תרגיל 1

נתונים n מספרים שלמים מהתחום $[0,n^2-1]$. הציעו דרך יעילה למיין אותם.

פתרון

דרך 1: מיון מבוסס השוואות ייקח $\Omega(n\log n)$ זמן.

דרך 2: Counting Sort ייקח $O(n+k)=O(n+n^2)=O(n^2)$ זמן.

דרך 3: Radix Sort כאשר המספרים בבסיס 10 ייקח זמן:

$$O\big(d\cdot(n+b)\big)=O\big((\log_{10}n^2)\cdot(n+10)\big)=O(n\log n)$$

דרך 4: נעביר את כל המספרים לבסיס n.

- לכל מספר בתחום $[0,n^2-1]$ יש לכל היותר 2 ספרות בבסיס n.
- העברת מספר בעל $O(\log n)$ ספרות לבסיס n לוקחת $O(1)$ זמן.

  בפועל – זה לא תמיד נכון (תלוי במעבד ובכמה n גדול).
- סה"כ: $O(n)$.

כעת נפעיל את אלגוריתם Radix Sort. פעולה זו תיקח:

$$O\big(d\cdot(n+b)\big)=O\big(2\cdot(n+n)\big)=O(n)$$

תרגיל 2

נתון מערך בגודל n עם $\log n$ איברים שונים זה מזה.

הציעו אלגוריתם יעיל למיון המערך.

פתרון

אלגוריתם מיון כללי ייקח $\Omega(n\log n)$. נתאר אלגוריתם משופר:

1. נעבור על כל האיברים ונכניס אותם לעץ חיפוש מאוזן (AVL, 2-3). כל צומת בעץ יחזיק את האיבר אותו הוא מייצג ומונה שסופר כמה פעמים איבר זה מופיע במערך.
   - אם האיבר לא קיים בעץ, נוסיף צומת חדש בעץ.
   - אחרת, נגדיל את המונה בצומת שמייצג אותו ב-1.
2. לבסוף, נבצע סיור Inorder על העץ. נדפיס כל איבר בעץ לפי ערך המונה בצומת.

סיבוכיות:

שלב 1 לוקח $O(n\log\log n)$, כיוון שמספר הצמתים בעץ בכל שלב הוא $\log n$ לכל היותר.

שלב 2 לוקח $O(n)$ (הסיור לוקח $O(\log n)$, הדפסת כל האיברים: $O(n)$ סה"כ).

סה"כ: $O(n\log\log n)$ במקרה הגרוע.

פתרון נוסף

נקצה טבלת ערבול בגודל $\log n$.

- בדומה לקודם, נכניס את כל האיברים לטבלת הערבול, כאשר כל תא בטבלה יחזיק מונה.

שלב זה יקח $O(n)$ בממוצע.

- כעת נמיין את הערכים המופיעים בטבלה בזמן $O(\log n \cdot \log\log n)$ במקרה הגרוע.
- לבסוף, נדפיס את הערכים תוך התחשבות במונים: $O(n)$ במקרה הגרוע.

<u>סה"כ:</u> $O(n)$ בממוצע.

פתרון שלישי ואחרון – הטוב משני העולמות

סיבוכיות הזמן של הפתרון הראשון היא $O(n \log\log n)$ במקרה הגרוע.

סיבוכיות הזמן של הפתרון השני היא $O(n)$ בממוצע.

<u>מהי סיבוכיות הזמן של הפתרון השני במקרה הגרוע?</u>

- כל הכנסה לטבלת ערבול במקרה הגרוע לוקחת $O(\log n)$.
- לכן, הפתרון השני במקרה הגרוע ייקח $O(n \log n)$.

אבל, אם נשתמש ב-Chain Hashing, אך במקום רשימות מקושרות נחזיק עצי חיפוש מאוזנים, כל פעולת הכנסה או חיפוש בטבלה תיקח $O(\log\log n)$ במקרה הגרוע.

מכאן, סיבוכיות הזמן של הפתרון האחרון היא $O(n \log\log n)$ במקרה הגרוע, אך $O(n)$ בממוצע. נסיק מכך שהפתרון האחרון טוב יותר.

חסם תחתון על אלגוריתמים

באמצעות המשפט העוסק בחסם תחתון על אלגוריתמי מיון מבוססי השוואות, ניתן למצוא חסמים תחתונים על אלגוריתמים אחרים.
לדוגמא, נוכיח שלא קיים אלגוריתם (מבוסס השוואות) הבונה עץ חיפוש בינארי מתוך מערך לא ממוין בזמן $o(n \log n)$.

כדי להוכיח טענות מסוג זה, נניח בשלילה שקיים אלגוריתם כנ"ל ונגיע לסתירה:

נניח בשלילה שקיים אלגוריתם כנ"ל. נראה שבאמצעות אלגוריתם זה ניתן למיין מערך A כלשהו בעל n איברים בזמן $o(n \log n)$.

- נשתמש באלגוריתם הנתון כדי לבנות עץ חיפוש בינארי המכיל את איברי A.
- כעת, נסרוק את העץ ע"י סיור Inorder ונדפיס את איברי העץ בסדר ממוין.

סה"כ: $o(n \log n) + O(n) = o(n \log n)$.

זו סתירה, כיוון שלא קיים אלגוריתם מבוסס השוואות שממיין מערך כלשהו בזמן $o(n \log n)$.
לכן, לא יתכן שניתן לבנות עץ חיפוש בינארי ממערך לא ממוין בזמן $o(n \log n)$.

תרגיל

יהי A מערך מספרים מגודל $n = 2^k - 1$, כאשר מתוכם רק $\log n$ שונים זה מזה.
נוסף על כך נתון: לכל $0 \le i < k$ קיים איבר במערך שמופיע בדיוק 2^i פעמים.

דוגמה: עבור $k = 3, n = 7$, מערך כזה הוא למשל: $[5,3,8,5,3,5,5]$, בו המספר 8 מופיע פעם אחת, 3 מופיע פעמיים ו-5 מופיע ארבע פעמים.

הראו כיצד ניתן למיין את המערך.

<u>סיבוכיות זמן:</u> $O(n)$

<u>סיבוכיות מקום:</u> $O(n)$