דגשים**: כל הפתרונות הן של סטודנטים מהקורס, מהמגה ומהדרייב, לא מתחייב על נכונות התשובות בצורה מלאה.** תודה רבה לכל מי שלקח חלק בדבר, אופיר, בתאל, לירן, עמרי ב', מאי, דניס, גל, אלי אם שכחתי מישהו אני מצטער, שיהיה לכולם בהצלחה! **לא לשכוח לא להשתמש בזכוכית מגדלת, הכתב תקין לגמרי ועבר בדיקות של הדפסה, אם יש בעיה וודאו את ההגדרות!**   
  
נוסחאות נסיגה – א' ועד י' (כולל חסמים ופונק')

לאחר מכן שילוב של אלג' ומבני נתונים.

1. – נובע משיקולי חישוב גבולות. בכל קריאה רקורסיבית נעשות לפחות פעולות. לכן חסם תחתון לנסוחת הנסיגה. נחפש חסם עליון. ננחש כי הוא . (נשתמש ב-3 כבסיס לפונקציה lgn. זה לא משנה מבחינה אסימפטוטית).

נניח כי: **לכן** – *נבחר* c=3 *ונקבל:*

1. *תהאנה פונ' עולות הוכח/הפרך: 1. אם אז או 2. אם אז* ***פתרון1:*** *הטענה לא נכונה, נסתור: עולה, עולה (כי אם לא אז קיים n0 כך שי g(n0+1)>g(n0) נניח כי n0 זוגי (אם אי זוגי אז עולה טרוו') אז:* ***לכן:******לכן*** *וזה מתקיים רק עם וזה לא בתחום לכן עולה(* ***כמו כן:*** *עולה.  
   ואז מתקיים:* ***וגם******מתקיים:*** *ולכן אין גבול ולכן וגם****פתרון2****: טריוויאלי: לכן כמו כן: נתון עבור : אם אזי לכן עבור קיים c2 כך ש לכן . עבור אזי* ***לכן*** *עבור קיים כך ש וגם* ***לכן******לכן******לכן*** *גם וגם ולכן . הטענה נכונה.*
2. *נתון בטבעיים,* ***1.*** *אם אז* ***פתרון1****: נגדיר , , מתקיים עבור כמו כן לכן לא נכון.* ***2.*** *אם אז* ***פתרון2:*** *אם לכן עפ הגדרה קיים כך שלכל מתקיים מכאן ש עולה, קיים מתקיים ניקח לכן לכל מתקיים ולכן ע"פ ההנחה בהתחלה מתקיים .*
3. **ולכן** כמו כן **לכן** .
4. יש בטבעיים **1.** או **פתרון1** נגדיר ו - נוכיח שעולות: מקרה 1 n זוגי לכן וגם מקרה 2 n אי זוגי לכן וגם  
    לכן הפונקציות עולות. נפריך את הטענה, יהיו c ו n0 חיוביים ונבחר אז וגם זוגי ואז **ולכן מתקיים** ש כמו גן ולכן ונכפול ב ולכן כלומר ולכן הפרכנו נוכיח בצורה דומה מהצד השני לגבי וסיימנו. **2.** **פתרון2** הטענה נכונה. נתבונן ב יהי אז מתקיים מהנתונים ש ולכן מתקיימת הטענה.
5. , נגדיר ולכן ואז נגדיר ואז לפי שיטת האב כלומר כאשר נשתמש בשיטת האב המורחב
6. יש בטבעיים הוכח הפרך: **1.** אם אז **פתרון1** דוגמה נגדית: נראה ש : לפי ההגדרה קיים c חיובי כך שלכל n0>n מתקיים  
   הדבר הבא: וכן לכל . נראה כי התוצאה הרצויה לא מתקיימת: לפי הגדרה - קיים c חיובי כך שלכל n0>n מתקיים: ואילו לכל c חיובי (מדובר בקבוע) לא יתקיים שוויון זה ובטח לא ל שכן: לפי הגדרת הגבול לכן הכל מ n מסוים הצד השמאלי יגדל לאין שיעור מאשר הצד הימני.  
   **2.** אם אז **פתרון2** לפי הגדרה שוב לכן קיים c חיובי כך שלכל n0>n מתקיים . נתון שהפונקציות חיוביות עולות וכמו כן גם החל מ ולכן נבצע לוגריתם על 2 חלקי המשוואה: מכך מבינים גם כי log c קבוע ובכלל הוספה של ערך קבוע לא משפיעה על חסם אסימפטוטי בכל מקרה נוכל לטעון כי והרי שעבור ו לדוגמה מתקיים לפי הגדרה : כנדרש.
7. לכן ולכן וגם ובכלליות ואז *לפי נוסחה ידוע: ולכן הפתרון :*
8. *נתבונן בנוסחת הנסיגה נוכל להסיק כי הביטוי נחלק את כל המשוואה ב : נגדיר ולכן   
   לכן: ואז נגדיר ולכן לכן: ולכן: כעת נפתור בשיטת האב:  
    ולכן: ואז מתקיים: מכיוון שזהו החסם ההדוק של אז נכפילו ב על מנת לקבל את החסם של ולכן:  
   החסם ההדוק הוא:*
9. *A מערך ובו n תאים (n זוגי) עם ערכים טבעיים מ 1 עד n, תת המערך השמאלי עד n/2 מכיל את האיברים האי-זוגיים והימני את הזוגיים, כל אחד מתתי המעריך הללו ממוין בעצמו, לדוגמה: 1,3,5,2,4,6.***1.** מה סיבוכיות מיון מהיר על מערך כזה? **ת:** (להדגים על המערך הנתון לדוגמה) ניתן לראות שהפעם הראשונה שמתבצעת חלוקה עם שגרת החלוקה, הקלט מתחלק לשני מערכים אחד בגודל 1 והשני בגודל n-1, לאחר מכן החלוקה היא למערך של 2 לעומת n-2 איברים וכן הלאה – החלוקה לא מאוזנת ולכן נוסחת הנסיגה המתקבלת היא ולכן סה"כ זמן הריצה במקרה זה הוא לפי שיטת האיטרציה. **2.** כמה החלפות בין איברים מבצע מיון הכנסה כאשר הוא מורץ על קלט כמו A? **ת:** ראשית נבחין כי בהתחלה האיברים האי-זוגיים ממוינים ויש לעשות הכנסה של האיברים הזוגיים למקומות הנכונים שלהם. נגדיר כי החלפה מתבצעת בשורות 6 ו 8 של הקוד, כל איבר יכנס למיקום הנכון שלו ועל מנת לבצע זו יש לבצע החלפות עד שיוכנס למיקום הנ"ל, לדוגמה, בקלט הניתן 2 גרום לביצוע של 2 החלפות ולאחר מכן העתקה של 2 מהמפתח החיצוני חזרה למיקום הנכון שלו לכן 3 העתקות, לאחר מכן עוד העתקה של 4 למיקומו הנכון שגורם להחלפה אחת והכנסה של 4 למקום הנכון ולכן 2 החלפות. סה"כ נבחין שהיו 5 החלפות. לבסוף נבחין כי כל איבר זוגי יעשה מספר החלפות זהה להפרש האינדקסים בינו לבין המיקום הנכון שלו ועוד אחד לכן הסכום הכולל יהיה וזהו מספר ההחלפות הכולל לקלט בגודל n. **3.** בונים עח"ב (לא עא"ש) ע"י הכנסה של האיברים של A לעץ ריק בזה אחר זה לפי הסדר ב A כמה פניות ימינה וכמה שמאלה מתבצעות בבניית העץ? **ת:** נבחין כי כל ה n/2 איברים הראשונים ילכו תמיד ימינה (ממוינים בסדר עולה) מלבד השורש ולכן יש פניות ימינה עד כה, לאחר מכן כל איבר מ n/2 ועד n-1 יעשו n/2 פניות ימינה (כדי להגיע לאיבר הגדול מהם) ואז יבצעו פניה אחת שמאלה שם יושרשו האיבר האחרון (n) יפנה ימינה עד סוף העץ ויושרש כעלה הימני האחרון – יבצע n/2 פניות ימינה (כולל השרשה). לבסוף נבחין כי יש פניות ימינה ו- פניות שמאלה.
10. *נתונים n קטעים בתוך הקטע על הישר הממשי מצאו שלם z השייך למספר מקסימלי של קטעים בזמן לינארי* ***ת:*** *שומרים מערך של מספרים שלכל מספר יש שדה נוסף שמגדיר אם אותו מספר פותח או סוגר קטע וממלאים את כל ערכי a בתור פותחים וכל ערכי b בתור סוגרים. לאחר מכן נעשה מיון בסיס לפי בסיס n על המערך. לפי שאלה ז-18 במדריך למידה העלות של כל המיון היא 2dn שזה לינארי מאחר ו d קבוע לפי נתון, כעת נעבור על הסדרה הממוינת ונשמור בכל איטרציה את מספר הקטעים הפתוחים באותה איטרציה (כלומר מספר הפתיחות עד כה פחות מספר הסגירות). בנוסף שומרים משתנה z נוסף ומשתנה שיחזיק את מספר הקטעים שמכילים אותו. בכל פעם שמספר הקטעים הפתוחים בנקודה גדול מזה של z מעדכנים את z לנקודה ואת המשתנה שאומר מה מספר הקטעים הפתוחים שמכילים אותו.*
11. *אלגוריתם מקבל ערימת מינימום עם n איברים ומספר שלם ומחזיר k איברים הקטנים ביותר ממוינים, סיבוכיות מקום של* ***1.*** *בזמן* ***ת1:*** *לפי עמ' 107 כל צומת i פרט לשורש מתקיים שהצומת קטנה או שווה לאביה, האיבר הקטן ביותר הוא השורש, נסרוק את הערימה, נתחיל בשורש ונבדוק מי מבין הבנים של i קטן יותר מאחיו ונמשיך את הסריקה לכיוון הבן הקטן ביותר, כלומר נעבור על k איברים ונעביר כל איבר מינימלי שנאתר למערך עזר בגודל k שיכיל את k האיברים הקטנים שמצאנו, מקרה שדורש התייחסות זה אם צמתי הבנים של צומת x בעלי ערך שווה, במקרה כזה נעביר את שני ערכי הבנים למערך הפלט ונבדוק את הנכדים של x (יש מקס' 4 כאלו) ונמשיך לכיוון הבן של הנכד שהוא הקטן מבין כולם, נמשיך לבצע זאת עד שנמלא את מערך העזר. במקרה הגרוע נעבור על k האיברים המינימליים ונכניס אותם למערך. סיבוכיות המקום היא כפי שנתבקשנו. את המערך שקיבלנו נוכל למיין במיון הכנסה בזמן ריבועי לפי עמ' 23 בספר כפי שנדרש.* ***2.*** *בזמן* ***ת:*** *על מנת שחיפוש k איברים יבוצע בזמן הרצוי אפשר להשתמש בפתרון של הסעיף הקודם ולמיין בעזרת מיון אופטימלי כגון מיון מיזוג/ערימה ובכך מיון של k האיברים שנמצאו יבוצע בזמן .*
12. *נתון מערך בגודל n כתוב אלגוריתם לינארי שבודק אם קיימים במערך שני איברים לא שווים שמופיעים יותר מ n/3 פעמים* ***ת:*** *מכיוון שידוע שיש לכל היותר 2 מספרים שחוזרים על עצמם יותר מ פעמים, אנו רצים על לולאה עם אינדקס מ – 1 ועד - 2 . בלולאה הפנימית נמצא את ערך המיקום של האיבר ה שיכול להופיע מעל ל פעמים (לכל הפחות פעמים). לאחר שמצאנו את ערך המיקום נספור כמה פעמים הוא מופיע באמת בסדרה. אם נמצא שהוא מופיע יותר מ פעמים נשמור בצד. לאחר מכן נמשיך לאיבר הבא שיכול להופיע מעל ל פעמים – עד סיום הלולאת for החיצונית לבסוף נבדוק שהם לא שווים, אם לא ערך אמת.*
13. *מבנה S שניתן לבצע את הפעולות: הכנסה בזמן lgn, מחיקה בזמן lgn ו – median-mode שמחזיר את ערך המפתח בעל השכיחות החציונית (כלומר, שכיחות המפתח הינו חציון כל n השכיחויות) בזמן O(1).****ת:*** *המבנה S יהיה מורכב מעץ אדום-שחור T שכל צומת בו מצביע לרשימה מקושרת דו כיווני ומצביע לערמה המתאימה, ערמת מקסימום וערמת מינימום.*

*תיאור הפעולות: הכנסה: מבצעים חיפוש בעץ T אחר המפתח k,אם המפתח k לא קיים בעץ T יוצרים צומת חדש בT עם רשימה של איבר אחד, מכניסים אותו לערמת המינימום עם הערך 1 ומבצעים*

*min-heapify, בדוקים את איזון הערימות, אם יש בערמת המינימום יותר מn/2 נמחק את השורש של ערמת המינימום ונעביר אותו על ערמת המקסימום.*

*במידה והצומת כבר קיים בעץ, מוסיפים אותו לרשימה המקושרת, מכיוון שלצמות יש מצביע לערמה המכילה את שכיחות האיבר, מגדילים את אותו איבר ב1, מבצעים heapify בתאם, ושוב בודקים את האיזון.*

*זמן הריצה הכולל הוא לוגריתמי. מחיקה: מבצעים חיפוש בעץT אחר המפתח k אם המפתח k לא קיים בעץ T נסיים את הפעלה, אחרת, מורידים איבר אחד מהרשימה של המפתח k מעדכנים בעזרת המצביע את הערמה המתאימה בכך ש מורידים 1, אם הערך שווה ל 0, מוחקים את הצומת מהעץ והערמה, מבצעים heapify ובודקים את איזון העץ ומסדירים אותו בהתאם.*

*זמן הריצה הכולל הוא לוגריתמי.*

*Median-mode: מחזירים את ראש ערמת המקסימום, זמן הריצה קבוע.*

1. נתון מערך בגודל n ידוע שקיים שלם k כך ש ומתקיים: המספר k לא ידוע, כתבו אלגו' רקורסיבי כמו מיון מיזוג הממין את המערך A בזמן לינארי, כנל נוסחת נסיגה ופתרון שלה. **ת:** נבחין כי המערך בנוי מ 2 מערכים עולים, הראשון גדול מהשני. האלגוריתם מתחיל בקריאה רקורסיבית למציאת האיבר k העלות היא ולאחר מציאת האיבר k נקרא לשגרת merge מעמ' 25 בספר, עם זמן ריצה לינארי, נוסחת הנסיגה:
2. נתון מערך A באורך n של מספרים **חיוביים ממשיים שונים זה מזה**, רוצים למצוא שני אינדקסים כל שמתקיים התנאי או **1.** כתבו שגרה למציאת האינדקסים שרצה ב במקרה גרוע **ת1:** ממיינים את המערך ב ואז לכל איבר x במערך מחפשים עם חיפוש בינארי אם קיים או , מתקיימים n חיפושים כאלו בעלות כוללת של . **2.** כתבו שגרה שמוצאת את האינדקסים שתוחלת זמן ריצתה לינארי **ת2:** מכניסים לטבלת גיבוב (שומרים את האינדקס המקורי של האיבר) ולכל איבר בטבלה מחפשים עבורו את או בטבלת הגיבוב, סה"כ מתקיימים n חיפושים וכל חיפוש בתוחלת זמן קבועה ולכן .
3. נתונה סדרה של n מספרים כתבו אלגוריתם שזמן ריצתו לינארי שמוצא וממיין את p האיברים הקטנים ביותר בסדרה, ידוע כי . **ת:** מצא את ערך מיקום p בעזרת אלגוריתם select בזמן לינארי, כעת נקבל כי כל האיברים בעלי ערך מיקום קטן מ p נמצאים משמאלו (בשל שגרת select) ונתבונן בתת-המערך 1..p – עליו נבצע מיון אופטימלי – מיון ערימה, מכיוון שמתקיים אז יתקיים שהמיון מתבצע בזמן לינארי.
4. *הצע מבנ"ת שמאפשר להכניס בזמן lgn , increase(p,d,S) מגדיל מפתח האיבר שאליו מצביע p בערך d>0 בזמן lgn, decrease מבצע דבר זהה רק שמקטין את המפתח בזמן lgn, ושגרה נוספת median-freq שמחזיר שכיחות חציונית (חציון שכיחויות של כל המפתחות) בזמן O(1). (n מציין מס' מפתחות שונים במבנ"ת)* ***ת:*** *המבנה S יהיה מורכב מעץ אדום-שחור T , ערמת מקסימום. כל מפתח T מגדיר צומת בT צומת ובH שני האיברים מחוברים ביניהם בעזרת מצביעים דו-כיווניים. בכל צומת בT נמצא מצביע אל רשימה מקושרת של איברים בעלי אותו מפתח k כל מפתח בערמה הינו השכיחות של מפתח k כלשהו בT תאור הפעולות: INSERT: מבצעים חיפוש בעץ Tאחר המפתח k אם המפתח k לא קיים בעץ T יוצרים צומת חדש בT עם רשימה של איבר אחד; יוצרים צומת חדש בH בעל המפתח 1 (בשורה האחרונה); אחרת, אם המפתח k כן קיים בעץ T מוסיפים את האיבר החדש לרשימה של המפתח k בצומת המתאים בH מוסיפים 1 למפתח ומבצעים MAX-HEAP-INCREASE-KEY זמן הריצה הכולל הוא lgn. INCREASE: בהנחה שהמפתח P קיים בעץ T מוחקים את הצומת שלו מT ומ - H מחליפים את המפתח P במפתח P+D ומכניסים אותו מחדש; אם המפתחp+d כבר קיים, ממזגים את שני הצמתים ומגדילים את השכיחות בH וגם מבצעים MAX-HEAP-INCREASE-KEY אחרת, מכניסים צומת חדש לT ולH זמן הריצה הכולל הוא lgn. DECREASE: בהנחה שהמפתח k קיים בעץ T מוחקים את הצומת שלו מT ומH מחליפים את המפתח k במפתח p-d ומכניסים אותו מחדש; אם המפתח p-d כבר קיים, ממזגים את שני הצמתים ומגדילים את השכיחות בH וגם מבצעים MAX-HEAP-INCREASE-KEY אחרת, מכניסים צומת חדש לT ולH זמן הריצה הכולל הוא lgn. Median-freq מחזירים את מפתח השורש של H זמן הריצה קבוע.*
5. *בניית ערמת מינימום המקיימת עבור כל כל איבר ברמה i-1 קטן מ (או שווה ל) כל איבר ברמה i (הרמה 0 מכילה שורש) הבנייה תתבצע בזמן לינארי.* ***ת:*** *נחשב את כובה הערימה ולאחר מכן נבצע select על מקום , כל מה שמימין לאיבר זה נדחף למקומות ועד n בערימה החדשה. נפעיל רקורסיבית את האלגוריתם על הצד השמאלי של המערך (שמכיל מחצית מהאיברים), איברים אלו יהיו מקומות 1 ועד בערימה. נוסחת נסיגה היא ופתרונה על ידי משפט האב הוא לינארי.*
6. *נגדיר את עץ צובר – כל צומת מכילה 2 שדות : key[z] ו accum[z] ניתן לבנות ממנו עץ בינארי רגיל כך: כל צומת z מחברים את accum[z] לכל המפתחות בתת העץ המושרש ב z.* ***1.*** *כתבו אלגו' הרץ בזמן לינארי הבודק אם העץ צובר A מייצג עץ חיפוש בינארי* ***ת1:*** *מכיוון שמעץ צובר ניתן לבנות עץ בינארי רגיל, מכך שמחברים את ערכי הaccum בהתאם. נובע שאנחנו צרכים לרוץ על העץ הצובר ולבדוק חיבור ערכי הaccum לכל צומת יתנו לנו עץ חיפוש בינארי תקין. נאגור באובייקט את הaccum של השורש, נקרא לבנו השמאלי, נוסיף להaccum שלו את הaccum של ההורה שלו, ונבדוק אם קטן יותר נתקדם לבן השמאלי שלו, ונחזור על הפעולה. אם גדול יותר נחזיר שגיאה ברגע שהגענו לעלה, נחזור לצמות ההורה ונבדוק את הבן הימני שלה, במידה ואחרי הוספת הaccum הבן הימני קטן מהורה נחזיר שגיאה. בסיום הבן השמאלי של השורש נעבור לבן הימני ונחזור, ונרוץ על ילדיו. כך נרוץ על כל הצמיתים, כל קריאה לצמות היא בעלות של O(1), סה"כ זמן הריצה O(n).* ***2.*** *כתבו שגרות עבור חיפוש, הכנסה ומחיקה עבור עץ צובר המייצג עץ חיפוש בינארי, זמן ריצה חייב להשאר כאשר h גובה העץ* ***ת2:*** *חיפוש – בעת החיפוש, לכל קריאה רקורסיבית שנעשה, נוסיף ל-K את ערך הaccum של אותה צומת.*

*הכנסה – בעת הכנסה של האיבר החדש, כל צומת שהוא "ירד" הלאה נוסיף לו את הaccum של ההורה שלו. מחיקה – אחרי פעולת המחיקה, אם אין לאותה צומת ילדים נמשיך כרגיל. אחרת נמצא את ערך הaccum של אותה צומת. נשתמש במשתנה עזר אשר יקבל את ערך הaccum של האיבר ה"מחליף" נחסיר מימנו את ערך הaccum של הצומת שמחקנו, מכאן נדע מה היה ערך הaccum ה"מקורי" של אותה צומות, ונחסיר את ערך זה מערך הaccum של הצומת המחליפה. וכך נעדכן גם את בניה של אותה הצומת.*

1. *מערך עם n איברים שכל איבר שייך לתחום כאשר k לא תלוי ב n. נתון בנוסף שלם (חיובי או שלילי) z. כתוב אלגו' למציאת מספר זוגות האינדקסים (i,j) שמקיימים המקיימים את התנאי זמן הריצה הוא .* ***ת:*** *ביצוע מיון מנייה שזמן ריצתו כנדרש ולאחר מכן בודקים עם שני מצביעים וסופרים כמה מספרים כאלו קיימים בכל פעם שמתקיים התנאי.*
2. *הצע מבנ"ת שבנייתו לינארית, הכנסה ב lgn, מחיקת ערך מיקום מהמבנה בזמן lgn, הגדלת מפתח איבר שאליו מצביע p בערך d>0 בזמן lgn.* ***ת:*** *המבנה בנוי מ 2 ערימות,*

*Build: משתמשים בselect למציאת 2n/3, לאחר מכן מבצעים חלוקה סביב הערך שקיבלנו*

*מ האיברים הקטנים ביותר בונים את ערמת המקסימום, ומה האיברים הגדולים יותר בונים את ערמת המינימום. בשני הערמת השתמשנו הbuild-heap, לכן זמן הריצה הכולל היה O(n).*

*Insert: אם המפתח קטן מערך 2n/3 נכניס אותו לערמת המקסימום אחרת הוא יכנס לערמת המינימום, במידה שמספר האיברים בערמת המקסימום גדול מ2n/3 מוחקים את השורש של ערמת המקסמום ומעבירם אותו לערמת המינימום, במידה ומספר האיברים בערמת המינימום גדול מ2n/3 מוחקים את השורש של ערמת המינימום ומעברים אותו לערמת המקסימום*

*זמן ריצה O(lgn).*

*DEL-OS-2-3: מוחקים את השורש של ערמת המינימום, לאחר מכן בודקים אם מספר האיברים בערמת המקסימום גדול מ2n/3, במידה וכן מוחקים את השורש של ערמת המקסימום ומעברים אותו לערמת המינימום.*

*INCREASE: אם מצבעים לאיבר בערמת המינימום מגדילים את האיבר ומפעלים את השגרה MIN-HEAPIFY.*

*אם מצביע לאיבר בערמת המקסימום, מגדילים את האיבר, בודקים אם הוא עדין קטן הוא שווה לשורש של ערמת המינימום, במידה וכן מבצעים את השגרה MAX-HEAPIFY.*

*אחרת מחליפים את האיבר בשורש של ערמת המינימום, ומבצעים על כל ערמה את ה HEAPIFY המתאים לה.*

1. *שגרה TRUE\_SQUEARES מקבלת מערך של ערכים בוליאניים, צריך להוכיח שכשהיא מסתיימת ערך התא A[i] הוא TRUE אמ"מ i ריבוע שלם* ***ת:*** *כל מקום במערך מתחיל ב FALSE ועל מנת שיסתיים בTRUE הוא צריך לעבור מספר אי זוגי של הפיכות באמצעות FLIP נריץ בדיקה על מערך באורך 9 ונראה שFLIP מתבצע בפעם הראשונה לכל אחד מהתאים בפעם השנייה לכל תא בקפיצות של 2 פעם שלישית לכל תא בקפיצות של 3 וכן הלאה, נראה שריבוע שלם הוא מספר טבעי המהווה ריבוע של מס טבעי כלשהו נראה כי 4 ו 9 הם מקיימים את התנאי 2 בחזקת 2 ו 3 בחזקת 2. כל מספר עובר היפוך כאשר i שלנו הוא מחלק של המספר. ניתן לראות שכל מספר שהוא ריבוע שלם הוא בעל מספר מחלקים אי זוגי (4 מתחלק ב 4 1 2 )(9 מתחלק ב 3 9 1) לפיכך כל מספר שיש לו כמות זוגית של מחלקים יחזור להיות FALSE וכל מספר שיש לו כמו אי זוגית של מחלקים יהיה בסוף TRUE לכן ערך התא יחזיר TRUE אם ורק אם i הוא ריבוע שלם.* ***לנתח סיבוכיות:*** *בשורה 1-2 רץ על כל המערך בזמן ליניארי שורה 3 עוד פעם זמן ליניארי רץ על כל המערך ובכל אחת מהקריאות תבצע lgn קריאות פנימיות בתוך הלולאה של WHILE כי רצים על המערך כמספר המחלקים של i ולא על כל המערך כולו סה"כ*
2. *יש n קטעים על ישר ממשי כך שקצות הקטעים שלמים בתחום כתוב אלגו' יעיל המחפש נק' מאוזנת – נק' בתחום שמס' הקטעים הנמצאים משמאלה שווה למספר הקטעים מימינה ושווה למס' הקטעים המכילים אותה, אם לא קיימת – מחזיר הודעה מתאימה, הוכח נכונות וסיבוכיות, ניתן להניח אין קצוות חופפים.* ***ת:*** *התחום שלנו הו כנתון, נבחר בבסיס n ובדוק אורך מקסימלי של מספר בכל הקטעים, ידוע שהכי גדול הוא ולכן אורך המס' הוא (בבסיס n כמובן) לכן יש 2 ספרות. נוכל למיין בעזרת מיון בסיס שיפיק זמן של ומכאן הפתרון פשוט : ראשית כדי שאכן תהיה נקודה כזאת – n חייב להתחלק ב 3 ללא שארית משום שכל קטע או מימין או משמאל או מכיל אותה. אם לא מתחלק ב 3 נחזיר תשובה שלילית. כעת נמיין לי נקודת סיום של הקטעים במיון בסיס ונלך למקום במערך, זו הנקודה בה עברנו קטעים משמאלנו, כלומר עם נסגרו, נשמור את הנקודה הזו. כעת נמיין לפי נקודת התחלה, נלך למקום ה במערך, כאן נפתחו הקטעים האחרונים, כלומר הקטעים שמימין. כעת עלינו לקחת את הקטעים הנותרים ( פשוט נעבור על הקטעים וניקח כל קטע שלא נסגר עד הנקודה הראשונה שמצאנו ולא נפתח אחרי הנקודה השנייה שמצאנו (בוודאות נקבל קטעים. כעת נבדוק מהי נקודת הפתיחה הגדולה ביותר (שוב, לפי מיון בסיס) ומה היא הנקודות סגירה הקטנה ביותר) אם נקודת הפתיחה קטנה מנקודת הסגירה נבחר כל נקודה ביניהם ונסיים, אם לא, לחלופין, 2 הנקודות הראשונות שמצאנו מקיימות נקודה ראשונה גדולה מהשנייה ונחזיר שאין נקודה מתאימה.* ***נכונות –*** *האלגוריתם מתבסס על למצוא את הקטעים הכי שמאליים ו הכי ימניים. לבדוק שיש ביניהם מקום (לא נחתכים), למצוא שיש נקודה שכל הקטעים הנותרים מכילים ולבחור אותה. סיבוכיות האלגוריתם היא , המיון צורך כפי שנאמר בהתחלה זמן לינארי ומציאת ערך מיקום צורך זמן לינארי גם עם אלגוריתם select, לכן לבסוף אחרי שימוש של 5 פעמים באלגוריתמים לינארים נקבל : . נכונות האלגוריתם מתבססת על כך שבהכרח קטעים הראשונים שנסגרים יהיו חייבים להיות משמאל לנקודה (אחרת לא היו מספיק משמאל) וכנ"ל על הימניים שנפתחו אחרונים. כמו כן, אם לא יהיה מרווח ביניהם לא נוכל למצוא נקודה ביניהם ולכן גם את זה וידאנו, כעת ברור שהקטעים שנשארו חייבים להיות המכילים ואם אין להם נקודה משותפת אין נקודה מתאימה.*
3. *מבנה ששומר מידע על כדורים בנפחים וצבעים שונים, מספר הצבעים האפשריים הוא C ונתון מראש, כל צבע מסומן 1 עד C וצריך לתמוך בפעולות: הכנסה, הוצאת כדור בעל נפח מקסימלי, ניפוח כל הכדורים בצבע מסוים ונפח גדול מ V כלשהו בערך d חיובי, כל הפעולות צריכות להתבצע בזמן כאשר k מסמל את כמות הצבעים בפועל ו n את כמות הכדורים הכוללת במבנה.* ***ת:*** *C+1 עא"ש-ים ה C הראשונים הם אחד לכל צבע והם בעלי שדה הרחבה, האחרון יכיל מקס' אחד מכל צבע – לכל מקס' בעץ של צבע יהיה בעל מצביע לצומת שלו בעץ המקסימליים ולהיפך. בעת ניפוח נלך לעא"ש של הצבע בו מנפחים ונחפש את V , יש לציין קודם כל שברירת המחדל של שדה ההרחבה היא 0. כעת, במהלך החיפוש אם נעים שמאלה נוסיף לשדה ההרחבה של העץ ממנו אנחנו נעים את d ואם נעים ימינה פשוט נעים ובמקרה ומצאנו את V נזוז ימינה, כך נעשה עד שנגיע לשורש, הדבר מציין שכל צאצא של השורש והשורש צריכים להיות הערך שלהם ועוד d . ניתן לתחזק פעולה זו בסיבובים באופן טריוויאלי, בסיבוב שמאלי כמתואר בעמ' 234 צריך להוסיף לערך ההרחבה ב y את של x ובסיבוב ימינה בדיוק ההיפך, לחסר. כעת הכנסה, בדיוק בצורה הפוכה מחיפוש, בכל פעם שהולכים שמאלה לא קורה כלום אך כאשר נעים ימינה צריך לחסר ערך הרחבה מהצומת המוכנסת, כמו כן, לפני שהולכים ימינה מוודאים שגם לאחר חיסור עדיין עלינו ללכת ימינה, אם לא – מבצעים חיסור והולכים שמאלה. בהכנסה ובניפוח עלינו לבדוק אם המקסימום השתנה/הניפוח תקף אליו ובמידה וכן לשנות את ערכו/להוסיפו לעץ אדום שחור של מקס' עם הערך החדש ולעדכן מצביעים. כעת מחיקה מקסימום היא פעולה פשוטה, הולכים לעץ המקס, מוציאים את המקס אך לפני כן הולכים להצבעה שלו בעץ של הצבעים ומוחקים גם משם ומכניסים את המקס החדש מהצבע הזה כולל ניפוחים מערכי ההרחבה! בכל אחת מהפעולות אנו מבצעים חיפוש/עדכון/מחיקה בשני עצים, עץ צבע ועץ מקסימום, לכן הסיבוכיות היא , k גודל עץ מקסימום ו n גודל עץ הצבע במקרה הגרוע. יש לציין שבכדי לגשת לכל אחד מעצי הצבע בנוחות ניתן לשמור מערך של מצביעים אליהם.*
4. *עבור מערך בגודל n נסמן ו - הוכיחו כי קיימים עבורם* ***ת1:*** *מתייחסים לכל המספרים כנקודות על ציר מספרים בקטע [m,M] מחלקים את הקטע הנ"ל ל n-1 תתי קטעים שווים בגודלם. בגלל שיש n נקודות על פי עקרון שובך היונים חייב להיות קטע אחד עם לפחות 2 נקודות לכן זה מתקיים.* ***מציאת זוג האינדקסים בזמן לינארי****: נממש שינוי של מיון דלי, נכניס את הערכים ל n-1 דליים אחרי נרמול של הקלט לטווח 0 עד 1, כעת נבדוק איזה דלי מחזיק 2 ערכים או יותר.*
5. *מס' השוואת בין מפתחות ומס' העתקות של מיון הכנסה על 2,1,4,3,…n-2,n-3,n,n-1* ***ת:*** *חוץ מהאיבר הראשון, כל איבר במיקום אי-זוגי יושווה רק פעם אחת לאיבר שלפניו וכל איבר במיקום זוגי יושווה פעמיים (לשני האיברים שלפיו) עד שיגיע למקום שלו.* ***העתקות:*** *רק המספרים האי-זוגיים לנמצאים במקום שלהם , ויועתקו רק פעם אחת שמאלה*
6. *נתונה ערימת מינימום בת n איברים ומשנים את המסלול השמאלי כך: לכל ו d חיובי.* ***1.*** *באיזה תנאים שינוי זה מוביל להפרת תכונות הערימה?* ***ת:*** *השינוי לא יוביל להפרת תכונות הערימה כל עוד האיבר ששונה עדיין יותר קטן מהבנים שלו – הבעיה צריכה להיות קיימת רק מול הבן הימני, כי מול השמאלי גם האב וגם הבן גדלו ב d.*
7. *נתון עץ חיפוש בינארי* ***מאוזן*** *המכיל n מפתחות, נוסיף לכל צומת שדה height שמאחסן את גובה תת העץ המושרש באותה צומת (גובה הצומת בתכלס)* ***1.*** *נניח שמתבצע בצומת z סיבוב ימני הראו שניתן לעדכן את כל שדות הגובה בעץ בזמן לוגריתמי* ***ת1:*** *נתבונן באיור בספר בעמ' 234, הערך של לא משתנה, הערך של y הופך להיות המקס' מהערכים של , הערך של x הופך להיות המקס' מהערכים של \*\*כל מה שנשאר זה לעדכן את העץ מ x ועד השורש כך שבכל צומת ניקח את המקסימום משני הבנים +1 . כל הפעולות חוץ מ \*\* מתבצעות בזמן קבוע, הפעולה \*\* מתבצעת בזמן לוגריתמי על גובה העץ שהוא מאוזן לפי הנתון ולכן כלל האלגוריתם פועל ב .* ***2.*** *נניח שבמקום השדה height מוסיפים שדה depth, המאחסן את עומק הצומת בעץ. האם ניתן לעדכן את כל השדות בעומק בזמן לוגריתמי?* ***ת2:*** *לא ניתן לבצע בזמן לוגריתמי מכיוון שבעת סיבוב הערך של depth נדרש להתעדכן עבור כל צומת בתת העץ ולא רק עבור הצמתים במסלול מסוים ולכן במקרה הגרוע יידרש לעדכן כל הצמתים בעץ ולכן העלות היא לינארית.*
8. *מבנה שתומך ב insert בזמן lgn, delete בזמן lgn, find-succ מציאת ערך מפתח עוקב למפתח שנכנס כפרמטר בזמן lgn, mode2 החזרת ערך המפתח בעל השכיחות השנייה בגובהה בזמן קבוע, n מס' המפתחות השונים.* ***ת:*** *נשתמש בעץ אדום שחור T עבור המפתחות השונים , כאשר כל צומת מכילה רשימה דו-מקושרת עבור כל האיברים עם אותו המפתח. נשתמש בעוד עץ אדום שחור Q שהמפתחות בו הינם השכיחות של המפתחות בעץ T , ונחזיר מצביע לערך השני בגודלו. בין העצים T ו-Q ישנן הצבעות דו כיווניות. Insert(S,k) – הכנסה לעץ T , אם נתקלים במסלול בצומת עם אותו הערך נוסיף את הערך לרשימה נעדכן את הערך של המפתח הזה בעץ Q ואם נדרש גם את מצביע q כעת, אם מגיעים לעלה אז יש לנו מפתח חדש נצטרך להוסיף המפתח לראש הרשימה בצומת החדש של העץ T לאחר מכן להוסיף את המפתח עם הערך 1 לעץ Q לעדכן את העץ Q לעדכן את המפתח q אם נדרש. כל הפעולות כאן הינן פעולת של הכנסה לעץ אדום שחור \ עדכן ראש רשימה מקושרת \ מצביעים ולכן העבודה הינה O(lgn) Delete(S,k) – חיפוש ערך המפתח ל p בעץ T לאחר מכן מחיקת p מהרשימה המקושרת, אם הוא האחרון מחיקת המפתח מ-T ומ-Q אחרת חיסור 1 מהערך המתאים ב-Q ועדכון העץ והמצביע qכל הפעולות הינן פעולות של מחיקה מעץ אדום שחור מחיקה מרשימה דו-מקושרת שיש לנו מצביע לערך הנמחק ולכן מחיקה מתבצעת ב-O(lgn) Findsuccessor(S,k) החזרת הערך העוקב בעץ אדום שחור עם הפונקציה הרגילה שלו Mode2(S,k) החזרת ערך מצביע q.*
9. *נתון מערך A[1…n], וידוע כי המערך הוא הפלט של סריקה תחילית של עח"ב T שכל המפתחות בו שונים זה מזה.* ***1.*** *כתבו אלגוריתם הבונה את העץ T בהינתן המערך A, נתחו את סיבוכיות זמן הריצה של האלגוריתם שהצעתם.* ***ת1:*** *מערך לדוגמה* ***{ 20, 10, 5, 1, 7, 15, 30, 25, 35, 32, 40 }*** *האיבר הראשון במערך הוא השורש של העץ, טריוויאלי מתוכנות סריקה תחילית, במקרה זה 20,נגדיר אותו כשרוש העץ, וניצור שני טווחים, מינימום ומקסימום אשר יתחמו את הבנים שלו השורש, כאשר באתחול הראשון המינימום יאתחל להיות המינימום הכי קטן שנתמך והמקסימום יאתחל להיות מקסימום הכי גדול שנתמך, כדי שהשורש יכול לקבל כל ערך בטווח האפשרי. כאשר נגדיר את הבן השמאלי של הצומת (לדוגמה השורש 20), צרכים להגדיר שערך המקסימלי שלו היה איבר הצומת (במקרה זה 20). אם האיבר הבא מערך נמצע בטווח הזה של המערך, נגדיר אותו להיות הבן השמאלי. אחרת, הוא חייב להיות הבן הימני, וכך אלה נעבור על זה באופן רקורסיבי.* ***2.*** *האם ניתן לבנות את T מתוך A אם ידוע ש T עץ בינארי אך לא נתון כי הוא עח"ב? הוכיחו את קביעתכם.* ***ת2:*** *לא, בעץ בינארי רגיל, לדוגמה 7 יכול להיות הבן השמאלי של הערך 1, לכן לא ניתן היה לדעת את מיקמו המדויק בעץ.*
10. *תכננו מבנה נתונים S שבאמצעותו ניתן לבצע את כל אחת מהפעולות הבאות בסיבוכיות זמן ריצה של Θ(logn) כאשר n מציין את מספר האיברים במבנה, הניחו כי כל האיברים במבנה שונים. Insert(S,x) – מכניס את המפתח x למבנה. Find(S,x) – בודק האם המפתח x נמצב במבנה ומחזיר מצביע לאיבר שמפתחו x אם כן אחרת מחזירה NIL. inversions(S,x) - אם קיים במבנה איבר בעל המפתח x השגרה מחזירה את מספר המפתחות y הגדולים מx והוכנסו לפניו.* ***ת:*** *תשובה: מבנה הנותנים היה עץ אדום שחור עם 2 שדות הרחבה. size – כמות הבנים הימנים של הצומת ביחס לאותה הכנסה. Inbefore – ערך האיברים הגדולים והוותיקים מהצומת. תיאור האלגוריתם: ברגע שנכניס איבר חדש לתוך העץ, נספור כמה איברים גדולים מאיבר שהוכנס. הנכונות של הרעיון הזה בא מקח שבמבנה הנותנים הזה אין פעולה של מחיקה לכן כל איבר שקיים כבר בתוך המבנה חייב להיות ותיק יותר מהאיבר שמוכנס. Insert – בעת הכנסה לעץ, במידה ואנו יורדים לעבר הבן הימני, השדה size שעברנו יגדל ב-1, אך לא נסכום אותו, כאשר יורדים לעבר הבן השמאלי סוכמים את ערך ה size של הצומת שעברנו ונוסיף לסכימה הזאת 1, במידה ושדה ההרחבה size של הבן השמאלי שלו קיים נחסר אותו מהסכימה. כאשר האיבר הגיע למקומו ערך שדה ה Inbefore יקבל לערך הסכימה שלנו. insert – חסם תחתון O(1) כלומר הכנסת השורש, חסם עליון O(logn) לכן סה"כ היה Θ(logn). find – חסם תחתון O(1) כלומר מציאת השורש, חסם עליון O(logn) לכן סה"כ היה Θ(logn). inversions- חיפוש רגיל, במידה ונמצא נחזיר את הערך Inbefore, מאותה סיבות כמו find סה"כ זמן הריצה יהיה Θ(logn). בעת פעולת הסיבוב הרגילות, לא תהיה פגיע במבנה, כי השדה size יעבור בהתאם ובעת ההכנסה הבא עדין ישמר מה מספר האיברים הגדולים מאותה צומת ולכן ידע כמה איברים ותיקים יותר קיימים מאותה הכנסה.*
11. *נתונים שני מערכים A[1…n] ו- B[1…n] וידוע כי המערך A הוא פלט של סריקה סופית, והמערך B הוא פלט של סריקה תוכית של עץ בינארי T שכל המפתחות בו שונים זה מזה.* ***1:*** *כתבו אלגוריתם הבונה את העץ T בהינתן שני המערכים, נתחו את סיבוכיות הריצה של האלגו'.* ***ת1:*** *דוגמה ל 2 מערכים כאלו: A-PostOrder - 9, 1, 2, 12, 7, 5, 3, 11, 4, 8  B-InOrder - 9, 5, 1, 7, 2, 12, 8, 4, 3, 11  לכן: מתכונות סריקה סופית,POSTORDER,נובע שתמיד השורש של העץ האיבר האחרון של המערך, במקרה זה A[n], הוא 8, נגדיר אותו כשורש. כעט נחפש את איבר השורש,8 במקרה זה, במערך של הסריקה התוכית, InOrder, כאשר מצאנו באיזה תא i הוא נמצא, נדע שכל האיברים שמתחת לi הם בצדו השמאלי וכל האיברים שמעל i הם בצדו הימני, במערך B כמובן. כעת נגדיר X, כמספר האיברים מB[1] ל B[i],  
    { 9, 5, 1, 7, 2, 12} . ונגדיר Y, כמספר האיברים מB[i] ל B[n],{ 4, 3, 11}. כעת באופן דומה, נבנה את תת עץ שמאלי של העץ, כי אם נרוץ עם X, על A, נראה ש A[X]=5, הוא "השורש" של בתת עץ השמאלי וכך נרוץ עד סיום כל תתי העץ. באופן זהה, נבע את התת עץ הימני עם Y. סה"כ זמן ריצה Θ(n²)* ***2:*** *האם ניתן לבנות את העץ T מתוך B בלבד? הוכיחו את קביעתכם.* ***ת2:*** *לא, לדוגמה 2 עצים שונים שיתנו תוצאת סריקה זהה שנהיים יתנו 1,2 ופעם אחת שורש 2 ובן שמאלי 1 ופעם אחרת שורש 1 ובן ימני 2.*
12. *תכננו מבנה נתונים S שבאמצעותו ניתן לבצע את כל אחת מהפעולות הבאות בסיבוכיות זמן ריצה של Θ(logn) כאשר n מציין את מספר האיברים במבנה Insert(S,x) – מכניס את המפתח x למבנה. Find(S,x) – בודק האם המפתח x נמצב במבנה ומחזיר מצביע לאיבר שמפתחו x אם כן אחרת מחזירה NIL. range(x,y) - השגרה מדפיסה את מספר האיברים במבנה בעלי ערך מפתח k המקייםx<=k<=y.****ת:*** *נשתמש בעץ אדום שחור מורחב שמכיל את מספר האיברים בתת העץ המושרש של אותה הצמות. Insert-בדומה לצורה הרגילה, פרט לתוספת הבאה, בעת ההכנסה כל צומת שעוברים מגדלים את שדה ההרחבה של אותה צומת ב1. חסם תחתון O(1) כלומר הכנסת השורש, חסם עליון O(logn) לכן סה"כ היה Θ(logn). find – בדומה לצורה הרגילה ,חסם תחתון O(1) כלומר מציאת השורש, חסם עליון (logn) לכן סה"כ היה Θ(logn). Range- נעזר בשדה ההרחבה כדי למצאו את המפתח X, כאשר מתחיל לחפש מהשורש ועובר כל צומת נבדוק: - אם המפתח בצומת הנוכחי קטן מX נסכום למשתנה זמני את הסכום שיש בשדה ההרחבה פחות הסכום שיש לבן הימני של הצומת ונתקדם ימינה. - אם המפתח בצומת הנוכחי גדול שווה לX נתקדם שמאלה בלי להוסיף דבר לסכום שצברנו.*

*במידה הצומת שווה ממש לX נוסיף לסכום 1. בסוף החיפוש נשאר עם סכום האיברים הקטנים ממש מX, מכיוון ששדה ההרחבה שלנו מצין כמה איברים יש בתת צומת, ולכן ברגע שמתקדים ימינה נובע שX, גדול יותר מהצומת שעברנו ולכן ניסכום כמה איברים קטנים מימנה קיימים. נבצע חיפוש דומה על Y, נחפש בעץ ונסכום במשתנה זמני אחר את סכום האיברים הקטנים שווים לו. במידה ונתקלנו בצומת אשר שווה ממש לY נוסיף לסכום 1. לבסוף נותרנו עם 2 משתנים זמנים SUMX וSUMY נחש את ההפרש ביניהם ונקבל את סכום האיברים הקטנים שווים מY וגדולים שווים לX. זמן הריצה, 2 חיפושים רגילים:*

*תמיכה בסיבוב: סכומו של Y משתה לסכומו של Y שהיה פחות הסכום של ϒ פחות 2 (צמתים X ו- ϒ עצמם). סכומו של X משתנה לסכומו של Y אחרי הסיבוב פלוס הסכום של ϒ פלוס 2 (בצמתים ϒ ו-Y עצמם).*

1. *בהינתן מערך A[1…n], ומספר שלם 1<=k<=n, כתבו אלגוריתם הבונה מערך 𝐵[𝑗] = max⁡{𝐴[𝑗],𝐴[𝑗 + 1],…, 𝐴[𝑗 + 𝑘 − 1]} המקיים כי 𝐵[1,…, 𝑛 − 𝑘 + 1] זמן הריצה של האלגוריתם הוא Θ(n).* ***ת:*** *נרוץ תחילה על המערך A, עד K ונמצא את האיבר המקסימלי באופן הבא, נכניס את האיבר הA[1] לתוך B[1], ונבצע השווה לכל איבר לעורך 1<=i<=k, נבדוק אם גדול מ B[1] אם כן נעדכן את התא, אחרת נמשיך, זמן הריצה O(k). נגדיר איבר J=2, וכעת נרוץ עם i על n-k האיברים הנותרים, n-k<=i<=n, במידה ו- A[i] גדול מ- B[j], התא B[j] יעודכן ב A[i] וJ יגדל ב1, אחרת B[j] יקבל הערך שבתא B[j-1] . זמן הריצה O(n-k). סה"כ זמן הריצה Θ(n).*
2. *נתונים שני עצי חיפוש בינאריים, העץ T1 ובו n צמתים והעץ T2 ובו m צמתים. כתבו אלגוריתם המקבל מצביעים לשני העצים ובונה בזמן עץ אדום שחור המכיל את צמתי שני העצים, זמן הריצה של האלגוריתם הוא Θ(max(m,n)), ניתן להניח שכל המפתחות בשני העצים שונים זה מזה.* ***ת:*** *תחילה נפעיל סריקה תוכית על שני העצים ונקבל שני מערכים ממוינים A[1….n] ו B[1….m]נמזג אותם למערך חדש C[1….n+m] בעזרת שיגרת המיזוג, כעת יש לנו מערך ממוין בגודל n+m נבנה את העץ מציאת החציון של C, מכיוון שממוין בעלות של Θ(1). מכניסים את החציון לשורש העץ, Θ(1) חוזרים באופן רקורסיבי על התהליך הנ"ל עבור 2 חצאי המערך, כאשר החלק השמאלי היה תת העץ השמאלי והחלק הימני היה תת העץ הימני. טיפול בעץ, יבוצע כרגיל בדיוק כמו שמתואר בספר. סיבוכיות זמן הריצה מקיים את נוסחת הנסיגה הבא: n=m+n עבור נוסחאות הנסיגה. לפי שיטת האב, כמו כן .*
3. *תכננו מבנה נתונים S שבאמצעותו ניתן לבצע את כל אחת מהפעולות הבאות בסיבוכיות זמן ריצה של Θ(logn) כאשר n מציין את מספר האיברים במבנה. Insert(S,x) – מכניס את המפתח x למבנה. Find(S,x) – בודק האם המפתח x נמצב במבנה ומחזיר מצביע לאיבר שמפתחו x אם כן אחרת מחזירה NIL. increment(x,y) - מוסיף ערך שלילי a<0 לכל אברי המבנה S שהמפתח שלהם קטן מ-x.*  ***ת:*** *מבנה הנותנים יהיה עץ אדם שחור עם שדה נוסף D , אשר יסמן את הערך שצריך להוריד מכל הצמתים אשר קטנים ממנו. Insert- הכנסה רגילה של עץ אדום שחור, רק שבכל צומת בדרך אם פונים ימינה נוסיף את הערך D ל-D שבאיבר שלנו. Θ(Logn) FIND – נשתמש במשתנה עזר d שיאותחל ב0, לכל צומת שמגיעים מוספים את ערך D של הצומת לd, נבצע השוואה key[K]+d מול הK שקיבלנו מהשגרה וK מייצג את הצומת הנוכחית. אם X קטן מהתוצאה שקבלנו נמשיך ללכת על התת עץ השמאלי, אם גדול נמשיך לסורק את התת עץ הימני, לא לפני שנעדכן את d באופן הבא, d=d-D, אם שווה נחזיר את הערך. סה"כ זמן הריצה הוא גובה העץ. Θ(Logn) Increment – סורקים את העץ כלפי מטה מהשורש, אם המפתח של הצומת קטן מX, נעדכן את הערך D הנוכחי שלו D=D-d, ונמשיך לסרוק את התת עץ הימני. אם גדול או שווה לX, נמשיך לסרוק את התת עץ השמאלי, סה"כ הריצה הוא גובה העץ. Θ(Logn)*
4. *כתבו אלגוריתם המקבל מצביע לעץ בינארי T ומחזיר את המספר d של הרמה בעץ שסכום איבריה מקסימלי, על האלגוריתם לרוץ בסיבוכיות זמן Θ(n), כאשר n הוא מספר האיברים בעץ, ובסיבוכיות מקום של Θ(n). הערה: רמה d בעץ מכילה את כל האיברים בגובה d בעץ (מוגדר בנספח ב.5).* ***ת:*** *נעבור מלמטה למעלה על כל צומת בעץ ונרחיב אותה בשדה sum הסוםם את סכום כל כלל הערכים הנמצאים מתחתיה (ממש כמו סעיף 3 במבחן זה) ובשדה גובה d המציין באיזה גובה הצומת נמצאת. לפי 14.1 ההרחבה בשדות אילו אינה משנה את זמן הריצה, ולכן זמן הריצה עד כה היה Θ(n). כי עוברים בדיוק פעם אחת על כל צומת (ויש n כאלה), מספיק המידע מהאבא,צומת,בניו בשביל לבצע את העדכונים. כעת נעבור שוב על כל צומת ונמצא את זה עם הערך sum המקסימאלי, ונחזיר את השתנה d שלו. גם כאן זמן הריצה היה Θ(n) כי אני צריכים לעבור בדיוק על כל הצומת פעם אחת לקבלת המקסימום. זמן הריצה הכולל הוא Θ(n)+ Θ(n)= Θ(n). לא השתמשנו בזיכרון נוסף פרט לn האיברים שבעץ ולכן גם כאן Θ(n) + משתנה קבוע אחד אשר זוכר את המקסימום.*
5. *תכננו מבנה נתונים S שבאמצעותו ניתן לבצע את כל אחת מהפעולות הבאות בסיבוכיות זמן ריצה של Θ(logn) כאשר n מציין את מספר האיברים השונים במבנה. Insert(S,x) – מכניס את המפתח x למבנה.*

*freq(S,x) – מחזיר את מספר הפעמים שהמפתח x מופיע במבנה (0 אם לא מופיע).smaller(S,y) - מחזיר את מספר המפתחות במבנה (כולל כפילויות) הקטנים או שווים לx.* ***ת:*** *מבנה הנתונים S היה מורכב מעץ אדום שחור מורחב בשני שדות freq: אשר מכיל את שכיחות (כמות הפעמים) המפתח במבנה הנותנים. size: אשר מכיל את מספר המפתחות בתת עץ המשורש בצומת, כולל כפילויות. Insert: מחפשים את המפתח במבנה הנותנים אם המפתח מופיע כבר בעץ השדה freq שלו יגדל ב1, ולכל האבות הקדומים שלו מגדלים את השדה size ב1. אם המפתח לא מופיע בעץ, יוצרים עבורו צומת חדשה, את שדה הsize מאתחלים ב1. ולכל האבות הקדומים שלו מגדלים את השדה size ב1. freq: מבצעים חיפוש על העץ עבור הערך x, נשתמש בTree-search, אם הערך לא קיים נחזיר null, אחרת נחזיר את השדהsize של הצומת. smaller: מבצעים חיפוש על העץ עבור הערך x, נשתמש בTree-search, בכל צומת z לאורך מסלול החיפוש אוגרים את הערך ב size[left[z]], במידה וleft[z] קיים ,בסיום מחזירים את סכום שדות אלה, אם המפתחx לא קיים נחזיר NULL.* *תמיכה בסיבוב: במידה ויש צורך לבצע סיבוב לעץ נשתמש בנוסחאות size[y]🡨size[x] וגם size[x]🡨 size[left[x]]+size[right[x]]+freq[x] זמני ריצה: insert – חסם תחתון O(1) כלומר הכנסת השורש, חסם עליון O(logn) לכן סה"כ היה Θ(logn). find – חסם תחתון O(1) כלומר מציאת השורש, חסם עליון O(logn) לכן סה"כ היה Θ(logn). smaller - חיפוש רגיל, במידה ונמצא נחזיר את סכום השדות, מאותה סיבות כמו find סה"כ זמן הריצה היה Θ(logn).*

1. *נתון מערך ממוין A[1…n], ונתון שלם k , 0<k<n, אנו משנים k מפתחות במערך A (במקומות כלשהם), אחר כך מפעילים על המערך A אלגוריתם מיון הכנסה או מיון מיזוג, עבור אילו ערכים של k עדיך להשתמש במיון הכנסה ואילו ערכים מיון מיזוג* ***ת:*** *בכל איטרציה של לולאת ה while הפנימית במיון הכנסה מתוקן היפוך שהיה במערך המקורי (היפוך זה כאשר i<j והערכים במערך A[i]>A[j]), כאשר אנחנו משנים ערך במערך ממוין אנחנו יכולים ליצור לכל היותר n-1 היפוכים כי האיבר יכול להיות רחוק לכל היותר n-1 מקומות ממיקומו האמיתי במערך ממוין, לכן כאשר אנחנו משנים את ערכם של k מפתחות במערך אנחנו יכולים ליצור להכל היותר סדר גודל של Θ(kn). היפוכים, לכן עבור k=O(logn) ירוץ מיון הכנסה במקרה הגרוע Θ(nlogn) ולכן נוכל להשתמש בו. לעומת זאת עבור ערכי k גדולים יותר (לדוגמה n/2) נשתמש במיון מיזוג המבצע תמיד Θ(nlogn) פעולות ללא תלות במערך הקלט*
2. *מערך בגודל n ברצוננו לבנות מבנה נתונים המאפשר לנו למצוא לכל זוג של אינדקסים 1<=i<=j<=m את האיבר המינימלי בתת המערך A[i…j].* ***ת:*** *בנייה בזמן לינארי כך שכל שאילתא תתבצע בזמן לוגריתמי - נבנה מערך בגודל 2m-1 ונאתחל את m-1 איבריו הראשונים להיות "אינסוף" , ואת M איבריו האחרונים להיות איברי המערך A לפי סדרם המקורי. כעת נבנה ערימת מינימום ממערך זה, נשים לב כי בתחילה כל איברי המערך A הם עלים ולאחר מכן כאשר אנו מתקדים לדוגמה את אביהם של הערכים A[1] ו- A[2] בערימה אנו מקבלים שאביהם הוא המינימום מביניהם.*
3. *כתבו אלגוריתם יעיל להכנסת n איברים הנתונים במערך לערמה בה יש 2n איברים – ליצור מערך בגודל 3n להכניס את כל האיברים וליצור ערימה מחדש בזמן לינארי*
4. *כתבו אלגוריתם יעיל להכנסת n איברים הנתונים במערך לעח"ב בה יש 2n איברים – מיון ערימה על מערך בגודל n סריקה תוכית על עחב מקבלים מערך ממוין, מיזוג של שניהם ויצירה בזמן לינארי.*
5. *כתבו אלגוריתם יעיל להכנסת n איברים הנתונים במערך לעא"ש בה יש 2n איברים – הכנסה לעץ א"ש*
6. *כתבו אלגוריתם יעיל להכנסת n איברים הנתונים במערך לטבלת גיבוב בה יש 2n איברים – פתרון התנגשויות לכן לינארי.*

|  |  |  |  |
| --- | --- | --- | --- |
| חיפוש סוף מערך המס' )המקום ה- n (:כל עוד לא עברנו את z קפיצות אינדקס i^2 . הגענו ל- ∞ , z לא במערך. עברנו את z – חיפוש בינרי על הקטע האחרון. )מקרה גרוע z לא במערך( | O(lgn) | חיפוש z במערך אינסופי עם n איברים ולאחריהם הערך ∞. המערך ממוין עולה | 1 |
| מחפשים x+y+w=z . לכל איבר w ב-A נחפש זוג x+y שמקיימים x+y=z-w ושהם שונים מ-w . ניתן לממש לולאה שרצה על ערכי w ועבור כל ערך להפעיל את אלג' מסעיף 2 )חפוש זוג( | O(n²) | חיפוש 3 איברים ממשיים שסכומם z במערך ממוין A | 3 |
| לכל j<n<1 רצים עם אינדקס 2 אינדקסים כרגיל . השוני הוא שהאיבר j בין שני האינדקסים ולכן אם הגענו עם אחד האינדקסים ל-j עוצרים ומקדמים את j . | O(n²) | חיפוש 3 אינדקסים i≤j≤k≤n≤1 כך שמתקיים A[i]+A[k]=2A[j] המערך ממשיים ממויין | 4 |
| הכנסה לטבלת גיבוב בזמן לינארי. מעבר בלולאה לינארית על איברי הסדרה ובכל איטרציה נחפש בטבלת הגיבוב את האיבר מנת החלוקה של z באיבר הנוכחי. )מקרה גרוע שבו כל האיברים מגובבים לתא אחד( | O(n) תוחלת | חפוש זוג שמכפלתו z במערך ממשיים לא ממויין | 6 |
| שלמים וחסומים מאפשר שימוש במיון מנייה בזמן לינארי ואז עובדים בשיטת שני האינדקסים. | O(n) | מציאת זוג שלמים שסכומו z ושכל איברי המערך קטנים מ-z . המערך לא ממויין. | 7 |
| פתרון 9.3.8 בספר . להעתיק מחוברת התרגילים את עיקרי הדרך לפתרון לספר. | O(lgn) | מציאת ערך מיקום של איחוד המערכים הממוינים A ו-B בגדלים זהים | 8 |
| מבוסס על 9.3.8 . להעתיק מחוברת התרגילים את עיקרי הדרך לפתרון לספר . נדרש O(lg(max(m,n))) |  | כנ"ל – גדלי המערכים שונים m , n | 9 |
| גרסה של חפוש בינרי עם תנאי מותאם לאיזור הבדיקה לפי הנתון בתרגיל | O(lgn) | חפוש אינדקס נק' מינימום במערך שחלקו הראשון ממויין עולה וחלקו השני ממויין יורד. יש גרסאות שונות לתרגיל . | 10 |
| הרעיון המרכזי- אם יש איבר רוב הוא חייב להיות החציון.  לכן נמצא את החציון של המערך ע"י select . נבדוק אם הערך שקיבלנו בחציון חוזר יותר מ- n/2 פעמים | O(n) | מציאת איבר רוב במערך A )ערכו מופיע יותר מ- n/2 פעמים במערך( | 11 |
| אילו המערך היה ממויין, כל מופעי z היו ברצף. התחלת הרצף הייתה אחרי המיקום n-3k . רצף כזה באורך גדול מ-k חייב לכלול את המיקום n-2k או n-k . ולכן נמצא ע"י select את הערך במיקום n-2k ונבדוק אם ערך זה מופיע לפחות kפעמים. אם כן מצאנו את z . אם לא נבדוק כנל לגבי הערך במיקום n-k. | O(n) | מערך A בגודל n של ממשיים . K טבעי המקיים 3k<n . מצא z שעבורו :A מכיל n-3k איברים קטנים מ-z . וגם z מופיע יותר מ-k פעמים ב-A. | 13 |
| נניח עומד לרשותנו מערך בגודל 2k . בכל פעם נכניס אליו את k האיברים הבאים. אחרי הכנסת 2k האיברים הראשונים למערך נבצע select למציאת ערך המיקום ה- k ונעשה סביבו partition . קיבלנו בצד אחד של המערך את k הגדולים. שוב נחליף את k הקטנים בחדשים מהקלט ונחזור על שוב על הפעולות הנל עד לסיום רשימת הקלט. סהכ בצענו n/k פעמים פעולות שכל אחת O(k) ובסה"כ O(n) . מיון גדולים klgk | O(n+ klgk) | מציאת kהאיברים הגדולים מרשימת קלט בגודל n באמצעות זכרון בגודל O(k) | 14 |
| קוראים את כל הרשימה למערך בגודל n . עושים select על ערך המיקום ה- n-k וסביבו מבצעים partition | O(n) | כנל עם זכרון בגודל O(n) | 15 |
| אפשר להראות ע"י עקרון שובך היונים. פתרון פורמלי : נסמן b0,b1,..,bn את הסדרה שלנו לאחר מיון נגדיר סדרת הפרשים 1(-di=bi-b(i .נשים לב ש (M-m)/n הוא ממוצע של איברי סדרת ההפרשים ולכן בכל קבוצת האיברים חייב להיות איבר שקטן מן הממוצע או שווה לו כלומר : di≤(M-m)/n מכאן נובעת הטענה. |  | נתונה סדרה a0, a1, ..,an . ערך האיבר המינימלי m וערך המקסימלי M הוכח שקיימים שני איברים  . |x-y|≤(M-m)/n המקיימים x,y | 16 |
| חלוקת סדרת המס' סביב החציון ולחפש את שני האיברים באופן רקורסיבי בחצי שבו ההפרש בין המינימום למקסימום קטן יותר. ההפרש הממוצע )בין האיברים הממוינים( בקבוצה זו קטן יותר מההפרש הממוצע בקבוצה המקורית, ולכן ע"פ סעיף קודם יש בה שני איברים שההפרש ביניהם עונה לדרישה .ממשיכים כך עד לתת מערך בגודל 2 . | O(n) | מציאת x,y מסעיף קודם | 17 |
| 2 דרכים: א. מיון כל התת קטעים לפי נק' ההתחלה שלהם ai . נבדוק עבור i≤n≤2 אם 1(-ai<b(i אז מחזירים יש חיתוך , אחרת אם סיימנו את הלולאה מתקיים קטעים זרים. ב. מיון כל הנקודות ai,bi לפי ערכן )אם ai=bj להקדים את ai במיון(.,סה"כ 2n ערכים ממוינים x1,x2..x2n. לכל ערך נשמור את סוגו )סוג a או סוג b( . נבדוק מ-1 עד n את x(2i) ו- 1(-x(2i אם הם לא מאותו קטע נחזיר יש חיתוך . | O(nlgn) | האם יש 2 תת קטעים שנחתכים בסדרת תת קטעים [ai,bi] של קטע [1,0] | 18 |
| נבנה מערך של 2n כל תא מכיל שדה ערך ושדה סוג ) ai או bi (. מיון ) במיון ערמה לדוגמה( המערך לפי שדה הערך.נעבור על המערך משמאל לימין ונשמור את מס' נק bi שעברנו במונה B עד שנגיע לנק' מסוג ai ואז נעדכן את מס' הקטעים הזרים בערכו של B . | O(nlgn) | ספירת תת קטעים זרים זה לזה בסדרת תת קטעים [ai,bi] של קטע [1,0] | 20 |
| נמיין את הקטעים לפי נק' התחלה ai ) הערה : אם שני קטעים מתחילים באותה נק' צריך לשים קודם את נק ההתחלה של הקטע הארוך(. נשמור משתנה bmax שנאתחל ל- b1 . לכל i מ-2 עד n נבצע: נבדוק אם קטע i מוכל ע"י אם bi≤bmax אז קטע i מוכל אחרת bmax←bi | O(nlgn) | בסדרת תת קטעים [ai,bi] של קטע [1,0] האם כל אחד מהקטעים מוכל בקטע אחר. אפשר להשתמש בזכרון עזר בגודל O(n) . ניתן להניח שהנק 'מאוחסנות ב-2 מערכים ai ב- A ו- bi ב-B | 21 |
| נכניס למערך ונמיין. נסרוק את המערך: בכל שלב נחזיק את הקצה הימני של הקטע המאוחד – אם הקטע הבא מתחיל לפני נק' זו אז מדובר באיחוד ונמשיך, אחרת נתחיל קטע מאוחד חדש . | O(nlgn) | המרת סדרת קטעים לסדרת קטעים זרים- כל קטע חדש הוא איחוד של כמה תת קטעים מקוריים | 23 |
| 0 לא כלול ולכן אפשר להשתמש בו בשגרת partition כאיבר הציר ולהפעיל את השגרה שתסדר כנדרש | O(n) | סדרת מספרים שונים מאפס- נדרש לסדר שליליים לפני חיוביים. בסיוע זכרון נוסף בגודל קבוע. | 24 |
| נשתמש במיון בסיס, בבסיס k=n למערך השמאלי. ובבסיס k=m למערך הימני. בשמאלי: ²d=1+lgn )לוג בסיס n ( נותן 3 ספרות . בימני ³d=1+lgm )לוג בסיס m ( נותן 4 ספרות. ולפי : O(d(N+k) נקבל למערך השמאלי O(3(m+k))=O(m+n) ולימני O(4(n+k)=O(n+m) . לבסוף נמזג עם merge גם O(m+n) | O(m+n) | מיון מערך p באורך m+n שלמים .m הראשונים בתחום [²n..1] ו-n האחרונים בתחום [³m..1] | 27 |
| מיון בסיס בבסיס n ,כמו ע '126 במדריך. ניתן להתייחס לתחום העליון כ- n²+n ולחשב רגיל את d יוצא 3 | O(n) | מיון סדרה של n שלמים בתחום [1-n…n²+n] | 28 |
| בונים את העא"ש בהתחשב בכפילויות של המפתחות. מתקבל עץ בגודל O(lg²n) שגובהו  O(lg(lg²n))=O(lglgn) ןאח"כ פשוט מבצעים סריקה תוכית על lg²n הצמתים. |  | מיון מערך A[1..n] ממשיים. לא קיימים במערך יותר מ- lg²n ערכים שונים. ב- O(nlglgn) | 29 |