(\sqrt{n}) + c$ $\quad m := \lg(\lg(n)) \Rightarrow n = 2^{2^m}$

$\Rightarrow T(2^{2^m}) = T(2^{2^{m-1}}) + c = T(2^{2^{m-2}}) + 2c = \cdots$

$= T(2)^{1} + mc = c\lg(\lg(n)) + 1$

$\Rightarrow T(n) \in \theta(\lg(\lg(n)))$

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

b. $T(n) = 2T(\sqrt{n}) + \lg n$ $\quad m := \lg(\lg n) \Rightarrow n = 2^{2^m}$

$\Rightarrow T(2^{2^m}) = 2T(2^{2^{m-1}}) + 2^m = 4T(2^{2^{m-2}}) + 2 \times 2^m$

$= \cdots = 2^m T(2)^{1} + m \times 2^m = 2^m(1+m) = \lg n(1 + \lg\lg n)$

$\Rightarrow T(n) = \lg n(1 + \lg\lg n)$ $\quad \Rightarrow T(n) \in \theta(\lg n \times \lg\lg n)$

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

تغییر متغیر $\rightarrow n = 2^m$

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

تغییر متغیر $\rightarrow T(2^m) = S(m)$

$\Rightarrow S(m) = 2S(\frac{m}{2}) + \frac{m}{\lg m}$

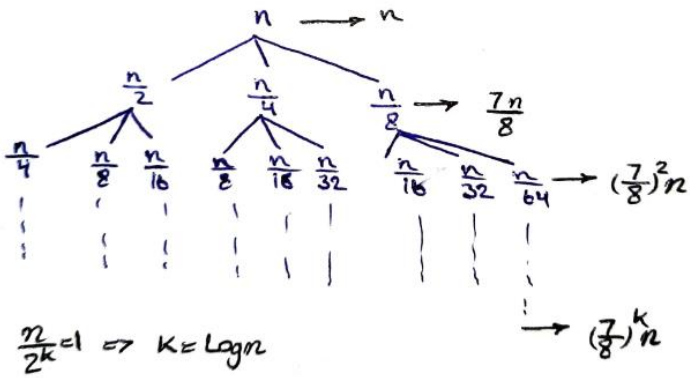
حال این مساله جوابی مشابه با قسمت d دارد. ضمنا از قضیه اصلی قابل حل نیست.

S(m) = 2S(m/2) + m/log(m)
= 2(2S(m/4) + m/2/log(m/2)) + m/log(m)
= 4S(m/4) + m/log(m/2) + m/log(m)
= 4(2S(m/8) + m/4/log(m/4)) + m/log(m/2) + m/log(m)
= 8S(m/8) + m/log(m/4) + m/log(m/2) + m/log(m)
= 16S(m/16) + m/log(m/8) + m/log(m/4) + m/log(m/2) + m/log(m)
= m * S(1) + m/log(2) + m/log(4) + ... + m/log(m/2) + m/log(m)
= m(1 + Sum[i = 1 to log(m)](1/log(2^i)))
= m(1 + Sum[i = 1 to log(m)](1/i))
~= m(1 + log(log(m)))
= m + m*log(log(m)))
~= m*log(log(m))
T(n)=log (n)* log(log(log(n)))

d. $T(m) = 3T\left(\frac{m}{3}\right) + \frac{m}{\lg m}$

T(m) = 3T(m/3) + m/log(m)
= 3(3T(m/9) + m/3/log(m/3)) + m/log(m)
= 9T(m/9) + m/log(m/3) + m/log(m)
= 9(3T(m/27) + m/9/log(m/9)) + m/log(m/3) + m/log(m)
= 27T(m/27) + m/log(m/9) + m/log(m/3) + m/log(m)
= 81T(m/81) + m/log(m/27) + m/log(m/9) + m/log(m/3) + m/log(m)
= m * T(1) + m/log(3) + m/log(9) + ... + m/log(m/3) + m/log(m)
= m(1 + Sum[i = 1 to log(m)](1/log(3^i)))
= m(1 + Sum[i = 1 to log(m)](1/i))*log 3
~= m(1 + log(log(m)))
= m + m*log(log(m)))
~= m*log(log(m))
T(m)=m* log(log(m))

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



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

$T(n) = \left[1 + \left(\frac{7}{8}\right)^1 + \left(\frac{7}{8}\right)^2 + \cdots + \left(\frac{7}{8}\right)^{\log n}\right] \cdot n = n$

$\Rightarrow \boxed{T(n) = \Theta(n)}$

f. $T(n) = \sqrt{n}T\left(\sqrt{n}\right) + n$

تغییر متغیر $\rightarrow n = 2^k$, $\sqrt{n} = 2^{k/2}$

$\Rightarrow T(2^k) = 2^{k/2} \cdot T(2^{k/2}) + 2^k \;/\div 2^k$

$\frac{T(2^k)}{2^k} = \frac{2^{k/2} \cdot T(2^{k/2})}{2^k} + 1$

تغییر متغیر $\rightarrow y(k) = \frac{T(2^k)}{2^k}$

$\Rightarrow \boxed{y(k) = y\left(\frac{k}{2}\right) + 1}$

$\begin{cases} a = 1 \\ b = 2 \\ d = 0 \end{cases}$ $a = b^d \Rightarrow 1 = 2^0 \Rightarrow$ حالت ۱

پس : $y(k) = k^d \cdot \log k = \log k$

طرفین : $T(2^k) = 2^k \cdot y(k)$

آنگاه $\Rightarrow T(2^k) = 2^k \cdot \log k \Rightarrow T(n) = n \log \log n$

$\Rightarrow \underline{T(n) = \Theta(n \log \log n)}$

بسمه تعالی

طراحی الگوریتم ها

پاسخنامه تمرین شماره یک

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

۷- دو آرایه مرتب شده از اعداد داریم (m, n عضوی). بهینه‌ترین الگوریتم را برای پیدا کردن عنصر k ام در ترکیب مرتب شده این دو آرایه پیشنهاد کنید. الگوریتم را از نظر پیچیدگی زمانی و حافظه ای تحلیل کرده و بهینه بودن آن را اثبات کنید. (۱۵ امتیاز)

از آنجایی که هر دو آرایه مرتب هستند لذا kامین عنصر حتما در بین kعضو ابتدایی یکی از دو آرایه قرار دارد. لذا از الگوریتم ادغام دو آرایه مرتب استفاده کرده و به محض رسیدن به kامین عضو ادغام را متوقف و عضو را اعلام می کنیم. پیچیدگی زمانی از O(k) خواهد بود. با لحاظ کردن شرایط خاص می توان در O(min{m,n}) نتیجه مورد نظر را بدست آورد.

۸- جدولی به ابعاد m x n برای ذخیره اعداد صحیح داریم. همواره هر یک از ستون ها از بالا به پائین به صورت صعودی مرتب هستند. همچنین همواره هر یک از سطرها از چپ به راست به صورت صعودی مرتب هستند. برخی از خانه های جدول خالی هستند. **برای هر الگوریتم بعد از توصیف و اثبات درستی، پیچیدگی الگوریتم را نیز تحلیل کنید.** (۳۰ امتیاز)

a. الگوریتمی بهینه طراحی کنید که یک عدد جدید را در جدول درج کند.
b. الگوریتمی بهینه طراحی کنید که کوچکترین عدد داخل جدول را یافته و حذف کند.
c. الگوریتمی بهینه طراحی کنید که عددی را گرفته و در جدول جستجو کند و وجود یا عدم وجود آن عدد را در جدول مشخص کند.

در این سوال محدودیتی برای اینکه حتما همه خانه ها پر شوند وجود ندارد. یعنی الگوریتم شما می تواند به صورت بهینه از تمامی خانه ها استفاده نکند اما به هیچ وجه نباید ساختار صعودی سطرها و ستونها را از بین ببرد. همچنین شرایط حاکم بر جدول موجب می شود که هر عضو x_{ij} که در خانه سطر iام و ستون jام قرار دارد، حتما از تمامی اعضای جدول که شماره شطرهای آنها بزرگتر از i و شماره ستون های آنها بزرگتر از j هستند کوچکتر خواهد بود. در بسیاری از ساختمان داده های پیچیده، همواره تعدادی عملیات مختلف همچون درج، حذف، بروزرسانی و ... برای ساختمان داده تعریف می شود. طراحی ساختمان داده معمولا به گونه ای است که الگوریتم های یک یاچند عملیات از نظر هزینه و پیچیدگی زمانی ممکن است برخلاف یکدیگر عمل کنند. یعنی مجبور باشیم یک عملیات که احتمالا کمتر از بقیه مورد استفاده قرار می گیرد را با روشی پر هزینه تر پیاده کنیم تا برای عملیات هایی که بیشتر تکرار خواهند شد هزینه کمتری بپردازیم. برقراری تعادل بین پیاده سازی عملیات های مختلف یکی از سخت ترین فعالیتهای طراحی یک ساختمان داده است.

بسمه تعالی

طراحی الگوریتم ها

پاسخنامه تمرین شماره یک

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

الگوریتم نمونه: اگر کل جدول را به صورت قطری و با شروع از خانه x_{00} پیمایش کرده و به صورت یک آرایه خطی ذخیره کنیم و همواره اعداد را به صورت صعودی در این آرایه ذخیره کنیم علاوه بر برآورده شدن شروط مربوط به جدول، الگوریتم های خواسته شده نیز قابل پیاده سازی خواهند بود.

۱	۲	۴	۷	۱۱	۱۶
۳	۵	۸	۱۲	۱۷	۲۲
۶	۹	۱۳	۱۸	۲۳	۲۷
۱۰	۱۴	۱۹	۲۴	۲۸	۳۱
۱۵	۲۰	۲۵	۲۹	۳۲	۳۴
۲۱	۲۶	۳۰	۳۳	۳۵	۳۶

ترتیب پیمایش جدول

جهت درج یک عدد جدید ابتدا آن را در آخرین مکان خالی در آرایه ذخیره کرده و سپس رو به عقب حرکت کرده و در اولین محل مناسب جایگذاری می کنیم. با توجه به ترتیب ورود اعداد در بدترین حالت پیچیدگی این عملیات برابر با کل اعداد وارد شده تا آن لحظه است.

جهت حذف کوچکترین عدد کافی است خانه ۱ را خالی کنیم. پیچیدگی آن مانند الگوریتم قبلی است.

با توجه به مرتب بودن جدول می توان از الگوریتم های جستجوی باینری برای یافتن عدد مورد نظر استفاده کرد.

* این تنها الگوریتم ممکن نیست و بسیاری از الگوریتم های مطرح شده در پاسخنامه ها صحیح هستند.