תרגיל 4- מבני נתונים 67109:

שם: דן קורנפלד

חלק א:

Counting Sort:

סעיף א:

1. כן, Counting-Sort הוא מיון יציב. הסבר: מכיוון שלאחר שהאלגוריתם סופר כמה פעמים כל מספר נמצא במערך הנתון, ולאחר מכן בודקים במערך עבור כל מספר "כמה מספרים קטנים או שווים לו", ועוברים על המערך המקורי מהסוף להתחלה, ומציבים את המספרים הללו במערך החדש שנוצר. מסיבה זו, המיון אכן מיון יציב, מכיוון שאם יש מספר כלשהו המופיע מספר פעמים, המופע האחרון שלו (הימני ביותר לצורך הדמיון) יהיה המופע האחרון שלו (הימני ביותר) במערך הממוין, וכך גם בתוך המופעים של מספר נתון (מכיוון שבכל פעם שמופיע המספר, נפחית ב-1 את האינדקס של המופע הבא שיהיה), נשמר סדר ההופעה במערך הממוין, ועל כן Counting-Sort אכן מיון יציב.
2. באופן פורמלי יותר: עבור 2 איברים j_1, j_2 עם ערך זהה כך ש $j_1 < j_2$ מכיוון שהאלגוריתם עובר בלולאה האחרונה מהסוף להתחלה, תחילה האלגוריתם ישים את j_2 ולאחר מכן ימשיך לסרוק עד שיגיע ל-j_1 וישים את j_1 משמאל ל-j_2 ומכאן ש-2 האיברים שומרים על הסדר שלהם, ולכן המיון יציב.

סעיף ב:

1. במידה ונשנה את סדר הלולאה האחרונה, שתעבור על המערך מההתחלה לסוף, ולא מהסוף להתחלה, אכן נקבל מערך ממוין, מכיוון שבכל פעם שיופיע מספר מסוים, האלגוריתם "יבדוק" במערך C היכן המספר צריך להיות מוצב במערך הממוין, יציב אותו במקום המיועד ויוריד את הערך במערך C אשר אחראי על הצבת המספרים ב-1, כלומר עבור הצבת האינדקסים במערך החדש (בפעם הבאה שאותו מספר יתקבל, נציב אותו באינדקס החדש שהצבנו).
   ולכן, מסיבה זו, נקבל שהאלגוריתם עדיין מציב את המופעים במקומות המיועדים לו (כלומר לפי הסדר),אך הוא לא מתחשב היכן המופע נמצא ביחס למערך המקורי, והוא מציב את המופע במערך החדש לפי סדר הריצה של הלולאה האחרונה.
2. מכיוון שאנחנו עוברים על המערך מההתחלה אל הסוף, ולא מהסוף להתחלה, אך אנחנו מציבים את המופעים בין המספרים הזהים מהסוף להתחלה, נקבל שהמיון לא יהיה מיון יציב, מכיוון שאם נסתכל על מספר ספציפי (לדוגמה x) מתוך המערך המקורי, במצב בו הלולאה עוברת מההתחלה אל הסוף, המופע הראשון של אותו מספר x יוצב במקום האחרון מבין כל המופעים של x.
   מהגדרת מיון יציב: לכל x, y אם $x = y$, וx קודם ל-y במערך המקורי, אזי x קודם ל-y גם במערך הסופי הממוין, ולכן שינוי סדר הסריקה של המערך האחרון יגרום למיון לא להיות יציב.

סעיף ג:

- נשאב השראה מאלגוריתם ה-Counting-Sort. עבור העיבוד המקדים של הקלט, התחיל ליישם את האלגוריתם המיון, עד לפני השלב שבוא מציבים את הערכים מהמערך המקורי, למערך החדש.
  כלומר: בסיבוכיות זמן ריצה של $O(n + k)$ ניצור מערך C המכיל לכל $0 \leqslant i \leqslant k$ את מס האיברים שקטנים\ שווים לו במערך המקורי.
- כך, עבור כל 2 ערכים a, b ניתן לגשת למקום ה-$C[a-1]$ כדי לקבל את מספר כל האיברים שקטנים ממש מ- a, וניגש ל- $C[b]$ כדי לקבל את מספר כל האיברים שקטנים או שווים ל-b, ובכך לחסר בין המספרים, כלומר: $C[b] - C[a-1]$. מכיוון שאנו ניגשים לאיברים ספציפיים במערך שלא תלוי בגודלו, פעולה זו היא בסיבוכיות של $O(1)$ כמבוקש.

תרגיל 4- מבני נתונים 67109:

שם: דן קורנפלד

חלק ב:

Counting Sort:

סעיף א:

- לא, דוגמה נגדית: ניקח את המערך $[8, 7, 9, 9, 9]$. נבחר את האיבר הימני ביותר להיות ה-Pivot, ונבדוק עבור כל שאר האיברים אם הוא קטן או שווה ל-Pivot (במקרה הזה, ל-8) או גדול ממנו. נבדוק מהאיבר השמאלי ביותר ונראה ש-$8 > 9$, נרוץ בלולאה עד האינדקס ש-7 נמצא, ונחליף בניהם כך שיתקבל:$[8, 9, 9, 9, 7]$, נמקם את ה-Pivot בין 2 הקבוצות שמצד אחד קטנה או שווה ל-Pivot, ומצד שני גדולה ממנו, ולכן יתקבל $[9, 9, 9, 8, 7]$
- נשים לב כי המערך ממוין אך סדר ההופעות של המספר 9 לא באותו הסדר, ולכן המיון לא יציב.

סעיף ב:

- נסמן את גובה העץ ב-h. מכיוון שהמערך מתחלק ל-α ול- $1-\alpha$, הענף הקצר ביותר יהיה $n \cdot \alpha^h \underset{\bigstar}{=} 1$.
- אלגברה:

$$n \cdot \alpha^h = 1$$
$$\alpha^h = \frac{1}{h}$$
$$h = log_\alpha\left(\frac{1}{h}\right)$$
$$h = log_\alpha(1)^{=0} - log_\alpha(n)$$
$$h = -log_\alpha(n) = -\frac{log(n)}{log(\alpha)} \Longrightarrow \theta\left(-\frac{log(n)}{log(\alpha)}\right)$$

- נשים לב: מדוע יש מינוס? הרי אנחנו רגילים למספרים חיוביים: $1 > \alpha = \frac{p}{q},\ p < q$
  לכן: $log(\alpha) = log\left(\frac{p}{q}\right) = log(p) - log(q) < 0$ ולכן ה-מינוסים מתבטלים.
- עבור הענף הארוך ביותר: באותו אופן במקום α, נציב $1-\alpha$ ונקבל $\theta\left(-\frac{log(n)}{log(1-\alpha)}\right)$ כמבוקש

• $\bigstar$ - n מס האיברים במערך, 1 העלה

תרגיל 4- מבני נתונים 67109:

שם: דן קורנפלד

חלק ג:

שמורת הלולאה ו- $Radix\ Sort\ +\ Bucket\ Sort$:

סעיף א:

1. הוכחה באמצעות אינדוקציה את הנכונות המיון:
נוכיח באינדוקציה את נכונות המיון כך ש n יהיה מספר הספרות המקסימלי במיון.
בסיס: עבור $n = 1$: אכן מיון פנימי כמו $Counting\ sort$ בסיבוכיות זמן ריצה של $O(n)$ ימיין את המערך כראוי.
הנחה: נניח שהמיון מתקיים בצורה תקינה עבור מספרים עם אורך מקסימלי $n-1$ ונוכיח את הטענה עבור n.
שלב: עבור 2 מספרים x_i, y_i כך שהמספר x לפני y
   (a) אם $x_i = y_i$ סדר האיברים x, y לא ישתנה, כלומר ישאר לפי הסדר המקורי, מכיוון שעד השלב הזה, x היה קטן יותר (קודם במערך) ולכן לא נשנה את מקומו ביחס ל-y. **אם x ו-y שווים, לכל אורך הדרך x יהיה לפני y ולכן המיון ישמור על המיקום היחסי בין 2 המספרים
   (b) אם $x_i > y_i$, נחליף בין המיקומים של x ו- y, ואכן נרצה לבצע זאת בגלל שאחד גדול מהשני.
   (c) אם $x_i < y_i$ לא נחליף את הסדר, שכן נרצה שהאלגוריתם ישמור על כך ש x קטן מ y.
ולכן האלגוריתם אכן ממיין כנדרש
2. היכן בהוכחה אתם משתמשים בטענה שהמיון על כל ספרה הוא יציב?: במיון הפנימי האלגוריתם דואג להשאיר את סדר האיברים לפי סדר הופעתם (כלומר יציב) בכל איטרציה חיצונית.
3. דוגמה עבור שימוש במיון לא יציב ששולל את נכונות $Radix\ Sort$:

$$\begin{bmatrix} 7 & 6 \\ 7 & 3 \\ 7 & 5 \\ 8 & 5 \\ 8 & 6 \\ 8 & 5 \end{bmatrix} \underset{\star}{\Longleftarrow} \begin{bmatrix} 7 & 3 \\ 8 & 5 \\ 8 & 5 \\ 7 & 5 \\ 8 & 6 \\ 7 & 6 \end{bmatrix} \underset{\bigstar}{\Longleftarrow} \begin{bmatrix} 7 & 5 \\ 8 & 5 \\ 7 & 6 \\ 8 & 5 \\ 7 & 3 \\ 8 & 6 \end{bmatrix}$$

- ★ - נמיין לפי עמודת האחדות באופן לא יציב
- ✶- נמיין לפי עמודת העשרות באופן לא יציב

אכן קיבלנו מערך לא ממוין, לכן חשוב לדאוג שהמיון הפנימי יהיה יציב, כדי ש-$Radix\ Sort$ ימיין בצורה תקינה.

סעיף ב:

1. אכן חשוב לוודא שאף אחת מהצלחות לא נשארת מקופחת ☺ , לכן נוכיח את הנכונות באמצעות שמורת הלולאה שלאחר n צלחות יש שוויון בין הצלחות.
2. שמורת לולאה: בתחום הלולאה ה-i, נעשה שימוש ב-i צלחות העליונות, ולא נעשה שימוש ב-$n-i$ צלחות התחתונות.
3. אתחול: תחילה עבור האיטרציה הראשונה, אחרי שימוש בצלחת 1, אכן השתמשנו ב-"1" צלחות

בצורה שווה.

4. שימור: נניח שהטענה נכונה לפני איטרציה כלשהי לאחר $n-1$ איטרציות, לפני האיטרציה הבאה ה-n כלומר שבכל איטרציה ניקח את הצלחת התחתונה ביותר, נאכל איתה ונכניס אותה ללמעלה.
5. סיום: באיטרציה האחרונה השתמשנו ב-n צלחות, ומהשימור אכן נראה שב-n ארוחות השתמשנו ב-n צלחות וכך נראה שאף צלחת לא קופחה, כמבוקש.

סעיף ג:

נוכיח באמצעות שמורת לולאה את נכונות $Binary-Search$.

שמורת לולאה: בתחילת האיטרציה ה-i, נצמצם את גבולות המערך לגודל $\frac{n}{2^{i-1}}$ מהגודל המקורי, במידה והאיבר שאנו מחפשים נמצא, הוא נמצא בגבולות הללו.

1. אתחול: לפני תחילת האיטרציה הראשונה, גודל המערך יהיה n. במידה והאיבר שאנו מחפשים נמצא, הוא יהיה בגבולות של המערך.
2. שימור: נניח שבתחילת האיטרציה ה-j, נצמצם את גבולות המערך לגודל $\frac{n}{2^{j-1}}$ מהגודל המקורי, במידה והאיבר שאנו מחפשים נמצא, הוא נמצא בגבולות הללו. באיטרציה ה-j נבדוק אם האיבר שאנו מחפשים הוא האיבר האמצעי, אם כן נסיים את הלולאה, אם לא, נצמצם את הגבולות כך שאם האיבר האמצעי גדול מהאיבר שאנו מחפשים, נגדיר את הגבולות החדשים להיות מ $[low, mid]$, אחרת נגדיר $[mid+1, up]$ וכך המערך מצטמצם ב-$\frac{1}{2}$.
3. סיום: בסוף באיטרציה האחרונה או שנגיע לאיבר במערך שאותו חיפשנו ובכך נחזיר "אמת", או שנגיע לאיבר האחרון לפי אלגוריתם החיפוש (כלומר ה-$low > up$), הגענו למערך ריק, לכן האיבר לא נמצא במערך.

מהסיום, הראנו שאם האיבר קיים, מצאנו אותו, ולכן האלגוריתם נכון.

סעיף ד בעמוד הבא

סעיף ד:

1. נסרוק את כל המספרים הנתונים ונבדוק אם הם שווים ל$1 = 2^0$. אם כן, נציב את המספר ישירות במערך הממוין החדש (לפי סדר ההופעה של המספרים). אם לא, נבצע על כל אחד מהמספרים הללו $log_2(x)$, כך שלאחר ביצוע ה-log, לכל מספר מהצורה 2^a נקבל: $log_2\left(2^a\right) = a$. לאחר שנבצע לכל אחד מהמספרים את ה-$log_2(x)$ נקבל שכל המספרים הם בין $k - 1$. נחלק כל מספר ב-$k + 1$ ($\theta\left(n^2\right)$, כלומר החזקה הגבוהה ביותר ועוד 1, כך נקבל שכל המספרים הנדגמים הם בקטע $[0, 1)$, עבור המספרים הללו נבצע $Bucket\ Sort$, כך שמספר ה-$Buckets$ יהיה n (✪התייחסות בהמשך). מותר להשתמש במיון זה מכיוון שנתון שהמספרים הנתונים נדגמים באופן אחיד ובין $[0, 1)$. בכל $Buckets$ יהיה מספר קבוע של מספרים (לפי ההתפלגות האחידה), ולכן בכל מיון פנימי של ה $Buckets$ הסיבוכיות זמן תהיה $O(1)$. לאחר ה- $Bucket\ Sort$ נעבור על המערך הממוין, נכפול כל איבר חזרה ב $k + 1$, ולאחר מכן נשנה את ערך האיבר להיות 2^x, כאשר x הוא המספר לאחר ההכפלה ב-$k + 1$.

✪- מספר ה-$Buckets$ תלוי במספר המספרים שנתונים, ולא ב-k, כך שאנו עוברים על כל $Buckets$ ומכניסים את המספרים בו למערך הממוין, אנו בסופו של דבר עוברים על n $Buckets$, כלומר $O(n)$.

כל $Buckets$ יהיה מהצורה $\left[\frac{i}{n}, \frac{i+1}{n}\right) \quad , \quad 0 \leqslant i < i+1 \leqslant n$

2. הסבר על הסיבוכיות:
   (a) מעבר על כל המספרים הנתונים, בדיקה אם הם שווים ל-1 (אם כן, מציבים אותם במערך החדש), אם לא מבצעים עליהם $log_2(x)$ הוא $O(n)$
   (b) ביצוע המיון הפנימי של כל $Bucket$ כאמור הוא $O(1)$,ולאחר מכן שעוברים ומשימים כל $Buckets$ במערך החדש סיבוכיות הזמן תהיה $O(n)$, ולכן סיבוכיות הזמן של $Bucket\ Sort$ הוא בזמן ממוצע $O(n)$.
   (c) כמו כן, מעבר על כל המערך לאחר $Bucket\ Sort$ וביצוע הפעולות המתמטיות הנ"ל כדי להחזירן לערכן המקורי הוא $O(n)$
   (d) לכן ביצוע של 3 הפעולות הללו יגרום למיון בסיבוכיות זמן ממוצע של $O(n)$ כמבוקש.