Subject
Date

98243050 پارسا محمدپور

۱- توابع بازگشتی زیر را حل کنید:

a) $T(n) = \sqrt{2}\, T(\frac{n}{2}) + O(\log n)$ $\leftarrow$ با استفاده از قضیه

تعمیم داده شده قضیه اصلی، به صورت زیر می باشد:

اگر داشته باشیم $T(n) = a\, T(\frac{n}{b}) + f(n)$ آنگاه 3 حالت ممکن است:

$$T(n) \in \begin{cases} \Theta(n^{\log_b a}) & \text{if } f(n) = O(n^{\log_b a}) \\ \Theta(n^{\log_b a} \log n) & \text{if } f(n) = \Theta(n^{\log_b a}) \\ \Theta(f(n)) & \text{if } f(n) = \Omega(n^{\log_b a}) \end{cases}$$

$f(n) = O(\log n)$

$a = \sqrt{2} \quad , \quad b = 2 \Rightarrow \log_b a = \log_2 \sqrt{2} = 1/2 \Rightarrow n^{\log_b a} = \sqrt{n}$

$\Rightarrow \log n \in O(\sqrt{n}) \Rightarrow f(n) = O(\sqrt{n})$

$\xRightarrow{\text{حالت اول}} T(n) \in \Theta(\sqrt{n})$

b) $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + O(\log n)$ ۱-

$$\left.\begin{array}{l} \Rightarrow T(n) \leq \sqrt{n}\, T(\sqrt{n}) + c * \log n \\ T(n) = n * g(n) \quad \text{I} \end{array}\right\}$$

$$\Rightarrow n * g(n) \leq \sqrt{n} * (\sqrt{n} * g(\sqrt{n})) + c * \log n$$

$$\Rightarrow n * g(n) \leq n * g(\sqrt{n}) + c * \log n$$

$$\Rightarrow g(n) \leq g(\sqrt{n}) + c * \left(\frac{\log n}{n}\right)$$

با توجه به اینکه سرعت رشد تابع n از $\log n$ بیشتر است یعنی $\log n \in O(n)$

است پس می‌توان گفت که $\frac{\log n}{n} \leq 1$ پس:

$$g(n) \leq g(\sqrt{n}) + c * 1$$

Subject
Date

۱- ادامهٔ b)

$g(n) \le g(\sqrt{n}) + c$

حال از تغییر متغیر استفاده می‌کنیم به طوری که:

(II)

$g(n) = t_i \quad , \quad n = 2^i \Rightarrow i = \lg n$

$\Rightarrow t_i \le t_{i/2} + c$

$\Rightarrow t_i \le t_{i/4} + c + c$

$\Rightarrow t_i \le t_{i/8} + 3 * c$

$\vdots$

$\Rightarrow t_i \le \overbrace{t_{\frac{i}{2^{\lg i}}}}^{O(1)} + \lg i * c$

(II)

$\Rightarrow g(n) \le O(1) + \lg(\lg n)$

$\Rightarrow g(n) \in O(\lg(\lg n))$ | (n توانی از 2 باشد)

برای اثبات g غیر کاهشی است

حال برای اثبات هموار بودن $\lg \lg n$ از قضایای حد استفاده می‌کنیم:

$$\lim_{n\to\infty} \frac{\lg(\lg bn)}{\lg(\lg n)} = \lim_{n\to\infty} \frac{\ln\left(\frac{\ln bn}{\ln 2}\right)}{\ln\left(\frac{\ln n}{\ln 2}\right)}$$

Hopital

$$\lim_{n\to\infty} \frac{\ln\left(\frac{bn}{\ln 2}\right) * \frac{bn}{\ln 2}}{\ln\left(\frac{n}{\ln 2}\right) * \frac{n}{\ln 2}} = \lim_{n\to\infty} b * \frac{\ln \frac{bn}{\ln 2}}{\ln \frac{n}{\ln 2}}$$

PAPCO

Hopital

$$\lim_{n\to\infty} b * \frac{\frac{bn}{\ln 2}}{\frac{n}{\ln 2}} = b^2 \longrightarrow \text{کران دار است}$$

هموار $\Rightarrow$

Subject

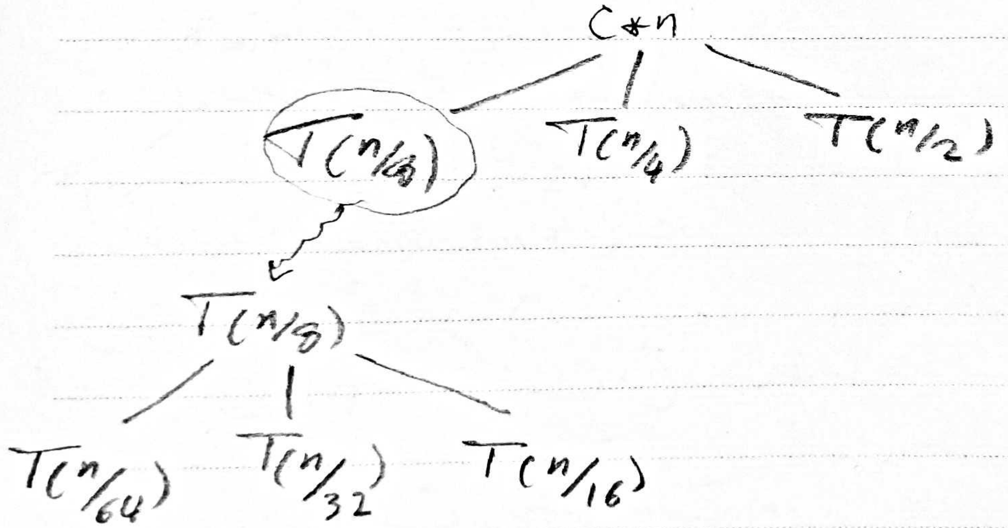
Date

$\Rightarrow g(n) \in O(\lg(\lg n))$ ۱ـ ط ادامه:)

(I) $\Rightarrow \frac{T(n)}{n} \in O(\lg(\lg n)) \Rightarrow T(n) \in O(n * \lg(\lg n))$

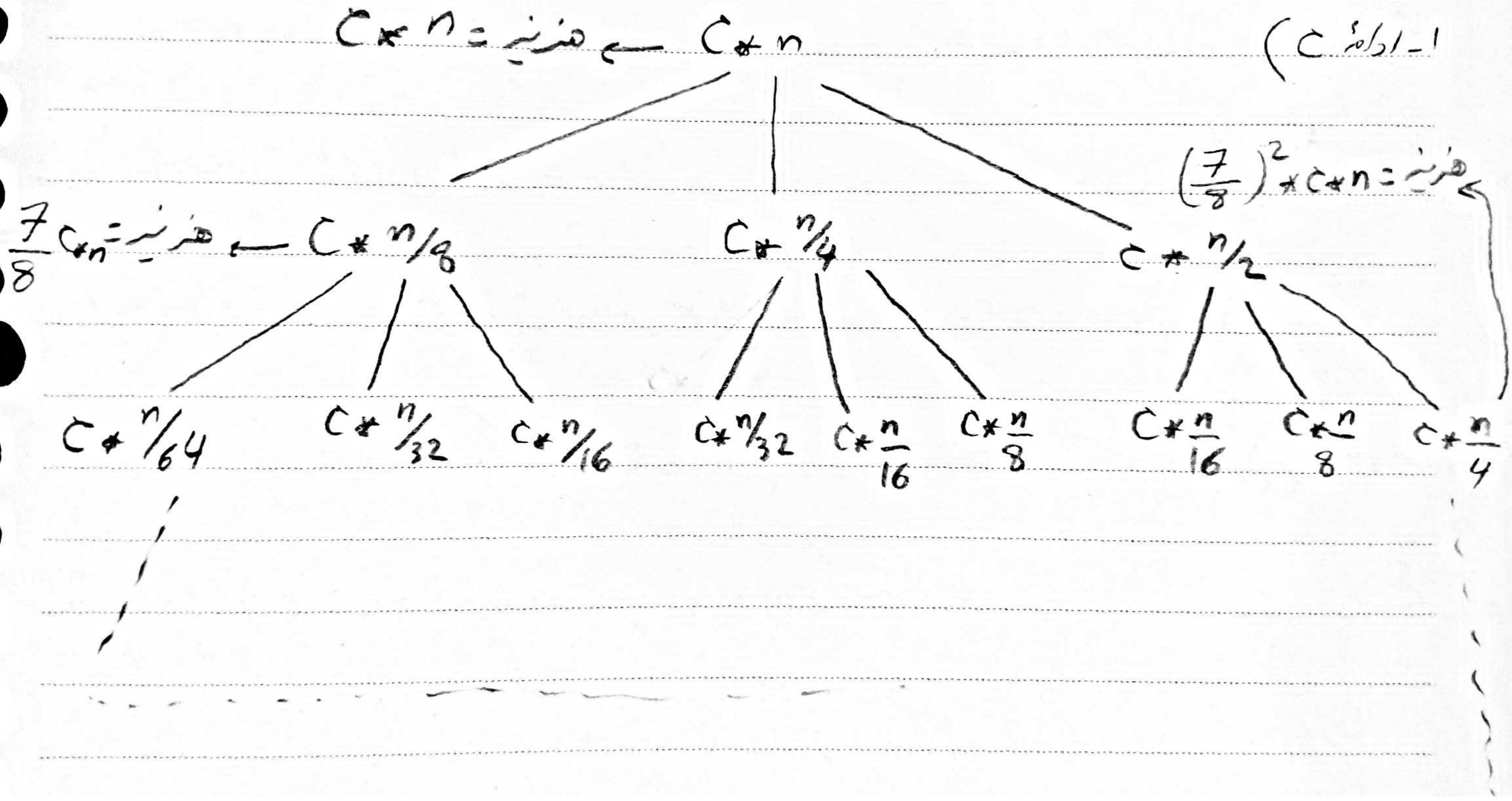
c) $T(n) = T(n/2) + T(n/4) + T(n/8) + \Omega(n)$ ۱ـ

$\Rightarrow T(n) \geq T(n/2) + T(n/4) + T(n/8) + c * n$



$\Downarrow$

۱- ادامه C)


C*n = هزینه ← C*n
7/8 C*n = هزینه ← C*n/8
C*n/4
C*n/2
(7/8)^2 * C*n = هزینه
C*n/64
C*n/32
C*n/16
C*n/32
C*n/16
C*n/8
C*n/16
C*n/8
C*n/4

Subject

Date

۱- ادامهٔ c) حال باید پیدا کنیم که اولین برگی که می‌بینیم (منظور از اولین برگ، برگ با کمترین عمق است) که مشخصاً آن برگ در سمت چپ‌ترین شاخه است پس :

$$\text{عمق اولین برگ} = l \Rightarrow \frac{n}{8^l} = 1 \Rightarrow n = 8^l \Rightarrow l = \log_8 n$$

$$\Rightarrow l = \frac{\lg n}{3}$$

هزینهٔ هر عمق (از اولین عمق تا عمق l که در بالا آن را پیدا کردیم) :

$$\text{هزینه در عمق } i = c * n * \left(\frac{7}{8}\right)^i$$

پس برای اینکه یک مینیمم برای هزینه‌ها پیدا کنیم، میاییم و هزینهٔ همهٔ عمق‌ها را از عمق صفر تا عمق l جمع می‌کنیم تا یک مینیمم به نسبت خوب بدست بیاوریم :

$$\Rightarrow \text{هزینهٔ کل} = \sum_{i=0}^{l} c * n * \left(\frac{7}{8}\right)^i = \sum_{i=0}^{\log_8 n} c * n * \left(\frac{7}{8}\right)^i$$

$$= c * n * \sum_{i=0}^{\log_8 n} \left(\frac{7}{8}\right)^i = c * n * \left(\frac{1 - \left(\frac{7}{8}\right)^{\log_8 n + 1}}{1 - \frac{7}{8}} \right)$$

$$= c * n * \left(\frac{1 - (7/8)^{\log_8 8n}}{1/8} \right) = c * n * \left(1 - (8n)^{\log_8 7/8}\right) * 8$$

$$= c * n * 8 * \underbrace{\left(1 - (8n)^{\overset{\text{منفی}}{\log_8 7 - 1}}\right)}_{\text{کوچکتر از یک}} \Rightarrow T(n) \in \Omega(n)$$

Subject
Date

d) $T(n) = 2T(\sqrt{n}) + \theta(\log(\log n))$ ⟵ smoothness !

تغییر متغیر می دهیم به گونه ای که:

$n = 2^x \quad , \quad T(n) = t(x)$ (I)

$n = 2^x \Rightarrow \sqrt{n} = \sqrt{2^x} \Rightarrow \sqrt{n} = 2^{x/2} \Rightarrow T(\sqrt{n}) = t(x/2)$

$n = 2^x \Rightarrow \log n = \log 2^x \Rightarrow \theta(\log(\log n)) = \theta(\log x + \log(\log 2)) = \theta(\log x)$

$\Rightarrow t(x) = 2t(x/2) + \theta(\log x) = 2t(x/2) + f(x)$

$f(x) \in \theta(\log x)$

حال از قضیه اصلی استفاده می کنیم:

$\begin{cases} a = 2 \\ b = 2 \end{cases} \Rightarrow x^{\log_b^a} = x^{\log_2^2} = x$

$\left. \begin{array}{l} f(x) \in \theta(\log x) \\ \log x \in O(x) \end{array} \right\} \Rightarrow f(x) \in O(x)$

$\Rightarrow t(x) \in \theta(x) \overset{(I)}{\Rightarrow} T(n) \in \theta(\log n) \mid$ (n توانی از 2 باشد)

بدیهی است که غیر کاهشی است

برای اثبات هموار بودن، از قضیه ی صدا استفاده می کنیم:

Subject

Date

۱- ادامهٔ د)

$$\lim_{n\to\infty} \frac{\lg 6n}{\lg n} = \lim_{n\to\infty} \frac{\frac{\ln 6n}{\ln 2}}{\frac{\ln n}{\ln 2}} = \lim_{n\to\infty} \frac{\ln 6n}{\ln 2}$$

$\xrightarrow{\text{Hopital}} \lim_{n\to\infty} \frac{6n}{n} = 6 \rightarrow$ عدد ثابت $\Rightarrow$ همواره

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

۱-

$$e)\ F(n) = \begin{cases} 0 & \text{if } n=1 \\ 2F(\lceil n/2 \rceil) + n^2 + n\log n & O.w \end{cases}$$

$$F(n) = 2F(\lceil n/2 \rceil) + n^2 + n\log n$$

با توجه به تابع و وجود $n/2$ در قسمت بازگشتی آن، از تغییر متغیر استفاده می‌کنیم به طوری که:

$$n = 2^i \quad , \quad F(n) = t_i \quad (I)$$

$$n = 2^i \Rightarrow n^2 = (2^i)^2 = 4^i$$

$$n = 2^i \Rightarrow \log n = \log 2^i = i\log 2 \Rightarrow n\log n = 2^i * i * \log 2$$

$$\Rightarrow t_i = 2t_{i-1} + 4^i + 2^i * i * \log 2$$

$$\Rightarrow t_i - 2t_{i-1} = 4^i + 2^i * i * \log 2$$ غیرهمگن خطی با ضریب ثابت

$$\Rightarrow p(x) = (x-2)(x-4)^{1+0}(x-2)^{1+1}$$

$$\Rightarrow p(x) = (x-2)(x-4)(x-2)^2 = (x-2)^3(x-4)$$

$\Rightarrow$ ریشه: $r_1 = 2$ $\quad r_2 = 4$
تکرار: $m_1 = 3$ $\quad m_2 = 1$

$$\Rightarrow t_i = C_1 * 2^i + C_2 * i * 2^i + C_3 * i^2 * 2^i + C_4 * 4^i$$

Subject
Date

$$t_i = C_1 * 2^i + C_2 * 2^i * i + C_3 * 2^i * i^2 + C_4 * 4^i \quad \text{II}$$ (۱- ادامه ج)

حال با توجه به تعریف تابع t_i در صفحهٔ قبل، از جایگذاری استفاده می‌کنیم:

$$t_i - 2t_{i-1} = 4^i + 2^i * i * \log 2 \quad \text{III}$$

$$\xrightarrow{\text{II},\ \text{III}} C_1 * 2^i + C_2 * 2^i * i + C_3 * 2^i * i^2 + C_4 * 4^i - 2 * (C_1 * 2^{i-1}$$

$$+ C_2 * 2^{i-1} * (i-1) + C_3 * 2^{i-1} * (i-1)^2 + C_4 * 4^{i-1}) = 4^i + 2^i * i * \log 2$$

$$\Rightarrow C_2 * 2^i * i + C_3 * 2^i * (1-2i) + C_4 * 4^i - \frac{C_4 * 4^i}{4} = 4^i + 2^i * i * \log 2$$

$$\Rightarrow C_5 * 2^i * i + C_3 * 2^i + C_4 * \frac{3}{4} * 4^i = 4^i + 2^i * i * \log 2$$

حال باید در نظر بگیریم که ضرایب 4^i و $i * 2^i$ و 2^i در دو طرف باید یکسان باشند:

$$\Rightarrow \begin{cases} 2^i: & C_3 = 0 \\ 4^i: & \frac{3C_4}{4} = 1 \Rightarrow C_4 = 4/3 \\ 2^i * i: & C_5 = \log 2 \end{cases}$$

$$\Rightarrow t_i = C_1 * 2^i + C_2 * 2^i * i + C_3 * 2^i * i^2 + \frac{4}{3} * 4^i$$

Subject

Date

۱- ادامه ج) حال با توجه به نحوهٔ تغییر متغیری که در اول کار انجام دادیم، t_i را به $F(n)$ تبدیل می‌کنیم. لازم است که تمام ضرایب C_1 و ... و C_5 را بیابیم انگر دهیم دو ضریب C_4 را که مهم‌ترین ضریب است چون 4^i بزرگ‌ترین اُردر را در بین این ها دارد، پس تبدیل می‌کنیم!:

$$n^2 = 4^i$$

$$\text{I}: \begin{cases} n = 2^i \Rightarrow i = \lg n \Rightarrow i^2 = (\lg n)^2 \\ F(n) = t_i \end{cases}$$

$$\Rightarrow F(n) = C_1 * n + C_2 * n * \lg n + C_3 * n * (\lg n)^2 + \frac{4}{3} * n^2$$

$$\Rightarrow F(n) \in O(n^2)$$

Subject
Date

2- پیچیدگی زمانی کدهای زیر را محاسبه کنید.

الف)

```
for(i=1 ; i<N ; i++){
    for(j=1 ; j<N ; j++){
        n++;
        N--;
    }
}
```

برای حل این مسئله از بارومتر و تعیین متغیری که تعداد اجرای بارومتر به آن وابسته است را تعیین می‌کنیم.

ابتدا بارومتر را مشخص می‌کنیم:

با توجه به تعریف بارومتر، دو تا راه داریم یا اینکه بیاییم و تیکه $j < N$ را در نظر بگیریم (که از لحاظ تعداد دفعات اجرا) از بقیه کد، بیشتر اجرا می‌شود. یا اینکه بیاییم و عبارت داخل بدنه for داخلی را در نظر بگیریم که از لحاظ تعداد دفعات اجرا، هر سری که وارد بدنه حلقه اول می‌شویم، یک بار کمتر اجرا می‌شود.

ما حالت دوم را در نظر می‌گیریم که البته فرق بین این دو تا حالت صرفاً این است که در مقدار تابع نهایی سقف $N/2$ را حذف کنیم یا کف $N/2$ را که البته تاثیری در جواب نهایی ما ندارد.

حال با توجه به تعریف بدنه حلقه دوم به راحتی متوجه می‌شویم که تعداد دفعاتی که بارومتر ما صدا زده می‌شود، در اصل $\left\lfloor \frac{N}{2} \right\rfloor$ است (اگر $j < N$ را در نظر می‌گرفتیم می‌شد $\left\lceil \frac{N}{2} \right\rceil$)

حال باید دید که به ازای هر مقدار، حلقه اول چند بار اجرا می‌شود. فرض می‌کنیم که وارد بدنه حلقه اول می‌شویم، در این حالت با توجه به توضیحات داده شده، مقدار N، می‌شود $\left\lfloor \frac{N}{2} \right\rfloor$

Subject
Date

۲- ادامهٔ الف) ولی در حلقهٔ بیرونی، هر سری N هم به مقدار i یکی اضافه می‌شود. پس

یعنی انگار که N را ثابت نه داریم ولی هر سری، بعد از اجرای کامل حلقهٔ دومی بیاییم و

$N--$ بذاریم. و این کار آنقدر تکرار می‌شود تا $\lfloor N/2 \rfloor - 1$ به یک برسد.

حال تابع را به صورت گفته شده بازنویسی می‌کنیم و از روی تابع جدید فرمول بازگشتی آن را می‌نویسیم.

```
for (i=1 ; i<N ; ){
  for (j=1 ; j<N ; j++){
     n++ ;
     N-- ;
```

$\Rightarrow$ زمان اجرا $\in O(1)$

$N--;$ $\longrightarrow$ تکهٔ اضافه‌شدهٔ مورد نظر

جایی که می‌توانستیم سقف بذاریم ولی کف گذاشتیم اینجا بود

$$\Rightarrow T(N) = T(\lfloor N/2 \rfloor - 1) + \lfloor N/2 \rfloor * O(1)$$

$$\Rightarrow T(N) = T(\underbrace{\lfloor N/2 \rfloor - 1}) + \lfloor N/2 \rfloor$$

با توجه به اینکه این تابع غیرکاهشی است پس در اصل می‌توان این آرگومان را با علاوه یک کرد، یعنی در اصل منفی یک را برداریم و علامت مساوی را به کوچکتر مساوی تبدیل کنیم:

((یعنی در اصل چنین داریم: $T(\lfloor N/2 \rfloor - 1) \le T(\lfloor \frac{N}{2} \rfloor)$ پس می‌توان نوشت:))

Date

۲- ادامه الف)

$\Rightarrow T(N) \leq T(\lfloor \frac{N}{2} \rfloor) + \lfloor \frac{N}{2} \rfloor$

حال اگر این شرط را داشته باشیم که $N = 2^k$ است پس می‌توان نوشت:

$T(N) \leq T(N/2) + N/2$

حال از قضیه استفاده می‌کنیم و اردر تابع را بدست می‌آوریم:

$\begin{cases} a = 1 \\ b = 2 \\ k = 1 \end{cases} \Rightarrow \begin{cases} b^k = 2 \\ a = 1 \end{cases} \Rightarrow b^k > a \Rightarrow T(N) \in O(N)$

Subject
Date

2_ قسمت ب)

```
i = n
while (i >= 1){
    j = n;
    while (j >= 1){
        j = j/2;
    i = i/3;
}
}
```

اگر در این قسمت اگه بعد ما وارد حلقهٔ اول شویم پس باید ن

به اینکه n بزرگتر از یک بوده است پس قطعاً j هم

بزرگتر از یک خواهد بود و به داخل بدنهٔ حلقهٔ دوم خواهیم رفت.

حال قبل از اینکه بدنهٔ حلقهٔ دوم اجرا شود مقدار i و j

با هم مساوی است و با هر بار اجرای بدنهٔ حلقهٔ داخلی، مقدار i قطعاً همیشه کمتر

از مقدار j خواهد بود، پس هر وقت از حلقهٔ داخلی خارج شویم چون $1 \geq j$ است

پس قطعاً i هم قطعاً باید کوچکتر یا مساوی یک باشد. پس به این ترتیب بعد از خروج

از حلقهٔ داخلی، از حلقهٔ اول هم خارج می‌شود. پس در اصل باید ببینیم بدنهٔ حلقهٔ داخلی

(که در اصل بار و متر است و در $O(1)$ اجرا می‌شود، چند بار تکرار می‌شود)). در هر مرحله

j تقسیم بر 2 می‌شود پس باید ببینیم که اولین توان 2 که بزرگتر از j است چیست.

یعنی:

$$\frac{j}{2^i} \leq 1 \Rightarrow j \leq 2^i \Rightarrow \lg j \leq i$$

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

البته حتی می‌توان گفت که: $T(n) \in \Theta(\lg n)$

Subject
Date

3- پیچیدگی زمانی قطعه کد زیر را کسب کنید و راجب عملکرد آن توضیح دهید.

```
int rec(vector<int> &v, int k, int start, int end){

    if(start > end) return 0;
    int mid = (start + end)/2;
    if(v[mid]<k) return rec(v, k, mid+1, end);
    if(v[mid]>k) return rec(v, k, start, mid-1);
    return rec(v, k, start, mid-1) + 1 + rec(v, k, mid+1, end);
}
```

عملکرد این تابع به نحوه ترتیب اعداد در vector وابسته است. حال ممکن است دو حالت رخ دهد:

1- اگر اعداد موجود در vector منظم باشند: در این حالت می‌توان گفت که این تابع عملکردی همانند Binary Search دارد با این تفاوت که در اینجا ما تعداد تکرار عدد k در vector را بررسی می‌کنیم، یعنی خروجی کلی این تابع در این حالت این است که چند بار عدد k در vector آمده است. در این حالت در بهترین حالت هیچ عدد kای در vector موجود نیست و در این حالت همیشه داریم ((منظور از n، تعداد اعداد موجود در vector است)):

$$T(n) = T\left(\frac{n}{2}\right) + O(1) \xrightarrow{\text{قضیه اصلی}} T(n) \in O(\lg n)$$

که در این حالت زمان مصرفی ما از $O(\lg n)$ است.

3- ادامه: اما بهترین حالت زمانی اتفاق می افتد که تمام عنصرهای عدد k باشد یعنی تک تک عناصر vector برابر با k باشند در آن صورت (منظور از n، تعداد اعداد موجود در vector است):

$$T(n) = T(n/2) + 1 + T(n/2)$$

$$\Rightarrow T(n) = 2T(n/2) + 1 \xRightarrow{\text{قضیه اصلی}} T(n) \in O(n)$$

که در این حالت زمان مصرفی ما از $O(n)$ است.

2- اگر اعداد موجود در vector منفی باشند: در این حالت عملکرد این تابع غیرقابل بررسی (اگر ورودی ها را نداشته باشیم قابل بررسی نیستند) می شود و باز هم ممکن است در بهترین و بدترین حالت با همان اردر های قبل کار کنند ولی عددی که تابع به عنوان خروجی برمی گرداند دیگر لزوما تعداد تکرار عدد k در بین عناصر vector نیست و عددی که برمی گرداند صرفا کوچک تر یا مساوی تعداد تکرار k در عناصر vector است. (برای مثال اگر 4=k را بخواهیم)

5	2	3	4	1

مثلا در اینجا اول به سراغ 3 می رود سپس چون کوچک تر از 4 است نیمه سمت چپ آرایه را بررسی می کند و الگوریتم به خانه ای که 4 هست نمی رسد) عدد صفر را برمی گرداند

5	2	4	3	1

ولی در حالت روبه رو هم عدد 1 را به عنوان خروجی برمی گرداند:

Subject
Date

۴_ نرخ رشد تابع زیر را حساب کنید.

$T(1) = 10$

for all $n \geq 1$

$T(n+1) = 2n + T(n)$

تغییر متغیر می دهیم به طوری که:

$i = n+1 \Rightarrow n = i-1$

$n \geq 1 \Rightarrow i \geq 0$

حال تابع $T(n)$ را می نویسیم:

$T(i) = T(i-1) + 2n$

$\Rightarrow T(i) - T(i-1) = 2(i-1) * 1^i$ (I)

با توجه به اینکه آن طرف تساوی صفر نشد پس یعنی یک تابع غیرهمگن خطی با ضریب ثابت داریم، پس:

$P(x) = (x-1)(x-1)^{1+1} = (x-1)^3 \Rightarrow \begin{cases} r_1 = 1 \\ m_1 = 3 \end{cases}$

$\Rightarrow T_{(i)} = C_1 * 1^i + C_2 * n * 1^i + C_3 * n^2 * 1^i$

$\Rightarrow T(i) = C_1 + C_2 * i + C_3 * i^2$ (II)

حال با توجه به تغییر جایگذاری خواهیم داشت: ((بازنویسی (I)))

$C_1 + C_2 * i + C_3 * i^2 - (C_1 + C_2 * (i-1) + C_3 * (i-1)^2) = 2(i-1)$

PAPCO

Subject
Date

ادامه سؤال 4 ـ

$\Rightarrow c_2 + 2i * c_3 - c_3 = 2(i-1)$

حال می‌دانیم که ضرایب باید در دو طرف یکی باشد. پس:

$$\begin{cases} c_2 - c_3 = -2 \\ 2i * c_3 = 2i \Rightarrow c_3 = 1 \end{cases} \Rightarrow c_2 = -1 \text{ , } c_3 = 1$$

حال با توجه به تعریف متغیر بازگشتی تابع خط (II) را بازنویسی می‌کنیم:

$\Rightarrow T(i) = c_1 - i + i^2$

$\xrightarrow{i=n+1} T(n+1) = c_1 - (n+1) + (n+1)^2 \quad , \quad n \geq 1$

$\Rightarrow T(n) = c_1 - n + n^2 \quad , \quad n \geq 2$

حال با توجه به تعریف تابع مقدار c_1 را به ازای $n=2$ بدست می‌آوریم (البته بدون این کار هم مشخص است که $T(n) \in O(n^2)$ است و حتی می‌توان گفت که $T(n) \in \Theta(n^2)$ می‌باشد):

با توجه به تعریف تابع $\Rightarrow T(1+1) = T(2) = 2*1 + T(1) = 12$

$T(2) = c_1 - 2 + 2^2 = c_1 + 2$

$\Rightarrow c_1 = 10$

$$\Rightarrow T(n) = n^2 - n + 10 \Rightarrow \begin{cases} T(n) \in \Theta(n^2) \\ T(n) \in O(n^2) \end{cases}$$

((می‌شد c_1 را هم از $T(1)$ بدست آورد که می‌شد 10 باز هم و فرقی نداشت))