1

به طور کلی به شکل زیر حل میشود

۱. یک سکه سنگین در log(n) پیدا میشود
۲. اگر تعداد سکه های سنگین و تعداد سکه های سبک زوج باشند ، سکه ها را به دو دسته هم وزن تقسیم کنید.
۳. در حالت کلی بفهمید که آیا تعداد سکه های سبک و سنگین زوج هستند و اگر زوج نبودند یکی را حذف کنید و دو دسته با وزن برابر بسازید
۴. T(n) = T(n/2) +O(logn) = > O(log(n)^2)

۲) اثبات کنید که $O(n\lg n)$ جواب $T(n) = T(\lfloor n/2 \rfloor + 17) + n$ است.(۰.۵ نمره)

$$\exists n_0 \in \mathbb{N}.\ \forall\, n \geq n_0 \quad T(n) \leq c(n-d)\log(n-d) \leq cn\log n \qquad c, d > 0$$

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor + 17\right) + n \ \leq c\left(\left(\left\lfloor \frac{n}{2} \right\rfloor + 17 - d\right)\log\left(\left\lfloor \frac{n}{2} \right\rfloor + 17 - d\right)\right) + n$$

$$\xrightarrow{\left\lfloor \frac{n}{2} \right\rfloor \leq \frac{n}{2}} T(n) \leq \ c\left(\left(\frac{n}{2} + 17 - d\right)\log\left(\frac{n}{2} + 17 - d\right)\right) + n$$

$$= c\left(\frac{n + 34 - 2d}{2}\right)\log\left(\frac{n + 34 - 2d}{2}\right) + n$$

$$T(n) \leq \ c\left(\frac{n + 34 - 2d}{2}\right)\log(n + 34 - 2d) \ - \ c\left(\frac{n + 34 - 2d}{2}\right)\log(2) + n$$

$$\xrightarrow{d \geq 34} T(n) \leq \ c\left(\frac{n - d}{2}\right)\log(n - d) \ - \ c\left(\frac{n - d}{2}\right)\log(2) + n$$

$$T(n) \leq \ c\left(\frac{n - d}{2}\right)\log(n - d) \ - \ c\left(\frac{n - d}{2}\right) + nT(n)$$

$$\leq \ c(n - d)\log(n - d) \ - \ \frac{c}{2}(n - d) + n$$

$$\xrightarrow{n_0 \geq d} T(n) \leq \ c(n - d)\log(n - d) \ + n$$

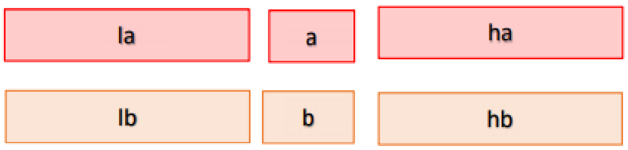
$$T(n) \leq c(n - d)\log(n - d) \leq n\log n \qquad\qquad c > 0\,,\ n_0 \geq d \ \geq 34$$

$$\Longrightarrow T(n) \ \leq \ \ cn\log n$$

$$\Longrightarrow T(n) = O(n\log n)$$

۳) دو آرایه مرتب شده از اعداد داریم .الگوریتمی با زمان $O(\log(n+m))$ برای پیدا کردن عنصر k ام در ترکیب مرتب شده این دو آرایه پیشنهاد کنید.(۱ نمره)

دو آرایه داریم به نام A و B عناصر وسط این دو آرایه را در نظر می‌گیریم نام آن ها را a و b در نظر می‌گیریم درنتیجه شکل کلی آرایه ما به این صورت خواهد بود



فرض میکنیم که $a < b$ در غیر این صورت جای این دو را عوض می‌کنیم

دو حالت پیش می‌آید:

۱- $k <= la.length + lb.length + 1$ باشد:

در این حالت عنصر k ام نمی‌تواند در hb باشد چرا که عناصر داخل hb از b و lb بزرگ تر هستند و از آنجایی که $a < b$ است عناصر داخل hb از a و lb نیز بزرگ تر هستند در نتیجه کوچک ترین عضو داخل hb از $la.length + lb.length + 2$ شروع می‌شود. پس اگر این حالت اتفاق بیافتد hb را حذف می‌کنیم و B را آپدیت می‌کنیم که دیگر hb داخل آن نباشد.

۲- $k > la.lenght + lb.length + 1$ باشد:

در این صورت عنصر k نمی‌تواند در la باشد. عناصر موجود در la هم از a کوچک‌تر هستند هم از b در نتیجه حداکثر می‌توانند $la.length + lb.length + 1$ را داشته باشند پس اگر این شرط برقرار باشند در نتیجه la را از A حذف میکنیم. و $k = k\text{-}la.length$ میگذاریم

در نهایت نیز این دو آرایه را آپدیت می‌کنیم و همین الگوریتم را دوباره روی آن اجرا میکنیم تا فقط a و b بمانند.

از آنجایی که هر مرحله ¼ از مجموع دو آرایه را حذف میکنیم در زمان ما از اردر $O(\log(m+n))$ خواهد بود.

1)

حدس می زنیم :

T(n)=$\theta(n \log_3 \log_3 n)$

برای کران بالا ابتدا اثبات میکنیم. اگر

$k<n \Longrightarrow T(k) \leq ck \log_3 \log_3 k - k$

$$T(n) \leq 3(c\frac{n}{3}\log_3 \log_3 \frac{n}{3} - \frac{n}{3}) + \frac{n}{\lg n}$$

$$\leq cn \log_3 \log_3 n - n + \frac{n}{\lg n}$$

$$\leq cn \log_3 \log_3 n \qquad \text{if n is large enough}$$

حد پایین به صورت مشابه اثبات می شود

2) $T(n) = 4T(n/2) + n^2\sqrt{n}$ $\qquad \Rightarrow T(n) = \theta(n^2\sqrt{n})$

$f(n) = n^2\sqrt{n},$

$a = 4, b = 2 \Rightarrow n^{\log_b a} = n^2$

$\exists c > 1\ \forall n > n_0 . \quad 4 * f\left(\frac{n}{2}\right) < c\, f(n) \Rightarrow 4\left(\frac{n}{2}\right)^2 \sqrt{\frac{n}{2}} = \frac{1}{\sqrt{2}} * n^2\sqrt{n} < c\, n^2\sqrt{n}$

$\Rightarrow$ for $c > \frac{1}{\sqrt{2}} \longrightarrow$ منظم

$\Rightarrow f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) = \Omega(n^2) \Rightarrow T(n) = \theta(f(n)) = \theta(n^2\sqrt{n})$

3) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$ $\qquad \Rightarrow T(n) = \theta(n)$

$$T(n) \leq cn \Rightarrow T(n) \leq c\left(\frac{n}{2}\right) + c\left(\frac{n}{4}\right) + c\left(\frac{n}{8}\right) + n$$

$$= n\left(\frac{c}{2} + \frac{c}{4} + \frac{c}{8} + 1\right) \xRightarrow{c \geq 8} T(n) \leq cn$$

Similarly :

$$T(n) \geq cn \Rightarrow T(n) \geq n\left(\frac{c}{2} + \frac{c}{4} + \frac{c}{8} + 1\right) \xRightarrow{c \leq 8} T(n) \geq cn$$

$$\Rightarrow T(n) = \theta(n)$$

4) $T(n) = T(n\text{-}2) + 1/\lg n$ $\Rightarrow T(n) = \theta(\frac{n}{\log n})$

$$T(n) = T(n-2) + 1/\log n = T(n-4) + 1/\log(n-2) + 1/\log n$$
$$= \cdots$$

$$T(n) = T(n-2r) + \sum_{0\le 2s<2r} \frac{1}{\log(n-2s)} \Rightarrow \begin{cases} T(0) + \sum_{k=1}^{\frac{n}{2}} \frac{1}{\log(2k)} & \text{n is even} \\ T(1) + \sum_{k=1}^{\frac{n}{2}-1} \frac{1}{\log(2k+1)} & \text{n is odd} \end{cases}$$

$$\sum \frac{1}{\log(n)} \le \int \frac{1}{\ln x} dx = \frac{x}{\ln x} \Rightarrow T(n) = \theta(\frac{n}{\log n})$$

5) $T(n) = \sqrt{n}\, T(\sqrt{n}) + n$ $\Rightarrow T(n) = \theta(n \log(\log(n)))$

$$T(n) = \sqrt{n}\left(\sqrt[4]{n}\left(\sqrt[8]{n}\, T\left(\sqrt[8]{n}\right) + \sqrt[4]{n}\right) + \sqrt{n}\right) + n$$

$$\xrightarrow{\text{smallest answer of } n^{\frac{1}{2^k}}<2} T(n) = n^{1-\frac{1}{2^k}}\, T(2) + n(1+k)$$

$$n^{\frac{1}{2^k}} < 2 \Rightarrow 2^k \ge \log(n-2) \Rightarrow k = \theta(\log(\log(n)))$$

$$\xRightarrow{n^{1-\frac{1}{2^k}} < n} T(n) \le n(T(2) + 1 + k) \Rightarrow T(n) = O(kn) = O(n \log(\log(n)))$$

$$\xRightarrow{n^{1-\frac{1}{2^k}}\, T(2) > 0} T(n) \ge n(1+k) \Rightarrow T(n) = \Omega(kn) = \Omega(n \log(\log(n)))$$

$$\Rightarrow T(n) = \theta(kn) = \theta(n \log(\log(n)))$$

الف) برای این کار ابتدا یک متغییر تعریف می‌کنیم و مقدار اولیه آن را صفر قرار می‌دهیم. این متغیر آخرین ارتفاع دیده شده از سایه هاست.

بعد از آن به ترتیب چپ ترین عنصر هر دو آرایه را با یکدیگر مقایسه می‌کنیم و آن عنصری را انتخاب می‌کنیم که x کمتری دارد

مقدار y نقطه انتخاب شده را با ارتفاع نقطه قبلی انتخاب شده مقایسه میکنیم اگر ارتفاع قبلی برای همین آرایه بود مقدار (x,y) را به آرایه جدید اضافه می‌کنیم در غیر این صورت مقدار $(x, \max(y, y_{prev}))$ را به آرایه اضافه می‌کنیم

در این حالت متغیر ارتفاع آخرین سایه را آپدیت می‌کنیم و مشخص می‌کنیم برای کدام آرایه بوده است

اگر x در هردو آرایه یکسان بود آن نقطه ای را برمیداریم که y بزرگ تری دارد

ایندکس آن آرایه ای که x را از آن انتخاب کردیم را یک عدد جلو میبریم

تمام این مراحل را تا آخر دوباره انجام می‌دهیم

ب) ایده الگوریتم شبیه merge sort است. برای حل این مسئله از divide and conquer استفاده می‌کنیم. آرایه مستطیل ها را به دو بخش تقسیم می‌کنیم آرایه سایه را برای هرکدام به صورت بازگشتی بدست می‌آوریم و با استفاده از الگوریتم قسمت الف هر دو آرایه سایه را با یکدیگر ترکیب میکنیم اردر زمانی این الگوریتم به این صورت خواهد بود

$T(n) = 2T(n/2) + O(n)$ $\qquad$ $T(n) = n\log n$

در هر سطر از heap به ترتیب :1,2,4,8,16 عنصر به طور ماکسیمم وجود دارد.همچنین تعداد برگ ها نیز $n/2$ میباشد.ارتفاع یک نود همان فاصله نود از یک برگ است.ارتفاع یک برگ نیز 0 می باشد و ارتفاع ریشه نیز همان ارتفاع درخت است.پس برای برگ ها ماکسیمم تعداد عنصرها طبق این فرمول میشود: $n/2 = \lceil n/2^{h+1} \rceil = \lceil n/2^{0+1} \rceil$

برای ریشه نیزارتفاع logn تعداد عناصر به این صورت است: $\lceil n/2^{h+1} \rceil = \lceil n/2^{\log n+1} \rceil = \lceil n/n+1 \rceil = 1$

فرض میکنیم این گزاره برای h-1 درست است.n_h تعداد عناصر در ارتفاع h میباشد و heap به نام T' همان heap به نامT با حذف برگ های آن است.پس T' دارای $n' = n - \lceil n/2 \rceil$ است که میشود $\lfloor n/2 \rfloor$.باید توجه کرد که عنصر با ارتفاع h در T میتواند با ارتفاع h-1 در T' باشد.n'_{h-1} تعداد عناصر در ارتفاع h-1 در T' را نشان میدهد.طبق نکات گفته شده داریم $n'_{h-1} = n_h$. طبق استقرا داریم:

$$n_h = n'_{h-1} = \lceil n'/2^h \rceil = \lceil \lfloor n/2 \rfloor / 2^h \rceil \le \lceil (n/2)/2^h \rceil = \lceil n/2^{h+1} \rceil$$

۷) الگوریتم HEAPSORT روی آرایه A با طول n که از قبل به طور صعودی مرتب شده در چه زمانی اجرا می‌شود؟ اگر به طور نزولی مرتب شده باشد چطور؟(هر کدام ۱ نمره)

هردو $\theta(n \log n)$ هستند.

مرتب شده به صورت صعودی:

BUILD_MAXHEAP در آن $O(n)$ چون آرایه مرتب شده است ولی در مقابل MAX_HEAPIFY آن n-1 بار صدا زده می‌شود که هر بار آن lgk مرحله طول میکشد.

$$\sum_{i=1}^{n-1} \log k = \log((n-1)!) = \theta(n \log n)$$

مرتب شده به صورت نزولی:

MAXHEAP در آن $O(n)$ چون آرایه مرتب شده است ولی در مقابل به خاطر حلقه درون خود HEAPSORT، باز هم زمان اجرا $\theta(n \log n)$ خواهد بود.

۸) یک الگوریتم از مرتبه O(nlgk) برای merge کردن k لیست مرتب شده به یک لیست مرتب شده ارائه دهید. n تعداد کل عناصر لیست ها است. (راهنمایی: از Minheap استفاده کنید)(۰,۵ نمره امتیازی)

۱- عناصر اول هر آرایه را برمیداریم و با همه آن ها یک Minheap درست می‌کنیم O(k)

۲- سپس ریشه Minheap را برمیداریم و در خروجی چاپ می‌کنیم و از Minheap نیز آن را حذف می‌کنیم

۳- از همان آرایه ای که ریشه عضو آن بود عنصر بعدی را برمیداریم و جای ریشه میگذاریم O(1)

۴- الگوریتم heapify را بر روی heap صدا میزنیم O(logk)

۵- مراحل بالا را تکرار می‌کنیم

اندازه minheap از k بیشتر نمی‌شود در نتیجه heapify همیشه O(logk) است و هر باری که عضوی را در خروجی می‌نویسیم و آن را از minheap حذف می‌کنیم یکبار باید heapify را صدا بزنیم در نتیجه کل زمان این الگوریتم O(nlogk) خواهد بود

در این سوال ابتد درخت بازگشت را رسم میکنیم.

این درخت به این صورت است که هر گره آن دو شاخه میشود که یکی -۱ شده‌ی والد است و دیگری نصف والد ... تا جایی که به $\mathrm{T}(1)$ برسیم.

سپس برای چند طبقه آن مجموع را حساب میکنیم تا بتوانیم پاسخ را حدس بزنیم.

$$h=0 \rightarrow n$$

$$h=1 \rightarrow (n-1) + \frac{n}{2}$$

$$h=2 \rightarrow (n-2) + \left(\frac{n-1}{2}\right) + \left(\frac{n}{2}-1\right) + \frac{n}{4} = (n-2) + \frac{(n-1)+(n-3)}{2} + \frac{n}{4}$$

$$h=3 \rightarrow (n-3) + \left(\frac{n-2}{2}\right) + \left(\frac{n-3}{2}\right) + \left(\frac{n-1}{4}\right) + \left(\frac{n-4}{2}\right) + \left(\frac{n-2}{4}\right) + \left(\frac{n-1}{4}\right) + \frac{n}{8}$$

$$= (n-3) + \frac{(n-2)+(n-3)+(n-4)}{2} + \frac{(n-1)+(n-2)+(n-3)}{4} + \frac{n}{8}$$

$$h=4 \rightarrow (n-4) + \frac{(n-3)}{2} + \left(\frac{n-4}{2}\right) + \frac{(n-2)}{4} + \left(\frac{n-5}{2}\right) + \left(\frac{n-3}{4}\right) + \left(\frac{n-5}{4}\right) + \left(\frac{n-1}{8}\right) + \left(\frac{n-6}{2}\right) + \left(\frac{n-4}{4}\right)$$

$$+ \left(\frac{n-6}{4}\right) + \left(\frac{n-2}{8}\right) + \left(\frac{n-8}{4}\right) + \left(\frac{n-4}{8}\right) + \left(\frac{n-8}{8}\right) + \frac{n}{16}$$

$$= (n-4) + \frac{(n-3)+(n-4)+(n-5)+(n-6)}{2} + \frac{(n-2)+(n-3)+(n-4)+(n-5)+(n-6)+(n-8)}{4}$$

$$+ \frac{(n-1)+(n-4)+(n-8)}{8} + \frac{n}{16}$$

با توجه به چندین محاسبه فوق، درخت بازگشت یک درخت دودویی کامل نیست اما همچنان به نظر نمی‌آید بتوانیم برای این عبارت‌ها پاسخی چندجمله‌ای پیدا کنیم، نشان می‌دهیم که اگر بخواهیم برای این تابع بخواهیم کران بالای چندجمله‌ای ارائه دهیم، دچار مشکل می‌شویم.

$$T(n) \leq c\, n^k \;\Rightarrow T(n) \leq c\,(n-1)^k + c\left(\frac{n}{2}\right)^k + n$$

$$\Rightarrow T(n) \leq c\,(n)^k + \cdots + c\,(-1)^k + c\left(\frac{1}{2}\right)^k n^k + n \xRightarrow{\alpha>0} c\,\left(1+\frac{1}{2^k}\right) + \alpha$$

$$\Rightarrow T(n) \leq c\,\left(1+\frac{1}{2^k}\right) + \alpha$$

اگر بخواهیم $T(n) \leq c\, n^k$ باشد، نمیتوانیم چنین ضریبی پیدا کنیم.

پس حدسمان را 2^n قرار میدهیم.

در این صورت:

$$T(n) \leq c\; 2^n - 4n$$

$$T(n) \leq c\; 2^{n-1} - 4n + 4 + \; c\; 2^{\frac{n}{2}} - 4\left(\frac{n}{2}\right) + n$$

$$\Rightarrow\; T(n) \leq c\; \left(2^{n-1} + \; 2^{\frac{n}{2}}\right) - 5n + 4 \xRightarrow{n \geq 2} T(n) \leq c\; (2^{n-1} + \; 2^{n-1}) - 4n$$

$$\Rightarrow\; T(n) \leq c\; (2^n) - 4n \Rightarrow T(n) = O(2^n)$$

پس بهترین کران بالایی که میتوانیم برای این سوال ارائه دهیم نمیتواند چند جملهای باشد و ثابت کردیم این کران یک کران نمایی $T(n) = O(2^n)$ است.

جواب دقیق تر

۹) با استفاده از درخت بازگشت، کران بالای حدی مناسب برای T(n) = T(n-1)+T(n/2)+n بیابید با روش Substitution جواب خود را اثبات کنید.(۱ نمره امتیازی)

$$T(n) = O(n^{logn})$$

$$T(n) \leq cn^{logn}$$

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

$$\leq c(n-1)^{\log(n-1)} + c\left(\frac{n}{2}\right)^{\log(n)-\log(2)} + n$$

$$\leq \;\; cn^{logn}$$

$$\Longrightarrow n \leq c(n^{logn} - \; (n-1)^{\log(n-1)} - \; \left(\frac{n}{2}\right)^{\log(n)-\log(2)})$$

میدانیم که اردر cn^{logn} از بقیه جملات بالاتر است در نتیجه به ازای یک n_0 میدانیم که عبارت بالا برقرار است.

$T(n) = O(n^{\log(n)})$