29/10 נ

בס"ד

①

# אלגוריתמים 1 - צנה שפירא

90% מבחן
10% מטלות

נושאים:
הפרד ומשול
אלגוריתמים חמדנים
תכנות דינמי
אלגוריתמי גרפים
פולינומים

## נושא 1 - הפרד ומשול

האלגוריתמים שנלמד בפרק זה הם אלגוריתמים שלוקחים בעיה גדולה ומפרקים אותה לחלקים, ומתמודדים עם כל חלק בנפרד, כך שכעת הבעיה יותר קלה לפיתרון. לבסוף מחברים את כל הפתרונות יחד ובכך פותרים את הבעיה הגדולה. לרוב אלגוריתמים אלו ישתמשו ברקורסיה אך לא בהכרח. נציג מספר אלגוריתמים בשיטת הפרד ומשול וננתח אותם:

### 1) מציאת מינימום ומקסימום במערך

נרצה לכתוב פונקציה שמקבלת מערך ומחזירה זוג סדור (a,b), כאשר a הוא הערך של האיבר המינימלי במערך, ו-b הוא הערך של האיבר המקסימלי. הדרך הפשוטה אך לא יעילה היא לעבור פעמיים על כל המערך ולמצוא ערכים אלו. אולם בדרך רקורסיבית בעזרת של הפרד ומשול נוכל לייעל זאת.

א) דרך רקורסיבית: נחלק את המערך כל פעם לשני חלקים שווים, כאשר תנאי העצירה הוא שבמערך יש איבר אחד או שניים (נפרט בהמשך). נפעיל את הפונקציה בדרך רקורסיבית על כל אחד מהחלקים, כאשר אנו מניחים שאנו יודעים למצוא את האיברים המינימלי והמקסימלי בכל אחד מהחלקים.

תנאי העצירה - אם יש איבר 1 נחזיר אותו פעמיים, ואם יש 2 איברים נחזיר הקטן ואז הגדול.

ב) פסאודו קוד:

```
minMax(i,j) {
    if (j=i)
        return(A[i], A[i])
    else if (j=i+1)
        if(A[i] < A[j])   return(A[i], A[j])
        else   return(A[j], A[i])
    else
        k=(i+j)/2
        (m1,M1)= minMax(i,k)
        (m2,M2)= minMax(k+1,j)
        return(min(m1,m2), max(M1,M2))
}
```

29/10 בס"ד

ג) <u>חישוב זמן ריצה:</u>

(2)
$$T(n)=\begin{cases}0 & n=1\\ 1 & n=2\\ \underset{\text{2 רקורסיות}}{2T\left(\frac{n}{2}\right)}+\underset{\text{השוואות בסוף}}{2} & n>2\end{cases}$$

$$T(n)=2T\left(\frac{n}{2}\right)+2=2\left(2T\left(\frac{n}{4}\right)+2\right)+2=2^2T\left(\frac{n}{2^2}\right)+2^2+2$$
$$=2^2\left(2T\left(\frac{n}{2^3}\right)+2\right)+2^2+2=2^3T\left(\frac{n}{2^3}\right)+2^3+2^2+2$$

לאחר k שלבים: $=2^kT\left(\frac{n}{2^k}\right)+2^k+\dots+2=2^kT\left(\frac{n}{2^k}\right)+2^{k+1}-2$

במקרה הגרוע לאחר k שלבים ישארו 2 איברים, לכן $\frac{n}{2^k}=2$. ומכאן $n=2^{k+1}$. נציב זאת במשוואה.

$$T(n)=\frac{n}{2}\underset{1}{T(2)}+n-2=\frac{n}{2}+n-2=\boxed{\frac{3n}{2}-2}\Rightarrow \text{יותר יעיל}$$

## 2) הכפלה בוליאנית

נרצה להכפיל שני מספרים בוליאניים $y,x$ כאשר כל אחד באורך $n$. לפי שיטת כפל ארוך נצטרך להכפיל כל ספרה של $x$-ב כל ספרה של $y$-ב, כך שבסך הכל נבצע $n^2$ פעולות כפל ו-$n^2$ פעולות חיבור $O(n^2)$-, כדי לקבל מספר באורך $2n$. ננסה לייעל זאת בשיטת הפרד ומשול.

א) <u>פתרון 1:</u> נפצל את $x$ לשני חלקים שווים $x=x_1,x_2$, ונרשום את $x$ בצורה הבאה: $x=x_1\cdot 2^{n/2}+x_2$, וכך נעשה גם ל-$y$. במספרים עשרוניים הדבר דומה לשקילה של 2745 ל-$27\cdot 10^2+45=2745$.

הכפלה ב-$2^k$ מתבצעת ב-$O(1)$ מכיוון ששקולה לפעולה shift שמאלה באורך $k$.

$$\begin{cases}x=x_1\cdot 2^{n/2}+x_2\\ y=y_1\cdot 2^{n/2}+y_2\end{cases}$$

כעת נכפיל בין $x$ ל-$y$ המיוצגים בצורה חדשה. $xy=2^n\cdot x_1y_1+2^{n/2}(x_1y_2+x_2y_1)+x_2y_2$.

נקבל חיבור של 4 איברים שכל איבר הוא מכפלה בין שני מספרים עם $\frac{n}{2}$ ספרות.

ב) <u>חישוב זמן ריצה:</u> נסמן ב-$T(n)$ מספר פעולות כפל בהכפלת שני מספרים באורך $n$.

נתמקד בספירת פעולות הכפל ולא בחיבור כי מבחינת סיבוכיות הוא יותר קריטי, וגם בוודאי מספר פעולות החיבור הוא קבוע, לכן נסמנו $cn$. תנאי העצירה הוא $n=1$ שאז יש לבצע רק הכפלה אחת.

$$T(n)=\begin{cases}1 & n=1\\ 4T\left(\frac{n}{2}\right)+cn & n>1\end{cases}$$

$$T(n)=4T\left(\frac{n}{2}\right)+cn=4\left(4T\left(\frac{n}{4}\right)+c\cdot\frac{n}{2}\right)+cn=4^2T\left(\frac{n}{2^2}\right)+2cn+cn$$
$$=4^2\left(4T\left(\frac{n}{2^2}\right)+c\cdot\frac{n}{2^3}\right)+2cn+cn=4^3\cdot T\left(\frac{n}{2^3}\right)+2^2cn+2cn+cn$$

לאחר k שלבים: $=4^kT\left(\frac{n}{2^k}\right)+cn\left(2^{k-1}+\dots+2^1+2^0\right)=4^k\left(\frac{n}{2^k}\right)+cn\left(2^k-1\right)$

נעצור כאשר $\frac{n}{2^k}=1$. ומכאן $k=\log_2 n$.

$$T(n)=\underset{2^{\log_2 n^2}}{4^{\log_2 n}}\cdot\underset{1}{T(1)}+cn\left(\underset{n}{2^{\log_2 n}}-1\right)=n^2+cn^2-cn=O(n^2)\Rightarrow$$

<u>מסקנה:</u> לא שיפרנו. ננסה דרך אחרת.

③

$$\begin{cases} A = x_1\cdot y_1 \\ B = x_2\cdot y_2 \\ C = (x_1+x_2)(y_1+y_2) \end{cases}$$

ג) <u>פתרון 2</u>: ננסה לשפר את המשוואה של $xy$ מפתרון הקודם ע"י שימוש בסימונים:
נשים לב כי עבור כל אחד מהסימונים $A,B,C$ דרוש מכפלה בין שני מספרים עם $n/2$ ספרות. כעת, בעזרת הסימונים נוכל להציג את $xy$ כך: $xy = 2^n\cdot A + 2^{n/2}(C-A-B)+B$.
סבירה כי אומנם יש לנו יותר חיבורים אך יש רק 3 מכפלות ($A,B,C$. לאחר שמכפלים פעם אחת ניתן להשתמש בערך במקום אחר). נסמן את מספר החיבורים $c'n$. תנאי העצירה לא משתנה. נחשב את זמן הריצה $T(n)$.

3) <u>חישוב זמן ריצה</u>:

$$T(n)=\begin{cases} 1 & n=1 \\ 3T\left(\frac{n}{2}\right)+c'n & n>1 \end{cases}$$

$$T(n)=3T\left(\frac{n}{2}\right)+c'n = 3\left(3T\left(\frac{n}{2^2}\right)+c'\frac{n}{2}\right)+c'n = 3^2T\left(\frac{n}{2^2}\right)+\frac{3}{2}c'n+c'n$$

$$= 3^2\left(3T\left(\frac{n}{2^3}\right)+c'\cdot\frac{n}{2^2}\right)+\frac{3}{2}c'n+c'n = 3^3T\left(\frac{n}{2^3}\right)+\left(\frac{3}{2}\right)^2c'n+\frac{3}{2}c'n+c'n$$

לאחר $k$ שלבים: $= 3^kT\left(\frac{n}{2^k}\right)+c'n\left(\left(\frac{3}{2}\right)^{k-1}+\dots+\frac{3}{2}+1\right) = 3^kT\left(\frac{n}{2^k}\right)+2c'n\left(\left(\frac{3}{2}\right)^k-1\right)$

נעצור כאשר $\frac{n}{2^k}=1$, ומכאן $k=\log_2 n$.

$$T(n)=3^{\log_2 n}\cdot T(1)+2c'\cancel{n}\cdot\frac{3^{\log_2 n}}{\cancel{2^{\log_2 n}}}-2c'n = 3^{\log_2 n}(1+2c')-2c'n$$

$$= 3^{\log_3 n\cdot\log_2 3}(1+2c')-2c'n = n^{\log_2 3}(1+2c')-2c'n = O(n^{1.584})$$

חישובי עזר:
(1) $\log_2 n = \frac{\log_3 n}{\log_3 2} = \log_3 n\cdot\frac{1}{\log_3 2} = \log_3 n\cdot\log_2 3$.
(2) $\log_2 3 = 1.584$

מסקנה: הצלחנו לייעל את זמן הריצה באמצעות אלגוריתם הפרד ומשול.

## 3) <u>מציאת חציון</u>

במערך לא ממוין נרצה למצוא את האיבר שחצי מהאיברים גדולים ממנו וחצי מהאיברים קטנים ממנו. איבר זה נקרא "חציון". נרצה למצוא את החציון לדוגמא במיון מהיר (QuickSort) כדי לקבוע אותו כאיבר הציר (Pivot), ובכך לקבל מיון יעיל יותר. דרך פשוטה למצוא חציון היא למיין את המערך ואז לקחת את האיבר האמצעי בזמן ריצה של $O(n\log n)$, אמנם זה כמו בעיית הביצה והתרנגולת שהרי אנו רוצים למצוא חציון לצורך מיון. ננסה להשתמש באלגוריתמים של הפרד ומשול לפתרון יעיל יותר.

א) <u>פתרון 1</u>: נחלק את המערך ל-2 חלקים שווים $A_1, A_2$, ונמצא ברקורסיה את החציונים $m_1,m_2$ בכל אחד מהם. אם שני החציונים שווים $m_1=m_2$, נחזיר את הערך השווה. אם אינם שווים, אזי: אם $m_1<m_2$ ניקח את האיברים שגדולים מ-$m_1$ ב-$A_1$ ואת האיברים

29/10 בס"ד

(4)

הקטנים מ-$m_2$ ב-$A_2$ ונמצא מהם את החציון, והוא יהיה גם החציון של A.
ואם $m_1 > m_2$ אז ניקח את האיברים שקטנים מ-$m_1$ ב-$A_1$ ואת האיברים שגדולים מ-$m_2$ ב-$A_2$, ומהם נמצא חציון.

ה) <u>פסאודו-קוד:</u>

```
median(A) {
    divide A into A1 and A2
    m1 = median(A1)
    m2 = median(A2)
    if(m1 == m2)
        return m1
    else if (m1 < m2)
        B = (a∈A1 | a > m1 + b∈A2 | b < m2)
    else // m1 > m2
        B = (a∈A1 | a < m1 + b∈A2 | b > m2)
    return median(B)
}
```

ו) <u>חישוב זמן ריצה:</u> בקוד זה אנו מפעילים 3 פעמים את הפונקציה median על מערך בגודל $\frac{n}{2}$. גם B בגודל $\frac{n}{2}$ שהרי אנו לוקחים חצי מ-$A_1$ וחצי מ-$A_2$ שהם כל אחד רבע מ-A. כדי לבנות את B צריך לעבור על כל המערך ולבחור את האיברים המקיימים את התנאי, בסיבוכיות $O(n)$.
תנאי העצירה הוא כאשר $n=1$, שאז הוא החציון של עצמו. לסיכום, זמן הריצה הוא:

$$T(n)=\begin{cases} 1 & n=1 \\ 3T\left(\frac{n}{2}\right)+n & n>1 \end{cases}$$

כבר באלגוריתם קודם חישבנו זמן ריצה זה שהוא $O(n^{1.584})$. אמנם כבר עדיף סיבוכיות של $O(n\log n)$, לכן זהו פתרון לא יעיל. ננסה פתרון שונה.

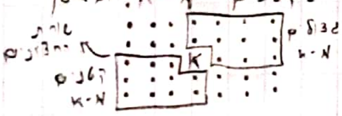
3) <u>פתרון 2:</u> נשתמש בפונקציה $Find(A,t)$ המקבלת מערך A ומחזירה את האיבר הכי גדול במקום $t$, כלומר שיש $t-1$ איברים בלבד במערך שגדולים ממנו. נרצה להשתמש בפונקציה עבור $t=\frac{n}{2}$ כדי למצוא את החציון. פונקציה זו עובדת כך שהיא בוחרת איבר ציר x (בהמשך נלמד שיטה טובה לבחירת איבר זה) ואז מחלקת את המערך A ל-2 חלקים: $S_1$ שיכיל את האיברים הגדולים מ-x,

⑤

ו- $S_2$ שיכיל את האיברים הקטנים מ-k. אם ב-$S_1$ יש $t-1$ איברים נחזיר את k. ואם גדול מ-$t-1$ אזי נפעיל עליו שוב את הפונקציה $Find(S_1,t)$. ואם $S_1$ קטן מ-$t-1$ איברים, אזי נפעיל על $S_2$ את הפונקציה $Find(S_2, t-|S_1|-1)$.

ה) __בחירת איבר ציר (pivot):__ אנו רוצים שאיבר הציר k שלנו יהיה כמה שיותר קרוב לחציון, כלומר שמבחינת גודל כל האיברים הוא איבר אמצעי יחסית. לשם כך, נחלק את A לקבוצות בגודל b כאשר b היא מספר אי זוגי קטן, סה"כ נקבל $\frac{n}{b}$ קבוצות. נמיין כל קבוצות אלו. ניקח את האיבר האמצעי מכל קבוצה ונכניס אותו למערך B, כך שבסוף במערך B יש $\frac{n}{b}$ איברים. נמצא את החציון במערך B על ידי פקודה $Find(B,\frac{n}{2b})$. איבר זה יהיה איבר הציר שלנו. נמצא שאנו משתמשים בפונקציה $Find(A,t)$ גם כדי למצוא חציון וגם כדי למצוא איבר ציר.

איבר זה הוא אכן איבר אמצעי יחסית. נוכיח כי יש לפחות $\frac{n}{4}$ איברים גדולים ממנו ובדרך דומה ניתן להוכיח שישנם לפחות $\frac{n}{4}$ איברים קטנים ממנו. במערך B יש $\frac{n}{b}$ איברים. מתוכם בדיוק $\frac{n}{2b}$ הקטנים מ-k. בנוסף, בכל קבוצה של כל איבר כזה יש $\frac{b}{2}$ איברים שקטנים מהחציון ולכן בוודאי קטנים מ-k. סה"כ יש לפחות $\frac{n}{2b}\cdot\frac{b}{2}=\frac{n}{4}$ איברים שקטנים מ-k. ובאופן דומה נוכיח גם שיש לפחות $\frac{n}{4}$ איברים שגדולים מ-k.




כיוון שבאלגוריתם זה אנו מבצעים חישובים רבים (אמנם בזמן ריצה יעיל כמו שנוכיח בהמשך), נבצע אותם רק כאשר $n\geq 50$. אם $n<50$ נמיין את A וניקח את האיבר האמצעי בזמן ריצה של $O$(חפסת).

ו) __פסודו קוד:__

```
              n/2
Find(A, t) {
    If |A| < 50 sort A and return the t elemen
    divide A into groups of size b, and sort each group
    B = medians of all the groups
    k = Find(B, n/2b)
    S1 = {x∈A | x>k} ,  S2 = {x∈A | x<k}
    if (|S1| = t-1) return k
    else if (|S1| > t-1)  Find(S1, t)
    else  Find(S2, t-|S1|-1)      //|S1| < t-1
}
```

5/11

6

ג) <u>חישוב זמן ריצה</u>: אנו מפעילים רקורסיבית את הפונקציה Find פעמיים: פעם אחת לצורך מציאת איבר הציר, שאז גודל המערך הוא $\frac{n}{5}$. ופעם שניה למציאת חציון על $S_1$ או $S_2$. מכיוון שאמרנו ש-$S_1$ ו-$S_2$ מכילים לפחות $\frac{n}{4}$ איברים כל אחד, במקרה הגרוע נפעיל את הפונקציה על מערך עם $\frac{3n}{4}$ איברים. בנוסף, חלוקת המערך לקבוצות של $b$ ומיונם, הכנסת החציונים ל-$B$, והכנסת האיברים הגדולים מ-$x$ ל-$S_1$ והקטנים ל-$S_2$, כולם מתבצעים בזמן ריצה לינארי, ולכן נסמנם ב-$c\cdot n$ כאשר $c$ קבוע.

בדרך כלל נבחר את $b=5$. זה יעזור לנו בהוכחת זמן ריצה וגם בגלל שלמיין מערך בגודל 5 דרושים 7 השוואות סה"כ. כי חישבנו לפי המקרה הגרוע

זמן הריצה הסופי הוא אם כן:

$$T(n)\le\begin{cases} n\log n & n<50 \\ T(\frac{n}{5})+T(\frac{3}{4}n)+c\cdot n & n\ge 50\end{cases}$$

נוכיח באינדוקציה חזקה שזמן ריצה זה הוא לינארי, כלומר שמתקיים: $T(n)\le cdn$, כאשר $cd$ קבוע.

<u>בסיס</u>: $n<50$. זמן הריצה הוא $T(n)\le n\log n$, ולכן $T(n)\le 50\log 50<6n$. שהוא אכן לינארי.

<u>הנחה</u>: נניח באינדוקציה חזקה כי עבור כל $k<n$ מתקיים $T(k)\le 20ck$.

נבחר $d=20$ כי הוא המכנה המשותף של 4 ו-5.

<u>הוכחה</u>: לפי הנחת האינדוקציה $T(\frac{n}{5})\le\frac{20cn}{5}$, $T(\frac{3}{4}n)\le\frac{20\cdot 3cn}{4}$. ומכאן נחשב:

$$T(n)\le T(\frac{n}{5})+T(\frac{3}{4}n)+cn\le\frac{20cn}{5}+\frac{20\cdot 3cn}{4}+cn$$

$$=4cn+15cn+cn=20cn \Rightarrow T(n)\le 20cn$$ מ.ש.ל.

הוכחנו כי לכל $n$, $T(n)\le 20ck$, ולכן זמן הריצה הוא אכן לינארי $O(n)$, שזהו זמן ריצה יעיל יותר לפעולה זו.