שאלה 1

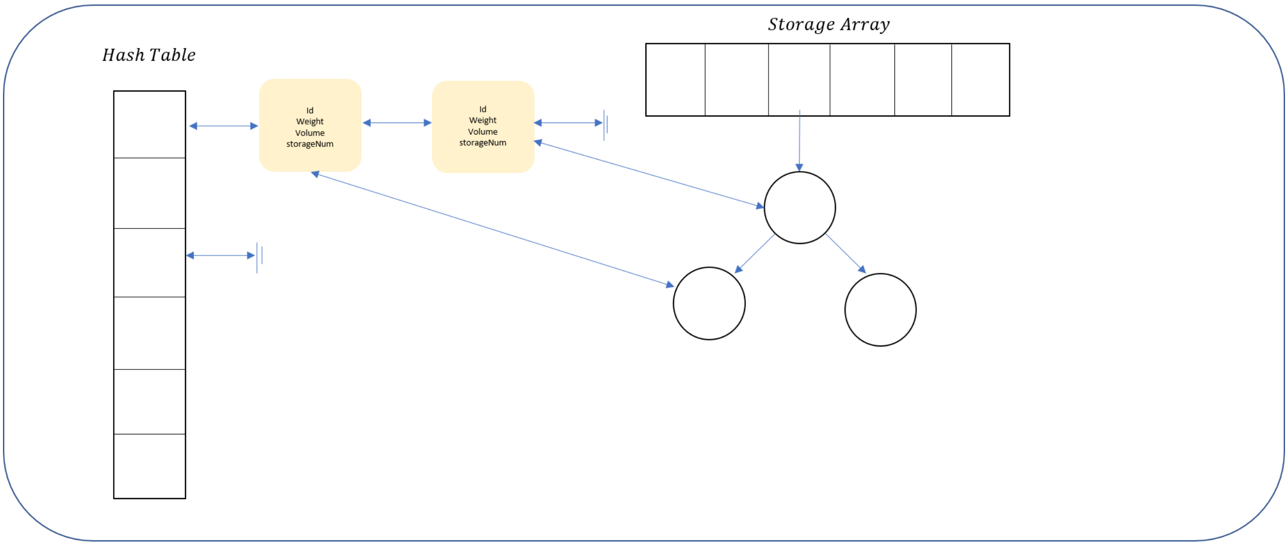
נשתמש במבני הנתונים הבאים:

- $Storage$ - מערך בעל k איברים המשמש לאיחסון המחסנים. כל מחסן מכיל מצביע לעץ דרגות AVL כאשר כל איבר הוא מצביע ל-Box בטבלת הערבול. עץ ה-AVL ממוין לפי הנפח. דהיינו, בכל צומת e, נפחם של האיברים בתת העץ השמאלי קטן מהנפח של ה-box המתאים לצומת e ונפחם של איברי תת העץ הימני גדול מהנפח של ה-box המתאים לצומת e.
- $BoxTable$ - טבלת ערבול המשמש להכנסת קופסאות (משתנה בשם Box). פונקציית הערבול תהיה מזהה הארגז מודולו גודל הטבלה.

המשתנה מסוג Box יכיל את השדות הבאים:

- id - מזהה הארגז
- $weight$ - משקל הארז.
- $volume$ - נפח הארגז.
- $storageNum$ - מספר המחסן אליו הארז משויך.

כמו כן נגדיר את הדרגה בעץ הדרגות AVL להיות זוג ערכים: כל צומת יחזיק בנוסף את המשקל המקסימלי בתת העץ הימני ואת המשקל המקסימלי בתת העץ השמאלי.



$Init(k)$ - נאתחל מערך ריק של k איברים במערך $Storage$ ובנוסף נקצה טבלת ערבול ריקה.

$AddBox(i, id, w, v)$ - ראשית, נבדוק בטבלת הערבול האם הארגז קיים לפי מזהה הארגז ע"י שימוש בפונקציית $Find$. במידה וקיים, נסיים את התוכנית - שימוש ב-$Find$ מתבצע ב-$O(1)$ משוערך.

אחרת, במידה ו-$1 \leq i \leq k$ ניצור קופסא המכילה את הפרטים הדרושים (מזהה, משקל, נפח והעץ אליו הוא משויך). נכניסו לטבלת הערבול על-ידי שימוש בפונקציית הערבול. כמו כן, נכניס מצביע (המתאים למחסן) ל-Box הנמצאת בטבלת הערבול ונעדכן את הדרגה המתאימה ע"י גישה לדרגות של הבנים של הצומת אשר הוכנסה. כעת, נעדכן את הדרגות אשר נמצאות במסלול במעלה העץ החל מה-Box שהכנסנו. כמו כן, על מנת שהוספת Box לטבלת הערבול תתבצע ב-$O(1)$ משוערך (דהיינו ש-$\alpha = \Theta(1)$ כפי שהוסבר בהרצאה) נבדוק ש-$m = O(n)$ כאשר n זה מספר האיברים ו-m זה גודל הטבלה. בפועל, כאשר יתקיים $2m \leq currSize$ כאשר $currSize$ זה מספר האיברים שקיימים בטבלת הערבול ברגע נתון נבצע הגדלת מערך פי 2 ובנוסף נכניס את כל האיברים (משתנים מסוג Box) לטבלת הערבול החדשה.

סה"כ הסיבוכיות של הכנסת מצביע לאיבר Box לעץ AVL תיקח $O\left(\log n\right)$ ובנוסף הכנסה של איבר Box לטבלת הערבול תיקח $O\left(1\right)$ ועדכון הדרגות עד שורש העץ תיקח גם כן $\log\left(n\right)$ (n הוא מספר הארגזים במבנה).

$TransferBox\left(i,j,V\right)$ - ניגש למחסן i ע"י גישה למצביע במערך המחסנים. נשמור משתנה אשר שומר מצביע ל-Box, נסמנו ב-tmp.
בתחילת הריצה נשמור את את שורש העץ ב-tmp. בכל צומת e בעץ ה-AVL נבצע את הפקודות הבאות באופן רקורסיבי:

- במידה ו-$V\left(e\right) > V$ נכנס לתת העץ השמאלי.
- אחרת, $V\left(e\right) \leq V$ ($V\left(e\right)$ הוא הנפח של המצביע ל-Box ו-V ניתן כפרמטר לפונקציה). נבדוק האם מתקיים: $W\left(e\right) > W\left(tmp\right)$.
  אם כן, נשנה את tmp להצביע ל-Box אשר e מצביע. נפריד למקרים:
  - במידה ומתקיים $W_{right}^{max} < Weight_{left}^{max}$ נכנס לתת העץ השמאלי (אנו בנוסף יודעים כי נפחם של איברי תת העץ השמאלי קטן בהכרח מ-$V\left(e\right)$.
  - אחרת, נכנס לתת העץ הימני.

$LocateBox\left(id\right)$ - ניגש לטבלת הערבול ונשתמש בפונקציית $Find$ (ב-$O\left(1\right)$ ממוצע על הקלט) ובכך נמצא את ה-Box בעל ה-id המתאים ונחזיר את המשתנה $storageNum$ המתאים לו.

שאלה 2

סעיף א'

נשתמש בתור בגודל k וברשימה מקושרת בה כל איבר הוא מצביע לערימת מקסימום בעל k איברים.

בנוסף עבור התור נשמור $currentSize$ אשר יהווה את מספר האיברים הנוכחי בתור ו-$maxSize$ שיהווה את מספר האיברים המקסימלי שניתן להכניס לתור. כמו כן, נשמור מצביע $currPtr$ לתא ברשימה המקושרת (אשר ישתנה תוך כדי הריצה). כמו כן, נגדיר מצביע נוסף $queuePtr$ אשר יצביע על תא במערך אשר נשתמש איתו ב-$CallNext()$

$Init\left(k\right)$ - נאתחל את התור ונאתחל רשימה ריקה ונתאחל את המצביע $currPtr$ ואת $queuePtr$ להיות $null$.

$AddToQueue\left(h\right)$ - נוסיף לתור אדם בעל גובה h ונגדיל את $currentSize$ ב-1. במידה ו$currentSize = maxSize$-, דהיינו התור מלא נוציא משם את כל האיברים ונבנה ערימה מקסימום. כעת נוסף איבר לרשימה המקושרת כך ש-$currPtr$ יצביע על תא זה. בהכנסת הערימת מקסימום הראשונה $queuePtr$ יצביע עליה. כמו כן בתא זה נצביע על הערימה שבנינו לעיל. בנוסף, נאפס את $currSize$. נשים לב כי בניית ערימה לוקחת $O\left(k\right)$ אבל מכיוון שפעולה זו מתבצעת רק לאחר הכנסת k אנשים נקבל כי סיבוכיות הזמן היא $O\left(1\right)$ משוערך.

$CallNext\left(\right)$ - ניגש למצביע $queuePtr$ ונבצע $FindMax\left(\right)$ - מתבצע ב-$O\left(1\right)$ ונוציא את האיבר בעל הגובה המקסימלי - מתבצע בסיבוכיות $O\left(\log k\right)$. לאחר מכן, במידה ויש עוד איברים בעץ עליו $queuePtr$ לא נעשה כלום. במידה והאיברים הסתיימו ניגש לערימת המקסימום הבא (למעשה התא מהרשימה המקושרת יוסר) ע"י גישה לאיבר הבא ברשימה המקושרת ב-$O\left(1\right)$. במידה ולא קיים תא נוסף ומתבצעת קריאה ל-$CallNext\left(\right)$ נסיק כי לא

קיימים k אנשים בתור ובמקרה זה הפעולה תיכשל.

סעיף ב'

נניח בשלילה שקיים מימוש של הפעולות מסעיף א' בו הפעולה $CallNext$ מתבצעת בסיבוכיות זמן $O\left(\log\log k\right)$ וזאת מבלי לשנות את שאר הפעולות. יהיו k איברים ממוספרים $n_1, n_2, \ldots, n_k$ (לא ממויינים). למעשה האיבר i יהוו את האיש בעל הגובה n_i. כעת נבצע $AddToQueue\left(n_i\right)$ לכל $1 \leq i \leq k$ - מכיוון שכל הכנסה היא $O\left(1\right)$ משוערך ואנו מבצעים k פעמים את הפעולה הזאת ולכן סיבוכיות הזמן של פעולה זאת היא $O\left(k\right)$. כעת, נקרא לפונקציה $CallNext\left(\right)$ ונקבל את האדם הגבוה ביותר. לאחר קריאה נוספת נקבל את האדם השני הגבוה ביותר. באופן איטרטיבי נקרא ל-$CallNext\left(\right)$ k-פעמים ובאופן זה נקבל k איברים ממויינים.

נבחין כי סיבוכיות הזמן מורכבת מהפעולות הבאות:

- הכנסת k אנשים לתור - $O\left(k\right)$.
- בניית ערימת מקסימום - $O\left(k\right)$.
- קריאה לפונקציה $CallNext$ k פעמים - מהנחת השלילה נקבל כי סיבוכיות הזמן היא $O\left(k \cdot \log\log k\right)$.

סה"כ, קיבלנו כי סיבוכיות הזמן לסידור k איברים היא $O\left(k \cdot \log\log k\right)$, בסתירה למשפט (אלגוריתם מיון מבוסס השוואות) שלמדנו בהרצאה לפיו מיון k איברים בכל הפחות ב-$k \log k$, דהיינו $\Omega\left(k \log k\right)$.

שאלה 3

סעיף א'

נשתמש במבנה הנתונים $Union - Find$. נקצה שני מערכים - מערך המכיל את n המתמודדדים ומערך המכיל n מפלגות שונות (כל מועמד הוא מפלגה בפני עצמו).

כמו כן, נשמור משתנה $numOfParties$ - המהווה את מספר המפלגות שקיימות ברגע נתון, אשר יאותחל בתחילה ל-n ומשתנה המכיל את מספר המתמודדים $numOfParticipants$.

$Init\left(n\right)$ -נבצע $Init$ של $UnionFind$ המכיל את איברים ואת הקבוצות כאשר כל איבר נמצא בקבוצה בגודל 1. כמו כן נעדכן את $numOfParties$ ואת $numOfParticipants$ להיות n.

$MergeParties\left(p_1,p_2\right)$ - נבצע $Union\left(Find\left(p_1\right),Find\left(p_2\right)\right)$ בסיבוכיות $\log^*\left(n\right)$ משוערכת כפי שלמדנו בהרצאה. כמו כן, נקטין את $numOfParties$ ב-1, שכן מספר הקבוצות קטן ב-1 לאחר שמבצעים איחוד מפלגות.

$IsSameParty\left(p_1,p_2\right)$ - במידה ו- $Find\left(p_1\right)=Find\left(p_2\right)$ נחזיר אמת, אחרת שקר. כמו כן, סיבוכיות הזמן היא $O\left(\log^*\left(n\right)\right)$ כפי שלמדנו בהרצאה.

$AveragePartySize\left(\right)$ - נחזיר את $\frac{numOfParties}{numOfParticipants}$. כמו כן, סיבוכיות הזמן במקרה זה היא $O\left(1\right)$.

סעיף ב'

לכל מועמד נוסיף משתנה נוסף בשם r אשר תחילה יאותחל (בפונקציית $init$) לאחד. כמו כן, נשמור משתנה אשר שומר את מספר המתמודדים בכל מפלגה ($numOfCurrentGroup$) (כאשר מבצעים איחוד מפלגות נסכום את מספרי המתמודדים בכל מפלגה ונשמור במפלגה שאותה אנו מאחדים - אז מספר המתמודדים החדש שיהיה במפלגה A יהיה סכום מספר המתמודדים של מפלגה A ומספר המתמודדים של מפלגה B).

כאשר אנו סוכמים את השדות לאורך מסלול ה-$Find$ תהיה שווה למיקום המועמד במפלגה בה הוא חבר.

ראשית, נסמן את האיבר בראש העץ של המפלגה p_1 ב-a. בנוסף, נסמן את האיבר בראש העץ של המפלגה p_2 ב-b.

על-מנת לבצע את $MergeParties\left(p_1,p_2\right)$ נפריד למקרים:

- במידה ומפלגה p_1 גדולה שווה ממפלגה p_2 (ההשוואה תיעשה לפי שדה "מספר המתמודדים במפלגה") נעדכן את הערכים הנוספים $r\left(a\right)_{new}=r\left(a\right)_{old}$, $r\left(b\right)_{new}=r\left(b\right)_{old}+numOfCurrentGroup\left(p_1\right)-r\left(a\right)_{new}$.
- אחרת, נשנה את הערכים הנוספים להיות: $r\left(b\right)_{new}=r\left(b\right)_{old}+numOfCurrentGroup\left(p_1\right)$, $r\left(a\right)_{new}=r\left(a\right)_{old}-r\left(b\right)_{new}$.

באופן זה, כאשר אנו רוצים את המיקום של מתמודד x כלשהו במפלגה כלשהי ניגש למתמודד ונסכום את השדה הנוסף r בכל צומת במעלה העץ עד אשר אנו מגיעים ל-$Find\left(x\right)$ - סכום זה יהווה את מיקום המתמודד במפלגה הנוכחית.

שאלה 4

ראשית, נסמן $value=|s|$. נבצע סיור $PostOrder$ כאשר בכל עלה נגדיר את הצלע המחוברת אלייה להיות $\left(value-currentValue+1,value\right)$. עבור כל עלה נעדכן את $value$ להיות $|s|-currentValue$. כעת, בכל עלייה בעץ ניקח את ה-$value$ המינימלי מבין כל הבנים ונגדיר מחדש $\left(value-currentValue+1,value\right)$ ונעדכן את $value$ בהתאם, באופן רקורסיבי.

נשים לב כי אנו עוברים על העץ בסיור $PostOrder$ ואנו מבצעים מספר פעולות ב-$O\left(1\right)$ בכל צומת ועל כן סיבוכיות הזמן היא $O\left(|S|\right)$, כנדרש.

שאלה 5

$Init\left(DNAs\right)$

ראשית, נבנה עץ סיומות מוכלל ודחוס עבור n רצפי ה-DNA - מתבצע ב-$O\left(|S|\right)$ ע"י אלגוריתם הקופסא השחורה. כמו כן בכל צומת נחזיק מערך בגודל n אשר יאתוחל ל-0 אשר בתא ה-i יהיה מספר העלים אשר מסתיים ב-$\$_i$ בתת העץ הנוכחי. כמו כן, נשמור $counter$ בכל צומת המכיל את מספר האיברים במערך השונים מ-0.

ע"י סיור $PostOrder$ נוסיף לעלים (אשר מסתיימים ב-$\$_i$) $counter++$. את שאר האיברים בעץ נעדכן תוך הסתכלות במערכים של הבנים באותו הצומת. בנוסף, נעדכן גם את ערכי ה-$counter$ בהתאם.

לצורך ביצוע הפונקציה $GetLongestpRep\left(p\right)$ נבצע את השגרה הבא כבר ב-$init$ על מנת לעמוד בסיבוכיות הזמן.

נסמן ב-$GetLongestpRepHelper\left(p\right)$ את הפונקציה הבאה:

ראשית, נחשב את $n\cdot p$ ובנוסף נאתחל שני משתנים $currentLength=0$ (אורך המחרוזת הנוכחית) $maxLength=0$ (אורך המחרוזת המקסימלית). כמו כן, נשמור מצביע שבסופו של דבר יצביע לתחילת המחרוזת הרצויה.

לאחר מכן, נבצע סיור $PostOrder$ כאשר בכל צומת נבדוק האם ה-$counter$ גדול/שווה או קטן מ-np. אם הוא גדול שווה מnp, באופן רקורסיבי, נבצע את הפונקציה שוב כאשר הצומת הנוכחית היא השורש. אחרת, נמשיך בסיור $PostOrder$. במשך הסיור נעדכן את $currentLength$ בהתאם ובמידה ואכן גדול מ-$maxLength$, נגדיל את אורך המחרוזת ואת המצביע כנדרש. לבסוף, נחזיר את המצביע לסוף המחרוזת.

סה"כ קיבלנו כי סיבוכיות הזמן היא $O\left(|S|\right)$.

נחזיק מערך arr בגודל n אשר בכל תא יחזיק מצביע לצומת בעץ הסיומות המוכלל.

נחזור על $GetLongestpRepHelper\left(p\right)$ n פעמים עם הקלטים $\frac{1}{n},\frac{2}{n},\ldots,\frac{n}{n}$ כאשר עבור הקלט $\frac{i}{n}$ נשים את ערך החזרה במקום ה-i. נשים לב כי יש n מחרוזות ובכך אנו מכסים את האפשרויות השונות ש-p מייצג יכול להיות.

סה"כ קיבלנו כי סיבוכיות הזמן היא $O\left(n\cdot|S|\right)$ ובנוסף סיור $PostOrder$ אשר לוקח $O\left(|S|\right)$. לכן נקבל כי סיבוכיות הזמן הכוללת היא $O\left(n\cdot|S|\right)$, כנדרש.

$GetLongestRep\left(p\right)$

ניגש למצביע במערך באינדקס ה-$p\cdot n$. כעת, נעלה במעלה העץ ונאחסן את האותיות בסדר הפוך (כי המצביע לסוף המחרוזת) עד אשר נקבל כי pn גדול מערך ה-$counter$ של הצומת הנוכחי בכל אחת מן האיטרציות.

לכן, סה"כ סיבוכיות הזמן היא למעשה מעבר על כל האותיות, דהיינו $O\left(S_p\right)$, כנדרש.