7. סיבוכיות של פעולות

7.1 - סיבוכיות מבני נתונים בייצוגים שונים

I רשימה

II מחסנית

III תור

7.2 - הערכת סיבוכיות

7.3 - מבנה הנתונים היעיל יותר

חישוב סיבוכיות של פעולות הוא נושא יסודי ומרכזי במדעי המחשב. חישוב סיבוכיות של פעולה נעשה על-ידי חישוב "עלות" הזמן והמקום הנדרשים לביצועה. בעיצוב תכנה אנו מחשבים תחילה את העלויות של הפעולות הבסיסיות, עבור מבני הנתונים השונים, ומשתמשים בעלויות אלה לחישוב העלויות של פעולות מורכבות יותר, הכוללות שימוש חוזר ונשנה בפעולות הבסיסיות. בפרק זה, ובחומר הלימוד בעיצוב תכנה אנו מתמקדים בעלות פעולות מבחינת זמן, כלומר מבחינת מספר ההוראות שיתבצעו במהלך ביצוע הפעולה.

צורת הביטוי הכללית של מספר ההוראות שיתבצעו נעשית תוך שימוש בסימן Big-O, המבטא סיבוכיות. כאשר מספר ההוראות שיתבצעו הוא מספר קבוע, שאינו מותנה בכמות הנתונים שבמבנה הנתונים, אנו מסמנים אותו בצורה $O(1)$. כאשר מספר ההוראות שיתבצעו הוא פונקציה של כמות הנתונים שבמבנה הנתונים, אנו משתמשים במשתנה n כדי לבטא באופן כללי את כמות הנתונים, ומתארים את מספר ההוראות כפונקציה של n. אם למשל, מספר זה הוא פונקציה לינארית של n (כגון, $3n$, או $3n+5$) נכתוב $O(n)$, שפירושו – "סיבוכיות בסדר גודל של n"; אם מספר זה הוא למשל פונקציה ריבועית של n נכתוב $O(n^2)$.

מספר ההוראות שיתבצעו במהלך ביצוע פעולה תלוי באלגוריתם לביצוע הפעולה ובאופן ייצוגו בפועל במחשב. בחומר הלימוד בעיצוב תכנה אנו עוסקים בייצוגים של מבני הנתונים רשימה, מחסנית ותור בשתי צורות – באמצעות מערך ובאמצעות שרשרת חוליות. בסעיף הראשון של פרק זה מוצגות טבלאות של סיבוכיות הפעולות הבסיסיות במבני נתונים אלו, לפי שני הייצוגים המוזכרים. בסעיף השני בפרק מודגמת הערכת סיבוכיות של אלגוריתמים נתונים, אשר בהם ישנם זימונים של פעולות בסיסיות. בדרך כלל ניעזר ישירות בסיבוכיות של הפעולות הבסיסיות בעת חישוב הסיבוכיות של פעולות מורכבות יותר, אך לפעמים כדאי להעמיק יותר בבחינת הסיבוכיות של פעולה מורכבת. יתר העמקה עשוי להוביל לחישוב מדוייק יותר, שלפעמים יוביל לסיבוכיות נמוכה יותר. מקרים כאלה מודגמים בסעיף השני בפרק.

במהלך פתרון בעיות בעיצוב תכנה, מוגדרים נתוני הבעיה והפעולות שיש לבצע עמם. יש לבחור מבני נתונים מתאימים לשמירת הנתונים, תוך התייחסות לפעולות לביצוע. מבני נתונים מתאימים יהיו כאלה שיאפשרו בהירות בשמירת הנתונים, פשטות באופן ביצוע הפעולות, ויעילות הפעולות מבחינת סיבוכיות החישובים שבהן. בחירות שונות של מבני נתונים עשויות להשפיע על יעילות הפעולות, מבחינת ערכי סיבוכיות, ורצוי לבחור את מבני הנתונים אשר מובילים ליעילות טובה יותר בביצוע הפעולות. בסעיף השלישי והאחרון בפרק זה מוצגת בעיה ומוצגים מבני נתונים אלטרנטיביים לשמירת הנתונים. עבור כל מבנה נתונים מוצג ניתוח סיבוכיות הפעולות הדרושות, המוגדרות בבעיה. בפרק האחרון בספר זה, פרק הבעיות המסכמות, מוצגים ניתוחים נוספים, של בעיות מגוונות.

7.1 - סיבוכיות מבני נתונים בייצוגים שונים

I רשימה

פעולה	ייצוג מערך	ייצוג שרשרת חוליות
אתחל-רשימה	*O(1)*	*O(1)*
עוגן-רשימה (L)	*O(1)*	*O(1)*
סוף-רשימה (L)	*O(1)*	*O(1)*
עוקב-ברשימה (L, p)	*O(1)*	*O(1)*
קודם-ברשימה (L, p)	*O(1)*	*O(n)*
הכנס-לרשימה (L, p, x)	*O(n)*	*O(1)*
הוצא-מרשימה (L, p)	*O(n)*	*O(n)*
עדכן-רשימה (L, p, x)	*O(1)*	*O(1)*
אחזר-מרשימה (L, p)	*O(1)*	*O(1)*
רשימה-ריקה? (L)	*O(1)*	*O(1)*

הערה: ניתן לבצע את הפעולה "הוצא מרשימה", כאשר הרשימה מיוצגת באמצעות שרשרת חוליות, בסיבוכיות של *O(1)*. זאת כאשר ניתן לזמן פעולה בסיסית זו עם שני מצביעים – מצביע לאיבר המוצא ומצביע לאיבר שלפניו. כדאי לכתוב ולהשתמש ישירות בפעולה כזו, המזומנת באמצעות שני המצביעים, בפתרון בעיות בהן פעולת ההוצאה מרשימה היא פעולה שכיחה. בפרק 4, אודות תבניות של רשימה ישנו פירוט נוסף בנקודה זו. ההצעה המוצגת כאן, היא "העשרה רעיונית" לתלמידי עיצוב תכנה, אך איננה ממש רלוונטית לפתרון בעיות בחומר הלימוד הנוכחי. זאת כיוון שפתרון בעיות בחומר הנוכחי מבוסס על הפעולה "הוצא מרשימה" שבה זימון עם מצביע בודד.

II מחסנית

פעולה	ייצוג מערך	ייצוג שרשרת חוליות
אתחל-מחסנית	*O(1)*	*O(1)*
מחסנית-ריקה? (S)	*O(1)*	*O(1)*
דחוף-למחסנית (S, x)	*O(1)*	*O(1)*
שלוף-ממחסנית (S)	*O(1)*	*O(1)*
הצץ-למחסנית (S)	*O(1)*	*O(1)*

III תור

פעולה	ייצוג מערך	ייצוג שרשרת חוליות
אתחל-תור	*O(1)*	*O(1)*
תור-ריק? (Q)	*O(1)*	*O(1)*
הכנס-לתור (Q, x)	*O(1)*	*O(1)*
הוצא-מתור (Q)	*O(n)*	*O(1)*
ראש-התור (Q)	*O(1)*	*O(1)*

הערות

- סיבוכיות הפעולות על תור הממומש באמצעות שרשרת חוליות, מתייחסת לייצוג של תור כרשומה/מבנה בעל/ת שני שדות: front (קדמי) המצביע על האיבר שנמצא בראש התור, ו- rear (אחורי) המצביע על האיבר האחרון בתור.
- סיבוכיות הפעולה **הוצא-מתור** בתור הממומש באמצעות מערך, זהה לסיבוכיות ההוצאה מרשימה הממומשת באותו אופן. תחילה יוּצָא האיבר שבראש התור/מערך, ולאחר מכן יוזזו שאר האיברים מקום אחד לכיוון תחילת המערך.

שאלות

שאלה 7.1.1

כיצד תשתנה סיבוכיות הפעולות על תור לו מימשנו התור באמצעות מחסנית?
זכור! פעולת הכנסה של איבר לתור נעשית בסוף התור, ואילו פעולה של הוצאת איבר נעשית מראש התור.

■

שאלה 7.1.2

כיצד תשתנה סיבוכיות הפעולות על תור לו מימשנו התור באמצעות רשימה?

■

7.2 - הערכת סיבוכיות

הערכת סיבוכיות תבנית 4.1 בנה-רשימה עבור רשימה הממומשת באמצעות מערך:

מימוש מערך	**בנה-רשימה**
$O(1)$	אתחל-רשימה ← L
$O(1)$	סוכן-רשימה (L) ← p
n *	כל עוד לא סוף-הסדרה בצע
$O(1)$	העתק הבא בסדרה ← x
$O(n)$	הכנס-לרשימה (L, p, x)
$O(1)$	סוקב-ברשימה (L, p) ← p
$O(1)$	החזר את L

לכאורה נראה כי פונקצית זמן הריצה של הפעולה היא: $f(n) = n^2 + 2n + 3$ וסיבוכיותה: $O(n^2)$, והיא נגזרת מהסיבוכיות הגבוהה של הפעולה **הכנס-לרשימה**.

התבוננות מעמיקה באלגוריתם, מראה שההכנסה לרשימה מתבצעת תמיד בסוף המערך, פעולה שאינה כרוכה בהזזת איברים ופינוי מקום לאיבר החדש. מכאן שסיבוכיות ההכנסה לרשימה בפעולה זו היא $O(1)$ בלבד, ולכן פונקצית זמן הריצה של **בנה-רשימה** היא: $f(n) = 3n + 3$ וסיבוכיות הפעולה כולה $O(n)$.

הערכת סיבוכיות של הפעולה מחק-רשימה(L) עבור רשימה הממומשת במימוש דינאמי של שרשרת חוליות:

מימוש דינאמי	**מחק-רשימה (L)**
$O(1)$	p ← (L) **שלוף-פשיאה**
$O(n)$ | $*n$	**כל זמן** p ≠ **שלוף-פשיאה**(L) **בצע**
	(L, p) **הוצא-מפשיאה**
$O(1)$	**החזר את** L

לכאורה נראה כי פונקצית זמן הריצה של הפעולה היא: $f(n) = n^2 + 2$ וסיבוכיותה: $O(n^2)$, והיא נגזרת מהסיבוכיות הגבוהה של הפעולה **הוצא-מרשימה**, המכילה בתוכה את הפעולה **קודם-ברשימה** שסיבוכיותה $O(n)$.

התבוננות מעמיקה באלגוריתם, מראה שפעולת ההוצאה מתבצעת תמיד בתחילת הרשימה. p מצביע לאורך כל האלגוריתם על האיבר הראשון ברשימה, ולכן הפעולה **קודם-ברשימה** היא החזרת מצביע לעוגן הרשימה, פעולה שסיבוכיותה היא $O(1)$. מכאן שסיבוכיות ההחזרה של הקודם ברשימה במקרה זה היא $O(1)$ בלבד, ולכן פונקצית זמן הריצה של הפעולה כולה היא: $f(n) = n + 2$ וסיבוכיותה $O(n)$.

שאלות

שאלה 7.2.1

חשב את סיבוכיות התבניות הבאות בכל אחד משני הייצוגים - ייצוג סטטי של מערך וייצוג דינאמי של שרשרת חוליות. הסבר את אופן החישוב.

א. תבנית 4.1 - **בנה-רשימה**

ב. תבנית 4.2 - **שרשור-רשימות** (L1,L2).

ג. תבנית 4.5 - **מחק-מרשימה** (L).

ד. תבנית 4.6 - **העברות-ברשימה** (L).

■

שאלה 7.2.2

א. פתח אלגוריתם לביצוע הפעולה היפוך-רשימה (L) באופן הבא: עבור כל איבר ברשימה L, יש להוציא את האיבר האחרון ברשימה, ולהכניסו חזרה לרשימה אחרי העוגן.

ב. השווה את סיבוכיות האלגוריתם שכתבת לסיבוכיות התבנית 4.7 **היפוך-רשימה** (L).

■

7.3 - מבנה הנתונים היעיל יותר

בתכנת משרד הבריאות קיים מודול בתי-חולים האוגר בתוכו את הנתונים הבאים על בתי החולים בארץ:
קיימים 100 בתי-חולים המקודדים לפי קוד 1-100, בכל בית חולים יש מחלקות, ולכל מחלקה משוייכים רופאים.

הנחות: רופא אינו יכול להיות רשום ביותר מבית חולים אחד.
רופא אינו יכול להיות משוייך ליותר ממחלקה אחת בבית החולים.

להלן חלק מממשק המודול **בתי-חולים**.

האם- רופא-בבי"ח? (Hosp, HospNum, DocName)	הפעולה מקבלת מאגר בתי חולים HOSP ומחזירה 'אמת' אם הרופא ששמו DocName עובד בבית חולים HospNum, ו'שקר' אחרת. הנחות: HOSP מאותחל, HospNum תקין.
האם-רופא-במחלקה? (Hosp,HospNum,DepName,DocName)	הפעולה מקבלת מאגר בתי חולים HOSP ומחזירה 'אמת' אם הרופא DocName עובד במחלקה DepName בבית חולים HospNum, ו'שקר' אחרת. הנחות: HOSP מאותחל, HospNum ו- DepName תקינים.
אחזר-רשימת-רופאים-לבי"ח (Hosp, HospNum)	הפעולה מקבלת מאגר בתי חולים HOSP ומחזירה את רשימת כל הרופאים העובדים בבית החולים HospNum. הנחות: HOSP מאותחל,HospNum תקין.
אחזר-רשימת-מחלקות (Hosp, HospNum)	הפעולה מקבלת מאגר בתי חולים HOSP ומחזירה את רשימת המחלקות הקיימות בבית החולים HospNum. הנחות: HOSP מאותחל, HospNum תקין.

הצע ייצוג מתאים למבנה הנתונים **בתי-חולים**, תוך התייחסות לפעולות הממשק.

ניתן לייצג את מערכת בתי החולים בדרכים שונות. כדי להעריך מהו הייצוג העדיף, יש לנתח את סיבוכיות הפעולות השונות בכל ייצוג.

להלן שתי דרכי ייצוג, וניתוח סיבוכיות הפעולות בכל אחת מדרכים אלו.

ייצוג 1:

מספר בתי החולים הוא סופי, לכן נבחר לייצג את בתי החולים במערך בגודל 100 שכל אחד מאיבריו הוא מטיפוס **בית-חולים**.

בכל אחד מבתי החולים מספר שונה, שחסמו לא ידוע, של מחלקות, לכן טיפוס הנתונים **בית-חולים** יכיל את השדות הבאים:

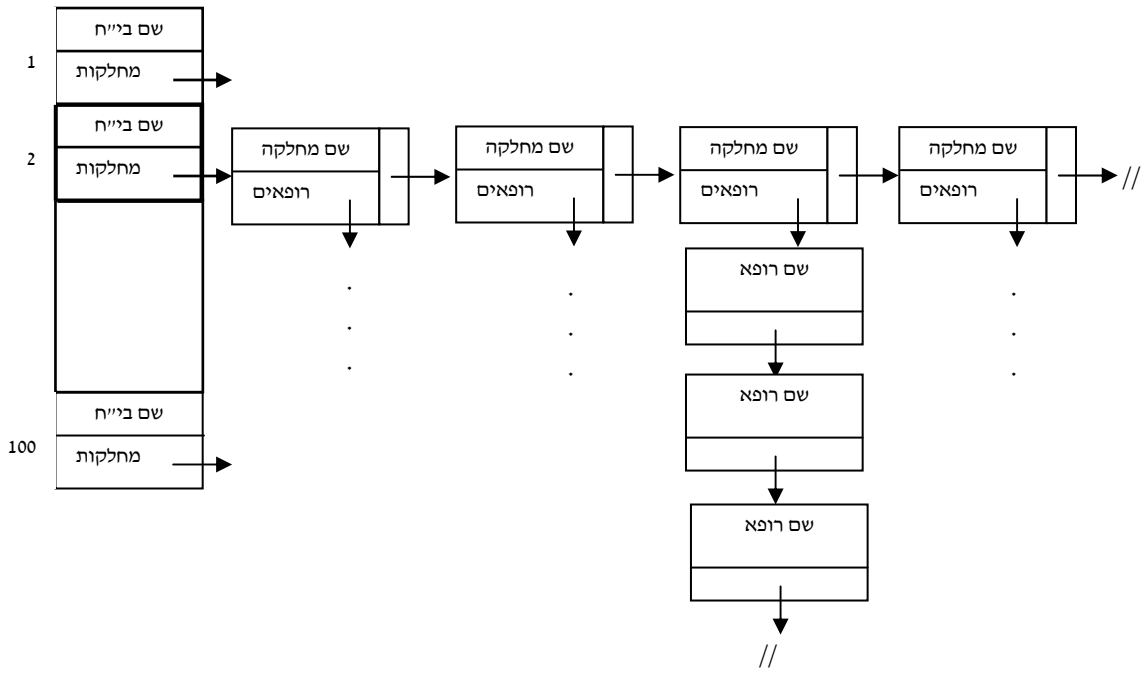
- שם בית החולים (אופציונאלי).
- מחלקות - רשימה שכל אחד מאיבריה הוא מטיפוס **מחלקה**.

טיפוס הנתונים **מחלקה** יכיל את השדות הבאים:

- שם-מחלקה DepName
- רופאים - רשימה שכל אחד מאיבריה הוא מטיפוס **רופא**.
- פרטים נוספים (רשימת עובדים אחרים, רשימת חולים וכד', נתונים שאינם רלוונטיים לשאלה זו).

טיפוס הנתונים **רופא** יכיל את השדות הבאים:

- שם-הרופא DocName
- פרטים נוספים (מספר רישיון, התמחות, מצב משפחתי וכדומה, נתונים שאינם רלוונטיים לשאלה זו).



ייצוג 2:
מספר בתי החולים הוא סופי, לכן נייצג את בתי החולים במערך בגודל 100 שכל אחד מאיבריו הוא מטיפוס **בית-חולים**.

בכל אחד מבתי החולים מספר שונה של מחלקות ומספר שונה של רופאים, לכן טיפוס הנתונים **בית-חולים** יכיל את השדות הבאים:

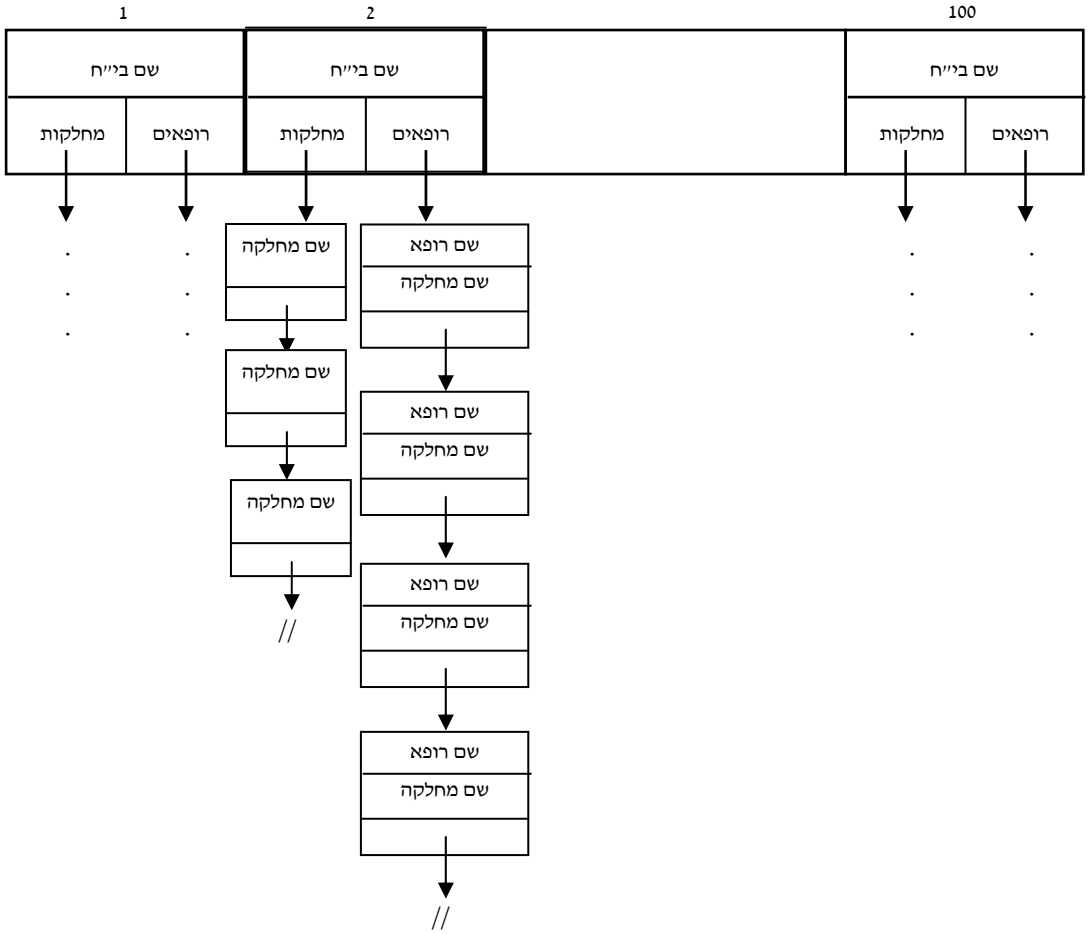
- שם בית החולים (אופציונאלי).
- מחלקות - רשימה שכל אחד מאיבריה הוא מטיפוס **מחלקה**.
- רופאים - רשימה שכל אחד מאיבריה הוא מטיפוס **רופא**.

טיפוס הנתונים **מחלקה** יכיל את השדות הבאים:

- שם-מחלקה DepName
- פרטים נוספים (נתונים שאינם רלוונטיים לשאלה זו).

טיפוס הנתונים **רופא** יכיל את השדות הבאים:

- שם הרופא DocName
- שם-מחלקה אליה משוייך
- פרטים נוספים (נתונים שאינם רלוונטיים לשאלה זו).



נבחן את סיבוכיות פעולות הממשק השונות בהנחה כי בכל בית-חולים קיימות n מחלקות ובכל מחלקה עובדים m רופאים, כלומר, בכל בית חולים רשומים $n*m$ רופאים.

האם- רופא-בבי"ח? (Hosp, HospNum, DocName)

ייצוג 1:
כדי לדעת האם הרופא DocName עובד בבית-חולים HospNum, יש לגשת לתא HospNum (פעולה מיידית) ולעבור על n המחלקות השונות, ובכל מחלקה לסרוק את m הרופאים השונים. כלומר, בפעולת הסריקה עוברים, במקרה הגרוע, על כל $n*m$ הרופאים בבית החולים, ובסה"כ סיבוכיות העבודה היא $O(n*m)$.

ייצוג 2:
בייצוג זה, יש לגשת לתא HospNum (פעולה מיידית) ולסרוק את רשימת כל הרופאים בבית החולים, רשימה שאורכה $n*m$, ובסה"כ סיבוכיות העבודה היא $O(n+m)$.

כלומר, לא חשוב באיזה ייצוג נבחר, כמות העבודה המתבצעת תהיה זהה בשני הייצוגים.

האם-רופא-במחלקה? (Hosp, HospNum, DepName, DocName)

ייצוג 1:
הגישה לבית החולים HospNum היא מיידית $O(1)$, ולכן יש לאתר את המחלקה DepName ברשימת n המחלקות, לסרוק את רשימת m הרופאים שבמחלקה ולבדוק האם הרופא DocName נמצא בה. ולכן סיבוכיות העבודה היא $O(n+m)$.

ייצוג 2:
הגישה לבית החולים היא מיידית $O(1)$. עתה יש לסרוק את רשימת כל הרופאים בבית-החולים כדי לאתר את הרופא DocName. אמנם מתבצע חיפוש לינארי ברשימה, אך יש לזכור שהרשימה כוללת את $n*m$ הרופאים, ולכן סיבוכיות העבודה היא $O(n*m)$.

כלומר, לייצוג הראשון עדיפות על פני הייצוג השני שכן סיבוכיותו טובה יותר.

אחזר-רשימת-רופאים-לבי"ח (Hosp, HospNum)

ייצוג 1:
הגישה לבית החולים HospNum היא מיידית $O(1)$. כדי לבנות ולאחזר את רשימת הרופאים בבית החולים, יש לשרשר את רשימת m הרופאים שבכל אחת מ- n המחלקות לרשימה אחת ארוכה. סיבוכיות העבודה היא $O(n*m)$.

ייצוג 2:
הגישה לבית החולים היא מיידית $O(1)$. עתה יש להעתיק את רשימת הרופאים לרשימה המוחזרת.

זוהי אמנם פעולה לינארית, אך כיוון שהרשימה כוללת את כל רופאי בית החולים, הרי שגם בייצוג זה עוברים על $n*m$ רופאים, ולכן סיבוכיות הפעולה היא $O(n*m)$.
בייצוג זה ניתן להחליט שמוחזרת הרשימה המקורית, ואז תהיה סיבוכיות הפעולה $O(1)$.

כלומר, היתרון של האלגוריתם השני נמצא בפשטות העבודה, למרות ששניהם עוברים על אותו מספר איברים.

אחזר-רשימת-מחלקות (Hosp, HospNum)

בשני הייצוגים יש גישה מיידית לבית החולים ואחר-כך סורקים את רשימה של n המחלקות. לכן בשני הייצוגים הסיבוכיות היא $O(n)$ אם מחזירים עותק של הרשימה, או $O(1)$ אם מחזירים מצביע לרשימה המקורית.

בסה"כ נראה שאין הבדלים גדולים בין שני הייצוגים, אם כי לפעולה **האם-רופא-במחלקה** נראה שמתאים יותר ייצוג 1, המקצר את תהליך החיפוש ואילו לפעולה **אחזר-רשימת-רופאים-לבי"ח** מתאים יותר ייצוג 2 המאפשר שימוש באלגוריתם פשוט יותר. להוספת פעולה של העברת רופא ממחלקה אחת לאחרת או מבית חולים אחד לאחר לפעולות הממשק, תהיה השפעה משמעותית על בחירת הייצוג, בגלל העלות הגבוהה של פעולות המחיקה וההוספה ברשימה במימושים השונים של רשימה, ובייצוג של כל אחד מבתי החולים בבעיה הנדונה.

שאלות

שאלה 7.3.1

חשב את הסיבוכיות של כל אחד מהאלגוריתמים הבאים:
ההוראה המרכזית הינה פעולה או קבוצת הוראות שמתבצעות בזמן קבוע *O(1)*.

1.
```
עבור I מ- 1 עד n בצע
    עבור j מ- 1 עד n בצע
        הוראה מרכזית
```

2.
```
עבור I מ- 1 עד n בצע
    j ← 1
    כל עוד j ≤ I בצע
        הוראה מרכזית
        j ← j + 1
```

3.
```
עבור I מ- 1 עד n בצע
    j ← 1
    כל עוד j ≤ n בצע
        הוראה מרכזית
        j ← j * 2
```

4.
```
I ← 1
כל עוד I ≤ n בצע
    עבור j מ- 1 עד I בצע
        הוראה מרכזית
        I ← I * 2
```

■

שאלה 7.3.2

חברת וקטורים ובניו בע"מ מוכרת חבילות תוכנה המבצעות פעולות מוזרות על וקטורים. יום בהיר אחד זומנו ויקטור ומתירצה, שני תוכניתנים מנוסים, למשרדו של המנכ"ל.

על ויקטור הוטלה המשימה הבאה: עליך לקלוט וקטור בעל N איברים, ולהתייחס אליו כאילו היה מורכב מ- N/7 וקטורים בני 7 איברים כל אחד. עליך למיין כל תת-וקטור כזה.

על מתירצה הוטלה משימה אחרת: עליך לקלוט וקטור בעל N איברים, להתייחס אליו כאילו היה מורכב מ- 7 תתי וקטורים שבכל אחד מהם N/7 איברים. עליך למיין כל תת-וקטור כזה.

דוגמא: עבור N = 21

הוקטור המקורי: 3 3 0 5 3 8 6 4 2 1 9 9 3 5 4 6 4 3 7 2 8 ⇦

טיפול ויקטור: 3 3 0 5 3 8 6 , 4 2 1 9 9 3 5 , 4 6 4 3 7 2 8 ⇦

טיפול מתירצה: 3 3 0 , 5 3 8 , 6 4 2 , 1 9 9 , 3 5 4 , 6 4 3 , 7 2 8 ⇦

שניהם מכירים שיטת מיון אחת בלבד והיא שיטת **מיון בועות** ($O(n^2)$) .

א. קבע את הסיבוכיות של הבעיות שהוטלו עליהם.

ב. קבע עבור אילו ערכים של n יהיה הפיתרון של ויקטור יעיל יותר ועבור אילו ערכים של n יהיה הפיתרון של מתירצה יעיל יותר.

■