• محمدحسین نظری - ۹۹۳۱۰۹۱

①

		هزینه	تعداد
1	`int j=0;`	c_1	1
2	`for(int i=0; i<n; i++){`	c_2	$n+1$
3	`if(A[i] != B[j]){`	c_3	n
4	`int k=j;`	c_4	n
5	`while(A[i] != B[k]){`	c_5	$\sum_{i=0}^{n-1}(n-i+1)$
6	`k++; }`	c_6	$\sum_{i=0}^{n-1}(n-i)$
7	`swap(B[j], B[k]); }`	c_7	n
8	`j++; }`	c_8	n

الف) درخط 4 تا 7 که داخل شرط if قرار دارد ، بدترین حالت ممکن درنظر گرفته شده و فرض شده هر n بار که شرط if چک می‌شود ، وارد حلقه if شده درصورتی که ممکن است به احتمال زیاد به این شکل نباشد.

درخط 5 تعداد تکرار حلقه while از i تا n-1 است بعلاوه ۱ حالت که false شدن شرط حلقه است.

$$T(n) = C_1 + C_2(n+1) + C_3 n + C_4 n + (C_5)\Big(\underbrace{\sum_{i=0}^{n-1}(n-i+1)}_{\frac{n^2}{2}+\frac{3n}{2}}\Big)$$

$$+ (C_6)\Big(\underbrace{\sum_{i=0}^{n-1}(n-i)}_{\frac{n^2}{2}+\frac{n}{2}}\Big) + C_7 n + C_8 n =$$

$$(C_1 + C_2) + (n)\Big(C_2 + C_3 + C_4 + \frac{3C_5}{2} + \frac{C_6}{2} + C_7 + C_8\Big)$$

$$+ (n^2)\Big(\frac{C_5}{2} + \frac{C_6}{2}\Big) = a + bn + cn^2$$

$$T(n) = O(n^2)$$

ب) چون برای حلقه‌ی while ، for گذاشته شده پس برای اینکه

برنامه تابع مناسبی طراحی شود، باید شرط پایه جای پایان یابد برای همه عناصر آرایه A ،

پس باید برابر آن ها در B وجود داشته باشد، پس احتمال میدهیم که آرایه B همان

A با چینش متفاوت است و این الگوریتم آرایه B را به فرم A بازسازی می کند

مثال:

$A = \{1, 2, 3\}$ $B = \{2, 1, 3\}$

$B' \longrightarrow \{1, 2, 3\} \Longrightarrow A = B$

۴) می‌دانیم در مجموعه n تایی . به $\log^n$ مرحله برای رسیدن به بدترین انتخاب داریم
(binary search طبق)

حال در الگوریتم داده شده برای انتخاب عدد رندوم $\log 1$ صفحات طی آسان

می‌افتد حال فرض کنیم برای اینکه مجموعه چند عضوی باشد احتمال بگذاریم که در این صورت

برابر $\frac{1}{n}$ است و (باید از بین ۱ تا n یک عدد برای تعداد مجموعه باشد) و در مرحله عدد رندوم

بدترین حالت را برای عدد او تعداد $\log 1$ می‌گذارم .

که در این صورت $T_1(n) = \log 1$, $T_2(n) = \log 2$, ... , $T_n(n) = \log(n)$

و

$$P_1(n) = P_2(n) = \dots = P_n(n) = \frac{1}{n}$$

$$\Rightarrow \underset{\text{میانگین}}{T(n)} = P_1(n)T_1(n) + \dots + P_n(n)T_n(n)$$

$$= \left(\frac{1}{n}\right)(\log(n!)) \xrightarrow{\log(n!) = O(n\lg n)} O(\lg n)$$

۳) الف) یک الگوریتم ساده search هم کارها را راه می‌اندازد

	هزینه	تعداد
1 `int k=0 ;`	c_1	1
2 `for(int i=0 ; i<n ; i++){`	c_2	$n+1$
3 `if(A[i] == T){`	c_3	n
4 `k=i ;`	c_4	1
5 `break; } }`	c_5	1
6 `return / print k ;`	c_6	1

$$T(n) = C_1 + C_2 n + C_2 + C_3 n + C_4 + C_5 + C_6$$

$$= (C_1 + C_2 + C_4 + C_5 + C_6) + (n)(C_2 + C_3) = O(n)$$

فرض می‌کنیم با آرایه A داریم و فقط یک مقدار T در آرایه وجود دارد پس خطوط 4 و 5 فقط یکبار اجرا می‌شوند

ب) از الگوریتم binary search استفاده می‌کنیم. این الگوریتم به این شکل است که یک بازه اعداد مرتب شده را تحویل گرفته و عدد دلخواه را در آن جستجو می‌کند با این شرایط که هر بار بازه مقایسه را نصف کرده و عدد دلخواه را با وسط مقایسه کرده و بنابراین

در هر مرحله نیمی از داده ها را حذف می کند و در بازه n عضوی ، تعداد حداکثری تکرار

آن از مرتبه $\lg n$ است . در این مسئله A[I] را بعنوان min و

A[I-1] را بعنوان max الگوریتم در نظر گرفته

binary Search (A , T , A[I] , A[I-1])

ج.

```
binary Search ( A, T , A[I] , A[I-1])

while ( A[I]  ≤ A[I-1]) {
    int mid = 1 + (A[I-1]-1)/2 ;
    if (A[mid] == T) {
        return mid ; }
    if (A[mid] < T) {
        A[I] = mid + 1 ; }
    if (A[mid] > T ) {
        A[I-1] = mid - 1 ; } }
```

بدترین حالت زمانی برای حلقه while زمانی است که تا آخرین تقسیم را

انجام دهیم یعنی به تعداد n را بر ۲ تقسیم کنیم که به ۱ برسیم که برابر است

با lgn

$$\frac{n}{2^k} = 1 \rightarrow 2^k = n \rightarrow \lg n = x$$

سایر خط های کدهم در بدترین حالت مجموعاً lgn بار اجرا شوند پس تاثیری بر

T(n) ندارند و $T(n) = O(\lg n)$ ..

Scanned with CamScanner

۴

الف)

	هزینه	تعداد
for $i \leftarrow n$ down to 1 do	C_1	$(i+1)$ / ۱
for $j \leftarrow 1$ to i do	C_2	$\sum_{i=1}^{n}(i)$ / n
if $A[j] > A[j+1]$:	C_3	$\sum_{i=1}^{n}(i)$ / n
swap$(A[j], A[j+1])$	C_4	$\sum_{i=1}^{n}(i)$ / ۰

بدترین حالت:

$$C_1 + C_2 n + (C_2 + C_3 + C_4)\left(\frac{n^2}{2}\right) + (C_2 + C_3 + C_4)\left(\frac{n}{2}\right)$$

$$= an^2 + bn + c = O(n^2)$$

بهترین حالت: در این حالت به حلقه اول تنها یکبار نیاز است و حلقه دوم n بار پیمایش روی آرایه را انجام می دهد

$$C_1 + C_2 n + C_3 n = an + b = O(n)$$

حالت متوسط:

بطور کلی برای یک آرایه n عنصری، n! حالت چینش اعضا وجود دارد

پس برای حالت متوسط مرتب سازی حبابی n! حالت را باید بررسی کنیم که از

$O(n)$ و $O(n \lg n)$ و $O(n^2)$ تشکیل شده اند.

اگر تعداد هر کدام را تقریبی $\frac{n!}{3}$ بگیریم در نهایت به شکل زیر در می آید

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

که به ترتیب می توان گفت از مرتبه $O(n^2)$ است.

5

a)

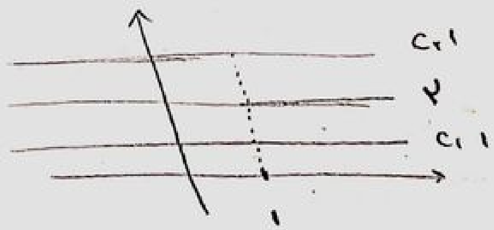
$$n^{\frac{1}{\lg n}} = n^{\log_n 2} = 2^{\log_n n} = 2$$

باید به ازای n_0 و c_1 و c_2 دنبال ثابت کنیم به ازای $n \geq n_0$

$$0 \leq c_1 1 \leq 2 \leq c_2 1$$

فرض می کنیم $c_1 = 1.5$ و $c_2 = 2.5$ که در این صورت تعریف اثبات می شود
و $n_0 = 1$

$$0 \leq 1.5 \leq 2 \leq 2.5$$



b)

باید حاصل حد و برد صفر شود

از تقریب استرلینگ استفاده می کنیم

$$\lim_{n \to \infty} \frac{2^n}{n!}$$

$$\lim_{n \to \infty} \frac{2^n \times e^n}{(\sqrt{2\pi n})(n^n)(1 + \Theta(\frac{1}{n}))}$$

$$\underbrace{\left(\lim_{n\to\infty}\left(\frac{1}{\sqrt{2\pi n}}\right)\right)}_{0}\underbrace{\left(\lim_{n\to\infty}\frac{1}{1+\Theta\left(\frac{1}{n}\right)}\right)}_{1}\left(\lim_{n\to\infty}\left(\frac{2e}{n}\right)^{n}\right)$$

حال اگر n_0 را بصورت مضرب ke بشرط $k>2$ بگیریم حاصل

حد سوم به شکل روبرو درمی‌آید که صفر است

$$\lim_{n\to\infty}\left(\frac{2}{k}\right)^{n}=0$$

پس حاصل حد کلی صفر است

روش دوم

بطور شهودی می‌توان گفت بعد از $n_0=4$ حاصل $n!$ همیشه از 2^n

بزرگتر است چون در هر مرحله ضرب $n!$ ، n از 2 بزرگتر است. پس

$n!$ هم از 2^n بزرگتر است.

$n\ge 4$, $c=1$

$$n_0! > 1\times 2^{n} \ge 0$$

ت

حاصل حد روبرو باید بی نهایت شود

از تقریب استرلینگ استفاده می کنیم

$$\lim_{n\to\infty} \frac{e^n}{n!} = \lim_{n\to\infty} \frac{e^n}{(\sqrt{2\pi n})(1+\Theta(\frac{1}{n}))}$$

اگر $(\sqrt{2\pi})(1+\Theta(\frac{1}{n}))_{n\to\infty} = c$ در تقریب بگیریم آنگاه

$$\lim_{n\to\infty} \frac{e^n}{c\sqrt{n}} = \frac{1}{c} \times \lim_{n\to\infty} \frac{e^n}{\sqrt{n}} \overset{HOP}{=}$$

از قاعده هوپیتال استفاده می کنیم

$$\frac{1}{c} \times \lim_{n\to\infty} \frac{e^n}{\frac{1}{2\sqrt{n}}} = \frac{1}{c} \times \lim_{n\to\infty} 2e^n\sqrt{n} = \frac{1}{c} \times \infty = \infty$$

راه حل دوم

به طور شهودی می توان گفت از $n \geq 2$ حاصل $n \times n \times \dots \times n$ قطعاً از $n \times (n-1) \times \dots \times 1$ بزرگتر است چون همه ی ترم های ضرب اول بزرگتر یا مساوی

ترم های ضرب دوم هستند

$n \geq 2 \quad , \quad c = 1$

$0 \leq n! \leq 1 \times n^n$

(4)

a $\times$

مثال نقض

$n = O(n^2)$ ولی $n^2 \neq O(n)$

b $\times$

مثال نقض

$2n = O(n)$ ولی $2^{2n} \neq O(2^n)$

برای اینکه $2^{2n} = O(2^n)$ باشد باید n_0 و c را پیدا کنیم به طوری که

$0 \leq 2^{2n} \leq c \times 2^n$ و با توجه به اینکه $2^{2n} = 2^n \times 2^n$ پس

$c \geq 2^n$ باشد که با توجه به اینکه c یک عدد ثابت است غیر ممکن است

c $\times$

مثال نقض $f(n) = \frac{1}{n} \rightarrow \frac{1}{n} \neq O(\frac{1}{n^2})$

در این مثال بعد از n_0 که جواب معادله $\frac{1}{n} = c\frac{1}{n^2}$ است، شرط

$\frac{1}{n} \le c\frac{1}{n^2}$ و در نتیجه $\frac{1}{n} = O(\frac{1}{n^2})$ نقض می‌شود.

d ✓

اثبات

$$f(n) = O(g(n)) \xrightarrow{n \ge n_0} 0 \le f(n) \le c\,g(n) \xrightarrow{\times\, c_2 = \frac{1}{c}}$$

$$0 \le c_2 f(n) \le g(n) \longrightarrow g(n) = \Omega(f(n))$$

e ✗

مثال نقض

$$2^{2n} \ne \Theta(2^n)$$

دلیل نادرستی $2^{2n} = \Theta(2^n)$ در b اثبات شد و شرط $f(n) = \Theta(g(n))$

این است که هم $f(n) = O(g(n))$، هم $f(n) = \Omega(g(n))$ درست باشد،

چون یکی از شروط ما نقض شد پس عبارت ما غلط است

$\checkmark$ f

اثبات : می‌دانیم

$$\underbrace{f(n) = \Omega(f(n))}_{۲} \quad \text{و} \quad \underbrace{f(n) = O(f(n))}_{۱}$$

همچنین

$$O(f(n)) + o(f(n)) \Rightarrow g_1(n) \geq c_1 f(n) + g_۲(n) > c_۲ f(n)$$

$$\rightarrow \underbrace{g_1(n) + g_۲(n)}_{g(n)} \geq \underbrace{(c_1 + c_۲)}_{c} f(n) \rightarrow g(n) \geq c f(n) \rightarrow = O(f(n))$$

در بالا اثبات کردیم

$$f(n) + o(f(n)) = O(f(n)) \stackrel{!}{=} \underline{O(f(n)) + o(f(n)) = O(f(n))}$$

$$\Omega(f(n)) + o(f(n)) \longrightarrow g_1(n) \leq c_1 f(n) + g_۲(n) > c_۲ f(n)$$

$$g_۲(n) > c_۲ f(n) \xrightarrow[-c_۲ = c_۳]{\times(-1)} -g_۲(n) < c_۳ f(n)$$

$$\xrightarrow{+} \underbrace{g_1(n) - g_۲(n)}_{g(n)} \leq \underbrace{(c_۳ + c_1)}_{c} f(n) \rightarrow g(n) \leq c f(n)$$

$$\rightarrow = \Omega(f(n))$$

$$f(n) + o(f(n)) = \Omega(f(n)) \overset{۲}{=} \Omega(f(n)) + o(f(n)) = \Omega(f(n))$$

دربالا اثبات شد

حال هردوی

$$f(n) + o(f(n)) = O(f(n))$$

و

$$f(n) + o(f(n)) = \Omega(f(n))$$

اثبات شد پس

$$f(n) + o(f(n)) = \Theta(f(n))$$

5

خط

۲	yes	yes	no	no	no
۳	yes	yes	no	no	no
٤	no	no	yes	yes	no
٥	yes	no	yes	no	yes
۶	no	no	yes	yes	no
۷	yes	no	yes	no	yes
۸	no	no	yes	yes	no
۹	no	no	yes	yes	no
۱۰	yes	no	yes	no	yes