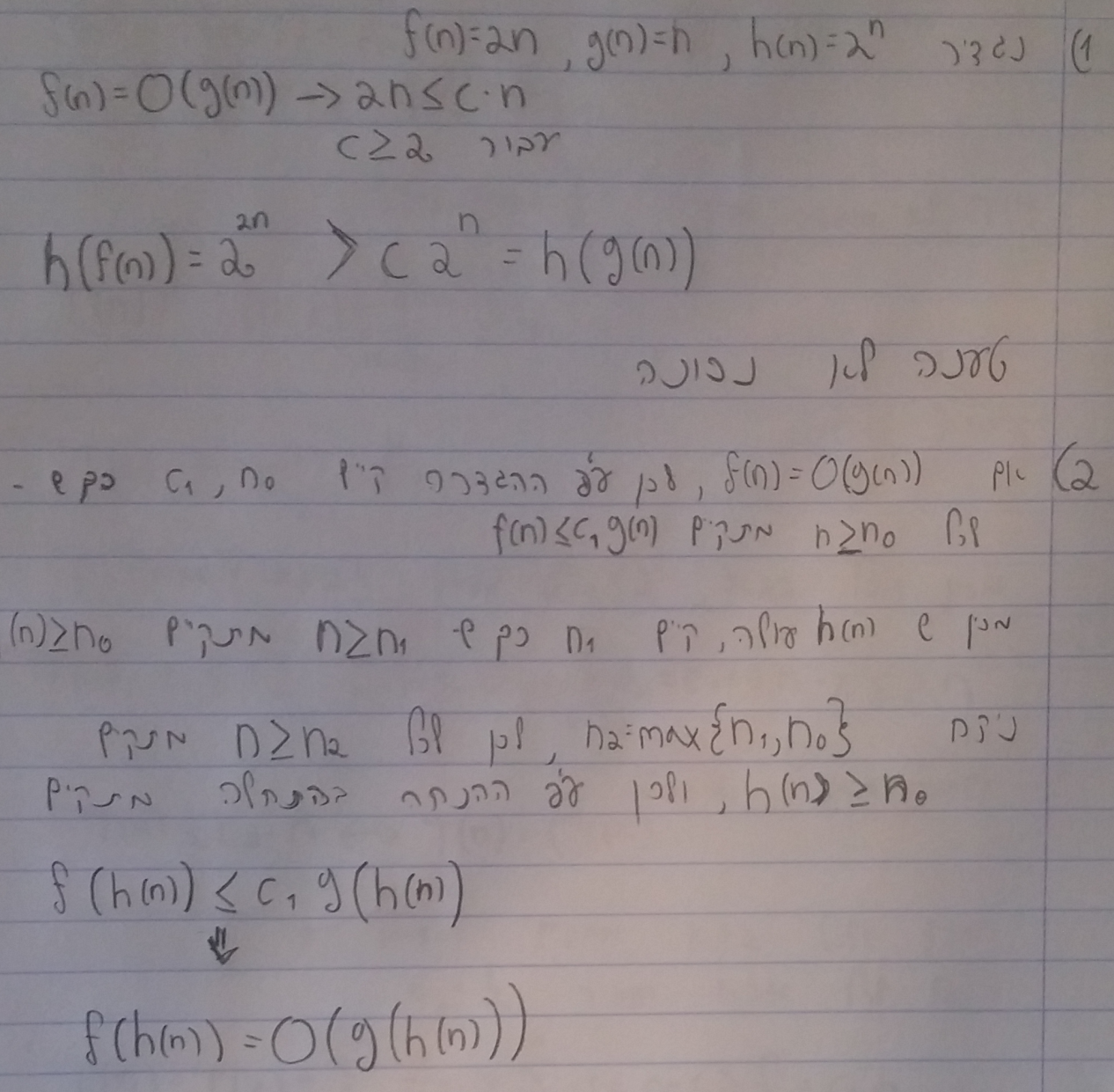
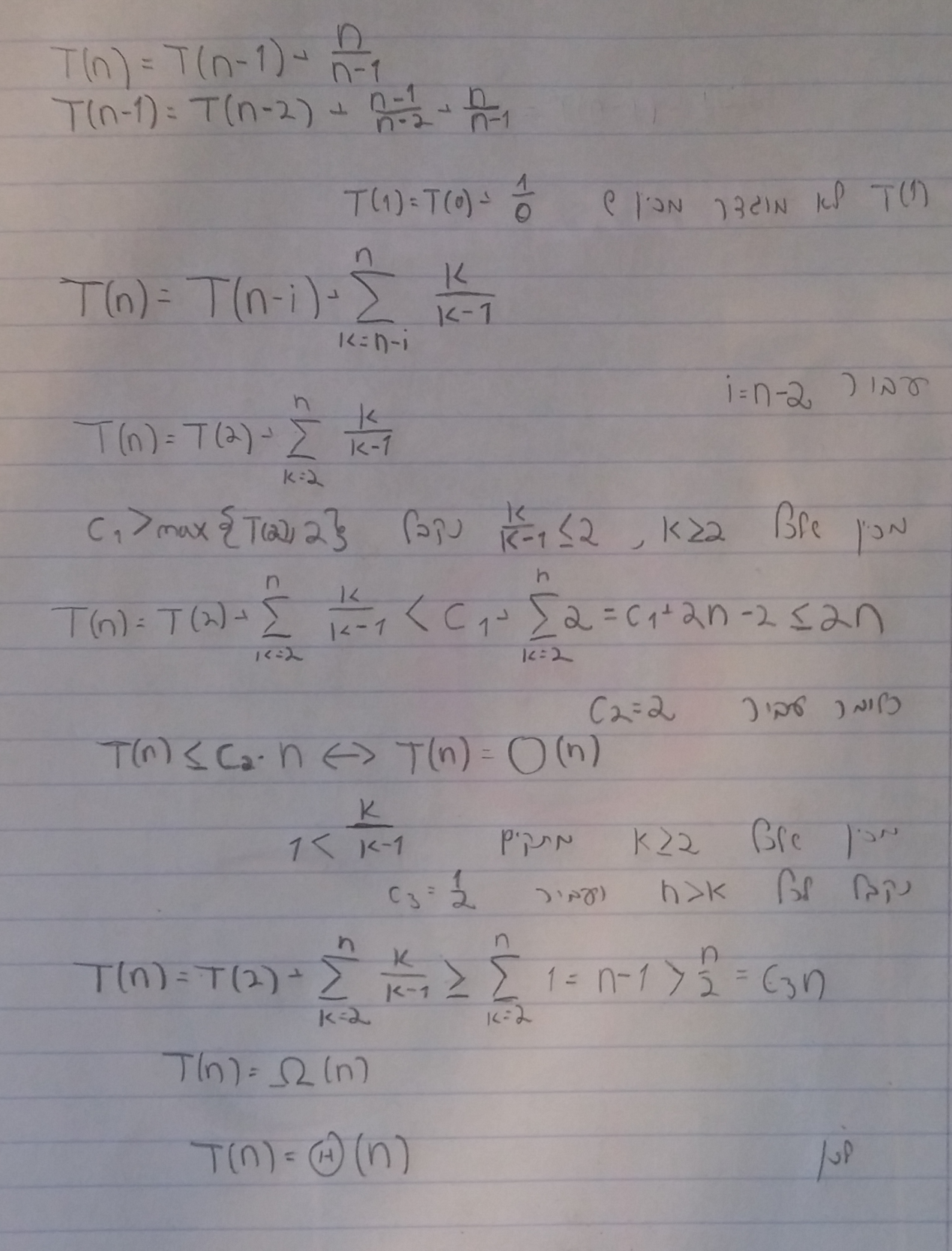
שאלה 1

א-

ב-



שאלה 2

1. תהיה f:N🡪N עולה ממש. יש לנו גישה לoracle אשר מקבלת מספר n טבעי ומחזירה את f(n) בסיבוכיות זמן קבוע.

יהי mϵN, נמצא חסן תחתון לביטוי f(m).

עבור 1, הערך המינימלי של f(1) הוא 1, מכיוון שf עולה ממש, נובע שעבור 2

1=f(1)<f(2). במקרה המינימלי, f(2)=2, כך ניתן להמשיך באינדוקציה לכל mϵN, ולהסיק שלכל mϵN, מתקיים m<=f(m).

נתאר אלגוריתם אשר מקבל מספר טבעי n, ומוצא האם קיים kϵn, כך שf(k)=n , יהי nϵN

נמצא את שני החסמים, העליון והתחתון לערכי k עבורם יכול להתקיים f(k)=n.

נניח בשלילה כי קיים kϵN כך ש n<k, וגם f(k)=n. מכיוון שf עולה ממש

f(n)<f(k), ע"פ n<k, ולכן f(k)=n>f(n). סתירה! לכן לכל nϵN n<=f(n) .

לכן תחום ערכי k עוברם יכול להתקיים f(k)=n הוא 1<=k<=n.

האלגוריתם יקבל את n, יגדיר את גבול החיפוש הימני, r🡨n, בהתאם למה שראינו, ואת גובל החיפוש השמאלי l🡨1, האלגוריתמיים ישתמש בלולאת while. בכל אינטרקציה של הלולאה האלגוריתם יגדיר m🡨└[(r+l)/2)] ┘

ויבדוק עם f(m)=n אזי יחזיר את m כפי שנידרש.

הבדיקה של ערכי f היא באמצעות oracle.

אם n<f(m) האלגוריתמיים יצמצם את גובל החיפוש הימני: r🡨m, מכיוון שלכל m<c יתקיים n<f(m)<f(m)<f(c), ולכן אין טעם לחפש בדווח ערכים זה את k.

אם f(m)<n האלגוריתמיים יצמצם את גובל החיפוש השמאלי: l🡨m

מכיוון שלכל c<m לפי ש-f עולה ממש מתקיים f(c)<f(m)<n ולכן אין טעם לחפש בדווח ערכים זה את k.

האלגוריתם ירוץ על הלולאה עד אשר f(m)=n או r=l.

צריך לשים לב שבהכרח אחד מביניים מתקיים מכיוון שאם קיים k כנדרש הוא יישאר בגובלות הגזרה l<=k<=r ולכן בהכרח נגיע אליו.

אם לא קיים k, מכיוון שאנו תמיד מצמצים את תחום החיפוש בחצי כל פעם ובנוסף יש מספר סופי של ערכים בין 1 ל-r בהכרח שבסופו של דבר יתקיים l=r מכיוון שאחרי הלולאה בהכרח r=l, מכיוון שתנאי העצירה f(m)=n נמצא בגוף הלולאה, נבדוק אם f(r)=n נחזיר את r, אחרת לא קיים k שמקיים f(k)=n.

סיבוכיות המקום קבועה.

סיבוכיות זמן הריצה

T(n)=T(n/2)+d=O(lgn), dϵN

1. לא מספיק לדרוש שf עולה במובן החלש, בשביל למצוא את k אנו מוכרחים לרוץ על ערכים "חשודים" ולמצאו את ערכי f עליהם בהכרח נוכל לחפש מספר סופי של ערכי k חשודים. מכיוון שאין מידע על הפונקציה f (לאלגוריתם). ניתן תמיד להגדיר פונקציה f שבכל הערכים שאותם האלגוריתם יבדוק, הפונקציה תמיד תהיה קבוע, במילים אחרות עבור:

f(k)=1 וn=2 לא נוכל לבדוק מספר סופי של ערכי k ולהוכיח שלא קיים k עבורו f(k)=2, לכן לא מפסיק לדרוש שf תעלה חלש.

שאלה 3

מבנה הנותנים s היה עץ אדום שחור עם הרחבה לערכי מיקום.

Insert: תתבצע בדיוק כמו השגרה RB-INSERT של עץ אדום שחור.

Find: תתבצע בדיוק כמו השגרה Tree-Search של עץ אדום שחור.

OSRange: נשתמש בOS-Rank לחפש את הערך המיקום של Y, במידה ולא קיים נחפש את האיבר המקסימלי שקטן מימנו, לאחר מכאן נחפש את X במידה ולא קיים אז את האיבר המינימלי שגדול מX.

כעת יש לנו את 2 ערכי המיקום שלהם, נבדוק שיש בנהיים לפחות m איברים y-x, אם אין נחזיר null אחרת נשתמש בselect כדי למצוא ולהחזיר את ערך המיקום הrank[x]+m-1.

שאלה 4

א-

Int dsum🡨depthsum(root,d)

int depthsum(T, d)

if (d==0)

return T🡪key;

else

return (depthsum(T🡪lchild,d-1))+(depthsum(T🡪rchild,d-1));

ב-

if d=1+treeHeight(T(

return T🡪key;

else

return (heightsum(ptr->lchild,d))+(heightsum(ptr->rchild,d));

גובה העץ, הוא כמה עלים יש מתחתיו, שכן הגובה נקבע מלמטה למעלה, לכן לכל צומת נבדוק באופן רקורסיבי מה הגובה שלה, כלומר מה המרחק המקסימלי בינה לבין העלים שלה, נוסיף אחד לתוצאה זו עבור אותה הצומת, כי הגובה מתחיל מ1, נבדוק עם המספר שקיבלנו שווה לגובה המבוקש, אם כן נסכום את האיבר לסכום שלנו, אחרת נרד במורד העץ ונבדוק לצמתים שמתחתיו,

זמן הריצה בשני האלגורייתם הוא Θ(n),מכיוון שאנחנו עברים על כל אברי העץ בשני האלגוריתמיים.

גם האלגוריתמיים אשר בדוק מה הגובה של אותה צומת רץ ב Θ(n).

סיבכויות המקום היא Θ(1) כי השתמשנו רק במשתנה עזר נוסף אחד.

שאלה 5

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]

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).