"به نام خدا"

Subject:

Year. .. Month. ۲ Day. ۱۹

حل سری دوم تمرینات الگوریتم

مصطفی فضلی شهری ۹۸۲۲۶۸۰۲

قرمول حل سؤالات بازگشتی: $T(n) = aT(\frac{n}{b}) + f(n)$

$T(n) \rightarrow \Theta(n^{\log_b a})$ if $f(n) < n^{\log_b a}$

$T(n) \rightarrow \Theta(f(n) \cdot \log n)$ if $f(n) = n^{\log_b a}$

$T(n) \rightarrow \Theta(f(n))$ if $f(n) > n^{\log_b a}$

1.a) $T(n) = 2T(\frac{n}{2}) + n^4$

$a=2 \quad b=2 \quad f(n) = n^4$

$n^4 > n^{\log_2 2} \Rightarrow f(n) > n^{\log_b a}$

$\Rightarrow \Theta(n^4)$

1.b) $T(n) = 7T(\frac{n}{10}) + n \quad a=7 \quad b=10 \quad f(n) = n$

$n \boxed{<} n^{\log_{10} 7} \Rightarrow n > n^{0.85} \Rightarrow f(n) > n^{\log_b a} \Rightarrow \Theta(n)$

1.c) $T(n) = 7T(\frac{n}{3}) + n^2 \quad a=7 \quad b=3 \quad f(n) = n^2$

$n^2 \boxed{?} n^{\log_3 7} \Rightarrow n^2 > n^{1.7} \Rightarrow f(n) > n^{\log_b a} \Rightarrow \Theta(n^2)$

1.d) $T(n) = \sqrt[4]{2}\, T(\frac{n}{4}) + \sqrt[5]{n} \quad a = \sqrt[4]{2} = 2^{1/4} \quad b=4 \quad f(n) = \sqrt[5]{n} = n^{1/5}$

$n^{1/5} \;?\; n^{\log_4 2^{1/4}} \Rightarrow n^{1/5} > n^{1/8} \Rightarrow \Theta(n^{1/5})$

دوتا دوتا کم میشه

1.e) $T(n) = 2T(n-2) + n^2 \Rightarrow 2(2T(n-4) + (n-2)^2) + n^2 = T(n-2) \Rightarrow \dots$

$\Theta(2^{n/2} \times n^2) = \Theta(2^{n/2} n^2)$

$T(n-4) = 2(2(2T(n-6) + (n-4)^2) + (n-2)^2) + n^2$

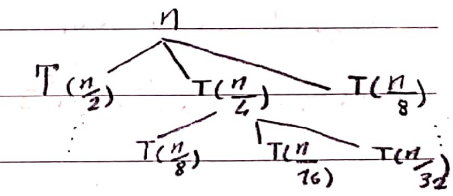
1.f) $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + n$

$T(n) = T(\frac{n}{a}) + T(\frac{n}{b}) + \dots + cn$

height

$= n \sum_{i=0}^{\log_a n} (\frac{1}{a} + \frac{1}{b} + \dots)^i \Rightarrow n \sum_{i=0}^{\log_2 n} (\frac{1}{2} + \frac{1}{4} + \frac{1}{8})^i \Rightarrow$

$n \sum_{i=0}^{\log_2 n} (\frac{7}{8})^i \Rightarrow \Theta(n \log n)$



1.g) $T(n) = 3T(\frac{n}{3} - 2) + \frac{n}{2} \quad a=3 \quad b=3 \quad f(n) = \frac{n}{2}$

$\frac{n}{2} \;?\; n^{\log_3 3} \Rightarrow \frac{n}{3} > n^{\log_3 3} \Rightarrow \Theta(\frac{n}{3}) \simeq \Theta(n) \qquad \Theta(3^{\log_3 2} \times \frac{n}{2}) \simeq \Theta(n \log n)$

1.h) $T(n) = 3T(\frac{n}{3}) + \frac{n}{\log n} \quad a=3 \quad b=3 \quad f(n) = \frac{n}{\log n}$

$\frac{n}{\log n} \;?\; n^{\log_3 3} \Rightarrow f(n) < n^{\log_b a} \Rightarrow \Theta(n)$

Subject:

Year. Month. Day.

1.i) $T(n) = 4T(\frac{n}{3}) + n\lg n \quad a=4 \quad b=3 \quad f(n) = n\log n$

$n\log n \overset{?}{<} n^{\log_3^{2a}} \Rightarrow f(n) < n^{\log_3^4} \Rightarrow \theta(n^{\log_3^4})$

1.j) $T(n) = 4T(\frac{n}{2}) + n^2\sqrt{n} \quad a=4 \quad b=2 \quad f(n) = n^2\sqrt{n} = n^{\frac{5}{2}}$

$n^{\frac{5}{2}} \overset{?}{<} n^{\log_2^4} \Rightarrow n^{\frac{5}{2}} > n^2 \Rightarrow \theta(n^2\sqrt{n}) = \theta(n^{\frac{5}{2}})$

1.k) $T(n) = 2T(\sqrt{n}) + O(1)$ تغییر متغیر $m = \log_2 n \Rightarrow n = 2^m$

$T(2^m) = 2T(2^{\frac{m}{2}}) + 1 \Rightarrow F(Y) = 2F(\frac{m}{2}) + 1 \quad a=2 \quad b=2 \quad f(n)=1$

$1 \overset{?}{<} n^{\log_2^2} \Rightarrow 1 < n \Rightarrow \theta(m) \overset{m=\log_2 n}{=\!=} \theta(\log_2 n)$

1.l) $T(n) = T(\sqrt{n}) + O(\log\log n) \quad m = \log_2 n \quad 2^m = n \quad a=1 \quad b=2 \quad f(n) = \log m$

$T(2^m) = T(2^{\frac{m}{2}}) + O(\log m) \Rightarrow S(m) =$

2 مرتب سازی سریع ← (در این مورد که تعداد زیادی داده را باید مرتب نمائیم، با توجه به تقسیم بندی مرتب سازی سریع به نفت و در نتیجه ایجاد وسکس مسئله به زیر مسئله های کوچکتر می توان از مرتب سازی سریع استفاده کرد. از طرفی به دلیل تکراری نبودن شماره های دانشجویی نیاز چندانی به پایداری الگوریتم نداریم. همچنین به دلیل درجا بودن این مرتب سازی سریع، حافظه مصرفی مستقل از طول آرایه است.

3 با توجه به maxheap یا minheap بودن الگوریتم مرتب سازی هرمی، ممکن است برعکس هم باشد، اگر maxheap باشد و آرایه به صورت صعودی مرتب شده باشد، پس نیازی به تغییر در آرایه نداریم و به صورت از پیش آماده آرایه ها مرتب است و مرتبه زمانی آن $O(1)$ است و اگر به صورت نزولی مرتب شده باشد باید همه عناصر را به صورت صعودی مرتب کنیم و این روند به صورت $O(n\log n)$ می باشد که بدترین حالت برای مرتب کردن در مرتب سازی هرمی هم همین حالت است.

4- $\overset{1}{0172}, \overset{2}{0173}, \overset{3}{0176}, \overset{4}{0164}, \overset{5}{0139}, \overset{6}{0120}, \overset{7}{0189}, \overset{8}{0153}, \overset{9}{0171}, \overset{10}{0142}$

$n=10, \lfloor 0172 \times 10 \rfloor = 719 - 7, \lfloor 0173 \times 10 \rfloor = 1, \ldots$

1	2	3	4	5	6	7	8	9	10
0113	0172	0139	0142	0153	0164	0171	0189		
0176						0177			

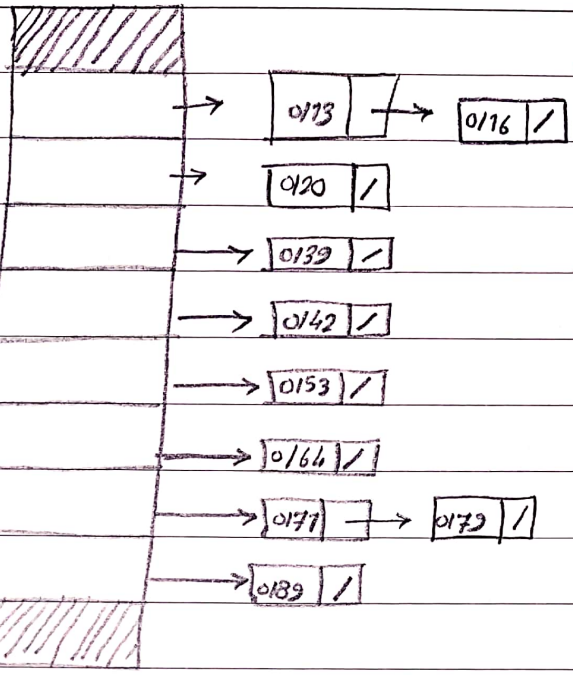
این الگوریتم بدین صورت کار می کند که حول پائین هر عنصر را در تعداد عناصر حساب کرده و روی خانه ی آن در جدول قرار داده و خانه کم شود و کنیم از ابتدا عناصر به صورت صعودی مرتب شده اند.

Subject:
Year: Month: Day.

(I)

1	0179
2	0173
3	0176
4	0164
5	0139
6	0120
7	0189
8	0153
9	0171
10	0142

(II)



ادامه سوال ۴ خانه‌های درون هر آرایه به صورت لینکلیست تعریف شده‌اند.

5 الف) برای رابطه بازگشتی در هر گذر یک عنصر محور انتخاب می‌شود و سپس عناصر آرایه به گونه‌ای که همه عناصر بزرگتر در سمت راست آن و همه عناصر کوچکتر در سمت چپ آن قرار گیرند. بدین ترتیب این عنصر هر بار در وسط قرار گرفته و دو قسمت دیگر آرایه نیز به همین روش و به صورت بازگشتی حل می‌شوند.

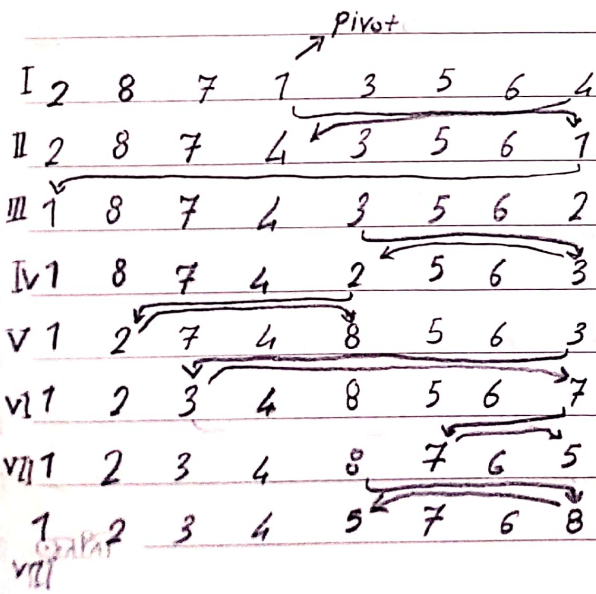
best → $T(n) = 2T(n/2) + \Theta(n) = O(n \log n)$

worst → $T(n) = T(n-1) + \Theta(n) = O(n^2)$

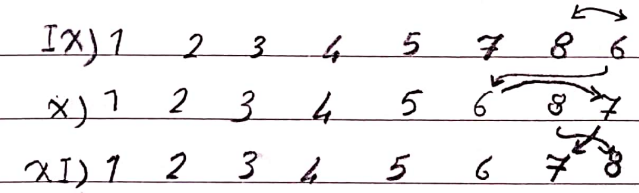
ب) این الگوریتم بدترین عملکرد را هنگامی خواهد داشت که آرایه ما به صورت نزولی مرتب شده باشد و هر بار هر عنصر نیاز به جابجایی با باشد، مثلاً اگر عنصر پایه یا محور برابر بزرگترین یا کوچکترین عنصر آرایه باشد.

ج) الگوریتم مرتب‌سازی ادغامی یک الگوریتم پایدار است و حافظه‌ی کمتری مصرف می‌کند اما پیچیدگی زمانی آن در بدترین حالت و حالت میانگین $O(n \log n)$ است که این مقدار در الگوریتم مرتب سازی سریع و حالت میانگین $O(n^2)$ است و زمان کمتری را برای مرتب‌سازی مصرف می‌کند همچنین الگوریتم مرتب‌سازی سریع در قیاس با ادغامی در حجم زیاد داده می‌تواند تفاوت زیادی داشته باشد و بسیار کارآمدتر باشد.

د)



compare Pivot



Subject:
Year. Month. Day.

6- الف) مرتب سازی شمارشی Counting sort (مرتب سازی سطلی bucket sort)

ب) مرتب سازی درجی insertion sort یا مرتب سازی پایه ای radix sort

ج) مرتب سازی درجی insertion sort

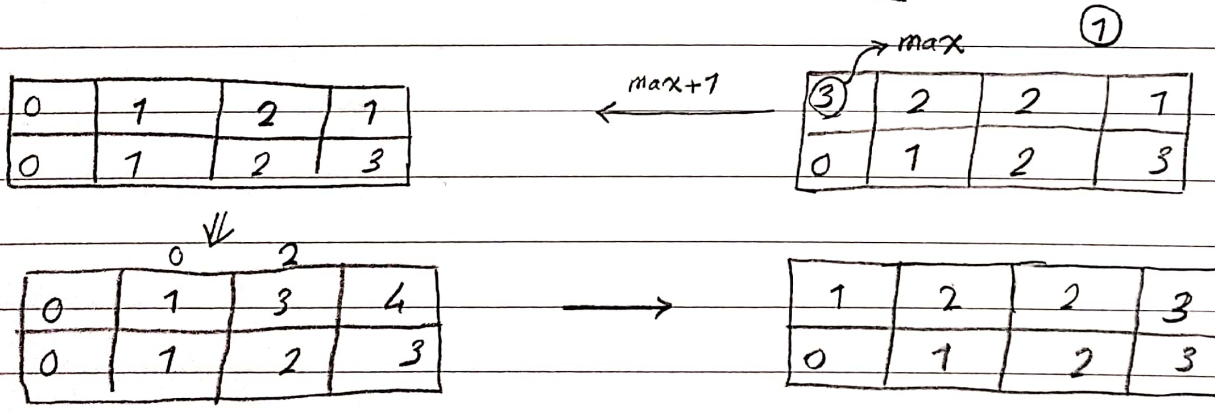
د) ترکیب این مرتب سازی پایه ای با سریع می توان این سوال را به صورت $O(bn)$ حساب کرد، بدین صورت که آرایه را از بالا به پایین با عناصری که با یک آغاز می شوند و از پایین به بالا با عناصری که با صفر آغاز می شوند مرتب کرده، گویی هر بار باجیب ترین عنصر آرایه را مرتب می کنیم.

همچنین این مرتب سازی پایه ای را می توان با مرتب سازی شمارشی تلفیق کرد، بدین صورت که با توجه به مرتبه وارزش هر عدد آن را در یک خانه از آرایه counting sort را انجام می دهیم و هرگاه می هر بار و b بار مرتب سازی شمارشی که از مرتبه زمانی $O(n)$ است را انجام می دهیم و هنگامی که این ارزش یکان، دهگان، صدگان و ... را به ترتیب مرتب شده توسط counting sort کنار هم بگذاریم b با $O(n)$ را تلفیق تا مرتب سازی را انجام دهیم. در آخر به مرتبه زمانی $O(bn)$ می رسیم.

(b تعداد بیت ها یا ارزش ارقام)

7- برای اینکار ابتدا بیشترین عنصر آرایه را می یابیم، پس از پیدا کردن بیشترین عنصر آرایه یک آرایه به طول یک خانه بیشتر از max آرایه درست می کنیم. پس این آرایه ای دوتایی که تشکیل داده ایم را با روش تعداد خانه ها در اندیس خانه ها حساب می کنیم، در مرحله سوم آرایه ای به طول max+1 تشکیل می دهیم و خانه های آرایه ی قبلی را دوبه دو جمع کرده و در آرایه ی بعدی قرار می دهیم در هر حله آخر از آخرین خونه آرایه اول شروع کرده و آن عدد را در اندیس آرایه ی دوم پیدا کرده و عدد آن را در آرایه سوم می نویسیم و مقدار آن را در آرایه دوم یکی کم می کنیم. بدین صورت مرتب سازی انجام شده است. با توجه به توضیحات مرتبه زمانی این الگوریتم از مرتبه $n+K$ است که با جدا کردن K داریم: $O(n)$ (این سوال در منابع اینترنتی نیز حل شده است)

مثلاً:



8- این سوال به چندین صورت حل می شود: روش ساده این است که حریک آرایه به صورت جدا خانه هایی که تغییر یافته اند را ذخیره می کنیم، پس معادل این خانه ها در آرایه اصلی حذف می کنیم، این آرایه ای که مرتب سازی آن از بین رفته است را با الگوریتم های همگف نظیر مرتب سازی سریع مرتب می کنیم و در هرحله آخر این خانه ها را در آرایه اصلی که این خانه ها در آن حذف شده بود قرار می دهیم!

روش دیگر استفاده از الگوریتمی به اسم timsort است که با $O(n)$ کار می کند.

همچنین با توجه به ریاضیات لهن، اجا بجایی حداکثر 20 خانه را از مرتب سازی ازبین می برد که با جابجایی آن ها و جا بجا کردن تغییر یافته ها به $O(n)$ می رسیم.