DA_HW3

ص 175

-5

a) $4T(\frac{n}{2})+n \xrightarrow[\text{theorem}]{\text{master}} 4>2^1 \longrightarrow T(n)=\Theta(n^2)$ | b) $4T(\frac{n}{2})+n^2 \longrightarrow 4=2^2 \longrightarrow T(n)=\Theta(n^2\log n)$

c) $4T(\frac{n}{2})+n^3 \longrightarrow 4<2^3 \longrightarrow T(n)=\Theta(n^3)$

۶ - از ایده marge-sort استفاده می کنیم. به این صورت که هر بار به صورت بازگشتی آرایه را به دو نیمه تقسیم می کنیم و جواب را برای چپ و راست حساب می کنیم. اینجا جواب به صورت [جواب چپ + جواب راست + تعداد inversion های چپ و راست] است.

A B C

برای حالت C: اگر اول دو متغیر در نیمه های چپ و راست باشند. اگر arr[i] < arr[j]، به تعداد mid-i تا inversion جدید داریم. چون چپ و راست مرتب شده اند و تمام عناصر چپ از arr[j] بزرگتراند. این را ادامه می دهیم تا به پایه برسیم.

هزینه این الگوریتم مثل خود mergesort، $O(n\log n)$ است.

ص ۱۵۱

8 - از ایده Quick sort استفاده می کنیم. به این صورت که 3 پارتیشن A، B، C داریم که در A همه عناصر منفی، در C همه مثبت و در B نامشخص اند. ابتدا پارتیشن های A و C خالی اند اما در هر دور، یکی از عناصر B به A یا C داده می شود. تا جایی که پارتیشن B خالی شود. پیچیدگی های زمانی و فضایی این الگوریتم شبیه Quick sort است.

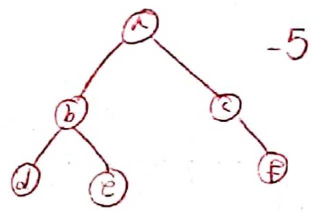
ص 185

2-

ایراد کد این است که برای هر درختی جواب صفر می دهد و در هیچ قسمتی برگ را در نظر نگرفته است. از ویژگی برگ که فرزندی ندارد استفاده کرد و کد را بصورت روبه رو اصلاح کرد:

```
LeafCounter(T)
    if T=∅    return 0;
else if T_L = ∅ and T_R = ∅    return 1;
    else return LeafCounter(T_L) + LeafCounter(T_R);
```

5-



a) Preorder: a, b, d, e, c, f

b) inorder: d, b, e, a, c, f

c) Postorder: d, e, b, f, c, a

7- پیمایش inorder یک لیست سورت شده خروجی می دهد. چون در BST هر نود Parent از فرزند چپ بزرگتر و از فرزند راست کوچکتر است. پس اگر آنها را سورت کنیم باید بصورت Left, Parent, Right بیایند. در پیمایش inorder دقیقا همین ترتیب رعایت می شود. پس لیست سورت شده خروجی می دهد.

ص ۱۹۱

2- 2101×1130

$C_2 : 21 \times 11$

$C_0 = 01 \times 30$

$C_1 = (21+01)\times(11+30) - (21\times 11 + 01\times 30) = 22\times 41 - 21\times 11 - 01\times 30$

$C_2 : 21\times 11$

$C_2 = 2\times 1 = 2$

$C_0 = 1\times 1 = 1$

$C_1 = (2+1)\times(1+1) - (2+1) = 3$

$\Rightarrow C_2 = 21\times 11 = 231$

$C_0 : 01\times 30$

$C_2 = 0\times 3 = 0$

$C_0 = 1\times 0 = 0$

$C_1 = (0+1)\times(3+0) - (0+0) = 3$

$\Rightarrow C_0 = 01\times 30 = 30$

$22\times 41:$

$C_2 = 2\times 4 = 8$

$C_0 = 2\times 1 = 2$

$C_1 = (2+2)\times(4+1) - (8+2) = 10$

$\Rightarrow 22\times 41 = 902$

$\Rightarrow 2101\times 1130 = 231\times 10^4 + (902-231-30)\times 10^2 + 30\times 10^0 = \underline{2,374,130}$, (why? :))

ص ١٩١

$m_1 + m_4 - m_5 + m_7 = (a_{00} + a_{11})(b_{00} + b_{11}) + a_{11}(b_{10} - b_{00}) - (a_{00} + a_{01})b_{11} + (a_{01} - a_{11})(b_{10} + b_{11})$

-6

$\rightarrow a_{00}b_{00} + a_{11}b_{00} + a_{00}b_{11} + a_{11}b_{11} + a_{11}b_{10} - a_{11}b_{00} - a_{00}b_{11} - a_{01}b_{11} + a_{01}b_{10} - a_{11}b_{10} + a_{01}b_{11} - a_{11}b_{11}$

$\rightarrow \underline{a_{00}b_{00} + a_{01}b_{10}}$

$m_3 + m_5 = a_{00}(b_{01} - b_{11}) + (a_{00} + a_{01})b_{11} = a_{00}b_{01} - a_{00}b_{11} + a_{00}b_{11} + a_{01}b_{11} = \underline{a_{00}b_{01} + a_{01}b_{11}}$

$m_2 + m_4 = (a_{01} + a_{11})b_{00} + a_{11}(b_{10} - b_{00}) = a_{10}b_{00} + a_{11}b_{00} + a_{11}b_{10} - a_{11}b_{00} = \underline{a_{10}b_{00} + a_{11}b_{10}}$

$m_1 + m_3 - m_2 + m_6 = (a_{00} + a_{11})(b_{00} + b_{11}) + a_{00}(b_{01} - b_{11}) - (a_{10} + a_{11})b_{00} + (a_{10} - a_{00})(b_{00} + b_{01})$

$\rightarrow a_{00}b_{00} + a_{11}b_{00} + a_{00}b_{11} + a_{11}b_{11} + a_{00}b_{01} - a_{00}b_{11} - a_{10}b_{00} - a_{11}b_{00} + a_{10}b_{00} - a_{00}b_{00} + a_{10}b_{01} - a_{00}b_{01}$

$\rightarrow \underline{a_{10}b_{01} + a_{11}b_{11}}$

7- ص ۱۵۱

$$A = \begin{array}{cc} A_{00} & A_{01} \end{array}$$
$$A = \left[\begin{array}{cc|cc} 1 & 0 & 2 & 1 \\ 4 & 1 & 1 & 0 \\ \hline 0 & 1 & 3 & 0 \\ 5 & 0 & 2 & 1 \end{array}\right]$$
$$\begin{array}{cc} A_{10} & A_{11} \end{array}$$

$$B = \begin{array}{cc} B_{00} & B_{01} \end{array}$$
$$B = \left[\begin{array}{cc|cc} 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 4 \\ \hline 2 & 0 & 1 & 1 \\ 1 & 3 & 5 & 6 \end{array}\right]$$
$$\begin{array}{cc} B_{10} & B_{11} \end{array}$$

$$M_1 = (A_{00} + A_{11})(B_{00} + B_{11}) = \begin{bmatrix} 4 & 0 \\ 6 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 7 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 8 \\ 20 & 14 \end{bmatrix}$$

$$M_2 = (A_{10} + A_{11})(B_{00}) = \begin{bmatrix} 3 & 1 \\ 7 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 2 & 8 \end{bmatrix}$$

$$M_3 = A_{00}(B_{01} - B_{11}) = \begin{bmatrix} -1 & 0 \\ -9 & 4 \end{bmatrix}$$

$$M_4 = A_{11}(B_{10} - B_{00}) = \begin{bmatrix} 6 & -3 \\ 3 & 0 \end{bmatrix}$$

$$M_5 = (A_{00} + A_{01}) B_{11} = \begin{bmatrix} 8 & 3 \\ 10 & 5 \end{bmatrix}$$

$$M_6 = (A_{10} - A_{00})(B_{00} + B_{01}) = \begin{bmatrix} 2 & 3 \\ -2 & -3 \end{bmatrix}$$

$$M_7 = (A_{01} - A_{11})(B_{10} + B_{11}) = \begin{bmatrix} 3 & 2 \\ -9 & -4 \end{bmatrix}$$

$$\Rightarrow C_{00} = M_1 + M_4 - M_6 + M_7 = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$$

$$C_{01} = M_3 + M_6 = \begin{bmatrix} 7 & 3 \\ 1 & 9 \end{bmatrix}$$

$$C_{10} = M_2 + M_4 = \begin{bmatrix} 8 & 1 \\ 5 & 8 \end{bmatrix}$$

$$C_{11} = M_1 + M_3 - M_2 + M_6 = \begin{bmatrix} 3 & 7 \\ 7 & 7 \end{bmatrix}$$

$$\Rightarrow A \times B = \begin{bmatrix} 5 & 4 & 7 & 3 \\ 4 & 5 & 1 & 9 \\ 8 & 1 & 3 & 7 \\ 5 & 8 & 7 & 7 \end{bmatrix}$$

۱- کافی است که mergesort روی آرایه بزنیم و سپس فاصله میان عناصر مجاور را بدست آوریم. mergesort یک عملیات کاملا Devide-and-Conquer است. همچنین این عملیات $O(n\log n)$ می شود چون صرفا یک سورت و یک بار پیمایش دارد.

3-

فرمول بازگشتی مسئله : $T(n) = 2\left(T\left(\frac{n}{2}\right) + \frac{n}{2}\log\frac{n}{2}\right) \xrightarrow{n=2^k} T(2^k) = 2T(2^{k-1}) + 2^k(k-1)$

$$\to T(2^k) = 2\left(2T(2^{k-2}) + 2^{k-1}(k-2)\right) + 2^k(k-1) = 2^2 T(2^{k-2}) + 2^k(k-2) + 2^k(k-1)$$

$$\vdots$$

$$T(2^k) = 2^{k-1}T(2^1) + 2^k + 2^k 2 + \dots + 2^k(k-1) = 2^k + 2^k(1+2+\dots+(k-1)) = 2^{k-1} + 2^k\frac{(k-1)k}{2}$$

$$\to 2^{k-1}(1 + k(k-1)) \xrightarrow[\log n = k]{n=2^k} \frac{n}{2}(1 + (\log n - 1)\log n) \to T(n) = \Theta(n\log^2 n)$$

7- تمام نقاط گفته شده در واقع رئوس سوم مثلث با رأس های P_n و P_1 است و دورترین نقطه، نقطه‌ای است که مساحت این مثلث را maximum کند. پس P_{max} در واقع نقطه‌ای با مختصات (i,j) است که مساحت مثلث را میتواند maximum کند.

از نظر ریاضی با فرمول دترمینان میتوان به مساحت مثلث رسید: ←

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_n & y_n & 1 \\ i & j & 1 \end{vmatrix}$$