نمونه سوالات مرتبه زمانی درس ساختمان داده

در زیر نمونه سوالات بسیار متنوعی از ساختمان داده را برای شما آورده ایم و سعی کردیم که از همه مباحث ساختمان داده نمونه سوالاتی با پاسخ تشریحی برای شما قرار دهیم، جامعیت و تنوع نمونه سوالات ساختمان داده ای که برای شما قرار دادیم حتما نیاز شما را بر طرف خواهد کرد.

متوسط خروجی تابع زیر کدام است؟ به دست آوردن مرتبه زمانی شبه کدها

$Func\,(n)$

$tmp = \circ;$

$for\ (i = \frac{n}{2}; i \leq n; i++)$ $\quad$ $for\ (j = 2; j \leq n; j = 2j)$ $\quad$ $tmp = tmp + \frac{n}{2};$

$return\ tmp;$

۱ $\Theta(nlgn)$

۲ $\Theta(n^2)$

۳ $\Theta(n^2lgn)$

۴ $\Theta(n^3)$

حلقه بیرونی $\frac{n}{2}$ مرتبه و حلقه داخلی $\Theta(lgn)$ مرتبه اجرا می‌شود، بنابراین تعداد دفعات اجرای عبارت $tmp = tmp + \frac{n}{2}$ برابر $\Theta(nlgn)$ می‌باشد و چون این عبارت با مقدار $\frac{n}{2}$ افزایش می‌یابد بنابراین خروجی از مرتبه $\theta(n^2lgn)$ می‌باشد.

متوسط تعداد تکرار جمله اصلی $x+=$ را در کد زیر به‌دست آورید؟ به دست آوردن مرتبه زمانی شبه کدها

$= n;$

۴ $۳^{log_{۲}^{n}}$

با هر بار تکرار حلقه بیرونی (حلقه‌ای که شرط خاتمه‌اش روی i است)، حلقه داخلی $log_{۳}^{n}$ بار اجرا می‌شود و در هربار اجرا حلقه داخلی؛ i تقسیم بر ۲ می‌شود و این بدان معناست که i در هر بار اجرا حلقه بیرونی تقسیم بر $۲^{log_{۳}^{n}}$ می‌شود و این کار تا زمانی ادامه دارد که $i > ۱$ باشد. بنابراین حلقه خارجی $log_{۲^{log_{۳}^{n}}}^{n}$ بار اجرا می‌شود که در هر بار اجرا آن حلقه داخلی $log_{۳}^{n}$ بار اجرا می‌شود، بنابراین تعداد اجزا جمله $x+ = ۱$ برابر است با:

$$log_{۲^{log_{۳}^{n}}}^{n} \times log_{۳}^{n} = \frac{۱}{log_{۳}^{n}} \times log_{۲}^{n} \times log_{۳}^{n} = log_{۲}^{n}$$

(نکته: $log_{c^d}^{a^b} = \frac{b}{a} log_{c}^{a}$)

دشوار رابطه بازگشتی زیر چه عملی را انجام می‌دهد؟

$$\begin{cases} G_n = \frac{G_{n-1}+H_{n-1}}{2}, H_n = \frac{A}{G_n}, |G_{n-1} - H_{n-1}| \geq ERR \\ G_1 = 1, H_1 = \frac{A}{G_1} \end{cases}$$

(ERR، مقدار کرانه خطای محاسبات و یا به عبارتی دقت جواب را بیان می‌کند.) به دست آوردن مرتبه زمانی شبه کدها

۱ $\sqrt{A}$ را حساب می‌کند.

۲ A^2 را حساب می‌کند.

۳ $\log_2^A$ را حساب می‌کند.

۴ $\log_e^A$ را حساب می‌کند.(e عدد نِپِر است.)

از روی گزینه‌ها می‌فهمیم که قرار است روی A یک عملیات انجام شود، پس مثلا به ازای A=۹ سوال را بررسی می‌کنیم.

$$G_1 = 1,\ H_1 = \frac{9}{1} = 9$$

$$G_2 = \frac{G_1+H_1}{2} = \frac{1+9}{2} = 5\ ,\ H_2 = \frac{9}{5} = 1/8$$

$$G_3 = \frac{5+1/8}{2} = \frac{34}{10} = 3/4\ ,\ H_3 = \frac{9}{3/4} \simeq 2/64$$

$$G_4 = \frac{3/4+2/64}{2} = 3/02\ ,\ H_4 = \frac{9}{3/02} \simeq 2/9$$

مشخص است که $A = H_n \times G_n$ است و هدف کم کردن اخ[illegible]س در واقع $A = G_n \times G_n$؛ داریم:

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

ورود | ثبت نام

الگوریتم A

```
function(int n){
if(n == 1)return;
printf(" * ");
for(int i = 1; i ≤ n; i + +){
for(int j = 1; j ≤ n; j + +){
break; }}
}
```

الگوریتم B

```
function(int n){
if(n == 1)return;
printf(" * ");
for(int i = 1; i ≤ n; i + +){
for(int j = 1; j ≤ n; j + +){
function(n − 3); }}
}
```

$A = O(n),\ B = O(n^3)$

$A = O(n),\ B = O(n^2 log\ n)$

$A = O(n^2),\ B = O(n^3)$

$A = O(n^2),\ B = O(n^2 log\ n)$

```
Function(int n){
    If(n==l) return;——O(l)
    for(int i=l;i ≤ n;i++){——O(n)
        for(int j=l;j ≤ n;j++){——O(l)
            Printf("*");
            Break;}}
}
```

برای الگوریتم B رابطه بازگشتی $T(n) = T(n-3) + n^2$ را داریم که با استفاده از تئوری مستر میزان پیچیدگی برابر با $O(n^3)$ می‌باشد.

یادآوری: تئوری مستر

$$T(n) = \begin{cases} c & , if n \leq 1 \\ aT(n-b) + f(n), & if n > 1 \end{cases}$$

به شرطی که $c, a > 0$ و $b \geq 0,\ k \geq 0$، اگر مرتبه تابع برابر با $f(n) = O(n^k)$ باشد، آن گاه داریم:

نمونه سوالات رشد توابع درس ساختمان داده

آسان توابع $f(n) = 4^{\log n}$ ، $g(n) = (\log n)^{\log n}$ و $h(n) = \log^2 n$ را در نظر می‌گیریم. کدام یک از گزاره‌های زیر صحیح است؟ رشد توابع

۱) $f(n) \in O(g(n)), f(n) \in \Omega(h(n))$

۲) $g(n) \in \Omega(h(n)), h(n) \in \Omega(f(n))$

۳) $f(n) \in \theta(h(n)), g(n) \in \Omega(f(n))$

۴) $h(n) \in O(g(n)), f(n) \in \theta(g(n))$

به سادگی باید توابع را با یکدیگر مقایسه کنیم:

یاددآوری) دقت کنید $\log^2 n$ با $\log n^2$ برابر نیست بلکه $\log^2 n = \log n * \log n$ و $\log n^2 = 2\log n$

نکته) $a^{\log_c b} = b^{\log_c a}$

اثبات نکته بالا:

یاددآوری: با توجه به فرمول روبه رو میتوان مبنا را تغییر داد: $\log_c b = \frac{\log_a b}{\log_a c}$

$$a^{\log_a b} = b^{\log_a a} \xrightarrow{\text{از دو طرف می‌گیریم } \log} \log(a^{\log_a b}) = \log(b^{\log_a a}) \to \log_c(b) * \log(a) = \log_c(a) * \log(b) \to \frac{\log_{10} b}{\log_{10} c} * \log(a) = \frac{\log_{10} a}{\log_{10} c} * \log(b)$$

حال با توجه به نکات بالا می‌فهمیم که $f(n) = 4^{\log n} = n^2$ حال به مقایسه‌ی این توابع می‌پردازیم.

$$f(n) = 4^{\log n} ? h(n) = \log^2 n \ \to \ n^2 > \log^2 n$$

چون هر توان ثابتی از n از هر توان ثابتی از $\log n$ رشد بیشتری دارد.

برای تشخیص $f(n) = 4^{\log n}$ ؟ $g(n) = (\log n)^{\log n}$ دو راه داریم (البته مشخص است که دو تابع با توان یکسان داریم و پایه‌های یکی عدد ثابت و دیگری $\log n$ است پس $g(n)$ بزرگتر است):

۱- با استفاده از حد نشان دهیم رشد کدام بیشتر است.

$$\lim_{n\to\infty} \frac{f(n)}{h(n)} = \lim_{n\to\infty} \frac{4^{\log n}}{(\log n)^{\log n}} = \lim_{n\to\infty} \left(\frac{4}{\log n}\right)^{\log n} = 0$$

$f(n) = 4^{\log n} ?\ g(n) = (\log n)^{\log n} \xRightarrow{\text{از دو طرف می‌گیریم } \log} \log(4^{\log n})\ ?\ \log((\log n)^{\log n}) \to \log(n) * \log(4) < \log(n) * \log(\log(n)) \to f(n) < g(n)$

پس داریم:

$g(n) \in \Omega(f(n))$ یا $f(n) \in O(g(n))$

برای تشخیص $h(n) = \log^2 n\ ?\ g(n) = (\log n)^{\log n}$ دو راه داریم (البته مشخص است که دو تابع با پایه یکسان داریم و توان یکی عدد ثابت و دیگری $\log n$ است پس $g(n)$ بزرگتر است):

۱- با استفاده از حد نشان دهیم رشد کدام بیشتر است.

$$\lim_{n\to\infty} \frac{g(n)}{h(n)} = \lim_{n\to\infty} \frac{(\log n)^{\log n}}{\log^2 n} = \lim_{n\to\infty} (\log n)^{\log n - 2} = \infty$$

از آنجا که نسبت یک برابر با بینهایت شد می‌فهمیم که $\log^2 n < (\log n)^{\log n}$

۲- با توجه به اینکه هردو تابع صعودی و بزرگتر از یک هستند از روی رشد $\log$هایشان نیز می‌توان مشخص نمود:

$h(n) = \log^2 n\ ?\ g(n) = (\log n)^{\log n} \xRightarrow{\text{از دو طرف می‌گیریم } \log} \log(\log^2 n)\ ?\ \log((\log n)^{\log n}) \to 2\log(n) < \log(n) * \log(\log(n)) \to h(n) < g(n)$

پس داریم:

$h(n) \in O(g(n))$ یا $g(n) \in \Omega(h(n))$ و در کل می‌فهمیم که $h(n) < f(n) < g(n)$

حال با توجه به توضیحات بالا می‌فهمیم که گزینه صحیح است.

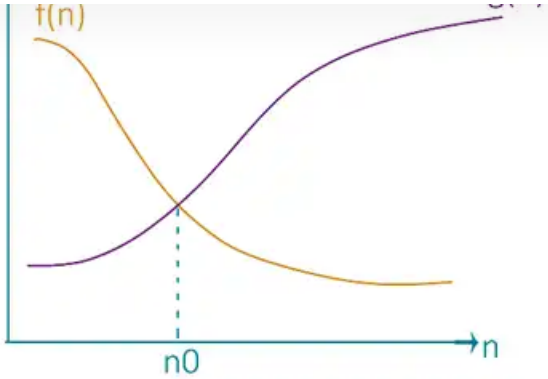
یاددآوری:

Big O: تابع f(n) از مرتبه O(g(n)) است هرگاه رشد f کمتر مساوی رشد g باشد.

$$f(n) \in O(g(n)) \to \exists c, n_0 > 0,\ \forall n \geq n_0 \Rightarrow \underbrace{0}_{\text{رشد برای توابع منفی معنی ندارد}} \leq f(n) \leq c.g(n)$$

O بزرگ کران بالا را مشخص می‌کند.

دقت شود O یک مجموعه است و در نتیجه توابع می‌توانند ... باشند اما به صورت اشتباه رایج از مساوی برای عضویت



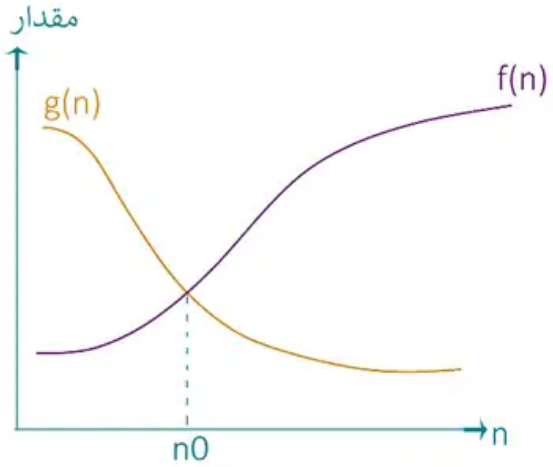
big Ω : تابع f(n) از مرتبه Ω(g(n)) است هرگاه رشد f بیشتر مساوی رشد g باشد.

$$f(n) \in \Omega(g(n)) \rightarrow \exists c, n_0 > 0 \ , \ \forall n \geq n_0 \Rightarrow 0 \leq c.g(n) \leq f(n)$$

Ω بزرگ کران پایین را مشخص می‌کند.

دقت شود Ω یک مجموعه است و در نتیجه توابع می‌توانند عضو این مجموعه باشند اما به صورت اشتباه رایج از مساوی برای عضویت استفاده می‌شود.

$$O(n^2) = \{\ldots, n^2, n^3, n!, n^2 \log(n), \ldots\}$$



با توجه به خواص مجانبی می‌توان عبارات زیر را نتیجه گرفت.

متوسط برای توابع زیر با مرتبه رشد داده شده، دنباله صعودی مرتب شده توابع به صورت $f_{i_1}, f_{i_2}, f_{i_3}, \ldots$ که در آن $f_{i_1} \in O(f_{i_2}),\ f_{i_2} \in O(f_{i_3}),\ f_{i_3} \in O(f_{i_4})$ می‌باشد، کدام است؟ رشد توابع

$$f_1 = lg[(lgn)^3],\ f_2 = (lgn)^{3lg3n};\ f_3 = 3^{lgn},\ f_4 = n^{3^{lgn}},\ f_5 = lg(3n^{n^3}),\ f_6 = (lglgn)^3$$

۱) $f_5, f_4, f_2, f_3, f_1, f_6$

۲) $f_2, f_5, f_4, f_3, f_1, f_6$

۳) $f_4, f_2, f_5, f_3, f_6, f_1$

۴) $f_4, f_3, f_6, f_2, f_1, f_5$

$f_1 = lg[(lgn)^3] = 3lglgn \in O(f_6 = (lglgn)^3)$ (خواص لگاریتم)

$f_6 = (lglgn)^3 \in O(f_3 = 3^{lgn} = n^{lg3})$ (توابع چندجمله‌ای از تمامی توابع لگاریتمی با هر توانی رشد سریع‌تری دارند)

$f_3 = 3^{lgn} = n^{lg3} \in O(f_5 = lg(3n^{n^3}) = n^3 lg(3n))$ (خواص لگاریتم)

$f_2 = (lgn)^{3lg3n} = (27n^3)^{lglgn} \in O(f_4 = n^{3^{lgn}} = n^{n^{lg3}} = f_5 = lg(3n^{n^3}) = n^3 lg(3n) \in O(f_2 = (lgn)^{3lg3n} = (lgn)^{lg27n^3} = (27n^3)^{lglgn})$

دشوار دو الگوریتم داده شده است که یکی دارای مرتبه زمانی $O(n^2)$ است (یعنی مرتبه زمانی آن برای $n \geq n_0$ کوچکتر و مساوی cn^2 است) و دیگری دارای مرتبه زمانی $O(n\ \lg\ n)$ است (یعنی مرتبه زمانی آن برای $n \geq n_1$ کوچک تر و مساوی $dn\ lg\ n$ است). برای چه n هایی $O(n\ \lg\ n)$ از $O(n^2)$ بهتر است؟ رشد توابع

۱) برای تمام مقادیر n

۲) $n \geq max\{n_0, n_1\}$

۳) $n \geq max\{n_0, n_1, \frac{c}{d}\}$

۴) $n \geq max\{n_0, n_1, \frac{d}{c}\}$

$$f(n) = O(n^2) \Rightarrow f(n) \leq cn^2;\ \forall n \geq n$$

$$\Rightarrow dn\ lg\ n \leq cn^2$$

$$g(n) \in O(n\ lg\ n) \Rightarrow g(n) \leq dn\ lg\ n;\ \forall n \geq n_1$$

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

مثلا فرض کنید c' برابر ۱۰۰۰ باشد و c برابر ۱ باشد($1000n \lg n ? n^2$)، خب هر چقدر هم که بیشتر شدن مقدار n باعث بشود که n^2 رشدش از n lg n بیشتر بشود ولی زورش به ۱۰۰۰ نمی‌رسد، پس برای این که n^2 بزرگتر بشه از $1000n \lg n$، باید n عدد بزرگی باشد. قرار شد $c'n \lg n \leq cn^2$ باشد، حالا اگر c از c' بزرگتر باشد که در این صورت cn^2 خیلی بهتر از $c'n \lg n$ است ولی اگر c کوچکتر از c' باشد، حالا وظیفه n است که با بزرگتر خودش، کوچکی c به c' را جبران کند.

متوسط کدام یک از موارد زیر درست می‌باشد؟ رشد توابع

الف) $(n+k)^m = \theta(n^m)$، زمانی که k و m هر دو ثابت باشند.

ب) $2^{n+1} = O(2^n)$ ج) $2^{2n+1} = O(2^n)$

۱ الف و ب

۲ الف و ج

۳ ب و ج

۴ الف و ب و ج

$(n+k)^m = n^m + C1^* n^{m-1} + \ldots + k^m = \theta(n^m)$

$2^{n+1} = 2^* 2^n = O(2^n)$

آسان کدام یک از مجموعه توابع زیر بر حسب زمان رشد صعودی از چپ به راست مرتب هستند. رشد توابع

۱ $(\log n)^{1000}, n^{1000}, (1/001)^n, n!$

۲ $(1/001)^n, (\log n)^{1000}, n^{1000}, n!$

۳ $(\log n)^{1000}, (1/001)^n, n^{1000}, n!$

۴ $(\log n)^{1000}, (1/001)^n, n!, n^{1000}$

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

متوسط برنامه A روی کامپیوتر اولی با اندازه ۳۲ در مدت ۵ میلی ثانیه اجرا می‌شود و برنامه B روی کامپیوتر دوم که سرعت آن ۱۲۸ برابر کندتر از کامپیوتر اولی می‌باشد و اندازه برنامه برابر با ۶ می‌باشد در این صورت زمان اجرای کامپیوتر دوم را محاسبه کنید. روابط بازگشتی

$$B) int\, Test(int\, n)\{$$
$$if(n \leq 1) return\, n;$$
$$return\, Test(n-3) + Test(n-3) + n;\}$$

$$A) for(\text{int}\, i = 1; i \leq n; i++)$$
$$for(\text{int}\, i = 1; j \leq n; j+=i)$$
$$x = x + 1;$$

۱ ۱ میلی ثانیه

۲ ۲ میلی ثانیه

۳ ۱۶ میلی ثانیه

۴ ۸ میلی ثانیه

ابتدا باید مرتبه زمانی هر دو برنامه را به دست بیاوریم. برای برنامه A داریم:

$\lambda = 1 \Rightarrow$ باز

$i = 2 \Rightarrow \frac{n}{2}$

$i = 3 \Rightarrow \frac{n}{3}$

. $\Rightarrow n + \frac{n}{2} + \frac{n}{3} + \ldots + n \Rightarrow n(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}) = \theta(n \log n)$

.

.

$i = 2 \Rightarrow \frac{n}{2}$

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

دشوار اگر زمان اجرای یک الگوریتم با رابطه بازگشتی $T(n) = \frac{2}{n}(T(0)+T(1)+T(2)+\ldots+T(n-1))+n$ مشخص شود کدام گزینه پیچیدگی زمانی الگوریتم را به درستی بیان می‌کند؟ روابط بازگشتی

۱ $\theta(n^2)$

۲ $O(nlogn)$

۳ $O(loglogn)$

۴ $O(\sqrt{n})$

$$T(n) = \frac{2}{n}(T(0)+T(1)+T(2)+\ldots+T(n-1))+n$$

ابتدا رابطه اصلی را در n ضرب می‌کنیم و سپس در رابطه بدست آمده به جای n، n+۱ قرار می‌دهیم:

$$nT(n) = 2(T(0)+T(1)+T(2)+\ldots+T(n-1))+n^2 \ (*)$$

$$(n+1)T(n+1) = 2(T(0)+T(1)+T(2)+\ldots+T(n))+(n+1)^2 \ (**)$$

سپس رابطه$(**)$ را از منفی رابطه$(*)$ می‌کنیم:

$$(n+1)T(n+1)-nT(n) = 2T(n)+2n+1 \rightarrow (n+1)T(n+1) = (n+2)T(n)+2n+1$$

$$T(n+1) = \frac{(n+2)T(n)}{n+1}+\frac{2n+1}{(n+1)} \xrightarrow{\text{تقسیم می‌کنیم n+2 را بر}} \frac{T(n+1)}{n+2} = \frac{T(n)}{(n+1)}+\frac{2n+1}{(n+1)(n+2)}$$

حال اگر تابع $g(n) = \frac{T(n)}{n+1}$ را به این صورت تعریف کنیم، با توجه به رابطه بازگشتی‌ای که در بالا بدست آوردیم خواهیم داشت:

$$g(n+1) = g(n)+\frac{2n+1}{(n+1)(n+2)} \xrightarrow{\text{اگر این رابطه بازگشتی را باز کنیم خواهیم داشت}}$$

$$g(n+1) = \frac{2n+1}{(n+1)(n+2)}+\frac{2n-1}{(n)(n+1)}+\cdots+\frac{3}{2\times 3}+g(1)$$

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

$$= g(n+1) = \left(\frac{(2n+1)}{n+1}-\frac{(2n+1)}{n+2}\right)+\left(\frac{(2n-1)}{n}-\frac{(2n-1)}{n+1}\right)+\ldots+\left(\frac{3}{2}-\frac{3}{3}\right)+g(1)$$

$$= 2\left(\frac{1}{n+1}+\frac{1}{n}+\cdots+\frac{1}{3}\right)+\frac{3}{2}-\frac{2n+1}{n+2} = O(logn) \rightarrow T(n) = O(nlogn)$$

۳ $T(n) = T(n-1) + O(n)$

۴ $T(n) = T(n-1) + 1$

واضح است که گزینه ۳ از $O(n^2)$ است و گزینه ۱و ۴ نیز $O(n)$.

می دانیم که گزینه دوم $\omega(\sqrt{n})$ زیرا اگر به رابطه دقت کنید، در سمت راست تساوی داریم $\sqrt{n} \times 99 \times T(\sqrt{n}) + 100 \times n$ ، از طرفی به کمک حدس زدن نشان خواهیم داد که:

$$Suppose:\ T(n) \sim T_k(n) = \sqrt{n} \times \sqrt{99 \times T(\sqrt{n})} + 100 \sim \sqrt{n} \times \sqrt{99 \times T(\sqrt{n}) + 100} \sim \sqrt{n} \times \sqrt{99 \times n^{\frac{k}{2}} + 100} \sim \sqrt{n} \times n^{\frac{k}{4}}$$
$$k = \frac{1}{2} + \frac{k}{4} \rightarrow k = \frac{2}{3} \quad T(n) \sim O\left(n^{\frac{2}{3}}\right)$$

روش دوم:

$$Suppose:\ T(n) \sim cn^{\frac{2}{3}}$$

جایگذاری کنید:

$$\sqrt{n} \times \sqrt{99 \times cn^{\frac{1}{3}} + 100} \geq cn^{\frac{2}{3}} \quad cn^{\frac{1}{2}+\frac{1}{6}} \leq cn^{\frac{2}{3}}$$

آسان در رابطه بازگشتی $T(n) = 9T(n/3) + f(n)$ به ازای چند تا از عبارت‌های $g(n)$ زیر، دست‌کم یک تابع برای $f(n)$ وجود دارد به طوری که $T(n) = \Theta(g(n))$ ؟ روابط بازگشتی

$$n \lg n \,||\, n^2 \,||\, n^2 \lg^2 n \,||\, n^3$$

۱ ۱

۲ ۲

۳ ۳

۴ ۴

با استفاده از قضیه مستر می‌دانیم که:

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

ورود | ثبت نام

نکته) اگر در $f(n)$، هر توانی از $lg(n)$ را کنار می‌گذاریم و بخش باقی‌مانده از آن را با سمت چپ مقایسه می‌کنیم سه حالت رخ می‌دهد

۴- اگر سمت چپ بزرگتر باشد مرتبه برابر سمت چپی می‌شود

۵- اگر سمت راستی بزرگتر باشد، مرتبه برابر با سمت راستی ضرب در lgn (یعنی همون $f(n)$) خواهد بود.

۶- اگر برابر باشند مرتبه برابر با $f(n)$ ضرب در lgn می‌شود

حال برای عبارت بالا داریم $n^{log_4 9} = n^2 \ ? \ f(n)$ در نتیجه برای اینکه این شرط برقرار باشد باید به صورت زیر عمل کنیم و هر تابع را امتحان کرده و $f(n)$ پیدا کنیم که شرط $T(n) \in \theta(g(n))$ برقرار می‌سازد.

$T(n) \in \theta(n\,lg(n)) \Rightarrow$ ندارد $T(n) \in \theta(n^2) \overset{f(n)=n}{\Longrightarrow} n^2 > n \Rightarrow T(n) \in \theta(n^2)$ $T(n) \in \theta(n^2 lg^2 n) \overset{f(n)=n^2 lg\, n}{\Longrightarrow} n^2 = n^2 \Rightarrow T(n) \in \theta(n^2 lg^2 n)$

$T(n) \in \theta(n^3) \overset{f(n)=n}{\Longrightarrow} n^3 > n \Rightarrow T(n) \in \theta(n^3)$

متوسط اگر محتوای تمامی خانه‌هایی یک آرایه n عنصری A به ازاء تمامی مقادیر $i \leq n$ برابر مقدار ثابت k و $k \leq n$ باشد $(\forall i \in \{1, 2, \ldots, n\}, A[i] = k)$ مرتبه زمانی تابع بازگشتی زیر با فراخوانی چیست؟ روابط بازگشتی

$f(A, i)$ { if$(A[i] + i \leq n)$ return$f(A, A[i] + i)$; return 1;}

۱ $\theta(log_k^n)$

۲ $\theta(\frac{n}{k})$

۳ $\theta(1) = \theta(k)$

۴ $\theta(nk)$

اگر همه عناصر آرایه مقدار ثابت K باشد $\overline{|K|K|\ldots|K|}$ و ما فراخوانی تابع f(A,i) را با i=۱ شروع کنیم می‌بینیم که شرط if دارد K تا زیاد می‌شود تا به n برسد بنابراین $\frac{n}{k}$ بار فراخوانی انجام می‌شود. بنابراین مرتبه برابر خواهد بود با $\theta(\frac{n}{k})$.

متوسط مرتبه رابطه بازگشتی $T(n) = T\left(\alpha \frac{n}{2}\right) + T\left((1-\alpha)\frac{n}{2}\right)$ که α ثابت و $0 < \alpha < 1$ است را به‌دست بیاورید. روابط بازگشتی

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

ورود | ثبت نام

برای این سوال درخت بازگشتی را رسم می‌کنیم.

جمع

$$n \qquad \Rightarrow n$$

$$\alpha\frac{n}{2} \qquad (1-\alpha)\frac{n}{2} \qquad \Rightarrow n$$

$$\alpha^2\frac{n}{4}+(1-\alpha)\alpha\frac{n}{4} \qquad (1-\alpha)\alpha\frac{n}{4} \qquad (1-\alpha)^2\frac{n}{4} \Rightarrow \frac{n}{4}$$

$$\Rightarrow \text{جمع}\ n+\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\ldots \Rightarrow n(1+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\ldots) \Rightarrow 2n \Rightarrow O(n)$$

تصاعد هندسی با جمله اول ۱ و قدرت نسبت $\frac{1}{2}$ $\Leftarrow a \times \frac{q^n-1}{q-1}$ $\Leftarrow$ اگر n خیلی زیاد باشد.

$$\frac{a}{1-q} \Rightarrow \frac{1}{\frac{1}{2}} = 2$$

متوسط برنامه A روی کامپیوتر اولی با اندازه ۳۲ در مدت ۵ میلی ثانیه اجرا می‌شود و برنامه B روی کامپیوتر دوم که سرعت آن ۱۲۸ برابر کندتر از کامپیوتر اولی می‌باشد و اندازه برنامه برابر با ۶ می‌باشد در این صورت زمان اجرای کامپیوتر دوم را محاسبه کنید. روابط بازگشتی

$$B)int\,Test(int\,n)\{$$
$$if(n \le 1)return\,n;$$
$$return\,Test(n-3)+Test(n-3)+n;\}$$

$$A)for(\text{int}\,i=1;i \le n;i++)$$

$$for(\text{int}\,i=1;j \le n;j+=i)$$

$$x = \ldots 1;$$

ابتدا باید مرتبه زمانی هر دو برنامه را به دست بیارویم. برای برنامه A داریم:

$\lambda = 1 \Rightarrow n$ با

$i = 2 \Rightarrow \frac{n}{2} = 3 \Rightarrow \frac{n}{3} \qquad \Rightarrow n + \frac{n}{2} + \frac{n}{3} + \ldots + n \Rightarrow n(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}) = \theta(n \log n) \Rightarrow \frac{n}{2}$

برای برنامه B داریم: $T(n) = 2T(n-3) + n$ بنابر قضیه مستر مرتبه اجرای این برنامه برابر است با $\theta(2^{\frac{n}{3}})$

بنابر مفروضات مسئله داریم $V_2 = \frac{1}{128} V_1$ حال با استفاده از رابطه $\frac{O(n_2)}{O(n_1)} = \frac{t_2}{t_1} \times \frac{v_2}{v_1}$

$$\frac{2^{\frac{n}{3}}}{n \log n} = \frac{t_2}{5m \ \sec} \times \frac{1}{128} \frac{V_1}{V_1} \Rightarrow \frac{2^{\frac{6}{3}}}{32 \log 32} = \frac{t_2}{5m \ \sec} \times \frac{1}{2^7} \Rightarrow t_2 = \frac{2^2 \times 5 \times 2^7}{2^5 \times 5} = 16m \ \sec$$

متوسط خروجی الگوریتم زیر با فراخوانی $Test(6, 4)$ چیست؟ روابط بازگشتی

```
int Test (int n, int k)
{
int i = 0;if(k > 0)i = 1;Test(--n, --k);cout << i << "," << k << ",";
```

۱ ۰, ۰, ۱, ۳, ۲, ۴, ۳, ۵, ۴, ۶

۲ ۰, ۰, ۱, ۳, ۲, ۴, ۳, ۵

۳ ۱, ۳, ۲, ۴, ۳, ۵, ۴, ۶

۴ خروجی تولید نمی کند.

با توجه به اینکه شرط if، else ندارد و تابع test مدام خودش را فراخوانی می‌کند و شرطی برای اتمام حلقه صدا زدن وجود ندارد و کنترل دستگاه هیچگاه به cout نمی‌رسد. لذا برنامه با پیام پر شدن پشته روی سیستم خطا می‌دهد.

نمونه سوالات درخت‌ها درس ساختمان داده

آسان یک درخت دودویی با ۶ گره داده شده است که هر گره فقط فرزند چپ دارد. با چند عمل دوران راست (بدون دوران چپ) می‌توان این درخت را به درختی تبدیل کرد که هر گره فقط فرزند راست داشته باشد. کم ترین مقدار ممکن را انتخاب کنید.

۴ | ۷

درخت اولیه یک درخت اُریب از چپ است که می‌خواهیم به یک درخت اُریب از راست تبدیل کنیم که کافی است هر دفعه از یک دوران به راست به محوریت گره فرزند مستقیم ریشه بزنیم تا درخت نهایی حاصل شود که با ۵ دوران این عمل انجام می‌گیرد. به طوری کلی، یک درخت اُریب از چپ با n گره را می توان با n-۱ دوران از راست (بدون دوران چپ) به درخت اُریب از راست تبدیل کرد.

متوسط پیمایش Preorder و Postorder یک درخت دودویی داده شده است. پیمایش inorder آن، کدام است؟ درخت‌ها

Preorder: fgbceda

Postorder: gedcabf

۱ gfecdba

۲ gfecabd

۳ gfecbda

۴ نمی‌توان به دست آورد.

حل تشریحی این تست را می‌توانید در تست ۶۸ نکته و تست تکمیلی ساختمان داده و الگوریتم ۹۹ استاد رضوی مشاهده کنید

متوسط دنباله‌ی jfcbadeghik پیش‌ترتیب یک درخت دودویی جست‌وجوی T است. کدام‌یک از گزینه‌های زیر دنباله‌ی پس‌ترتیب T است؟ درخت‌ها

۱ abdecihgfkj

۲ badecihgfkj

۳ abdecighfkj

۴ abedcihgfkj

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

۱

	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
A	۸۷	۴۸	۵۲	۲۹	۲۳	۴۸	۴۹	۲۱	۱۲	۲۴	۱۷

۲

	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
A	۸۰	۳۵	۵۲	۳۰	۲۳	۴۸	۴۹	۲۱	۱۲	۱	۲

۳

	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
A	۸۷	۳۰	۵۲	۳۱	۲۳	۴۸	۵۱	۲۹	۱۲	۲۲	۱۷

۴

	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
A	۸۰	۳۱	۵۲	۳۰	۲۳	۵۰	۴۹	۲۱	۱۲	۱۹	۴۷

در بین گزینه فقط آرایه‌ی

	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
A	۸۰	۳۵	۵۲	۳۰	۲۳	۴۸	۴۹	۲۱	۱۲	۱	۲

یک MaxHeap می‌باشد.

متوسط فرض کنید اعداد ۱ تا ۱۰۰ به صورت تصادفی در یک BST درج شده‌اند. اولین عدد (ریشه) ۵۰ است. با چه احتمالی در روند اضافه شدن اعداد به درخت، دو عدد ۵ و ۱۲ با هم مقایسه می‌شوند؟ درخت‌ها

۱ $\frac{1}{3}$

۲ $\frac{1}{4}$

۳ $\frac{1}{8}$

۴ $\frac{2}{5}$

اعداد ۵ تا ۱۲ را در نظر بگیرید که !۸ جایگشت دارند. اگر عدد ۵ یا ۱۲ زودتر از ۴ تا ۱۱ وارد شوند حتماً ۵ در ۱۲ با هم مقایسه می‌شوند.

تعداد حالاتی که ۵ زودتر وارد شود !۷ و تعداد حالاتی که ۱۲ زودتر وارد شود نیز !۷ است پس:

احتمال $= \frac{2\times 7!}{8!} = \frac{1}{4}$

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

۳ $O(lg\, n)$

۴ متغیر است.

درج هر عنصر در ساختمان داده Red-Black Tree یا منجر به تغییر رنگ می‌شود و یا حداکثر ۲ دوران انجام می‌شود.

کدام گزینه پیمایش preorder یک درخت جستجوی دودویی BST با پیمایش postorder به صورت زیر است؟ درخت‌ها

PostOrder: ۵, ۶, ۱۵, ۱۰, ۲۳, ۲۴, ۲۲, ۲۶, ۲۰

۱ ۲۰, ۱۰, ۶, ۵, ۱۵, ۲۶, ۲۴, ۲۳, ۲۲

۲ ۲۰, ۱۰, ۶, ۵, ۱۵, ۲۶, ۲۲, ۲۴, ۲۳

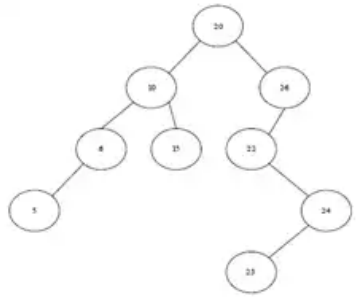
۳ ۶, ۵, ۱۰, ۱۵, ۲۰, ۲۲, ۲۴, ۲۳, ۲۶

۴ ۲۰, ۲۳, ۱۵, ۵, ۶, ۱۰, ۲۶, ۲۴, ۲۲

با توجه به این که پیمایش inorder یک درخت BST صعودی است با داشتن دو پیمایش inorder و preorder درخت قابل ترسیم است.

Post :۵, ۶, ۱۵, ۱۰, ۲۳, ۲۴, ۲۲, ۲۶, ۲۰

In :۵, ۶, ۱۰, ۱۵, ۲۰, ۲۲, ۲۳, ۲۴, ۲۶



مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

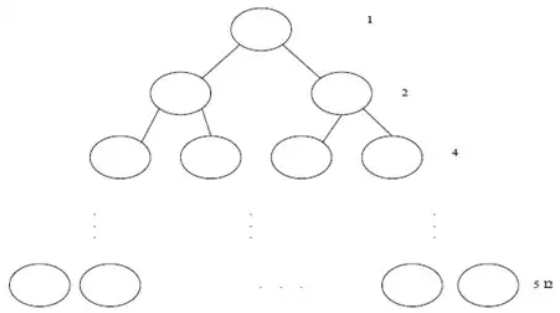
۱ ۸

۲ ۹

۳ ۱۰

۴ ۱۱

با توجه به مفروضات مسئله ریشه در عمق صفر قرار دارد. همان طور که در شکل زیر نشان داده شده است بیش ترین عمق درخت برابر با ۹ می‌باشد.



دشوار در یک درخت دودویی کامل با n گره و ارتفاع h مجموعه تمامی ارتفاع گره‌ها برابر است با: درخت‌ها

۱ O(n)

۲ O(n-h)

۳ O(n log n)

۴ O(log n)

در یک درخت باینری کامل هر سطح شامل ۲i گره می‌باشد (همچنین هر گره در سطح i دارای عمق i و ارتفاع h-i می‌باشد) حال می‌خواهیم مجموعه تمامی ارتفاع‌ها را حساب کنیم.

$$S=\sum_{i=0}^{h}2^{i}(h-i)\Rightarrow S=h+2(h-1)+4(h-2)+\ldots+2^{h-1}(1)$$

با توجه به این که مجموع تمامی گره‌ها در یک درخت کامل برابر n است داریم $n = 2^{h+1}-1$ بنابراین داریم $h = \log(n+1)$ در نهایت با جایگذاری داریم

$$S = n - (h-1) \Rightarrow O(n - \log n) \Rightarrow O(n-h)$$

متوسط کدام گزینه صحیح است؟

الف) حداکثر تعداد گره‌های یک درخت دودویی با L برگ

ب) حداقل و پ) حداکثر تعداد گره‌های یک درخت دودویی محض با ارتفاع h (درختی که هر گره آن یا فرزند ندارد یا دو فرزند دارد.) درخت‌ها

۱ الف: 2^{L+1}، ب: $h+1$ و پ: 2^h-1

۲ الف: 2^L، ب: $2h+1$ و پ: $2^{h+1}-1$

۳ الف: $2L-1$، ب: $2h+1$ و پ: 2^h-1

۴ هیچکدام

الف) ارتباطی بین تعداد برگ‌ها و تعداد گره‌های یک درخت وجود ندارد و بنابراین سه گزینه اول رد می‌شوند.

ب) حداقل تعداد گره‌های یک درخت دودویی محض با ارتفاع h زمانی رخ می‌دهد که درخت شبیه یک مسیر باشد که از هر گره‌ی آن یک فرزند اضافه نیز خارج شده باشد. در این صورت تعداد گره‌های هر سطح ۲ است که به علاوه گره‌ی ریشه تعداد کل گره‌ها برابر h۲ +۱ خواهد بود.

پ) حداقل تعداد گره‌های یک درخت دودویی محض با ارتفاع h زمانی رخ می‌دهد که درخت پر باشد که یک درخت پر با ارتفاع h دارای $2^{h+1}-1$ گره می‌باشد.

نمونه سوالات مرتب سازی درس ساختمان داده

آسان در یک آرایه A به اندازه n ، اگر $i < j$ و $A[i] > A[j]$ می‌گوییم که زوج (i,j) یک «زوج- معکوس» (inversion) در A است بیشترین تعداد زوج- معکوس‌ها در یک آرایه n عضوی چندتاست؟ مرتب سازی

راه حل ۱ عدد گذاری)

با توجه به اینکه گزینه ها رابطه دقیقی از n داده اند کافیست برای n=۲ چک کنیم و گزینه صحیح را پیدا کنیم:

برای n=۲ و یک مجموعه دو عضوی، بیشترین تعداد زوج معکوس به صورت A=[max min] خواهد بود (جای max و min عدد هم می‌توانید بذارید مثلا A = [۲۵ ۱۴]) در نتیجه بیشترین تعداد زوج معکوسی برابر با ۱ است و فقط گزینه ۱ صحیح است.

راه حل ۲ اثبات)

به توضیحات زیر توجه کنید:

وارونگی نسبت به صعودی بودن:

وقتی یک آرایه داریم و برای دو عنصر A[i] , A[j] با جایگاه (اندیس) به ترتیب i, j شرط زیر برقرار است می‌گوییم که یک وارونگی (زوج-معکوس بر اساس این تست یا inversion) داریم.

$$j > i but A[i] > A[j]$$

برای پیدا کردن وارونگی ها از سمت چپ به راست، حرکت می‌کنیم تمامی عناصری که از یک عنصر کوچکتر هستند با آن عنصر وارونه هستند.

مثال) آرایه زیر چند وارونگی دارد:

$$\mathrm{A}\,[1\ldots5] = [12\ 4\ 9\ 7\ 5]$$

وارونگی ها:

$$(12,4)\ (12,9)\ (12,7)\ (12,5)$$
$$(9,7)\ (9,5)$$
$$(7,5)$$

در کل ۷ وارونگی داریم.

نکته) آرایه صعودی وارونگی ندارد.

نکته) آرایه نزولی max تعداد وارونگی دارد.

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

ورود | ثبت نام

$n-1$ قبلی این وارونگی ها به حساب آورده ایم نباید$n-1$ وارونگی هم برای عنصر آخر جمع کنیم.

راه اثبات ۲)

هر دو عنصری که انتخاب کنیم چون آرایه نزولی است قطعا شرط j > i but A[i] > A[j] برقرار است پس تعداد وراونگی برابر تعداد انتخاب بدون ترتیب دو عنصر از n عنصر است.

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

موارد گفته شده را در آرایه نزولی زیر می‌توانید امتحان کنید.

$$A\,[1\ldots4] = [12\ 8\ 5\ 3\ 1]$$

نکته) متوسط تعداد وارونگی ها در یک آرایه برابر با$\frac{n(n-1)}{4}$ است.

اثبات:

روش ۱:

فرض کنید L یک لیست دلخواه از عناصر است وL_R لیستی با عناصر L است که به صورت برعکس چیده شده است برای مثال:

$$L = [b\ e\ f\ a\ g]\ ,\ L_R = [g\ a\ f\ e\ b]$$

حالا دو عضو از لیست L انتخاب می‌کنیم، برای این کار$\binom{n}{2}$ حالت وجود دارد. دو عضوی که انتخاب کردیم در حداقل یکی از لیست های L یاL_R با یکدیگر وارونگی دارند. بنابراین می‌توان نتیجه گرفت در دو لیست L وL_R در مجموع$\binom{n}{2}$ وارونگی وجود دارد.

حال برای اعضای هر لیستی $n!$ جایگشت وجود دارد که این تعداد جایگشت را به دو مجموعه ی مجزا که یکی شامل L ها و دیگری شاملL_R ها است تقسیم می‌کنیم (به عبارت دیگر نصفی از جایشگت ها برعکس شده ی نصفه ی دیگر هستند). در نتیجه می‌توان فهمید که میانگین وارونگی ها در کل جایشگت های برابر است با$\frac{\frac{n(n-1)}{2}}{2} = \frac{n(n-1)}{4}$.

روش ۲ (استقرا):

در استقرا ابتدا برای تعدادی عدد کوچک درستی قضیه$P(n)$ را حک می‌کردیم و سپس قضیه را برای$P(k)$ درست فرض می‌کردیم و

نکته) با مرتب سازی درجی و ادغامی می‌توان تعداد وارونگی ها شمرد.

نحوه شمردن مرتب سازی درجی:

در الگوریتم مرتب سازی درجی یک شمارنده اضافه می‌کنیم. هر جابه‌جایی که انجام شد یکی به شمارنده اضافه می‌کنیم به معنی اینکه هر این جابه‌جایی به دلیل یک وارونگی بوده که به وسیله الگوریتم حل شده.

یاددآوری:

در مرتب سازی درجی هربار از عنصر دوم شروع می‌کنیم هر عنصر را در انتهای آرایه درج می‌کنیم با عناصر قبلی مقایسه می‌کنیم اگر جایشان درست نبود باید جابه‌جایی انجام میدادیم برای مثال:

$$\mathrm{A}\,[1\ldots5] = [12\ 4\ 9\ 7\ 5]$$

ابتدا ۴ را درج می‌کنیم پس جایش باید با ۱۲ تغییر کند.$[4\ 12]$

سپس ۹ را درج می‌کنیم و دوباره ۹ جایش را با ۱۲ تغییر می‌دهد.$[4\ 9\ 12]$

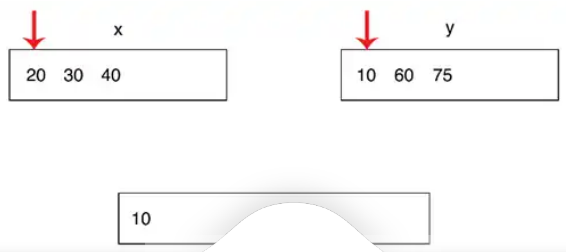
۷ را درج می‌کنیم به ترتیب جایش با ۱۲ و ۹ تغییر می‌دهد.$[4\ 7\ 9\ 12]$

۵ را درج می‌کنیم جایش را با ۱۲، ۹ و ۷ تغییر می‌دهد$[4\ 5\ 7\ 9\ 12]$

در مجموع ۷ جابه‌جایی و در نتیجه ۷ وارونگی داشتیم.

نحو شمردن مرتب سازی ادغامی:

در این الگوریتم نیز یک کانتر می‌گذاریم. و هر وقت کسی از لیست y انتخاب شد کانتر را با تعداد تمامی عناصر موجود در لیست x جمع میزنیم و این کار را در تمامی مراحل مرتب سازی ادغامی انجام می‌دهیم. (از مقایسه آرایه های کوچک شده و تک عنصری تا آخرین مرحله.)



۱ $T(n) = O(n), M(n) = O(n)$

۲ $T(n) = O(n), M(n) = O(1)$

۳ $T(n) = O(n^2), M(n) = O(1)$

۴ $T(n) = O(n \log n), M(n) = O(n)$

حل تشریحی این تست را می‌توانید در تست ۱۵۹، نکته و تست ساختمان داده و الگوریتم استاد رضوی مشاهده کنید.

متوسط در مرتب‌سازی سریع برای مرتب کردن آرایه‌ای با n عنصر، برای انتخاب محور از میان n عنصر $\frac{n}{4}$ عنصر اول آرایه را انتخاب و با مرتبه O(n) آن را مرتب می‌کنیم و محور را عنصر میانه این عناصر می‌گیریم مرتبه‌ی زمانی الگوریتم مرتب‌سازی سریع بعد از اعمال این موارد برابر است با: مرتب سازی

۱ $\theta(n)$

۲ $\theta(n \log n)$

۳ $\theta(n^2)$

۴ $\theta(n^2 \log n)$

بنابر صورت سوال به رابطه بازگشتی زیر می‌رسیم.

$$T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + O(n)$$

که با استفاده از تئوری مستر و یا درخت مرتبه اجرا برابر با $O(n \log n)$ می‌شود.

آسان کدام‌یک از گزینه‌های زیر به‌درستی ارتباط بین رابطه زمانی در بهترین حالت و موارد مطرح شده را به‌درستی بیان می‌کند

مرتب سازی

$A)\ O(\log) \quad B)\ O(n) \quad C)\ O(n \log n) \quad D)\ O(n^2)$

۴) $A-R,\ B-S,\ C-R,\ D-Q$

مرتبه اجرا در بهترین حالت برای مرتب‌سازی انتخابی برابر با $O(n^2)$ ادغامی برابر $O(n \log n)$، مرتب‌سازی درجی برابر $O(n)$ و جست‌وجوی دودویی برابر $O(\log n)$ می‌باشد.

آسان آرایه S که شامل n عنصر عدد صحیح مرتب شده می‌باشد. اگر t(n) نشان دهنده‌ی کارآمدترین الگوریتم باشد تا نشان دهد که دو عددی وجود دارد که مجموع اعداد آن‌ها کمتر از ۱۰۰۰ باشد مقدار t(n) برابر است با: مرتب سازی

۱) $t(n) = O(1)$

۲) $n \log n < t(n) < \binom{n}{2}$

۳) $n < t(n) < n \log n$

۴) $t(n) = \binom{n}{2}$

چون آرایه مرتب شده است فقط کافی است دو عنصر اول را جمع کنیم تا ببینیم درست است یا خیر، پس مرتبه اجرایی آن برابر O(۱) می‌باشد.

دشوار فرض کنید عناصر آرایه $P[1..n]$ به صورت $P[1] < P[2] < \cdots < P[i] > P[i+1] > \ldots$ هستند (یعنی تا درایه با اندیس i به صورت صعودی و از این درایه به بعد نیز به صورت نزولی مرتب شده‌اند). کاراترین الگوریتم برای یافتن اندیس i در بدترین حالت از چه مرتبه‌ی زمانی است؟ مرتب سازی

۱) $\Theta(n)$

۲) $\Theta(lg^* n)$

۳) $\Theta(lg\, n)$

۴) $\Theta(lg\, lg\, n)$

مرتب بودن دو طرف عنصر P[i] استفاده از ایده جست‌وجو [illegible] فراهم می‌کند. در واقع می‌توانیم مشابه الگوریتم جست‌وجوی

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

در این حالت اندیس مورد جست‌وجو پیدا شده است و پاسخ مسئله می‌باشد.

بنابراین در هر فاز الگوریتم، یا پاسخ مسئله را به دست می‌آوریم و یا باید در یکی از نیمه‌های آرایه به جست‌وجو را ادامه دهیم. پس اگر T(n) تابع پیچیدگی زمانی این الگوریتم باشد، رابطه بازگشتی آن به صورت زیر خواهد بود:

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

که مرتبه زمانی این رابطه بازگشتی O(lgn) می‌باشد، یعنی در بدترین حالت مرتبه الگوریتم $\Theta(lgn)$ است.

آسان اگر در الگوریتم BucketSort برای مرتب‌سازی هر bucket از الگوریتم (الف) InsertionSort با زمان اجرای $O(n^2)$ در بدترین حالت و (ب) MergeSort با زمان اجرای $\Theta(n\ lg\ n)$ در همه‌ی حالات، استفاده کنیم. امید ریاضی زمان اجرای این الگوریتم کدام خواهد بود؟ مرتب سازی

۱ الف: $\Theta(n)$ و ب: $\Theta(n)$

۲ الف: $\Theta(n^2)$ و ب: $\Theta(n)$

۳ الف: $\Theta(n^2)$ و ب: $\Theta(n\ lg\ n)$

۴ الف: $\Theta(n\ lg\ n)$ و ب: $\Theta(n\ lg\ n)$

در هر دو مورد (الف) و (ب) امید ریاضی زمان اجرای الگوریتم BucketSort از مرتبه‌ی $\Theta(n)$ خواهد بود.

دشوار فرض کنید n کوزه‌ی آب قرمز رنگ و n کوزه‌ی آب سبز رنگ داده شده است که شکل و اندازه‌ی تمام آنها متفاوت است. تمامی کوزه‌های قرمز میزان آب متفاوتی نسبت به یک دیگر نگه می‌دارند و همین طور تمامی کوزه‌های سبز میزان آب متفاوتی نسبت به یک دیگر نگه می‌دارند. علاوه براین به ازای هر کوزه‌ی قرمز یک کوزه‌ی سبز با همان ظرفیت وجود دارد و بالعکس. هدف گروه‌بندی کوزه‌ها به صورت جفت کوزه‌های قرمز و سبز با ظریف یکسان می‌باشد. برای انجام این می‌توان به این صورت عمل کرد که یک جفت کوزه انتخاب می‌کنیم که یکی قرمز و دیگری سبز باشد. کوزه‌ی قرمز را با آب پر می‌کنیم و سپس آب درون آن را در کوزه‌ی سبز می‌ریزیم. با این عمل می‌توان تشخیص داد که آیا ظرفیت این دو کوزه برابر است و یا کدام یک از کوزه‌ها ظرفیت بیشتری دارد. اگر چنین مقایسه‌ی به یک واحد زمان نیاز داشته باشد، کران پایین تعداد مقایسه‌های مورد نیاز در هر الگوریتمی که این مسئله را حل کند از چه مرتبه‌ای است؟ مرتب سا

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

ورود | ثبت نام

۴ $\Omega(n^2)$

برای به دست آوردن کران پایین این مسئله، محاسبات را از دید درخت تصمیم‌گیری این مسئله نگاه می‌کنیم. هر گره‌ی داخلی این درخت از یک جفت کوزه (یک قرمز و یک سبز) که مقایسه می‌کنیم تشکیل شده است و دارای سه یال خروجی دارد. (کوزه‌ی قرمز ظرفیت کمتر، ظرفیت برابر و یا ظرفیت بیشتر نسبت به کوزه‌ی سبز). برگ‌های این درخت، گروه‌بندی‌های یکتای کوزه‌ها را نشان می‌دهد. ارتفاع درخت تصمیم این مسئله برابر تعداد مقایسه‌های الگوریتم در بدترین است. تعداد برگ‌های این درخت برابر n! می‌باشد چون هر جایگشت از کوزه‌های قرمز به همراه کوزه‌ی سبز متناظرش یک نتیجه متفاوت را حاصل می‌کند. هر شاخه درخت از درجه ۳ می‌باشد و بنابراین این درخت حداکثر 3^h برگ خواهد داشت. بنابراین خواهیم داشت:

$$3^h \geq n! \Rightarrow h \geq log_3(n!) \Rightarrow h \in \Omega(n\ lg\ n)$$

نمونه سوالات جداول درهم ساز درس ساختمان داده

دشوار یک جدول درهم‌سازی پویا (Dynamic) با نام DT را در نظر بگیرید که اندازه‌ی آن در ابتدا ۱ است و آن درایه نیز خالی است. درهم‌سازی با روش بسته (Closed Hashing) است و با یک تابع درهم‌سازی ساده و Linear-Probing انجام می‌شود. پویایی این جدول به این صورت تامین می‌شود که اگر هنگام درج یک عنصر Load Factor $(\alpha(DT))$ جدول یک باشد، جدول را گسترش داده و عنصر جدید را درج می‌کنیم. استراتژی گسترش به گونه است که جدول جدید با سایز دو برابر جدول فعلی ایجاد می‌کنیم، تمامی عناصر جدول قبلی را در آن درج می‌کنیم و جدول قبلی را پاک می‌کنیم. اگر هزینه واقعی یک عمل درج مستقیم هر عنصر در جدول را ۱ و هزینه واقعی عمل گسترش را برابر تعداد درج‌های مورد نیاز فرض کنیم، کدام گزینه در مورد هزینه سرشکن یک عمل درج و حدود Load Factor این جدول صحیح است؟ جداول درهم ساز

۱ $\frac{1}{4} < \alpha(DT) \leq 1$,2

۲ $0 < \alpha(DT) \leq 1$,n

۳ $\frac{1}{2} < \alpha(DT) \leq 1$,3

۴ هیچ کدام

مجموع هزینه‌ها به ازای دنباله‌ای از n عمل درج برابر $\sum_{i=1}^{n} c_i \leq n + \sum_{i=1}^{\lfloor lgn \rfloor} 2^i < n + 2n$ است، بنابراین هزینه سرشکن عمل درج $\hat{c}_i \leq \frac{3n}{n} = 3$ می‌باشد. همچنین چون همواره نیمی از [illegible] است، Load Factor حداقل $\frac{1}{2}$ می‌باشد.

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

چه تعداد از دنباله‌های متفاوت درج کلیدها در این جدول وجود دارد که جدول بالا را نتیجه می‌دهد؟

جداول درهم ساز

۱ ۱۰

۲ ۲۰

۳ ۳۰

۴ ۴۰

در این جدول، کلیدهای ۴۲، ۲۳، ۳۴ و ۴۶ در مکان صحیح خود هستند و با هر ترتیبی می‌توانند درج شده باشند. ولی کلید ۵۲ باید حتماً بعد از عناصر ۴۲، ۲۳ و ۳۴ درج شده باشد، چون در زمان درج آن این مکان‌ها پر بوده است که ۵۲ در خانه‌ی ۵ قرار گرفته است. همچنین کلید ۳۳ نیز باید بعد از تمامی کلیدها درج شده باشد. بنابراین کل تعداد دنباله‌های مختلف درج برابر $3! \times 5 = 30$ می‌باشد.

متوسط اعمال Insert، Search و Extract-Min را بر روی یک جدول در هم‌سازی باز (OpenHashing) انجام می‌دهیم. کدام‌یک از گزینه‌های زیر زمان اجرای یک پیاده‌سازی کارا (Efficient) از این جدول با n عنصر می‌باشد؟ جداول درهم ساز

۱ Insert: O، Search: O(n)(۱) و Extract-Min: O(۱)

۲ Insert: O، Search: O(lgn)(۱) و Extract-Min: O(۱)

۳ Insert: O، Search: O(n)(۱) و Extract-Min: O(n)

۴ Insert: O، Search: O(lgn)(lgn) و Extract-Min: O(lgn)

با Augment کردن جدول درهم‌سازی زنجیرهای به صورت زنجیره‌هایی به شکل یک جدول جست‌وجوی متوازن، گزینه ۴ نتیجه می‌شود. گزینه ۱ و ۲ کران پایین الگوریتم‌های مرتب‌سازی مقایسه‌ای در بدترین حالت را نقض می‌کنند.

نمونه سوالات آرایه، جستجو در آرایه، لیست پیوندی، پشته، صف درس ساختمان داده

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

۱ ۲

۲ ۲۰۴۸

۳ ۱۰۲۴

۴ ۱

این الگوریتم نودها را یک در میان حذف می‌کند. اگر اولین نود حذفی، شماره ۲ باشد آنگاه برای یافتن آخرین عدد کافی است ۲۰۴۸ را باینری بنویسیم و چرخش به سمت چپ دهیم (اژولف) (یعنی: $2048 = (1\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ) \xrightarrow{left\ rotate} (\circ\circ\ldots\ldots\ldots 1)$ عدد ۱ باقی می‌ماند ولی با توجه به موقعیت p ، اولین نود حذفی شماره ۳ است پس عدد باقیمانده ۲ می‌باشد.

دشوار کلیدهای k_1 تا k_n براساس احتمال جست‌وجو شدن به‌صورت نزولی مرتب شده‌اند. یعنی احتمال جست‌وجو شدن کلید k_i از احتمال جست‌وجو شدن k_{i+1} بیشتر است. با شروع از k_1، این کلیدها را به‌ترتیب در یک درخت جست‌وجوی خالی درج می‌کنیم در مورد این الگوریتم کدام گزینه نادرست است؟ آرایه، جستجو در آرایه، لیست پیوندی، پشته، صف

۱ درخت حاصل کم‌ترین میانگین تعداد مقایسه‌ها را در جست‌وجوی موفق دارد.

۲ زمان اجرای این الگوریتم در بدترین حالت است.

۳ کلید k_n در برگ درخت قرار خواهدگرفت.

۴ کلید k_1 در ریشه درخت قرار خواهدگرفت.

به عنوان مثال نقض فرض کنید احتمال جست‌وجو شدن کلید $k_1 = 5$ برابر ۰/۴۱، کلید $k_2 = 4$ پا برابر ۰/۳۰ و کلید $k_3 = 2$ برابر ۰/۲۹ باشد. در این‌صورت الگوریتم درخت T1 را تولید می‌کند در حالی‌که درخت جست‌وجوی بهینه درخت T2 است.

متوسط خروجی الگوریتم زیر بر روی لیست پیوندی با داده‌های ۱, ۲, ۳, ۴, ۵, ۶, ۷ که به ترتیب پشت سرهم از چپ به راست در لیست پیوندی قرار دارند چیست؟ آرایه، جستجو در آرایه، لیست پیوندی، پشته، صف

```
struct node
{
int value;
```

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!


```
int temp;
if ((!list) || !list -> next)
    return;
p=list; q=list-> next;
while(q)
{
    temp = p-> value;
    p-> value= q-> value;
    q-> value= temp;
   q=-> next;
  p=p ? q-> next:. ;
      }
}
```

۱ ۱, ۲, ۳, ۴, ۵, ۶, ۷

۲ ۲, ۱, ۴, ۳, ۶, ۵, ۷

۳ ۱, ۳, ۲, ۵, ۴, ۷, ۶

۴ ۲, ۳, ۴, ۵, ۶, ۷, ۱

الگوریتم مورد نظر اقدام به تعویض محتویات هر گره با گره بعدی خود می‌کند و این کار را با شروع از گره اول می‌کند.

متوسط نوعی خاصی از پشته در اختیار داریم که با هر عمل push دو عنصر بالای پشته خود را در پشته قرار می‌دهد و در هر عمل pop کردن اگر تعداد popهایی که تا قبل از عمل pop انجام شده است فرد باشد عنصر بالای پشته را برمی‌گرداند و اگر تعداد popها زوج باشد دو عنصر بالای پشته را به ترتیب برمی‌گرداند با فرض خالی بودن پشته در ابتدا و داده ورودی زیر کدام خروجی را نمی‌توان تولید کرد. (ورودی داده‌ها از چپ به راست می‌باشد) آرایه، جستجو در آرایه، لیست پیوندی، پشته، صف

ABCDEFGH

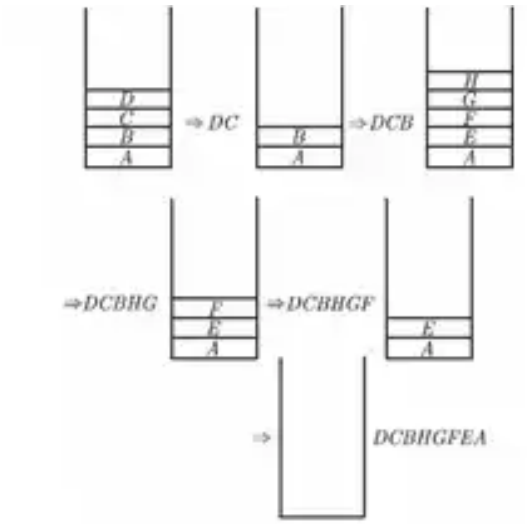
۱ BADFECHG

مسیر ۶ ماهه تا کنکور: مشاوره خصوصیت با استاد رضوی رو رزرو کن!

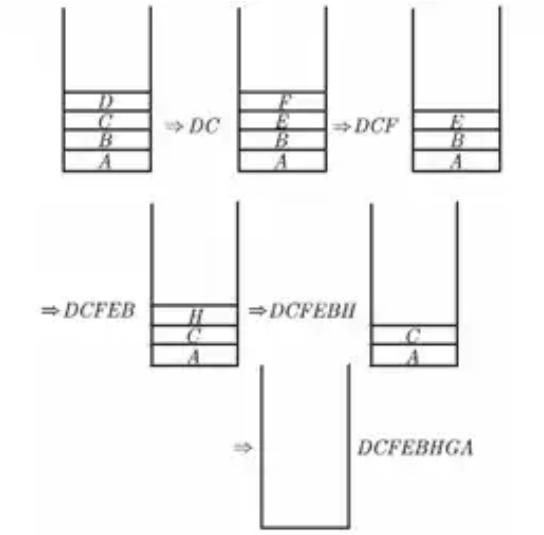
ورود | ثبت نام

می‌شود سپس D و C را داخل پشته قرار می‌دهیم و دوباره pop می‌کنیم، چون مقدار pop فرد است فقط مقدار D بر می‌گردد و دوباره عمل push برای F و E انجام و بلافاصله pop که دو عنصر بالا را بر می‌گرداند و سپس مجددا pop که فقط یک عنصر بر می‌گرداند و در نهایت H و G، push می‌شود و دوباره pop می‌شود و گزینه ۱ در نهایت تولید می‌شود.

برای مورد ۲ داریم:



برای مورد ۴ داریم:



فقط برای گزینه ۳ نمی‌توان رسم کرد چون DC با هم آمده که امکان‌پذیر بنابر مسئله نمی‌باشد.