שאלה 1 סעיף ג':

כאשר קיימת הגבלה מערכתית על זמן חיפוש ב-FIND, נסמן את מספר החיפושים בטבלת הפרפיקסים באורך עד 32 ביטים ב-x.

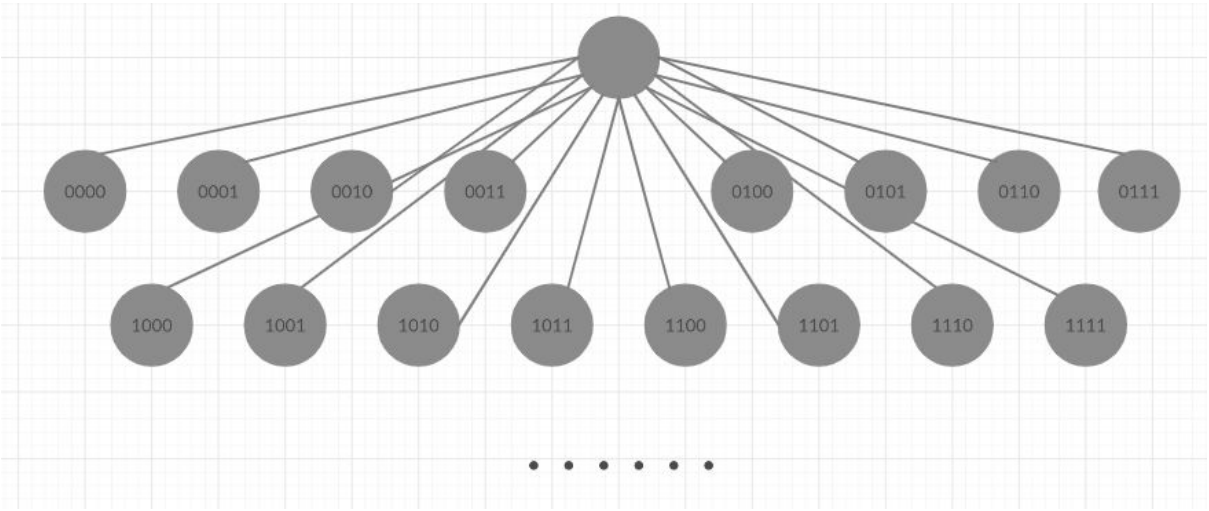
נחלק לשני מקרים:

1.כאשר x זוגי, נעשה את החישוב הבא: $32/x$ התוצאה תהיה מספר הסיביות שיש בכל קודקוד.

2.כאשר x אי-זוגי נעשה את החישוב הבא: $32/x$ עם ערך עליון והתוצאה תהיה מספר הסיביות המקסימלי שיש בכל קודקוד. (יהיו קודקודים השונים בכמות הסיביות מאחיהם כי מספר זוגי לא מתחלק באופן שווה למספר אי-זוגי.)

מספר הקודקודים שיהיו בכל רמה: $2^{32/x}$ (החזקה תהיה בערך עליון כאשר ה-x הוא אי-זוגי) והיו x רמות, שזהו מספר החיפושים המבוקש.

דוגמא עם 8 חיפושים כמו שיש בדוגמא במטלה: $32/8 = 4$ ולכן יהיו לנו 4 סיביות בכל קודקוד,
בכל רמה יהיו לנו $2^4 = 16$ קודקודים (נסמן את הסיביות של הקודקוד כמספר הקסדצימלי של הפרפיקס,
למשל: קודקוד 0 יהיה 0000, קודקוד 15 יהיה 1111 , במקום שמאל וימין שיש לנו עכשיו) ויהיו 8 רמות.
ציור לדוגמא לרמה הראשונה:
מכל קודקוד יצאו 16 בנים שהן כל האופציות של הסיביות הקיימות,
וכך נגיע ל-4 סיביות כפול 8 רמות = 32 סיביות סה"כ.



כך נגביל את כמות החיפושים ל-x חיפושים.
אפשר לעשות אופטימיזציה הדומה לסעיף ב, אך לא לסיבית אחת אחרונה אלא לארבע סיביות אחרונות, זאת אומרת שאם נרצה לצמצם את העץ, נצטרך שלצומת פנימית יהיו את כל הפרמוטציות האפשריות להשלמה ל-32 ביטים (לכל היותר) עם אות אחרונה (למשל A) זהה לכולם, ורק אז נוכל "לעלות" רמה ולהעביר את הסיומת רמה אחת מעל הרמה שהיינו בה והחיפוש יקטן ברמה אחת.
כך זה נראה בדוגמא שלנו:

X 0 0 0 0 A
X 0 0 0 1 A
....
X 1 1 1 1 A

שאלה 1 סעיף ד':

-נדבר על מערך סטטי, מערך דינאמי ורשימה מקושרת, על היתרונות והחסרונות שלהם.

ישנה דרך שבה נוכל להגיע לסיבוכיות שהינה $O(1)$ בכל 3 הפונקציות ADD, REMOVE ו-FIND, אולם פתרון זה בזבזני מבחינת מקום, אך נוכל לעשות לו אופטימיזציה קטנה שתחסוך קצת מקום.
על מנת להגיע לסיבוכיות זו, ניצור **מערך סטטי** בגודל 2^{32}.
כאשר האינדקס ה-i במערך מייצג את המספר i בבינארית בגודל 32 סיביות, ולהפך, כלומר האינדקס ה-0 בטבלה, ייצג את 0000...0 סה"כ 32 פעמים בבינארית.

אתחול: נבנה מערך בגודל 2^{32} כאשר כל התאים "ריקים" כלומר NULL.

הכנסה: בהינתן X ו-Y, כאשר X זה כתובת בינארית ו-Y זה הפריקס, אז נקבע כי המערך במיקום ה-X בדיצמלי יהיה שווה ל-Y.

מחיקה: בהינתן X ו-Y, כאשר X זה כתובת בינארית ו-Y זה הפירפקס, אז נקבע כי המערך במיקום ה-X בדצימלי יהיה שווה ל-NULL.

חיפוש: בהינתן X ו-Y, כאשר X זה כתובת בינארית ו-Y זה הפירפקס, אז נחזיר את הכתובת הבינארית ואת הפריפקס במידה והתא ה-X בדיצמלי במערך לא שווה ל-NULL.

קיבלנו כי סיבוכיות המקום הינה 2^{32} (על יצירת המערך) אבל על מנת לבצע את הפעולות אז הסיבוכיות היא סה"כ קבוע, כאשר הקבוע הוא הפעולת השמה, והפעולת המרה של המספר הבינארי לאינדקס המתאים.

יתרונות:
1. יעיל מבחינת זמן ריצה עבור כל הפעולות.
2. קל למימוש.

חסרונות:
1. לא יעיל מבחינת זיכרון.
2. לא ישים פרקטית בגלל גודל המערך (תלוי בחומרה).

אופטימיזציה:
הבעיה שנוצרה לנו, זה בעצם ה"טרייד אוף" שקיים בין זמן חיפוש לבין זיכרון מוקצה.
אם נשתמש בכמות מקום גדולה אז נוכל להגיע לזמן חיפוש נמוך יותר, ניתן בעצם לבצע כמה אופטימיזציות שחוסכות מקום של זיכרון, אך פוגעות בזמן החיפוש.

אפשרות א:
ניצור במקום מערך סטטי בגודל 2^{32} , ניצור **מערך דינמי** בגודל 0 התחלתי.
כאשר תכנס כתובת בינארית X כלשהי, אז נחשב את הכתובת בדצימלי, ונעשה resize למערך לגודל שקבלנו במידת הצורך (אם גודל המערך כרגע קטן מהכתובת הדצימלית).
קיבלנו כי הזיכרון המוקצה שלנו אפשרי להיות קטן יותר משמעותית, לדוגמא אם הכתובת המקסימלית היא 2^{10} אז נצטרך לבנות מערך משמעותית קטן יותר בגודל 2^{10} ולא 2^{32} כמו שהיה בשיטה הקודמת.
אבל כל כתובת גדולה שנכנסת, נצטרך לבצע העתקה של המערך הישן כדי לשמור את המידע (הפריפקסים) אשר קיימים כבר, ולכן הסיבוכיות זמן ריצה נפגשת בפעולת ההכנסה, (חיפוש ומחיקה לא משתנה).

אפשרות ב:

ליצור **רשימה מקושרת**, כאשר כל איבר ברשימה מכיל כתובת בינארית ופריפקס, באופן זה כאשר נבצע הכנסה אז האלגוריתם יהיה פשוט להכניס לראש הרשימה כתובת בינארית ופרפיקס מתאים, כאשר נרצה לעשות חיפוש אז נצטרך לעבור איבר איבר ברשימה המקושרת ולבצע השוואות, כנל לגבי מחיקה כאשר פעולת המחיקה לאחר החיפוש הינה קבועה (למחוק מצביע).

ולכן קיבלנו סיבוכיות של הכנסה $O(1)$, חיפוש ב- $O(n)$ ומחיקה ב- $O(n)$.

היתרון בשיטה זו, שאם מספר הכתובות קטן (כלומר הקלט הוא קובץ קטן) אז עדיף לעבור אחד אחד מאשר ליצור מערך גדול, לדוגמא נשקול את הקלט:

קובץ עם 2 שורות, אחת זה ADD עם כתובת בגודל 32 ביטים והשנייה היא FIND של הכתובת הראשונה, אז בשיטה הראשונה וגם כן באופטימיזציה נצטרך לשמור מערך בגודל 2^{32} , אולם ברשימה המקושרת נחזיק רק איבר יחיד עם כתובת בינארית ופרפיקס.