دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

10/28/2020

Homework 2

Heap Sort - Divide and Conquer

ALGORITHM DESIGN

امیرحسین علی بخشی – ۹۷۳۱۰۹۶
دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

بخش تئوری

۱) سکه داریم که برخی از آن ها سنگین تر هستند. تعداد سکه های سنگین تر را با انجام $O(\log^2 n)$ بار وزن کردن بیابید. توجه کنید که وزن همه سکه‌های سنگین و همچنین وزن همه سکه‌های سبک مساوی هم است.(۱ نمره)

دو متغیر m و c را در نظر میگیریم. برای حل سوال از BST استفاده میکنیم. دسته ها را به دو قسمت تقسیم میکنیم و هر قسمت را آنقدر باز هم به دو قسمت تقسیم میکنیم تا به دسته هایی با تعداد یک سکه برسیم. در این حالت سکه ها را دو به دو مقایسه میکنیم. اگر وزن دو سکه متفاوت بود یعنی یکی از سکه های سنگین پیدا شده پس c را ++ میکنیم. در غیر این صورت m را بعلاوه ۲ میکنیم. در صورت تساوی ناچاریم به مراحل قبل برگردیم تا مقایسه را با سایر شاخه ها انجام دهیم. تا زمان برقرار بودن این تساوی این m آنقدر با سایز شاخه ها انتقال میابد و زیاد میشود تا اینکه متوجه شویم نسبت به شاخه های دیگر چه وضعیتی دارد. اگر مقدار m از قسمت دیگر بیشتر باشد یعنی اون دو سکه ی یکسان سنگین بوده اند و مقدار c با m جمع شده و آپدیت میشود. وقتی به بالای BST رسیدیم تعداد سنگین ها مشخص شده اند.

۲) اثبات کنید که $O(n\lg n)$ جواب $T(n) = T(\lfloor n/2\rfloor+17)+n$ است.(۰.۵ نمره)

$$\begin{cases} \text{مثال رابطه بازگشتی}: T(n) = T\left(\left\lfloor\frac{n}{2}\right\rfloor + 17\right) + n \\ \text{حدس}: T(n) = O(nlgn) \end{cases} \Rightarrow \begin{cases} \text{فرض استقرا}: T\left(\left\lfloor\frac{n}{2}\right\rfloor + 17\right) \le c\left(\left\lfloor\frac{n}{2}\right\rfloor + 17\right) lg\left(\left\lfloor\frac{n}{2}\right\rfloor + 17\right) \\ \text{حکم استقرا}: T(n) \le cnlgn \end{cases}$$

$$T(n) = T\left(\left\lfloor\frac{n}{2}\right\rfloor + 17\right) + n$$

$$\le c\left(\left\lfloor\frac{n}{2}\right\rfloor + 17\right)\lg\left(\left\lfloor\frac{n}{2}\right\rfloor + 17\right) + n$$

$$\cong c\left\lfloor\frac{n}{2}\right\rfloor \lg\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + n \qquad \left(n \to \infty \Rightarrow \left\lfloor\frac{n}{2}\right\rfloor + 17 \cong \left\lfloor\frac{n}{2}\right\rfloor\right)$$

$$\le c\frac{n}{2}\lg\frac{n}{2} + n \qquad (\lfloor\alpha\rfloor \le \alpha)$$

$$= \frac{c}{2}n(\lg n - \lg 2) + n = \frac{c}{2}n\lg n - \frac{c}{2}n + n$$

$$= \frac{c}{2}n\lg n + (1 - \frac{c}{2})n$$

$$\boxed{T(n) \le cnlgn} \qquad (\forall c \ge 2)$$

۳) دو آرایه مرتب شده از اعداد داریم .الگوریتمی با زمان O(log(n+m)) برای پیدا کردن عنصر k ام در ترکیب مرتب شده این دو آرایه پیشنهاد کنید.(۱ نمره)

آرایه ها را باید در هر مرحله به دو قسمت تقسیم کنیم به طوری که اختلاف حاصل جمع هر یک از دو قسمت کمترین مقدار ممکن را داشته باشد.(شاید نتوانیم برابری را تضمین کنیم) شرط دیگر این تقسیم بندی این است که بزرگترین عنصر سمت چپ یکی از کوچکترین عنصر سمت راست دیگری کوچکتر باشد.

I. $maxLeftX \leq minRightY$
II. $maxLeftY \leq minRightX$

در این روش سه حالت ممکن است رخ دهد.

یک حالت این است که تمامی شرط ها برقرار باشند. حالت دیگر این است که یکی از دو شرط نقض شوند. که برای شرط باید برای x به سمت چپ و برای دومی باید به سمت راست حرکت نماییم. در صورت نقض شدن مراحل تکرار میشوند تا عنصر میانه پیدا شود. (در صورت خالی شدن یکی از طرفین از بینهایت استفاده میکنیم تا بتوان مقایسه را انجام داد)

حال باید جای میانه را با k مقایسه میکنیم تا بدانیم چه باید بکنیم.

- اگر k با مکان میانه برابر بود که به جواب میرسیم.
- در غیر این صورت دو حالت داریم:
  - اگر جای میانه از k بزرگتر باشد در سمت چپ تکه اعداد به دنبال k میگردیم.
  - اگر جای میانه از k کوچکتر باشد باید در سمت راست تکه اعداد به دنبال k بگردیم.

۴) برای هر T(n) بهترین کران بالا و پایین حدی پیدا کنید. فرض کنید T(n) برای n های کوچک ثابت است. جواب های خود را توجیه کنید. (هر کدام ۰.۵ نمره)

1) $T(n) = 3T(n/3) + n/lgn$
2) $T(n) = 4T(n/2) + n^2\sqrt{n}$
3) $T(n) = T(n/2) + T(n/4)+T(n/8)+n$
4) $T(n) = T(n\text{-}2) + 1/lgn$
5) $T(n) = \sqrt{n}\, T(\sqrt{n}) + n$

$$\boldsymbol{T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{\lg n}}$$

$$n^{\log_b a} = n^{\log_3 3} = n^1 = n$$

$f(n) = \frac{n}{\lg n} \cong n^{0.78} = O\left(n^{\log_b a - \varepsilon}\right) = O(n^{1-\varepsilon}), where\ 0 < \varepsilon \leq 0.22$

master theorem بند اول : $T(n) = \Theta(n^{\log_b a}) = \Theta(n)$

$$\boldsymbol{T(n) = 4T\left(\frac{n}{2}\right) + n^2\sqrt{n}}$$

$n^{\log_b a} = n^{\log_2 4} = n^2$

$f(n) = n^2\sqrt{n} = n^{\frac{5}{2}} = \Omega\left(n^{\log_b a + \varepsilon}\right) = \Omega(n^{2+\varepsilon}), where\ 0 < \varepsilon \leq \frac{1}{2}$

$af\left(\frac{n}{b}\right) = 4\frac{n^2}{4}\sqrt{\frac{n}{2}} = \frac{\sqrt{2}}{2}n^2\sqrt{n} \leq cf(n) = cn^2\sqrt{n}, where\ \frac{\sqrt{2}}{2} \leq c < 1$

master theorem بند سوم : $T(n) = \Theta(n^2\sqrt{n}) = \Theta(n^{\frac{5}{2}})$

$$\boldsymbol{T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n}$$

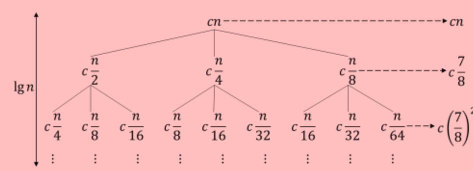
$T(n) \leq d\frac{n}{2}\lg\frac{n}{2} + d\frac{n}{4}\lg\frac{n}{4} + d\frac{n}{8}\lg\frac{n}{8} + cn$

$= d\frac{n}{2}(\lg n - \lg 2) + d\frac{n}{4}(\lg n - \lg 4) + d\frac{n}{8}(\lg n - \lg 8) + cn$

$= dn\lg n\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8}\right) - dn\left(\frac{1}{2} + \frac{2}{4} + \frac{3}{8}\right) + cn$

$= dn\lg n\left(\underbrace{\frac{7}{8}}_{<1}\right) - \left(\underbrace{\frac{11}{8}d - c}_{constant}\right)n$

$\leq dn\lg n \Rightarrow T(n) = O(n lg n)$



تعداد سطور: $\left(\frac{1}{2}\right)^k n = 1 \to k = \log_2 n = \lg n$

$Total = O(\lg n), T(n) \overset{?}{\leq} dn\ lg\ n$

$$\boldsymbol{T(n) = T(n-2) + \frac{1}{\lg n}}$$

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

$$= T(n-4) + \frac{1}{\lg(n-2)} + \frac{1}{\lg n} = \cdots = T(n-2k) + \sum_{i=0}^{k-1} \frac{1}{\lg(n-2i)}$$

$$= \cdots = c + \sum_{i=0}^{\left\lfloor \frac{n}{2} \right\rfloor} \frac{1}{\lg(n-2i)} \le c + \sum_{i=0}^{\left\lfloor \frac{n}{2} \right\rfloor} \frac{1}{b} = c + \frac{n}{2b}$$

$\Rightarrow T(n) = O(n)$

$\boldsymbol{T(n) = \sqrt{n}\, T(\sqrt{n}) + n}$

$$n = 2^k : T(2^k) = 2^{\frac{k}{2}} T\left(2^{\frac{k}{2}}\right) + 2^k \overset{\div 2^k}{\Longrightarrow} \frac{T(2^k)}{2^k} = \frac{2^{\frac{k}{2}} T\left(2^{\frac{k}{2}}\right)}{2^k} + 1 = \frac{T\left(2^{\frac{k}{2}}\right)}{2^{\frac{k}{2}}} + 1 \xRightarrow{\frac{T(2^k)}{2^k} = T'(k)} T'(k) = T'\left(\frac{k}{2}\right) + 1$$

$k^{\log_2 a} = k^{\log_2 1} = k^0 = 1$

$f(k) = 1 = \Theta(1)$

master theorem بند دوم : $T'(k) = \Theta\left(k^{\log_b a} \lg k\right) = \Theta(lgk)$

$$\xRightarrow{\frac{T(2^k)}{2^k} = T'(k)} T(2^k) = 2^k \Theta(\lg k) = \Theta(2^k \lg k) \overset{n=2^k}{\Longrightarrow} T(n) = \Theta(n\lg(\lg(n)))$$

۵) در یک صفحه مختصات تعدادی مستطیل روی محور x داریم. مستطیل B_i با (L_i, H_i, R_i) مشخص می‌شود که L_i و R_i مختصات چپ و راست و H_i ارتفاع را مشخص می‌کند. تابع **سایه** روی این مستطیل ها اعمال و یک لیست از نقاط که با مختصات طول و ارتفاع مشخص می‌شوند خروجی میدهد. (حداکثر 2n نقطه)

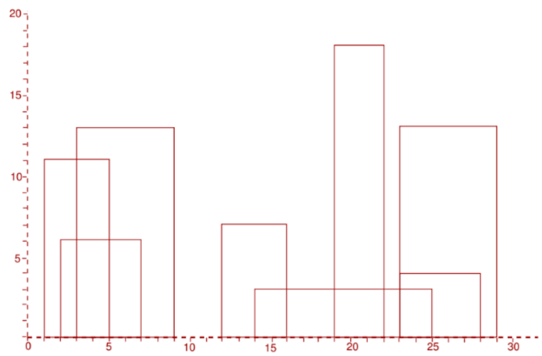
برای مثال **سایه** مستطیل‌های زیر:

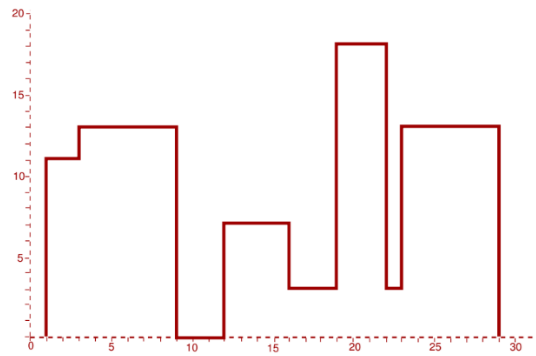
{(3,13,9),(1,11,5),(12,7,16),(14,3,25),(19,18,22),(2,6,7),(23,13,29),(23,4,28)}

برابر لیست زیر می‌شود.

{(1,11),(3,13),(9,0),(12,7),(16,3),(19,18),(22,3),(23,13),(29,0)}

امیرحسین علی بخشی – ۹۷۳۱۰۹۶

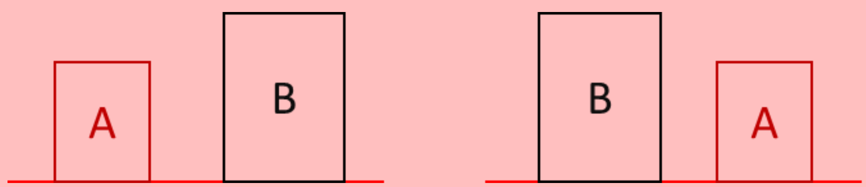




الف) الگوریتمی ارائه کنید که **سایه** A با طول n_1 را با **سایه** B با طول n_2 را به یک **سایه** S با اندازه $O(n_1+n_2)$ تبدیل کند. الگوریتم شما باید در زمان $O(n_1+n_2)$ اجرا شود.(۰.۷۵ نمره)

باید overlapping را بررسی کنیم؛ اگر $L_B > R_A$ و $R_A > L_B$ یا $L_B < R_A$ و $R_A < L_B$ آنگاه overlapping خواهیم داشت، با یان تفاوت که در حالت اول مستطیل A از B جلو تر است و در حالت دوم برعکس. درهمچوشانی های دو طرفه نیز شروع و پایان یکی از مستطیل ها بین نقاط شروع و پایان دیگر قرار دارد که مشخص است طول مستطیل داخلی باید کمتر باشد که در این حالت ارتفاع اهمیت پیدا میکند.

در صورت عدم وجود overlapping:



میتوان بدون مشکلی مستطیل ها را رسم کرد. تقاطعی نخواهیم داشت.

در صورت overlapping یکطرفه :

که در حالت چپ:

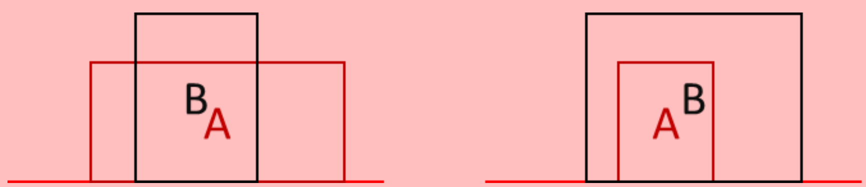
$$\begin{cases} n_1' = n_1 - (R_A - L_B) \\ n_2' = n_2 - (R_A - L_B) \\ \quad from\ L_A\ to\ R_B \end{cases}$$

و در حالت راست:

$$\begin{cases} n_2' = n_2 - (R_B - L_A) \\ n_1' = n_1 - (R_B - L_A) \\ \quad from\ L_B\ to\ R_A \end{cases}$$

بطور کلی از چپ تا نقطه نقاطع به طول جدید با ارتفاع مستطیل چپی رسم میکنیم و از تقاطع تا راست ترین نقطه هم همین کار را انجام میدهیم.

در صورت *overlapping* دو طرفه:

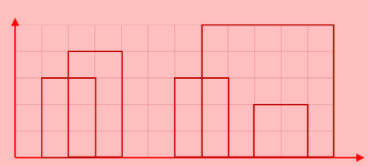


در حالت سمت راست تنها باید مستطیلی که مستطیل دیگر را احاطه کرده رسم کنیم.

در حالت چپ ۲ تقاطع خواهیم داشت که میتوان مشابه هم پوشانی یک طرفه نقاط تقاطع را محاسبه کرد و سایه را با یک مرحله بیشتر رسم کرد.

ب) الگوریتمی از O(nlogn) برای پیدا کردن **سایه** n مستطیل پیشنهاد کنید.(۰.۷۵ نمره)

داده ها را در یک وکتور ذخیره میکنیم. فرض میکنیم که تقطه سمت چپ بازه ارتفاع منفی و عدد سمت راست ارتفاع مثبت دارد.و نقاط را sort شده داخل این وکتور میریزیم. بعنوان نمونه:



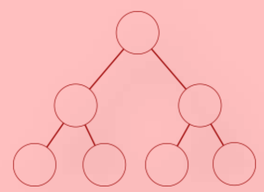
x	h
1	-3
2	-4
3	3
5	4
6	-3
7	-5
8	3
9	-2
10	2
11	5

اعداد را به ترتیب نگاه میکنیم و اگر سمت چپ بود آن را به داخل یک stack میفرستیم و اگر دیواره راست بود آن را خارج میکنیم. ارتفاع ماکسیمم را در متغیر max ذخیره میکنیم و در هر مرحله ارتفاع نقطه مورد نظر را با آن مقایسه میکنیم. اگر بیشتر بود آن را رسم میکنیم در غیر این صورت آن را ignore میکنیم. هنگام رسیدن به دیواره سمت راست اگر با برداشتن آن max تغییر کند نقطه را رسم میکنیم.

۶) ثابت کنید حداکثر $\lceil n/2^{h+1} \rceil$ عنصر با ارتفاع h در یک heap با n عنصر داریم.(۱ نمره)

امیرحسین علی بخشی – ۹۷۳۱۰۹۶

هر heap با n گره دارای $\lceil \frac{n}{2} \rceil$ برگ خواهد بود. اگر فرض کنیم درخت ما پر شده و تمامی برگ ها در سطر آخر قرار گرفته اند، میتوان گفت در $\text{height} = 0$ ، $\lceil \frac{n}{2} \rceil$ گره خواهیم داشت. حال میخواهیم یک سطر بالا بیاییم؛ برای پیدا کردن تعداد گره های این سطر میتوان تمامی گره ها (برگ ها) ی حالت قبل ($height = 0$) را از درخت حذف کرد. در این حالت $n - \lceil \frac{n}{2} \rceil = \lfloor \frac{n}{2} \rfloor$ گره در heap جدید باقی میماند که $\left\lceil \frac{\lfloor \frac{n}{2} \rfloor}{2} \right\rceil$ گره در پایین ترین سطر آن ($\text{height} = 1$ در درخت اولیه) وجود دارد که برگ های درخت جدید هستند. اگر برای $\text{height} = 2$ همین کار را بکنیم $= \left\lceil \frac{\left\lfloor \frac{\lfloor \frac{n}{2} \rfloor}{2} \right\rfloor}{2} \right\rceil$ $\left\lceil \frac{\lfloor \frac{n}{2} \rfloor}{2^2} \right\rceil$ گره خواهیم داشت. برای $\text{height} = h$ داریم:



$$\left\lceil \frac{\left\lfloor \begin{matrix} \lfloor \frac{n}{2} \rfloor \\ \vdots \end{matrix} \right\rfloor}{2} \right\rceil = \left\lceil \frac{\lfloor \frac{n}{2} \rfloor}{2^h} \right\rceil \le \left\lceil \frac{\frac{n}{2}}{2^h} \right\rceil = \left\lceil \frac{\boldsymbol{n}}{\boldsymbol{2^{h+1}}} \right\rceil$$

۷) الگوریتم HEAPSORT روی آرایه A با طول n که از قبل به طور صعودی مرتب شده در چه زمانی اجرا می‌شود؟ اگر به طور نزولی مرتب شده باشد چطور؟(هر کدام ۱ نمره)

صعودی:

در این حالت Build-Max-Heap دارای $O(n\lg n)$ خواهد بود. چون در اصل این heap در ابتدا یک Min-Heap میباشد و ما قصد داریم آن را به Max-Heap تبدیل کنیم. پس نیاز است که از گره $\lceil \frac{n}{2} \rceil$ به قبل، برای همه‌ی گره ها Max-Heapify را صدا بزنیم. هر حلقه که در Heap-Sort انجام میشود، $O(\lg n)$ هزینه دارد. بنابراین کل آن هزینه $O(n\lg n)$ را به همراه خواهد داشت.

نزولی:

در این حالت، Build-Max-Heap دارای $O(n)$ میباشد. (از ابتدا Max-Heap بوده است) اما در هر مرحله از تکرار حلقه که بزرگترین عنصر خارج میشود و آخرین برگ جایگزین آن میشود،کماکان نیاز داریم از Max-Heapify هایی استفاده کنیم که هر کدام $O(\lg n)$ دارند. در نتیجه در کل $O(n\lg n)$ خواهیم داشت

نتیجه:

آرایه ی ورودی در ابتدا چه به صورت نزولی مرتب شده باشد و چه به صورت صعودی، **الگوریتم Heap-Sort در هر دو حالت با مرتبه زمانی $\boxed{\mathbf{O(n\lg n)}}$ اجرا خواهد شد.**

امیرحسین علی بخشی – ۹۷۳۱۰۹۶

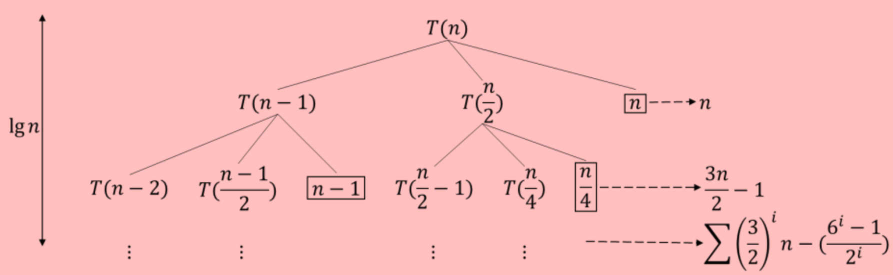
۸) یک الگوریتم از مرتبه O(nlgk) برای merge کردن k لیست مرتب شده به یک لیست مرتب شده ارائه دهید. n تعداد کل عناصر لیست ها است. (راهنمایی: از Minheap استفاده کنید)(۰.۵ نمره امتیازی)

ابتدا یک min-heap با سایز k میسازیم و عناصر شماره ۱ (دارای کمترین مقدار) هر کدام از k لیست را داخل آن قرار میدهیم. حال تابع **Heap-Extract-Min()** را فراخوانی میکنیم تا کوچکترین عنصر این heap، از آن خارج شود و سپس آن عنصر را داخل لیست نتیجه ی این merge کردن append میکنیم. حال، جای خالی heap را با عنصر بعدی همان لیستی پر میکنیم که در مرحله قبل توسط تابع Heap-Extract-Min() آن تا از heap حذف کردیم و به انتهای لیست نتیجه افزودیم.

ساخت heap به $O(k)$ نیاز دارد. هر یک از عناصر برای هر تابع Heap-Extract-Min() به $O(lg\ k)$ و برای جایگذاری عنصر جدید در heap نیز به $O(lg\ k)$ احتیاج خواهیم داشت.

$$O(k + n\ lg\ k) = O(n\ lg\ k)$$

۹) با استفاده از درخت بازگشت، کران بالای حدی مناسب برای T(n) = T(n-1)+T(n/2)+n بیابید با روش Substitution جواب خود را اثبات کنید.(۱ نمره امتیازی)



$$T(n) = \sum_{i=0}^{\lg n}\left(\frac{3}{2}\right)^i n - \left(\frac{6^i-1}{2^i}\right) \le \sum_{i=0}^{\lg n}\left(\frac{3}{2}\right)^i n$$

$$\begin{cases} \text{مثال رابطه بازگشتی}: T(n) = T(n-1) + T\left(\frac{n}{2}\right) + n \\ \text{حدس}: T(n) = O(n^{\lg n}) \end{cases} \Rightarrow \begin{cases} \text{فرض استقرا}: \begin{cases} T(n-1) \le c.(n-1)^{\lg(n-1)} \\ T\left(\frac{n}{2}\right) \le c.\left(\frac{n}{2}\right)^{\lg\left(\frac{n}{2}\right)} \end{cases} \\ \text{حکم استقرا}: T(n) \le c.n^{\lg n} \end{cases}$$

$$T(n) \le c(n-1)^{\lg(n-1)} + c\left(\frac{n}{2}\right)^{\lg\left(\frac{n}{2}\right)} + n$$

$$\le cn^{\lg(n-1)} + cn^{\lg n - \lg 2} + n$$

$$\le cn^{\lg(n-1)} + cn^{\lg n} + n \le c'n^{\lg n} \Rightarrow T(n) = O(n^{\lg n})$$

بخش عملی:

۱ ـ یکی از موضوعات مهم در الگوریتم های هندسی پیدا کردن نقاط نزدیک به هم می‌باشد. در این سوال هدف پیدا کردن کمترین فاصله بین n نقطه در فضای دو بعدی با استفاده از الگوریتم divide and conquer می‌باشد. به بیانی دیگر ورودی شامل n نقطه با مختصات x و y است و در خروجی کوتاه ترین فاصله ی اقلیدسی بین این نقاط را در خروجی چاپ کنید.

الف) الگوریتم خود در این خصوص را نوشته و تحلیل زمانی خود را بیان کنید.

ب) الگوریتم خود را با استفاده از یکی از زبان های c،c++،java و python پیاده سازی کنید و در کلاس کوئرا که آدرس و نحوه‌ی عضویت در آن در سایت درس قابل مشاهده است، بارگذاری کنید. (تعدادی تست نمونه در کوئرا قرار داده شده است.)

پ) یکی از روش های دیگر حل این مسئله استفاده از روش brute force است. به این معنی که برای یافتن نزدیک ترین جفت فاصله ی همه ی جفت ها از یکدیگر محاسبه کرده و نزدیک ترین آن ها را چاپ کنید. این الگوریتم را نیز پیاده سازی کرده و با استفاده از داده های تست واقعی که در کنار فایل تمرین وجود دارد، از نظر موارد زیر دو الگوریتم را با یکدیگر مقایسه کنید:

۱- خروجی

۲- مدت زمان اجرای هر الگوریتم (برای درک بهتر تفاوت زمان اجرای هر الگوریتم، هر دو الگوریتم را با یک زبان پیاده سازی کنید.)

فایل این قسمت را نیز که شامل پیاده سازی دو الگوریتم و قسمت تحلیلی مقایسه‌ی این دو الگوریتم است را به صورت یک فایل زیپ در کوئرا بارگذاری کنید. (فایل قسمت تحلیل باید به صورت pdf باشد.)

- مهلت ارسال تمرین ساعت ۲۳:۵۵ روز **دوشنبه ۱۹** آبان‌ماه می‌باشد.
- سوالات **بخش تئوری** را می‌توانید از طریق ایمیل از تدریسیاران بپرسید.
  - m.masumi@aut.ac.ir
- سوالات مربوط به **بخش عملی** را نیز از طریق کوئرا یا از طریق آدرس زیر مطرح نمایید.
  - mr.mim1377@gmail.com
- ارائه پاسخ تمرین به دو روش ممکن است:
  1) تایپ داخل همین فایل و ارائه فایل Pdf
  2) چاپ تمرین و پاسخ دهی به صورت دستنویس خوانا
- ارائه تمرین به روش اول شامل ۱۰٪ نمره امتیازی می‌گردد.
- فایل پاسخ تمرین را تنها با قالب **HW2-9531747.pdf** در مودل بارگزاری کنید.
- فایل زیپ در مودل ارسال **نکنید**.