**יבש 3**

מגישים: נמרוד קדיש ואלכס בלגודרסקי

**שאלה 1**

מבנה הנתונים יהיה עץ דרגות AVL בעל k צמתים("עץ הצבעים") ולכל צומת יהיה עץ AVL("עץ הכדורים") בעלי מצביע לאיבר המקס' בעץ. בעץ הדרגות האיברים הם הצבעים 1,2,..,k והם יהיו מסודרים בעץ לפי הסדר (כלומר עבור סיור Inorder נקבל את הצבעים בסדר 1,2,…,k). המידע הנוסף שנשמור בעץ הוא השדה (Xi)∆ עבור כל צומת Xi ,שמייצג את ההפרש בין ערך הצומת לערך צומת האב (כאשר הערך הוא הנפח), ומידע נוסף maxVol שהוא הנפח המקס' בתת העץ(כולל עצמו) של כדור עבור צבע i.

בכל עץ AVL פנימי – עץ הכדורים - נשמור את הכדורים מצבע מסויים i מויינים לפי הנפח שלהם.

נתאר את הפעולות ונראה כי הם עומדות בדרישות הסיבוכיות:

* Init(k) : ניצור ונאתחל את מבני הנתונים הבאים:

-ניצור עץ בינארי כמעט שלם או שלם בעל k איברים על פי האלגוריתם שהוצג בתרגול, ונאתחל לכל צומת Xi את(Xi)∆ ואת maxVol להיות שווים ל-0. סיבוכיות הפעולה הינה O(k).

-לכל צומת בעץ הדרגות ניצור עץ AVL ריק .O(k)

סה"כ סיבוכיות הפעולה הינה O(k) כנדרש.

* AddBall(i,v) : נוסיף כדור למבנה בצבע i בעל נפח v:

-נחפש תחילה את הצבע הi- בעץ הצבעים בעזרת find(i) פעולה זו לוקחת O(log(k)), נסכום על מסלול החיפוש שלו את כל (Xi)∆ עבור כל צומת Xi במסלול החיפוש וניגש לעץ הכדורים שלו ושם נוסיף איבר חדש בעל נפח d v+(O(log(n) במקרה הגרוע). (כאשר d הוא סכום כל (Xi)∆ במסלול החיפוש אל הצומת)

-נעדכן את מצביע max בעץ הכדורים ע"י סיור "ימינה עד הסוף" בעץ. O(log(n))

-בעץ הצבעים, נמשיך ונעלה במעלה מסלול החיפוש עד השורש, ולכל צומת במסלול, נעדכן את הmaxVol במידת הצורך. נשים לב כי אנו מסיירים על מסלול החיפוש בלבד ותיקון המידע הוא רק עליו ולא משנה ערכים אחרים בעץ ולכן O(log(k)).

סה"כ סיבוכיות הפעולה הינה O(log(k))+ O(log(n))= O(log(k)+log(n)) כנדרש.

* PopMax: נרצה להוציא את הכדור בעל הנפח המקס':

-נסייר מן השורש כלפי מטה לפי הערך maxVol באופן הבא:

נסייר בצומת הנוכחי ונבדוק מי מהבנים שלו בעל ערך maxVol כמו של הצומת הנוכחי, ונמשיך אליו. כך נתקדם באופן רקורסיבי עד שנגיע לצומת שהmaxVol של שני בניה קטנים מהmaxVol שלה(עבור בן שלא קיים הmaxVol מוגדר להיות 0), הרי הגענו אל הצומת שמסמל צבע בעל נפח הכדור המקס' במבנה. אורך מסלול החיפוש הוא לכל היותר O(log(k)) ולכן זוהי גם הסיבוכיות. כעת ניגש לעץ הכדורים הרלוונטי O(1) ונסיר ממנו את הכדור המקס' בעזרת המצביע O(logn) ועל ידי סיור ימינה "עד הסוף" ונדאג לעדכן את הmax החדש O(log(n)). בנוסף נעדכן על מסלול החיפוש, בעץ הצבעים, מהאיבר עד לשורש כלפי מעלה את המקס' החדש, במידה וקיים, בהתאם. (O(log(k).

סה"כ סיבוכיות הפעולה הינה O(log(k))+ O(log(n))= O(log(k)+log(n)) כנדרש.

* Increase(I,j,d) : נבצע את האלגוריתם הבא:

-אם i>j נחזיר שגיאה ונסיים.

-אחרת נחפש בעזרת שני איטרטורים את הצמתים i ו j בעץ הצבעים. O(log(k)).

-כעת נחפש את האב הקדמון הקטן ביותר המשותף שלהם – כלומר הצומת האחרון במסלולי החיפוש של שניהם שהינו משותף לשניהם (ניתן למצוא זאת למשל ע"י שמירה של מסלולי החיפוש של שניהם במערך ובמעבר פשוט על המערך למצוא את הצומת הנ"ל - אורך המערך הוא לכל היותר אורך מסלול החיפוש בעץ ולכן O(log(k)) )

-כעת נוסיף ל(Xc)∆ עבור Xc ,האבהקדמון המשותף שמצאנו הרגע, את הערך d ונמשיך להתקדם על מסלולי החיפוש של i ואל j באופן הבא:

- במסלול החיפוש של j:

\*לכל צומת Xk שבה k>j נחסיר מ(Xk)∆, את d. לאחר ההחסרה, נוסיף לצומת הבא במסלול החיפוש d ל(Xl)∆ עבור Xl האיבר הבא במסלול החיפוש.

\*עבור צומת Xk שבה k=j נחסיר מ(Xk->right)∆, במידה וקיים, את d. ונסיים.

-במסלול החיפוש של i:

\*לכל צומת Xk שבה k<i נחסיר מ(Xk)∆, את d. לאחר ההחסרה, נוסיף לצומת הבא במסלול החיפוש d ל(Xl)∆ עבור Xl האיבר הבא במסלול החיפוש.

\*עבור צומת Xk שבה k=iנחסיר מ(Xk->left)∆, במידה וקיים, את d. ונסיים.

-בנוסף, נעלה שוב מצומת האב הקדמון המשותף ונעדכן את ערך maxVol החדש כלפי מעלה אל השורש בהתאם. (כלומר maxVol של הצומת ובנוסף d הנתון שקיבלנו , "נחלחל" כלפי מעלה).

נשים לב שסיבוכיות הפעולה הינה O(log(k) מכיוון שמבצעים סיור על שלושה מסלולי חיפוש בעץ בגובה O(log(k)) ומס' פעולות קבוע O(1) לכל צומת במסלול החיפוש.

נכונות הפעולה: השמורה בעץ הנ"ל היא שלכל צומת Xi ההפרש בין ערכו לבין ערך האב הינו (Xi)∆. לכן דרוש שסכימת ערכי ה∆ במסלול החיפוש של צומת הינה הגודל המקורי של הכדורים בתת העץ + ההפרש שנוסף לאחר מס' פעולות מסוים של addToRange. כלומר אם ברצוננו לדעת את גודלו של כדור בצבע מסוים, על ידי מעבר וסכימה של ערכי ה∆ במסלול החיפוש שלו נוכל לדעת את גודלו העדכני. נשים לב בנוסף כי עבור האלגוריתם המתואר למעלה, צבעים שאינם בטווחים המגודרים, עבורם סכימת ערכי ה∆ במסלול יהיה שווה ל-0 – כלומר נשמרת השמורה וגודלם אינו משתנה כנדרש.

בנוסף, שמורת המקס' בעץ זהה לשמורת הmax שראינו בעץ הדרגות בתרגול – כלומר, כל צומת מחזיק את ערך המקס' של תת העץ-שלו(כולל הוא עצמו), השינוייים שמתבצעים משפיעים רק על צמתים במסלול החיפוש שלו כלפי מעלה (=כאלה שהוא חלק מתת העץ שלהם) ולכן העדכון נעשה אך ורק על מסלול החיפוש שלו – דבר שאינו פוגע בסיבוכיות הפעולה.

**שאלה 2**

1. מס' האיברים בערימה עם זנב מטיפוס k:

עבור עץ שלם בגובה h, מס' הצמתים נתון ע"י n=2^(h+1)-1, ולכן עבור k=h-1:

2^(k)-1+1\* = 2^k

-כאשר \* הוא צומת הזנב.

1. נבדוק מיהו השורש המינימאלי מבין 2 הערימות, הוא יהיה השורש החדש בערימה....

**שאלה 3**

1. נשתמש ברשימת ערבול עם ערבול אוניברסלי ועם chain-hashing, צמתי הרשימות יכילו את המספר המוכנס לטבלה ותא counter שיספור כמה פעמים הוכנס האיבר לטבלה. בנוסף נחזיק תא שיקרא max. בכל פעם שנכניס איבר לטבלה נוסיף צומת חדש לשרשרת אם האיבר לא קיים ואם הוא קיים (נדע זאת ע"י השוואה מהסוג = ) נגדיל את הcounter ב1. סיבוכיות הזמן של בניית טבלה כזאת הינה O(n) בממוצע הסתברותי כאשר הטבלה היא בגודל n (כלומר פקטור העומס הוא O(1) ). כעת נעבור על כל הצמתים ברשימות הנמצאות בטבלת הערבול (יש n צמתים כאלה במקרה הגרוע), בסיור זה נמצא את האיבר בעל הכי הרבה חזרות ע"י השוואה של תא counter ועדכון max בהתאם (תא max שומר את מס' החזרות המקס' ואת ערך האיבר שחוזר מס' מקס' של פעמים). לבסוף נבצע בדיקה האם max.accurrences>n/3. אם התנאי מתקיים נחזיר את המס' ששמור בmax, במידה והתנאי לא מתקיים נחזיר הודעה מתאימה. כנזכר למעלה, גודל הטבלה הוא n ובנוסף מס' הצמתים הכולל ברשימות הוא n ולכן אנחנו עומדים בדרישות סיבוכיות המקום O(n).
2. נשים לב שאם איבר מופיע יותר מn/3 פעמים במערך, אז במערך הממויין הוא בוודאות יהיה או באינדקס n/3 או באינדקס 2n/3 בדומה לטענה מהתרגול. לכן, נשתמש באלגוריתם select למציאת האיברים במקומות n/3 וה2n/3. אם איברים אלו שווים, אז האיבר הזה חוזר n/3 פעמים, נדפיס אותו. אם האיברים האלו שונים, לכל אחד מהם נסרוק את המערך ונספור את מס' החזרות של כל אחד מהם. אם אחד מהם מופיע יותר מn/3 פעמים, נדפיס אותו. אם אף אחד מהם לא מופיע יותר מn/3 פעמים, נחזיר הודעה מתאימה.

סיבוכיות האלגוריתם select היא O(n) וסריקת המערך מס' קבוע של פעמים הינה גם בסיבוכיות O(n), ולכן בסה"כ סיבוכיות האלגוריתם המוצא הינו O(n). נשים לב כי סיבוכיות המקום של select ושל 2 המערכים לכל היותר הינם O(n) ולכן אנו עומדים בדרישת סיבוכיות המקום.

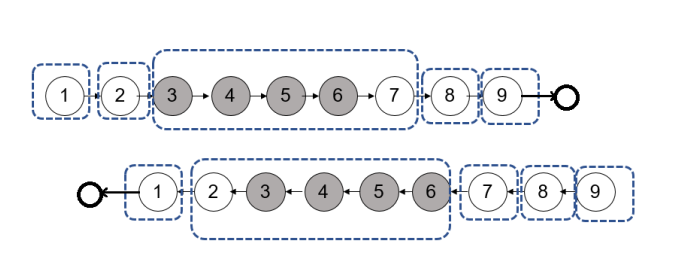
שאלה 4

מבנה הנתונים יכיל שני Union-Find, כאשר כל אחד מהם יתפקד בשיטת Master-Close כמוצג בתרגול. כל צומת תייצג חדר ממוספר לפי מספר העובד שלו כך שאם הצומת כבויי אזי החדר תפוס, בנוסף תהיה צומת שורש שאינה תייצג חדר.

ב Union-Find הראשון שיסומן בR ה Master של כל קבוצה יהיה החדר הפנוי הראשון מימין.

בUnion-Find השני שיסומן ב L ה Master של כל קבוצה יהיה החדר הפנוי הראשון משמאל.

(\*הערה: צומת השורש של הרשימות אינה יכולה להכבות, על פי מימוש Master-Close). מצורפת תמונה של מבנה הנתונים המתואר:



Init(): נאתחל את שני ה Union-Find(עם כיווץ מסלולים ואיחוד לפי גודל) בדומה לשיטת Master-Close במוצגת בתרגול, כך שכל צומת תהיה בקבוצה נפרדת. כלומר, כל חדר הוא החדר הפנוי הקרוב ביותר של עצמו. בנוסף, נאתחל את צומת השורש להיות צומת פנוי. סה"כ O(2n)=O(n) כנדרש.

WorkerArrival(k): בשני ה Union-Find נבצע Close(k). (פעולה זאת תאחד בR את הקבוצה שהMaster שלה הוא k עם הקבוצה שהMaster שלה הוא החדר הפנוי הראשון שמספרו גדול מk-, ב-L תתבצע באופן סימטרי).

ClosestRoom(k): בשני ה Union-Find נבצע R.Master(k) ו L.Master(k). אם החדר הקרוב ביותר הוא חדר השורש, אזי אין חדרים פנויים (משמאל ומימין). לאחר מכן, נחשב את המרחק המינימאלי של כל אחד מהחדרים הפנויים מk. החדר שעבורו מתקבל המרחק המינימלי הוא החדר הפנוי הקרוב ביותר לk, נחזיר את מס' חדר זה.

נכונות: עבור כל שלב, כל מבנה של Union-Find מחזיק קבוצות שהMaster שלהן הוא החדר הפנוי הקרוב ביותר. בביצוע WorkerArrival(k), אנו מחדים את הקבוצות כך שלקבוצה החדשה יש את החדר הפנוי הקרוב ביותר, זהו החדר הפנוי הקרוב ביותר לכל החדרים שנמצאים בקבוצה החדשה. בעזרת שימוש בפונקציית Master, אנו מקבלים את השורש של הקבוצה – כלומר את החדר הקרוב ביותר משמאל או מימין, ומציאת המרחק המינימלי תתן לנו את החדר הקרוב ביותר ללא תלות בכיוון.

סיבוכיות: לפי טענה מהתרגול הסיבוכיות המשוערכת של Master ו Close יחד הינה O(log\*(n)), (סיבוכיות המקום הינה O(n)).