מבני נתונים – תרגיל 1

מאור אסייג 318550746

רפאל שטרית 204654891

המחלקה להנדסת חשמל ומחשבים, התכנית להנדסת מחשבים

מבני נתונים 202.1.1011

תשובות

1.

$$f_4 \le f_1 \le f_9 \le f_{11} \le f_2 \le f_{10} \le f_{12} \le f_3 \le f_5 \le f_7 \le f_8 \le f_6$$

את ההוכחה חילקנו להוכחה חלקית בכל מקטע של אי השוויון ומכלל המעבר זה מוכיח את כל האי שוויון הגדול. ברוב הסעיפים הסתמכנו על הידע שצברנו בחדוו"א 1 ומכללי קצב גדילה ושאיפה של מנת פונקציות בשאיפת גבולות אינסופיים.

תחילה נוכיח כי $f_8 \le f_6$

$$\lim_{x\to\infty} \frac{3^{2^n}}{2^{3^n}} = 0$$

$f_7 \le f_8$

$$\lim_{x\to\infty} \frac{n^n}{3^{2^n}} = 0$$

מכיוון האספוננט של 3 הוא בעצמו 2^n אז קצב הגדילה שלו גדול בהרבה מקצב הגדלה של n^n

$f_5 \le f_7$

$$\lim_{x\to\infty} \frac{3^n}{n^n} = 0$$

כאן אני רואים שכאשר $n > 3$ המכנה גדול מהמונה ולכן הגבול שלו הוא 0

$f_3 \le f_5$

$$\lim_{x\to\infty} \frac{2^{\sqrt{n}}}{3^n} = 0$$

ברור לנו שגם אם היה 2^n הגבול היה הולך לאפס ולכן גבול זה הולך לאפס גם.

$f_{12} \le f_3$

$$\lim_{x\to\infty} \frac{n^2 + \log n + n}{2^{\sqrt{n}}} = \frac{lop}{lop} = 0$$

$f_{10} \le f_{12}$

$$\lim_{x\to\infty} \frac{\log 2^n * n^2}{n^2 + \log n + n} = \frac{n + 2\log n}{n^2 + \log n + n} = 0$$

מכיוון של $n^2 > \log n$ אז הפונקציה למטה גדלה הרבה יותר מהר מהפונקציה למעלה

$f_2 \le f_{10}$

$$\lim_{x\to\infty}\frac{2^{\log_{\sqrt{2}} n}}{\log 2^n * n^2} = \frac{\sqrt{n}}{\log 2^n * n^2} = 0$$

מכיוון של $n > \sqrt{n}$ אז הגדילה של המכנה גדולה מהגדילה של המונה

$f_{11} \leq f_2$

$$\lim_{x\to\infty}\frac{\log n^{10}}{2^{\log_{\sqrt{2}} n}} = \frac{\log n^{10}}{\sqrt{n}} = 0$$

מלופיטל נמצא שהגבול הוא אפס

$f_9 \leq f_{11}$

$$\lim_{x\to\infty}\frac{\log\sqrt{n}}{\log n^{10}} = \frac{0.5}{10}$$

באי שוויון הזה נוכיח שהם שניהם שייכים ל $O(\log n)$

$\log n^{10} = 10\log n \leq c_1 \log n \mid c_1 \geq 10$

$\log\sqrt{n} = \frac{1}{2}\log n \leq c_2 \log n \mid c_2 \geq \frac{1}{2}$

כלומר שניהם שייכים ל$\mathrm{O}(\log n)$

$f_1 \leq f_9$

$$\lim_{x\to\infty}\frac{2017}{\log\sqrt{n}} = 0$$

ברור שהגבול הזה הוא 0 זה קבוע חלקי משהו ששואף לאינסוף

$f_4 \leq f_1$

$$\lim_{x\to\infty}\frac{\frac{1}{n}}{2017} = 0$$

2. הוכיחו או הפריכו את הטענות הבאות :

א. קיימת פונקציה $f(n)$ כך ש- $f(n-k) \neq \Theta(f(n))$. כאשר $k \geq 1$ הוא קבוע חיובי.

נכון

$for\ example\ given\ f(n) = n!,$
$assuming\ that\ n! \in \theta\big((n-k)!\big)\ which\ yelid\ for\ every$
$n_0 < n,\ \ c * n! < (n-k)!\ \ for\ c > 0\ \ then\ we\ get$
$c < \dfrac{1}{n(n-1)..(n-k+1)}\ and\ for\ big\ enough\ n\ its\ yelid$
$that\ c < 0, so\ the\ assumption\ is\ wrong \rightarrow claim\ is\ truth$ ■

ב. קיימת פונקציה $f(n)$ כך ש- $\big(f(n)\big)^2 = O(f(n))$ וגם $f(n) = \Omega(\text{logn})$.

לא נכון

$assuming\ there\ is\ 0 < c_1, 0 < n_0\ which\ yelid\ for\ every$
$n_0 < n \rightarrow\ f^2(n) \leq c_1 f(n)\ for\ c > 0\ \ then\ we\ get\ f(n) \leq c_1$
$according\ to\ the\ figure\ in\ the\ question\ we\ get$
$\log(n) \leq c_2 f(n) \leq c_2 c_1\ , there\ is\ no\ such\ constant\ c_1$
$so\ the\ assumption\ is\ wrong.$ ■

ג. יהיו $f(n), g(n)$ פונקציות כך ש $f(n), g(n) \geq 1$ לכל n. אזי $f(n)+g(n) = O(f(n)\cdot g(n))$.

נכון

$according\ to\ the\ figure\ in\ the\ question\ we\ get$
$f(n) + g(n) \leq c_1(f(n)g(n))$
$\dfrac{1}{f(n)} + \dfrac{1}{g(n)} \leq c_1 \rightarrow consider \dfrac{1}{f(n)} < 1, \dfrac{1}{g(n)} < 1$
$we\ can\ choose\ c_1 = 3\ which\ yelid\ this\ for\ each\ n.$ ■

3. מצאו חסם עליון וחסם תחתון אסימפטוטיים עבור $T(n)$ בכל אחת מנוסחאות הנסיגה שלהלן. הניחו כי $T(n)$ קבועה עבור n קבוע. מצאו חסמים הדוקים ככל שתוכלו ונמקו את תשובותיכם.

א. $T(n) = T(\sqrt{n}) + 1$

Iteration method

$$T(2) = 1, T(n) = T\left(n^{\frac{1}{2}}\right) + 1 = T\left(n^{\frac{1}{4}}\right) + 1 + 1 = \cdots =$$

$$T\left(n^{\frac{1}{2^i}}\right) + i, \quad for\ i = loglogn$$

$$= T\left(n^{\frac{1}{2^{loglog(n)}}}\right) + loglog(n) = T(2) + loglog(n)$$

$$= loglog(n), from\ class \rightarrow \mathrm{T(n)=\theta(log(log(n)))}.$$

$$we\ get\ i\ by: [n^{\frac{1}{2^i}} = 2 \rightarrow \quad \log n^{\frac{1}{2^i}} = \log 2 = 1 \rightarrow$$

$$\left(\frac{1}{2}\right)^i logn = 1 \rightarrow \quad 2^i = \log n \rightarrow \quad i = loglog(n)].$$

ב. $T(n) = 5T\left(\frac{n}{2}\right) + n^3 \log n$

Master (type 3)method

$c = \log_b a = \log 5, \varepsilon = 1\ then:$
$(1)\ f(n) = n^3 logn = \Omega(n^{log5+1})$
$proof: 0 \le n^{log5} n \le c_1 n^3 \log(n) \rightarrow n^{log5} \le c_1 n^2 \log(n)$
$and\ for\ c_1 = 1, log5 < 2\ then\ its\ true.$

$$(2)\ \frac{5}{8} f\left(\frac{n}{2}\right) \le df(n) \rightarrow \frac{5}{8} n^3 \log\left(\frac{n}{2}\right) = \frac{5}{8} n^3 (logn - log2) =$$
$$\frac{5}{8} n^3 (logn - 1) \le \frac{5}{8} n^3 logn \rightarrow \ for\ d = \frac{5}{8} < 1$$

$thus\ T(n) = \theta(n^3 logn).$

ג. $T(n) = T(cn) + T((1-c)n) + 1$, $0 < c < 1$.

Iteration method, *assuming* $T(1) = 1$

$given\ 0 < c < 1\ we\ mark\ a = \max(c, c-1)\ , clearly\ a \geq \frac{1}{2}$

$T(n) = T(cn) + T\big((1-c)n\big) + 1 \leq 2T(an) + 1 = 2[T(acn)$
$+T(a(1-c)n) + 1] \leq 2[T(a^2n) + 1] + 1 \leq \cdots \leq$

$\leq 2^i\big[T\big(a^i n\big)\big] + \sum_{k=0}^{k=i-1} 2^k.$

$For\ a^i n = 1 \rightarrow i = \frac{1}{(\log_a n)} = \log_a \frac{1}{n}$

$T(n) \leq 2^{\log_a \frac{1}{n}}\left[T\left(a^{\log_a \frac{1}{n}} n\right)\right] + \sum_{k=0}^{k=\log_a \frac{1}{n} - 1} 2^k$

$= \left(\frac{1}{n}\right)^{\log_a 2} + \left(\frac{1}{n}\right)^{\log_a 2} - 1 = 2n^{-\log_a 2} - 1 = 2n^{\log_a \frac{1}{2}} - 1 \leq^* 2n - 1$

$*\ given\ that\ a \geq \frac{1}{2}\ then \log_a \frac{1}{2} \leq 1$

$\rightarrow T(n) = \mathrm{O}(\mathrm{n}).$

$given\ 0 < c < 1\ we\ mark\ b = \min(c, c-1)\ , clearly\ b \leq \frac{1}{2}$

$T(n) = T(cn) + T\big((1-c)n\big) + 1 \geq 2\mathrm{T}(\mathrm{bn}) \geq \cdots \geq 2^i\big[T\big(b^i n\big)\big]$

$For\ b^i n = 1 \rightarrow i = \log_b\left(\frac{1}{n}\right) = -\log_b n$

$then\ T(n) \geq 2^{-\log_b n} T(1) = \left(\frac{1}{n}\right)^{\log_b 2} = n^{-\log_b 2} \geq^* n$

$*\ given\ that\ b \leq \frac{1}{2}\ then \log_b 2 \leq -1$

$\rightarrow T(n) = \Omega(\mathrm{n}).$

$Thus\ altogether\ \mathrm{T}(n) = \theta(n).$

$$T(n) = T\left(\frac{3n}{5}\right) + 2T\left(\frac{n}{5}\right) + n$$.ד

Iterative tree method, $assuming\ T(1) = 1$

$for\ the\ left\ side\ \left(of\ T\left(\frac{3n}{5}\right)\right)\ we\ can\ calculate\ the\ hight\ mark\ by\ i:$

$\left(\frac{3}{5}\right)^i n = 1 \to \log_{\frac{5}{3}} n = i$

$for\ the\ right\ side\ \left(of\ 2T\left(\frac{n}{5}\right)\right)\ we\ can\ calculate\ the\ hight\ mark\ by\ k:$

$\left(\frac{1}{5}\right)^k n = 1 \to \log_5 n = k$

$Thus, n\log_{\frac{5}{3}} n \le T(n) \le nlog_5 n \to since\ log_{\frac{5}{3}} n = O(\log_5 n)$

$= \mathrm{O}(\log_2 n), we\ get:$

$\mathrm{T(n)} = \theta(\mathrm{nlogn})$

$$T(n) = 2T(n-1) + 1$$.ה

Iteration method, $assuming\ T(0) = 0$

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

$$2^i T(n-n) + \sum_{k=0}^{k=n-1} 2^k = \frac{1(2^{n-1}+1)}{2-1} = \frac{1}{2}2^n - 1 =^* \theta(2^n)$$

$*\ from\ class$

4. סיבוכיות זמן ריצה :

א.

a) <u>**function**</u> **BubbleSort(A[1..n])**

```
for i←1 to n -1
    for j← n downto i+1
        if A[j-1] > A[j]
            temp ←A[j-1]
            A[j-1] ← A[j]
            A[j] ← temp
```

$i \ is \ runnig \ n-1 \ times, j \ is \ runnig \ backward's \ from \ n \ to \ i+1;$
$worst \ case \ yeild: T(n) = 3(n-2+n-3+\cdots+n-n+1+0) =$

$$= \frac{3(n-1)[2(n-2)+(n-2)(-1)]}{2} = \frac{3}{2}(n-1)(n-2) =^{*} \theta(n^2)$$

$* \ from \ class.$

ב.

b) <u>**function**</u> **exp(base , power)**

```
if (power = 0)
    return 1
else if (power = 1)
    return base
else
    return base · exp(base,power-1)
```

$The \ recursive \ function \ is \ called \ power \ times,$
$thus \ for \ T(base = k, power = n) \ we \ get \ T(k,n) = \theta(n)$

ג.

c) <u>**function**</u> **exp2(base , power)**

```
if (power = 0)
    return 1
else if (power = 1)
    return base
else if (mod(power, 2) = 0)
    tmp ← exp2(base, power/2)
    return tmp · tmp
else
    return  base · exp2(base,power-1)
```

$assuming\ T(1) = \theta(1)$
$In\ case\ that\ the\ power\ from\ the\ first\ place\ is\ \%2$
$= 0\ then\ the\ recursive\ function\ will\ keep\ the\ condition$
$until\ we\ hit\ the\ stop\ condition's, thus$
$T(base = k, power = n) = \log_2 n$

$$for\ the\ other\ cases : T(n) = T\left(\frac{n-1}{2}\right) + \theta(1)$$

$$by\ \text{Iteration method} : T(n) = T\left(\frac{n}{4} - \frac{1}{4} - \frac{1}{2}\right) + 2\theta(1) = \cdots =$$

$$T\left(\frac{n}{2^i} - \sum_{k=1}^{k=i} \frac{1}{2^k}\right) + i\theta(1) = \mathrm{T}\left(\frac{n}{2^i} + \frac{1}{2^i} - 1\right) + i\theta(1)$$

$$we\ can\ find\ i\ by : \frac{n}{2^i} + \frac{1}{2^i} - 1 = 1 \rightarrow 2 * 2^i = n + 1 \rightarrow$$

$$i = \log\left(\frac{n+1}{2}\right) = \log(n+1) - log2 = \log(n+1) - log2$$

$altogether\ T(n) = T(1) + [\log(n+1) - log2]\theta(1) = \theta(\log_2 n).$

.5

א. הציעו אלגוריתם המקבל שני פרמטרים: מערך ממוין A וערך x, ומחזיר אינדקס של x במערך. אם x לא נמצא במערך A יש להחזיר ‎-1.

זמן ריצה הדרוש הינו $O(\log d)$, כאשר d הוא מספר האיברים מופיעים לפני איבר x במערך המבוקש במידה ו-x (היה) קיים במערך A (ראו דוגמה בסוף השאלה).

$Find\ (Arr[]\ A, int\ x)$
$\quad n = size(A)$
$\quad k = 1$
$\quad while\ (Arr[k] < x\ \&\&\ k < n)$
$\qquad\quad k = 2k$
$\quad if\ (k < n)$
$\qquad indexOfx = BinarySearch(A, x, k/2, k)$
$\quad else$
$\qquad indexOfx = BinarySearch(A, x, 1, n)$
$\quad return\ indexOfx$

$BinarySearch\ (A, x, low, high)$
$\qquad while(low \le high)$
$\qquad\qquad mid = (low + high)/2$
$\qquad\qquad if(A[mid] == x)$
$\qquad\qquad\quad return\ mid$
$\qquad\qquad else\ if\ (A[mid] < x)$
$\qquad\qquad\qquad low = mid + 1$
$\qquad\qquad else$
$\qquad\qquad\qquad high = mid - 1$
$\qquad if\ (low == high)$
$\qquad\quad return\ (-1)$
$\quad return\ BinarySearch\ (A, x, low, high)$

$worst\ case:$
$Find\ Function: x\ in\ the\ last (= d) index\ in\ a\ sorted\ array\ (the\ size$
$of\ the\ array\ is\ n, assuming\ n \ge d)\ while\ condition\ is$
$still\ holding\ until\ we\ hit\ k \ge d\ with\ exponential\ growing\ of\ k,$
$therefor\ while\ occures\ \log_2 d\ times,$
$\rightarrow Find = T(\ indexOfx = d, sizeOfArray = n\ ) = \theta(\log_2 d)$

$BinarySearch\ Function\colon we\ call\ this\ function\ with\ array$
$in\ size\ of\ \frac{k}{2}\ index's, assuming\ n\%2 = 0, we\ get\ the\ size\ of\ \frac{d}{2},$
$\rightarrow BinarySearch\ as\ shown\ at\ class\ is\ T(n) = \theta(\log_2 n),$
$Thus\ for\ our\ case\ we\ get \rightarrow BinarySearch = T\left(\frac{d}{2}\right) = \theta(\log_2 d)$

$altoghter$ the $timerun\ for\ our\ alogorithem\ \rightarrow T(n) = \theta(\log_2 d).$

ב.

$Algorithem:$

$A\ array\ size\ is\ n, A = \{a_1, a_2..M_A..a_n\}\ when\ M_A\ is\ the\ median\ of\ A.$
$B\ array\ size\ is\ m, B = \{b_1, b_2..M_B..b_m\}\ when\ M_B\ is\ the\ median\ of\ B.$
$if\ M_A = M_B \rightarrow then\ the\ median\ is\ one\ of\ them (= M_A = M_B).$

$else$
$\rightarrow\ we\ call\ the\ recursive\ function\ Find\ with\ (A, B, 1, size\ of\ A, 1, size\ of\ B):$

- $stop\ condition: if\ we\ left\ with\ a\ combined\ size\ of\ 4\ or\ less\ we\ sort$ $thus\ numbers\ and\ return\ the\ one\ that\ his\ index\ fit\ with\ the\ mathmetical$ $conditon\ we\ found\ by\ the\ median\ defintion.$
- $due\ to\ the\ math\ the\ median\ of\ a\ single\ Arr\ (A\ or\ B)\ calculate\ by\ the\ formula$ $\frac{high+low1-1}{2} + k.$
- $if\ we\ deal\ with\ even\ (n + m) than\ the\ median\ of\ the\ merged\ arr\ is\ the\ index\ \frac{n+m}{2} of\ the\ merged\ arr.$
- $if\ we\ deal\ with\ odd\ (n + m) than\ the\ median\ of\ the\ merged\ arr\ is\ the\ index\ \left(\frac{n+m}{2} + 1\right) of\ the\ merged\ arr.$
- $each\ time\ we\ cut\ the\ the\ size\ of\ the\ leftover\ array\ in\ half.$
- $we\ found\ the\ the\ recursive\ change\ by:$
  $if\ the\ current\ M_A \geq M_B\ then\ the\ Mid\ is\ in\ x\ when:$
  $in\ the\ merged\ arr\ \{b_1, b_2..M_{b-1}, \{elements\ from\ A\}, M_B, \{..x..\}, M_A, \{elements\ from\ A, B\}\}$
  $if\ the\ current\ M_A < M_B\ then\ there\ are\ a\ symmetery.$

בודק נכבד שלום רב, נהיה כנים עמך – זו אחת הבעיות הכי מאתגרות שאני ושותף שלי נתקלנו בהן. לאחר שהבנו את הרעיון הכללי של האלגוריתם, ניסינו המון גרסאות אפשריות של לקיחת חציונים חדשים ותנאי עצירה, עד שהגענו לנוסחה המתמטית שהצגנו. השתדלנו להשאיר את הקוד נקי ביותר, ללא אופטימיזציה של מקרי קיצון – על מנת לשמור על טוהר הרעיון.

כמו כן אנו מודעים לכך שאין צורך להקצות זיכרון חדש למערך בגודל מקסימלי 4 (תנאי העצירה) – ניתן לכתוב תנאים מסורבלים על פי האינדקס שנקבל Result ומציאתו המידית של החציון, תוך שימוש בתכונות המערכים הממוינים.

צירפנו דוגמא להרצת האלגוריתם הזה בסוף הקוד (בנוסף בדקנו כל ווראציה אפשרית של מערכים [זוגי+אי זוגי וכו']).

Pseudo code : $= \textit{is the counter} \rightarrow$ הפוך סימון

$global\ parameters - k, mid;$
$Median\ (Arr[\ \ ]\ A, Arr[\ \ ]\ B)\{$
  $n = size(A)$
  $m = size(B)$
  $k = (m + n)\%2$
  $Mid = \left\lceil \frac{n+m}{2} \right\rceil$
  $if\ (A[\left\lceil \frac{n}{2} \right\rceil] == B[\left\lceil \frac{m}{2} \right\rceil])$
    $return\ A[\left\lceil \frac{n}{2} \right\rceil];$
  $return\ Find\ (A, B, 1, n, 1, m);$
$\}$

$Find\ (Arr[\ \ ]\ A, Arr[\ \ ]\ B, int\ low1, int\ high1, int\ low2, int\ high2)\ \{$
  $size1 = high1 - low1 + 1$
  $size2 = high2 - low2 + 1$
  $Mid1 = \frac{high1 + low1 - 1}{2} + k$
  $Mid2 = \frac{high2 + low2 - 1}{2} + k$
  $if\ (size1 + size2 \leq 4)\{$
    $New = new\ Arr[size1 + size2]$
    $Result = Mid - (low1 + low2) + 2$
    $for\ (i = 1; i \leq size1 + size2; i++)\{$
      $if\ ((A\ [low1] \leq B[low2\ ]\ \&\&low1 \leq high1)||low2 \geq high2)$
        $New[i] = A[low1]$
        $low++$
      $else$
        $New[i] = B[low2]$
        $low2++$
      $\}$
    $return\ New[Result]$
  $\}$

  $else\ if\ (A\ [Mid1] \leq B[Mid2])\{$
    $return\ Find(A, B, Mid1, high1, low1, Mid2)$
  $\}$
  $else\ if\ (A\ [Mid1] > B[Mid2])\{$
    $return\ Find(A, B, low1, Mid1, Mid2, high2)$
  $\}$
$\}$

$Run\ time\ Analysis\ (worst\ case)$:

$every\ recursive\ call\ we\ decrease\ the\ size1$
$+size\ 2\ (merged\ arrays)\ in\ most\ of\ the$
$cases\ in\ half, if\ we\ get\ into\ the\ stop\ condition\ we\ get\ \theta(1)$
$to\ sort\ 4\ numbers\ and\ return$
$an\ exacat\ indes\ in\ the\ leftover\ merged\ array(the\ only\ new\ array\ we\ make).$

$$T(n+m) = c + T\left(\frac{n+m}{2}\right) = \cdots = c + T\left(\frac{n+m}{2^i}\right)$$
$$= cc_2 \log(n+m) + T(n+m \le 4)$$
$$= \theta(1) + \theta(\log(n+m)) = \theta(\log(n+m)).$$

- $if\ n+m \le 4\ then\ anyway\ T(n+m \le 4) = \theta(\log(n+m)) = c\log(n+m)).$

$Example$

A

1	5	6	10	13	15	20

B

2	3	7	11	17	25

A+B

1	2	3	5	6	7	10	11	13	15	17	20	25

$The\ median\ is\ in\ index\ \left\lceil\frac{size(A+B)}{2}\right\rceil = 7, then\ the\ median\ is\ 10.$

$starting\ the\ algorithem\ we\ get\ that\ size(A+B) is\ odd, then\ k=1.$

$for\ now\ on, the\ median\ of\ each\ array\ we\ get\ is\ choosen\ by\ the\ formula$

$$int\ Mid1 = \frac{high1 + low1 - 1}{2} + k.$$

$for\ now. Mid1\ is\ the\ index\ of\ the\ median\ in\ the\ left\ arr\ A\ etc,$

1. $Mid1 = 4, Mid2 = 4 \rightarrow A[Mid1] < B[Mid2]$
   $we\ left\ with\ A\{10,13,15,20\}\ \ B\{2,3,7,11\}$
2. $Mid1 = 3, Mid2 = 3 \rightarrow A[Mid1] \ge B[Mid2]$
   $we\ left\ with\ A\{10,13,15\}\ B\{7,11\}$
3. $Mid1 = 2, Mid2 = 2 \rightarrow A[Mid1] \ge B[Mid2]$
   $we\ left\ with\ A\{10,13\}\ \ B\{11\}$
4. $we\ left\ with\ size\ of\ A+B = 3 \le$
   $4\ then\ we\ calculte\ the\ merge\ arr\ to\ New[\quad] =$
   $\{10,11,13\}. in\ the\ recursive\ function\ we\ also\ get\ low1$
   $= inedx\ of\ 10\ in\ A = 4, low2 = index\ of\ 11\ in\ B = 4.$
   $the\ median\ is\ the\ number\ in\ the\ index:$
   $Result = Mid - (low1 + low2) + 2 = 7 - (4+4) + 2 = 1$
   $New[Result] = the\ median\ of\ the\ orginal\ sorted\ merged\ array!$

Until next time, thank you.