**ממ"ן 14 – מנמ"א**

**אציין מראש, ראשית כל, תודה רבה על דחיית ההגשה, אין זה מובן מאיליו לקבל כזו הארכה.  
שנית, אני חייב לציין עקב סיטואציה בחיי האישיים, לצערי לא הספקתי באמת להסביר לעומק את אשר נעשה.  
אשתדל לפרט ככל הניתן, עקב מגבלות הזמן.  
בנוסף, הקוד נכתב ב, משמע טבלאות הגיבוב הינן ממוינות מראשיתו של התהליך.  
עוד אציין, כי אני משתמש ב אשר מתוך ספרייה בשם , תפקידו הוא להריץ את הקוד ב שורה אחרי שורה, ובכך מדמה את פעולת הקומפיילר ובכך אמור להגיע למהירויות של שפות אשר נחשבות מהירות ממנה, אך עם זאת, הוא לא עובד בצורה הכי יעילה, ומציג לי שגיאות, לכן, אוסיף שני קבצים, האחד ללא ה והשני איתו. בכדי להימנע משגיאות עשיתי להן .  
בנוסף, אציין כי מגבילה את יכולת השימוש ברקורסיה, דבר שגם אותו שיניתי בהגדרות למעלה.**

ראשית, על פי דרישת הממ"ן, עלינו ליצור מערך שהגדרת גודלו נעשית על ידי המשתמש. אשר מתבקש להכניס את כמות האיברים אשר ברצונו להריץ עליהן אלגוריתם זה, בנוסף, נדרשנו להגריל את המספרים בתחום שבין 1-100.   
אציין בנוסף, כי נקודת ההנחה שלי הינה, שחייב להיכנס קלט שהינו חיובי (משמע ).  
נדרשנו בממ"ן זה לממש את האלגוריתם המקורי מממ"ן 11 שאלה מס'2 סעיף א' ע"י אלגוריתמים אותם למדנו לאורך הסמסטר.

לכל אחד מן האלגוריתמים אשתדל להסביר מה צורת הפתרון וכיצד האלגוריתם מיישם את פעולתו.

1. בשאלה הראשונה התבקשנו להציג את האלגוריתם מממ"ן 11 ולממשו, באלגוריתם זה, מתקבל מערך על ידי הפונקציה, וסדר המערך משתנה, ע"י כך שבכל פעם אשר מתקיימת בדיקה בין זוג איברים במערך במיקומים ה וה- ובמידה מתגלה כי האברים שונים, אז גודל ה יגדל ובכך האלגוריתם יוכל לגלות את כמות האיברים השונים במערך. אלגוריתם זה פועל בתצורת לולאה מקוננת אשר רצה על איברי המערך הן על ידי הלולאה הפנימית והן על ידי הלולאה החיצונית, מכאן שיעילות האלגוריתם הינה .
2. האלגוריתם השני הינו מיון הכנסה, במיון זה, נעבור בלולאה על האיברים, עד אשר האיבר יימצא את מקומו, ראשית הוא ירוץ האלגוריתם על כל איברי המערך, אך בלולאה הפנימית הלולאה תרוץ על כל איבר ובכך איבר עלול לרוץ על כל המערך מתחילתו עד לסופו עד אשר יימצא את מקומו, וסיטואציה זו עלולה לקרות לכל איבר במערך, מכאן, שיעילות האלגוריתם הינה .
3. האלגוריתם השלישי הינו מיון מהיר בעל פיבוט רנדומאלי. אלגוריתם זה הינו בין האלגוריתמים השמישים ביותר היום בתעשייה, אלגוריתם זה על אף שאינו בהכרח היציב מכולם ותלוי הן במזל והן במצביעו, הוא עדיין מהיר מאוד לכל הדעות ונחשב לבין האלגוריתמים הרקורסיביים המהירים ביותר ביחס לאלגוריתמים המוכרים לנו כיום. אציין כי השיקול העיקרי שלי בזמן בחירת האלגוריתם היה בינו לבין , אשר נחשב גם הוא כאלגוריתם יעיל ועל אף שהמקרה הגרוע של ה שווה בזמן פעולתו הממוצע לזמן פעולתו הגרוע ביותר הוא לרוב עדיין איטי מ- אשר גם זמן ריצתו הינו .

האלגוריתם מתחיל עם סמנים משני הקצבות של המערך ומחלק אותו לשני חלקים, האלגוריתם בודק האם האיבר קטן מהפיבוט והאם האיבר השמאלי גדול מהפיבוט, במידה וזה מתקיים, הם יחליפו מקומות בינהם, במידה ולא, כל איבר ייתקדם אל עבר הכיוון הנגדי לו, עד אשר המצביעים של אחד מהצדדים ייפגשו, כאשר זה קורה, המערך יחולק רקורסיבית לחלקים קטנים יותר ובכך יצליח להתמיין. יש לציין כי כאשר המצביעים נפגשים, ייבחר פיבוט חדש בכדי לבדוק מספרים שונים שעלולים להיות קטנים\ גדולים מן הפיבוט. תהליך זה יצליח יפרק את המערך לחלקים קטנים וימינם עד אשר המערך ימוין עד סופו. אלגוריתם זה, נמצא יעיל יותר מאשר אלגוריתם מיון מהיר המקורי, מהסיבה הפשוטה שלא תמיד הפיבוט הטוב ביותר יימצא דווקא במרכז, מכאן שסטטיסטית האלגוריתם יכול להתייעל רבות רק מבחירה רנדומאלית של פיבוט, אלגוריתם זה מצליח לעשות זאת על ידי חלוקה של כל אחד מאיבריו ל2 חלקים קטנים יותר בכל פעם, ועל ידי כך, אלגוריתם זה מצליח להגיע ליעילות של .

1. האלגוריתם הרביעי הינו מערך של מיון מניה, אלגוריתם זה יעיל בתצורה גבוהה מאוד, לכל איבר במערך, ישנו מיקום אשר מייצג אותו, משמע, כל איבר יודע לאן להשתייך. באלגוריתם זה כל איבר המתקבל על ידי המערך, ייספר ויוכנס למקום המתאים לו על פי מערך אחר אשר מייצג את האיברים הנספרים על ידו יעילותו של אלגוריתם זה הינה יעילות גבוהה מכיוון שהיא תלויה אך ורק בסדרי הגודל של המערכים הנתונים לנו. מכאן שיעילות אלגוריתם זה הינה חיבור אורך המערכים משמע: .
2. למימוש האלגוריתם בעזרת טבלאות גיבוב השתמשתי בשני תצורות מימוש. אציין כי האלגוריתם השני אשר מימשתי הינו ע"י מיעון ישיר, בנוסף אציין כי האלגוריתם במיעון ישיר מציג תוצאות טובות הרבה יותר, מכיוון שאנו יודעים כמה איברים אמורים להימצא בטבלת הגיבוב בתצורה המקסימאלית שלה ואנו מקצים מקום לכל אחד מהם מקום מראש (100 איברים) ובכך הפעולה היחידה שעל האלגוריתם לעשות הינה להכניס להגדיל את ייצוגו של כל מפתח בכל איטרציה בה ספרת האחדות מתקיימת.

* אלגוריתם זה הינו טבלת גיבוב אשר משתמשת בשיטת החילוק, ע"י שימוש בטבלת גיבוב שבעת הגדרתה, מוגדרת טבלת הגיבוב ל גיבובים שונים, אשר מייצגים את ספרת האחדות בכל אחד מן איברי המערך, לדוגמא: בספרה , ספרת האחדות הינה וספרת העשרות הינה . טבלת הגיבוב תיוצג לנו בצורה הבאה:

0:{}  
1:{9:1}  
2:{}  
3:{}  
4:{}  
5:{}  
6:{}  
7:{}  
8:{}  
9:{}

כפי שניתן לראות המספר נכנס לטבלת הגיבוב, כך שהוא סופר את כמות הפעמים שמופיעה ספרת העשרות כאשר ספרת האחדות הינה . דוגמא נוספת: נוסיף לטבלת הגיבוב שלנו את המספר אשר ואת המספר נוסיף פעמיים בכדי להציג מצב בו מתקיים מספר פעמיים במערך אותו אנו בוחנים. נציג זאת כך:

0:{10:1}  
1:{9:1}  
2:{9:2}  
3:{}  
4:{}  
5:{}  
6:{}  
7:{}  
8:{}  
9:{}

כפי שניתן לראות, המספרים אשר מיוצגים לנו על ידי טבלת הגיבוב הינם .

*משמע, אנו משתמשים בטבלה זו בכדי לספור את כמות האיברים ובעזרת מצביעים אנו יכולים לראות את הפירוט המלא של האיברים שקיימים לנו בתוך טבלת הגיבוב.  
אלגוריתם זה, מאפשר לנו גישה נוחה מאוד לאיברים הנתונים לנו במערך, אלגוריתם זה של טבלת גיבוב מגיע עד לזמן ריצה של , מכיוון שעליו באיטרציות רבות ליצור הן את ספרת העשרות מתוך המערך וליצור אותן וכל זאת תלוי גודל המערך.  
בנוסף, באלגוריתם זה, עלינו להבין כי קיימות השוואות ובדיקות רבות בכדי להקצות מקום לנתונים אשר אמורים להיות מוזנים אל תוך טבלת הגיבוב. מכאן שגודל זמן הריצה הינו .*

* אלגוריתם זה הינו טבלת גיבוב במיעון ישיר, אלגוריתם זה הינו היעיל ביותר בתצורת שימושו על ידי טבלאות גיבוב מהסיבה הפשוטה כי אנו מודעים לכמות האיברים הסופיים, מכאן, אנחנו יכולים להקציב מיקום עבור כל אחד מן האיברים, בצורה הזו, היעילות הינה היעילות הטובה ביותר שנוכל להגיע לקבל תחת טבלת גיבוב, ניתן לראות כי, לכל איבר במערך, מתקיים איבר אשר סופר את כמות הופעותיו באופן בלעדי של אותו המספר, ומכיוון שידוע לנו כי כמות האיברים הסופית של האיברים הינה 100 איברים, אנו יודעים כי כמות האיברים אשר יוזנו על ידי המשתמש הינו הפרמטר העיקרי, אך לאחר קבלת מספר זה ניתן לומר כי: , זאת מכיוון ואנו יודעים כי מקיים שהאיבר בהכרח איבר סופי, משמע, סדר גודל יעילותו של מספר זה הינה בהכרח .

דוגמאות הרצה:  
יש לשים לב כי בדוגמא בה , הריצה של האלגוריתם השני (( הייתה איטית בצורה שהייתה בלתי אפשרית להרצה בזמן סביר, מכאן שלא הצלחתי להציג ריצה של אלגוריתם זה.

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generated