א׳ תפקידם של האלגוריתמים בעולם המחשוב (פרק 1); התחלה (פרק 2)

=====================

קראו את ההקדמה

ואת **פרק 1** מספר הלימוד

=====================

בעיות חישוביות ואלגוריתמים

אלגוריתם הוא מרשם לפתרון בעיה חישובית. בדרך כלל, לכל בעיה שנפתרה ידועים מספר אלגוריתמים, ותמיד קיימת האפשרות שנצליח לגלות או להמציא אלגוריתמים נוספים. גם לאלגוריתם ידוע ניתן להמציא גרסאות חדשות. בכל המאמץ הזה, מה שמעניין אותנו בעיקר זה למצוא אלגוריתמים **יעילים** יותר. שני היבטים חשובים ביותר: יעילות הזמן של אלגוריתם (כמות הזמן) ויעילות המקום שלו (כמות הזיכרון). אלגוריתם נקרא יעיל מבחינת זמן הריצה שלו אם הוא דורש זמן ריצה קצר יחסית; הוא נקרא יעיל מבחינת המקום אם הוא דורש כמות קטנה יחסית של זיכרון בנוסף לנתוני הקלט והפלט. גורמים רבים יכולים להשפיע על זמן הריצה של אלגוריתם: גודל הבעיה, מהירות הביצוע של החומרה (המחשב), ביצועי המהדר (האיכות שלו), מיומנות המתכנת, ואחרים.

בחיפושינו אחר אלגוריתם הפותר בעיה נתונה, אנחנו מעוניינים למצוא אלגוריתם הצורך כמה שפחות משאבים (זמן וזיכרון). ברור שאם אלגוריתם צורך יותר מדי מקום בזיכרון, לא נוכל לפתור את הבעיה על המחשב שברשותנו מעבר לגודל מסוים של הקלט. נוכל אולי להחליף את המחשב במחשב גדול יותר, אבל מהר מאוד ניתקל באותה בעיה עם גידול הקלט. אם האלגוריתם צורך יותר מדי זמן, לא נוכל להשלים את משימתנו כמתוכנן מעבר לגודל מסוים של הקלט. גם במקרה הזה, החלפת המחשב במחשב מהיר יותר תפתור לנו את הבעיה רק לזמן מוגבל. לכן חשוב להשוות בין ביצועי האלגוריתמים הפותרים אותה בעיה. אנחנו נתעניין יותר בהיבט זמן הריצה מאשר בזה של כמות הזיכרון.

השוואת אלגוריתמים מבחינת יעילותם

במהלך ריצתו, כל אלגוריתם מבצע כמה סוגים של פעולות יסוד, בעיקר פעולות חיבור, חיסור, כפל, חלוקה, השוואה והעתקה. למשל, אלגוריתם חיפוש מבצע בעיקר פעולות השוואה; אלגוריתם מיון מבצע גם פעולות העתקה. הזמן שכל סוג של פעולה צורך תלוי בסוג המחשב: החומרה יכולה להיות איטית יותר עבור פעולות מסוג אחד ומהירה יותר עבור פעולות מסוג אחר; המצב משתנה מסוג אחד של מחשב לסוג אחר. לכן, אין טעם למדוד את זמני הריצה של אותו אלגוריתם על מחשבים שונים. גם אין ביכולתנו האפשרות להשפיע על מהירות ביצוע המחשב שבידינו. לכן, חשוב יותר למדוד את ביצועי אלגוריתמים שונים על אותו מחשב. אבל מדידות אלה נותנות לנו תוצאות מוגבלות ביותר; לעולם לא נוכל להעריך את ביצועי אלגוריתם נתון על כל המחשבים הקיימים. לכן חשוב לנו מאוד לחפש מדדים להערכת ביצועי האלגוריתמים שאינם תלויים במאפיינים של החומרה שבשימוש. הדרך הפשוטה ביותר היא לספור, או לפחות להעריך את מספר פעולות היסוד (רצוי עבור כל סוג של פעולה בנפרד, אם אפשר).

חשוב מאוד שנכתוב אלגוריתמים יעילים ככל שאפשר; ביכולתנו להשפיע, במידה מסוימת, על מהלך פתרון הבעיה, כלומר, על מספר פעולות היסוד שהאלגוריתם שלנו מבצע. אם ברצוננו להשוות בין שני אלגוריתמים הפותרים אותה בעיה, נוכל להשוות בין מספר פעולות היסוד שכל אלגוריתם מבצע עבור קלט בגודל נתון.

שתי השאלות הבאות מומלצות רק לבעלי רקע מתמטי (חומר רשות).

שאלה א-1

ברצוננו להשוות את ביצועי שני אלגוריתמים, A ו-B, בריצתם על אותו מחשב. ידוע לנו שעבור קלט בגודל n, אלגוריתם A מבצע $8n^2$ פעולות השוואה ו- $32n\lg n$ פעולות העתקה; עבור קלט באותו גודל, אלגוריתם B מבצע $256n\lg n$ פעולות השוואה ו- $128n\lg\lg n$ פעולות העתקה. עבור אלו ערכים של n אלגוריתם A "מנצח" את אלגוריתם B באופן וודאי (גם מבחינת ההשוואות וגם מבחינת ההעתקות)? לא תוכלו לתת תשובה כוללת, מכיוון שלא ידוע לנו היחס בין זמן הביצוע של פעולת השוואה לבין זה של פעולת העתקה.

שאלה א-2

השלימו את התשובה לשאלה הקודמת: בהנחה שפעולת העתקה צורכת פי 8 זמן מפעולת השוואה, עבור אלו ערכים של n אלגוריתם A "מנצח" את אלגוריתם B?

=============================

קראו ולמדו את ה**סעיף 2.1** מספר הלימוד

=============================

השימוש בפסֵידוקוד

נשתמש במוסכמות המתוארות בספר הלימוד לצורך קידוד האלגוריתמים. **פסֵידוקוד** הוא שיטת קידוד מופשטת שלא ניתן להדר ולהריץ על מחשב, אך ניתן לתרגם בקלות לתכנית בשפת תכנות. בניגוד למה שנהוג בשפות התכנות התקניות, בפסֵידוקוד לא מגדירים את המשתנים, מתעלמים מתהליכי הקלט\פלט ומאפשרים חופש מסוים במבנה. כל זה כדי להשתחרר מהמאפיינים היחודיים של שפות התכנות ולאפשר לנו להתרכז במבנה האלגוריתמים עצמם. חשוב לציין מגבלה אחת שחייבים להקפיד עליה: ל**הזחת** שורות הקוד (אינדנטציה) תפקיד חשוב מאוד בבניית השגָרות ובהגדרות הבלוקים של הוראות בתוכן, כמו מבני ההסתעפות **if-then-else** ובמבני הלולאות **while**, **for**, **repeat**. השורות ממוספרות לצורך התיחסות אליהן.

האלגוריתם מיון-הכנסה

אלגוריתם זה הוא הראשון המופיע בספר הלימוד. פה גם מוזכר בפעם הראשונה נושא הוכחת נכונות אלגוריתם איטרטיבי בעזרת **שמורת לולאה**. אחרי קריאת הסעיף, מומלץ לפתור את התרגילים 2.1-1 עד 2.1-4.

שאלה א-3

האלגוריתם מיון-הכנסה בונה את המערך הממוין החל מהקצה השמאלי שלו. כתבו גרסה של מיון-הכנסה הבונה את המערך הממוין החל מהקצה הימני שלו.

שאלה א-4

ניתן לשפר את זמן הריצה של מיון-הכנסה במחיר של הוספת תא זיכרון אחד למערך.

נוסיף למערך A את התא מספר 0, כלומר נעבוד במערך $A[0..n]$. איברי המערך יהיו מאוחסנים, כמו קודם, בתאים $A[1..n]$. ב- $A[0]$ נאחסן את הערך $-\infty$ (נקרא לו **זקיף**).

הראו שהוספת הזקיף מאפשרת לנו לבצע פחות פעולות השוואה בשגרת המיון. כתבו את הגרסה המתאימה בפסֵידוקוד.

חיפוש לינארי

בתרגיל 2.1-3 מתוארת בעיית החיפוש אחר ערך נתון v במערך $A[1..n]$. האלגוריתם הפשוט ביותר הפותר את בעיה זו הוא אלגוריתם החיפוש הלינארי. האלגוריתם משווה את v לאיברי המערך, החל מהראשון (השמאלי ביותר); החיפוש מסתיים אם הערך v נמצא במערך - חיפוש מוצלח – או אם החיפוש עובר את האיבר האחרון (הימני ביותר) – חיפוש כושל. נתאר את האלגוריתם בפסֵידוקוד (NIL מסמן תוצאה לא מוגדרת, במקרה הזה חיפוש כושל):

```
LINEAR-SEARCH(A, v)
i ← 1
while i ≤ length[A] and A[i] ≠ v
   do i ← i + 1
if i > length[A]
   then return NIL
   else return i
```

הנחנו פה כי $length[A] > 0$.

שאלה א-5

בהנחה שהמערך A ממוין בסדר עולה (או לא יורד), ניתן לשפר את אלגוריתם החיפוש הלינארי כך שהחיפוש הכושל ייפסק מיד אחרי שהוא נתקל באיבר גדול מ-v. כתבו את השגרה החדשה בפסֵידוקוד.

שאלה א-6

כתבו גרסה של אלגוריתם החיפוש הלינארי במערך ממוין (שאלה א-5) כך שהחיפוש יתבצע בסדר יורד (מימין לשמאל). כתבו את השגרה המתאימה בפסֵידוקוד.

שאלה א-7

הראו שניתן לשפר את ביצועי השגרה הקודמת (שאלה א-6) על חשבון תא זיכרון אחד. כתבו את השגרה המתאימה בפסֵידוקוד.

רמז: הוסיפו זקיף למערך A (הערך $-\infty$ בתא $A[0]$).

==========================

קראו ולמדו את ה**סעיף 2.2** מספר הלימוד

==========================

אם חסר לכם הרקע האלגברי, קראו את **נספח א'**.

ניתוח זמן הריצה של האלגוריתמים

נתחיל בכמה כללים עבור ניתוח (חישוב, הערכת) זמני הריצה של האלגוריתמים. החלק העיקרי מורכב מחישוב (או הערכת) מספר פעולות היסוד (מספר ה"צעדים") שמבצע האלגוריתם כ**פונקציה של גודל הקלט**. אנו מניחים פה שכל פעולת יסוד מתבצעת ביחידת זמן אחת (פעולות מסוגים שונים מתבצעות בזמנים שונים, אז אפשר לקחת את זמן הריצה הארוך ביותר של פעולת יסוד כלשהי כחסם עליון על זמני הריצה של כל הפעולות ואת זמן הריצה הקצר ביותר של פעולת יסוד כלשהי כחסם תחתון על זמני הריצה של כל הפעולות).

בדרך כלל לא ניתן לחשב במדויק את מספר פעולות היסוד, לכן ננסה בכל מקרה למצוא חסם עליון או חסם תחתון (או את שניהם) על מספר הפעולות.

אם הגענו לביטוי אלגברי המסמן את מספר הפעולות, נוכל לבחור את האיבר המוביל (האיבר השולט) ולהתעלם מהאיברים האחרים. למשל, אם הגענו לביטוי

$$T(n) = 3n^2 + 5n \cdot \lg n + 8n + 6\lg n$$

נוכל להסתפק באיבר $3n^2$; אם הגענו לביטוי

$$T(n) = 2n^2 \cdot \lg n + n^3 \cdot \lg\lg n + n \cdot (\lg n - 1)$$

נוכל להשאיר את האיבר $n^3 \cdot \lg\lg n$ בלבד. בכל מקרה, הגורם הקבוע זניח בהשוואה ל- n ואינו מוסיף לנו שום ידע חשוב, לכן אפשר להוריד אותו; מה שנשאר לנו הוא ביטוי המייצג את **סדר**

הגודל של פונקצית זמן הריצה. לכן, נוכל לכתוב במקרה הראשון $T(n)=\Theta(n^2)$ ובמקרה השני $T(n)=\Theta(n^3\cdot\lg\lg n)$. (אנשים רבים מעדיפים לכתוב $O(f(n))$ במקום $\Theta(f(n))$.)

דבר נוסף שחייבים להבהיר: זמן הריצה של אלגוריתם תלוי לא רק בגודל הקלט אלא גם, במידה רבה, בקלט המסוים עצמו. נסמן ב- $\mathrm{K}(n)$ את קבוצת כל הקלטים בגודל n של האלגוריתם A. לכל קלט $I\in\mathrm{K}(n)$, $T(I)$ יציין את זמן הריצה של האלגוריתם הרץ על הקלט הזה. נסמן:

$T_{worst}(n)=\max\{T(I)\,|\,I\in\mathrm{K}(n)\}$ (זמן הריצה הארוך ביותר של A על קלט באורך n)

$T_{best}(n)=\min\{T(I)\,|\,I\in\mathrm{K}(n)\}$ (זמן הריצה הקצר ביותר של A על קלט באורך n)

$T_{average}(n)=(\sum_{I\in\mathrm{K}(n)}T(I))/|\mathrm{K}(n)|$ (זמן הריצה הממוצע של A על הקלטים באורך n)

אם האלגוריתם A רץ על קלט I עם זמן ריצה שסדר הגודל שלו הוא $T_{worst}(n)$, אזי אומרים שהקלט I הוא **מקרה גרוע** (ביותר) עבור האלגוריתם; גם אומרים שזמן הריצה של האלגוריתם במקרה הגרוע הוא $T_{worst}(n)$.

אם האלגוריתם A רץ על קלט I עם זמן ריצה שסדר הגודל שלו הוא $T_{best}(n)$, אזי אומרים שהקלט I הוא **מקרה טוב** (ביותר) עבור האלגוריתם; גם אומרים שזמן הריצה של האלגוריתם במקרה הטוב הוא $T_{best}(n)$.

בהנחה שכל הקלטים עשויים להופיע בסיכויים שווים, הפונקציה $T_{average}(n)$ נקראת זמן הריצה של האלגוריתם בממוצע, או ב**מקרה הממוצע**, או **תוחלת** זמן הריצה של האלגוריתם.

אחרי קריאת ה**סעיף 2.2** מומלץ לפתור את התרגילים 2.2-1 עד 2.2-4.

האלגוריתם מיון-בחירה

ראו את ה**תרגיל 2.2-2**.

האלגוריתם בפסֵידוקוד:

```
SELECTION-SORT(A)
1  for i ← 1 to n−1
2     do min ← i
3        for j ← i+1 to n
4           do if A[min] > A[j]
5                 then min ← j
6        exchange A[i] ↔ A[min]
```

נגדיר את **שמורת הלולאה** הבאה: "בהתחלת כל איטרציה של לולאת **for** בשורות 1-6, התת-מערך $A[1..i-1]$ מכיל את $i-1$ האיברים הקטנים ביותר ב- A בסדר ממוין."

נוכיח את נכונות האלגוריתם:

אתחול: לפני האיטרציה הראשונה, התת-מערך $A[1..i-1]$ ריק, לכן שמורת הלולאה מתקיימת באופן ריק.

תחזוקה: בשורות 2-5 נמצא האיבר המינימלי של התת-מערך $A[i..n]$ (דרושה הוכחה פורמלית גם עבור החלק הזה, אבל זה קל להשלים); בשורה 6, האיבר המינימלי הזה הועבר למקום i; מכיוון שהאיברים ב- $A[1..i-1]$ כולם קטנים או שווים ל- $A[i]$, מתקבל התת-מערך $A[1..i]$ המכיל את i האיברים הקטנים ביותר ב- A בסדר ממוין.

סיום: אחרי האיטרציה האחרונה מתקבל המערך השלם $A[1..n]$ ממוין בסדר עולה.

שאלה א-8

כתבו את הגרסה הבאה של האלגוריתם מיון-בחירה המשתמשת, חוץ מהמערך $A[1..n]$, גם במערך נוסף $B[1..n]$: תחילה, מוצאים את האיבר הקטן ביותר ב- A ומעתיקים אותו ל- $B[1]$; לאחר מכן, מוצאים את האיבר השני הקטן ביותר ב- A ומעתיקים אותו ל- $B[2]$; ממשיכים באותו אופן עד שכל n האיברים ב- A נמצאים ב- B.

כתבו את האלגוריתם בפסֵידוקוד. השוו את ביצועיו לאלה של האלגוריתם מיון-בחירה המקורי.

===========================

קראו ולמדו את ה**סעיף 2.3** מספר הלימוד

===========================

בסעיף זה מתוארת פרדיגמת הפרד-ומשול, אחת התבניות העיקריות לתכנון ופיתוח אלגוריתמים. שיטת התכנון מבוססת על שימוש ברקורסיה : מחלקים את הבעיה לתת-בעיות (נפרדות) – שלב ההפרדה ; פותרים את התת-בעיות באופן רקורסיבי – שלב השליטה ; ממזגים את התוצאות – שלב הצירוף.

אחרי קריאת הסעיף, מומלץ לפתור את התרגילים 2.3-1 עד 2.3-7.

שגרת המיזוג

שגרה זו מבצעת בעצם את כל העבודה באלגוריתם מיון-מיזוג. קיימות כמה גרסאות אפשריות עבור השגרה. אחת מהן, המשתמשת בזקיפים, מתוארת בספר הלימוד. גרסה אחרת היא זו שנדרשת ב**תרגיל 2.3-2** :

```
MERGE(A, p, q, r)
m ← r − p + 1
▷ create array B[1..m]
i ← p
j ← q + 1
k ← 1
while i ≤ q and j ≤ r
   do if A[i] ≤ A[j]
          then B[k] ← A[i]
               i ← i + 1
          else B[k] ← A[j]
               j ← j + 1
      k ← k + 1
while i ≤ q
   do B[k] ← A[i]
      i ← i + 1
      k ← k + 1
while j ≤ r
   do B[k] ← A[j]
      j ← j + 1
      k ← k + 1
for k ← p to r
   do A[k] ← B[k]
```

גם כאן משתמשים במערך עזר B . לולאת **while** הראשונה משווה בין איבר בתת-מערך השמאלי לבין איבר בתת-מערך הימני ; הקטן מביניהם מועתק ל- B . קידום האינדקסים מתבצע כמו בשגרה המקורית. אחרי שכל איברי תת-מערך אחד הועברו כבר ל- B , החלק שנשאר מהתת-מערך האחר מועבר כמו שהוא בהמשך ; זה קורה באחת מלולאות **while** השנייה או השלישית (בכל מקרה, אחת הלולאות האלה לא מבצעת דבר והאחרת מעבירה את האיברים שנשארו).

אין צורך ליצור מחדש את מערך העזר $B[1..r-p+1]$; אפשר ליצור מערך עזר $B[1..n]$ ולהשתמש בתת-מערך $B[p..r]$ (במקרה זה, עלינו לשנות את השורה 5 : $k \leftarrow p$).

מיון ורקורסיה

האלגוריתם מיון-מיזוג פועל בצורה רקורסיבית. גם אלגוריתמי מיון אחרים ניתן לכתוב בצורה רקורסיבית, למשל את מיון-הכנסה (ראו את ה**תרגיל 2.3-4**) :

השגרה הרקורסיבית

```
RECURSIVE-INSERTION(A, j)
1  if j > 1
2     then RECURSIVE-INSERTION(A, j − 1)
3          key ← A[j]
4           ▷ Insert A[j] into the sorted sequence A[1..j − 1]
5          i ← j − 1
6          while i > 0 and A[i] > key
7              do A[i + 1] ← A[i]
8                 i ← i − 1
9          A[i + 1] ← key
```

והשגרה הראשית (קריאת ההפעלה)

```
RECURSIVE-INSERTION-SORT(A)
1   RECURSIVE-INSERTION(A, length[A])
```

חיפוש בינרי

שיטה דומה ל״הפרד-ומשול״ ניתנת להפעלה גם באלגוריתמי חיפוש (רק שכאן מתאים יותר הכינוי ״הפרד-והסר״). בהינתן מערך ממוין $A[1..n]$ וערך v, אפשר לבצע חיפוש יעיל אחר v במערך A (ראו את ה**תרגיל 2.3-5**). כמו באלגוריתמים אחרים, גם במקרה הזה אפשר לכתוב גרסאות רקורסיביות וגרסאות איטרטיביות. נכתוב בהמשך גרסה רקורסיבית:

השגרה הרקורסיבית

```
BINARY-SEARCH(A, p, r, v)
if p > r
    then return NIL
q ← ⌊(p + r)/2⌋
if v < A[q]
    then return BINARY-SEARCH(A, p, q − 1, v)
else if v > A[q]
    then return BINARY-SEARCH(A, q + 1, r, v)
else return q
```

והשגרה הראשית (קריאת ההפעלה)

```
RECURSIVE-BINARY-SEARCH(A, v)
    BINARY-SEARCH(A, 1, length[A], v)
```

האלגוריתם מבצע שתי פעולות השוואה בכל קריאה רקורסיבית. כדי לחשב את מספר הקריאות הרקורסיביות, נפעל כדלקמן: הקריאה הראשונה פועלת על מערך באורך n, הקריאה השנייה (קריאה בעומק 1) פועלת על תת-מערך באורך $\lfloor n/2 \rfloor$ לכל היותר; וכן הלאה, הקריאה ה-$k+1$ (קריאה בעומק k) פועלת על תת-מערך באורך $\lfloor n/2^k \rfloor$ (ראו את הנוסחא (3.5) מספר הלימוד). התהליך נמשך כל עוד $\lfloor n/2^k \rfloor \geq 1$. בקריאה האחרונה, $\lfloor n/2^k \rfloor \geq 1$ וגם $\lfloor n/2^{k+1} \rfloor < 1$, וזה מתקיים אם ורק אם $2^k \leq n < 2^{k+1}$; נפעיל את פונקצית הלוגריתם ונקבל $k \leq \lg n < k+1$; מזה נובע $k = \lfloor \lg n \rfloor$. לכן, זמן הריצה של האלגוריתם חיפוש בינרי מקיים $T(n) = \Theta(\lg n)$.

שאלה א-9

ראו את ה**תרגיל 2.3-6** בספר הלימוד :

נחליף את החיפוש הלינארי שבשורות 7-5 בשגרה INSERTION-SORT בחיפוש בינרי. האם שינוי זה ישפר את ביצועי שגרת המיון כך שזמן הריצה יהיה $\Theta(n \cdot \lg n)$?

שאלה א-10

ראו את ה**תרגיל 2.3-7** בספר הלימוד :

נתונה סדרה S בת n שלמים ושלם נוסף z . כתבו אלגוריתם שזמן ריצתו $\Theta(n \cdot \lg n)$, הקובע האם קיימים ב- S שני שלמים a ו- b המקיימים $a + b = z$.

מיון-בועות

מיון-בועות הוא אחד האלגוריתמים הידועים יותר למיון מערכים. אחת הגרסאות שלו מופיעה בספר הלימוד (ראו את ה**בעיה 2-2**) :

```
BUBBLE-SORT(A)
1  for i ← 1 to length[A]
2      do for j ← length[A] downto i + 1
3             do if A[j] < A[j − 1]
4                   then exchange A[j] ↔ A[j − 1]
```

כדי להוכיח את נכונות האלגוריתם, עלינו להראות שפעולת האלגוריתם לא חורגת מגבולות המערך A , ושהפלט שלו ממוין. מעקב אחר פעולת האלגוריתם מראה שהוא פועל באופן אינקרמנטלי ושהוא בונה (בדומה למיון-הכנסה ולמיון-בחירה) תת-מערך ממוין בצד השמאלי של A . בכל איטרציה של הלולאה הראשית (שורות 2-4) מתווסף איבר חדש לתת-מערך הממוין ; איבר זה מתגלה כאיבר המינימלי בתת-מערך הלא ממוין שבצד הימני של A . לכן, לצורך הוכחת הנכונות, נוכל להגדיר את שמורת הלולאה עבור הלולאה הפנימית ($n = length[A]$) :

"בהתחלת האיטרציה ה- k של לולאת ה-**for** שבשורות 2-4, האיבר $A[j]$ ($j = n - k + 1$) הוא האיבר המינימלי בתת-מערך $A[j..n]$ ",

ואת שמורת הלולאה עבור הלולאה החיצונית:

"בהתחלת כל איטרציה של לולאת ה-**for** שבשורות 1-4, התת-מערך $A[1..i-1]$ מכיל את $i-1$ האיברים הקטנים ביותר ב- A בסדר ממוין."

שאלה א-11

הוכיחו באופן פורמלי ששמורת הלולאה הפנימית מתקיימת.

השתמשו בתנאי הסיום של שמורת לולאה זו כדי להוכיח באופן פורמלי ששמורת הלולאה החיצונית מתקיימת.

שאלה א-12

חשבו את מספר פעולות ההשוואה (בין איברי המערך בלבד) באלגוריתם מיון-בועות הרץ על מערך בגודל n.

חשבו את המספר המכסימלי של פעולות החלפה באלגוריתם מיון-בועות ואת מספר פעולות ההעתקה (פעולת החלפה מורכבת משלוש פעולות העתקה).

מהו זמן הריצה של האלגוריתם מיון-בועות במקרה הגרוע?

שאלה א-13

אפשר לשפר את האלגוריתם מיון-בעות באופן הבא:

בכל איטרציה של הלולאה החיצונית בודקים אם בוצעו פעולות החלפה; אם כן, ממשיכים את פעולת הלולאה, אחרת מפסיקים.

כתבו את גרסה זו של האלגוריתם בפסֵידוקוד.

חישוב ערכי פולינומים

נתון פולינום במעלה n (הארגומנט x והמקדמים a_i , $i=0,1,\ldots,n$, יכולים להיות כולם מספרים ממשיים, או מרוכבים, או מסוג אחר)

$$P(x)=\sum\nolimits_{i=0}^{n} a_i\cdot x^i = a_0+a_1\cdot x+\ldots+a_n\cdot x^n$$

ברצוננו לחשב את הערך $P(x)$ עבור ערך נתון של x.

הדרך הפשוטה ביותר (אבל לא היעילה ביותר) תהיה לחשב את x^i לכל $i=0,1\ldots,n$, בעזרת השגרה

```
POWER(x, n)
p ← 1
for i ← 1 to n
    do p ← p * x
return p
```

שרצה בזמן $\Theta(n)$. נניח שהמקדמים של P מאוחסנים במערך $A[0..n]$ ($a_i=A[i]$, לכל $i=0,1\ldots,n$). אז נוכל לחשב את $P(x)$ בעזרת השגרה

```
POLYNOMIAL(A, x)
n ← length[A] − 1
pn ← 0
for i ← 0 to n
    do pn ← pn + A[i] * POWER(x, i)
return pn
```

השגרה מבצעת $n+1$ קריאות לשגרה $\mathrm{POWER}(x,i)$ שתרוץ בזמן $\Theta(i)$, $i=0,1\ldots,n$; זמן הריצה של כל קריאות יהיה $\sum_{i=0}^{n}\Theta(i)=\Theta(\sum_{i=0}^{n} i)=\Theta(n^2)$.

נוכל לשפר את זמן הריצה אם נוסיף מערך עזר $X[0..n]$ שבו נאחסן את החזקות $X[i]=x^i$, $i=0,1\ldots,n$; גם נוכל לחשב את כל החזקות בזמן לינארי אם נשתמש בשגרה

```
POWERS(x, n)
  ▷ define array X[0..n]
  X[0] ← p ← 1
  for i ← 1 to n
      do X[i] ← p ← p * x
  return X
```

בתוך השגרה

```
POLYNOMIAL1(A, x)
  n ← length[A]
  X ← POWERS(x, n)
  pn ← 0
  for i ← 0 to n
      do pn ← pn + A[i] * X[i]
  return pn
```

זמן הריצה של השגרה הזאת הוא $\Theta(n)$.

אפשר לבצע את חישוב הפולינום ללא זיכרון נוסף, כפי שנראה בהמשך (ראו גם את ה**בעיה 2-3** בספר הלימוד).

כלל הורנר (Horner rule)

נגדיר את סדרת הפולינומים $P_i(x) = \sum_{j=i}^{n} a_j \cdot x^{j-i}$, $i = 0,1,...,n$. ניתן לבדוק בנקל את העובדות הבאות:

$P_n(x) = a_n$;

$P_{i-1}(x) = a_{i-1} + x \cdot P_i(x)$, לכל $i = 1,...,n$;

$P(x) = P_0(x)$.

נחשב את $P(x)$ בעזרת השגרה

```
HORNER(A, x)
1  n ← length[A]
2  pn ← A[n]
3  for i ← n downto 1
4      do pn ← A[i − 1] + x * pn
5  return pn
```

זמן הריצה של שגרה זו הינו $\Theta(n)$.

נוכיח עכשיו את נכונותה של השגרה $\text{HORNER}(A,x)$. לצורך זה, נגדיר את השמורה הבאה עבור לולאת ה-**for** שבשורות 3-4: "בתחילת האיטרציה ה-($n-i+1$) של לולאת ה-**for**, מתקיים שערכו של pn הוא $P_i(x)=\sum_{j=i}^{n} a_j \cdot x^{j-i}$". נבדוק את קיום התנאים:

אתחול: בתחילת האיטרציה הראשונה, ערכו של pn הוא $P_n(x)=a_n$;

תחזוקה: אם בתחילת האיטרציה ה-($n-i+1$) ($2 \le i \le n$) ערכו של pn הוא $P_i(x)=\sum_{j=i}^{n} a_j \cdot x^{j-i}$, אזי בתחילת האיטרציה ה-($n-i+2$) ערכו של pn הוא $P_{i-1}(x)=\sum_{j=i-1}^{n} a_j \cdot x^{j-i+1}$;

סיום: בסופה של האיטרציה ה-n (האחרונה) ערכו של pn הוא $P(x)=P_0(x)$.

תשובות לפרק א'

שאלה א-1

נשווה בין ערכי הפונקציה $C_A(n)=8n^2$ לבין ערכי הפונקציה $C_B(n)=256n\lg n$, עבור ערכי הפרמטר $n=2^k$, $k=0,1,2,...$, ונקבל $C_A(n)<C_B(n)$ לכל $n=2^k$, $0<k<8$; בנוסף, $C_A(256)=C_B(256)$. בעזרת נגזרות הפונקציה $c(x)=8x^2-256x\cdot\lg x$ ניתן להוכיח כי $C_A(n)<C_B(n)$ לכל $1<n<256$.

נשווה בין ערכי הפונקציה $D_A(n)=32n\lg n$ לבין ערכי הפונקציה $D_B(n)=128n\lg\lg n$, עבור ערכי הפרמטר $n=2^k$, $k=0,1,2,...$, ונקבל $D_A(n)<D_B(n)$ לכל $n=2^k$, $3<k<16$; בנוסף, $D_A(65536)=D_B(65536)$. בעזרת נגזרות הפונקציה $d(x)=32x\cdot\lg x-128x\cdot\lg\lg x$ ניתן להוכיח כי $D_A(n)<D_B(n)$ לכל $15<n<65536$.

לכן, האלגוריתם A מנצח לכל $15<n<65536$.

שאלה א-2

נשווה בין ערכי הפונקציה $S_A(n)=8n^2+8*32n\lg n=8n^2+128n\lg n$ לבין ערכי הפונקציה $S_B(n)=256n\lg n+8*128n\lg\lg n=256n\lg n+1024n\lg\lg n$, עבור ערכי הפרמטר $n=2^k$, $k=0,1,2,...$. נשתמש בנגזרות הפונקציה $s(x)=8x^2-1024x\cdot\lg\lg x$.

שאלה א-3

עלינו לבצע שני שינויים : היפוך כיוון לולאת ה-**for** והיפוך כיוון האי-שוויון ; נקבל

```
BACKW-INSERTION-SORT(A)
1  n ← length[A]
2  for j ← n − 1 downto 1
3      do key ← A[j]
4          ▷ Insert A[j] into the sorted sequence A[j + 1..n]
5          i ← j + 1
6          while i ≤ n and A[i] < key
7              do A[i − 1] ← A[i]
8                  i ← i + 1
9          A[i − 1] ← key
```

שאלה א-4

הזקיף חוסך לנו את ההשוואה $i > 0$ שבשורה 5.

הצורה החדשה של השגרה תהיה

```
SENTINEL-INSERTION-SORT(A)
1  for j ← 2 to length[A]
2      do key ← A[j]
3          ▷ Insert A[j] into the sorted sequence A[1..j − 1]
4          i ← j − 1
5          while A[i] > key
6              do A[i + 1] ← A[i]
7                  i ← i − 1
8          A[i + 1] ← key
```

שאלה א-5

השגרה החדשה תהיה

```
SORTED-LINEAR-SEARCH(A)
i ← 1
while i ≤ length[A] and A[i] < v
    do i ← i + 1
if i > length[A] or A[i] > v
   then return NIL
   else return i
```

הלולאה נפסקת מיד אחרי מציאת האיבר הראשון במערך הגדול מ- v.

שאלה א-6

נהפוך את כיוון הלולאה ואת כיוון האי-שוויון ונקבל את השגרה

```
BACKW-SORTED-LINEAR-SEARCH(A)
i ← length[A]
while i > 0 and A[i] > v
    do i ← i − 1
if i = 0 or A[i] < v
    then return NIL
    else return i
```

שאלה א-7

השגרה לחיפוש לינארי במערך עם זקיף תהיה

```
SENTINEL-BACKW-LINEAR-SEARCH(A)
i ← length[A]
while A[i] > v
    do i ← i − 1
if A[i] < v
    then return NIL
    else return i
```

הזקיף מאפשר לנו לחסוך בהשוואות $i > 0$ ו-$i = 0$.

שאלה א-8

```
SELECTION-SORT-1(A)
1  ▷ create array B[1..n]
2  for i ← 1 to n − 1
3     do min ← i
4        for j ← i + 1 to n
5           do if A[min] > A[j]
6                 then min ← j
7        B[i] ← A[min]
8        A[min] ← A[i]
```

לכל $i=1,\ldots,n-1$, השגרה מבצעת שתי העתקות בשורות 8-7, במקום ההחלפה (3 העתקות) בשגרה המקורית.

שאלה א-9

השינוי בשגרת מיון-הכנסה רק יקטין את מספר פעולות ההשוואה; מספר פעולות ההעתקה יישאר ללא שינוי (שורה 6 בשגרה המקורית). לכן זמן הריצה במקרה הגרוע יישאר $\Theta(n^2)$.

שאלה א-10

נעתיק את הסדרה S למערך $A[1..n]$ (ללא שינוי בסדר האיברים).

נמיין את הסדרה S בעזרת מיון-מיזוג, בזמן $\Theta(n \cdot \lg n)$.

לכל איבר $a=A[i]$, $i=1,\ldots,n$, נבצע ב-$A[1..n]$ חיפוש בינארי אחר הערך $b=z-A[i]$, בזמן $\Theta(\lg n)$. זמן הריצה של כל פעולות החיפוש הינו $\Theta(n \cdot \lg n)$, וכך גם זמן הריצה הכולל.

שאלה א-11

נכונות הלולאה הפנימית

אתחול: לפני האיטרציה הראשונה, $k=1$ ו- $j=n$; התת-מערך $A[j..n]$ מכיל איבר אחד בלבד, לכן שמורת הלולאה מתקיימת באופן ריק.

תחזוקה: נניח ששמורת הלולאה הפנימית מתקיימת בהתחלת האיטרציה ה- k, כלומר $A[n-k+1]$ הוא האיבר המינימלי בתת-מערך $A[n-k+1..n]$; אחרי ההשוואה בשורה 3, אם $A[n-k+1]<A[n-k]$, מתבצעת גם ההחלפה שבשורה 4; בסוף האיטרציה ה- k (ובהתחלת האיטרציה ה- $(k+1)$), $A[n-k]$ הוא האיבר המינימלי בתת-מערך $A[n-k..n]$, כלומר שמורת הלולאה מתקיימת גם לפני האיטרציה ה- $(k+1)$.

סיום: אחרי ביצוע האיטרציה ה- $(n-i)$, האיבר $A[i]$ הוא האיבר המינימלי בתת-מערך $A[i..n]$.

נכונות הלולאה החיצונית

אתחול: לפני האיטרציה הראשונה, $i=1$ והתת-מערך $A[1..i-1]$ ריק, לכן שמורת הלולאה מתקיימת באופן ריק.

תחזוקה: נניח ששמורת הלולאה החיצנית מתקיימת בהתחלת האיטרציה ה- i, כלומר התת-מערך $A[1..i-1]$ מכיל את $i-1$ האיברים הקטנים ביותר בסדר ממוין; כפי שראינו, הלולאה הפנימית מעבירה את האיבר המינימלי של התת-מערך $A[i..n]$ ל- $A[i]$; לכן, בסוף האיטרציה ה- i (בהתחלת האיטרציה ה- $(i+1)$) התת-מערך $A[1..i]$ מכיל את i האיברים הקטנים ביותר בסדר ממוין, כלומר שמורת הלולאה מתקיימת גם לפני האיטרציה ה- $(i+1)$.

סיום: אחרי ביצוע האיטרציה ה- $(n-1)$, התת-מערך $A[1..n-1]$ מכיל את $n-1$ האיברים הקטנים ביותר בסדר ממוין; אבל אז, $A[n]$ הוא האיבר הגדול ביותר, לכן כל המערך $A[1..n]$ הוא עכשיו ממוין.

שאלה א-12

פעולות ההשוואה מופיעות בשורה 3 של השגרה. לכל $n = length[A]$, $i = 1, 2, \ldots,$ מתבצעות $n - i$ פעולות השוואה. סה"כ, מספר ההשוואות הוא $\sum_{i=1}^{n} n - i = \sum_{i=0}^{n-1} i = \frac{1}{2} n(n-1)$. מספר השוואות זה נכון תמיד, ללא תלות בסדר האיברים במערך $A[1..n]$.

פעולות ההחלפה מופיעות השורה 4 של השגרה. פעולות אלה לא תמיד מתבצעות ; במקרה הגרוע, למשל כאשר המערך $A[1..n]$ ממוין בסדר יורד, מתבצעת החלפה אחת אחרי כל השוואה. לכן, במקרה הגרוע, מספר ההחלפות הוא $\frac{1}{2} n(n-1)$, ומספר ההעתקות הוא $\frac{3}{2} n(n-1)$. זמן הריצה של מיון-בועות במקרה הגרוע הוא $\Theta(n^2)$.

שאלה א-13

```
BETTER-BUBBLE-SORT(A)
for i ← 1 to length[A]
    do swap ← FALSE
       for j ← length[A] downto i + 1
           do if A[j] < A[j − 1]
                 then exchange A[j] ↔ A[j − 1]
                      swap ← TRUE
       if swap = FALSE
          then return
```