מבני נתונים – תרגיל רטוב 2

נמרוד קדיש ואלכס בלגודרסקי

18.06.17

תיאור מבנה הנתונים:

מבנה הנתונים הראשי ייקרא Xmen ולצורך מימוש הדרישות השתמשנו במספר מבני נתונים נוספים שיתוארו להלן:

- **Ranked AVL Tree** – מבנה נתונים הממומש עץ דרגות מסוג AVL – הממויין לפי דרגת הצמתים. כל צומת, פרט לדרגתו, מחזיק את ה-id שלו(יחיד – משמש לצורך השוואה והחזרה), את רמתו(POWER) ואת הרמות המקסימליות בתתי העץ השמאלי והימני שלו. לאחר כל פעולה בעץ נשמרת השמורה הבאה:
  בכל צומת מעודכנות הרמות המקסימליות של תתי-עציו. (פירוט בתיאור הפעולות)
  סיבוכיות מקום – $O(n)$
  כאשר n הוא מספר הצמתים.

- **Union-Find** – מבנה נתונים התומך בקבוצות זרות של מספרים, כאשר הוא מאותחל ע״י מספר האיברים n. משתמש בעצים הפוכים עם איחוד לפי גודל וכיווץ מסלולים.
  תומך בפעולות:
  - Find(x) – מוצא ומחזיר את נציג מחלקת השקילות שבקבוצה עם x.
  - Union(x, y) – מאחד את הקבוצות של מחלקות השקילות של x ו-y.
  - areConnected(x, y) – מחזיר משתנה בוליאני הבודק האם x ו-y נמצאים באותה מחלקת שקילות.

  סיבוכיות זמן הפעולות – כפי שלנמד בכיתה:
  O(n) – Init
  Find, Union, areConnected – $O(log^*n)$ משוערך.
  סיבוכיות מקום – O(n)
  כאשר n הוא מספרים האיברים השונים ביצירה.

- **Dynamic Hash-Table** – מבנה נתונים הממש טבלת ערבול דינאמית.

הערבול מתבצע באמצעות שיטת double-hashing ובאמצעות מערך דינמי. גודל המערך משתנה באופן דינמי, ע"י threshold.

תומך בפעולות:

- Insert(key, value) – הכנסת איבר לטבלה.
- Remove(key) – הוצאת איבר מהטבלה.
- Find(key) – החזרת ה-value המשויך ל-key.

סיבוכיות זמן הפעולות – O(1) בממוצע על הקלט באופן משוערך, כפי שנלמד בכיתה.

סיבוכיות מקום – *מוסברת לאחר אופן מימוש הפעולות למטה.

***מצורף איור המבנה בסוף החלק היבש**

פרטי המימוש של מבני הנתונים

- **עצי ה-AVL – סטודנטים**
  - כל צוות מחזיק עץ עם הסטודנטים ששייכים אליו. העצים שמורים במערך כאשר האינדקס במערך הוא מזהה הקבוצה ההתחלית. בנוסף כל עץ יחזיק 2 תאי זיכרון של MaxID ו-MaxPower עבור הסטודנט החזק ביותר בעץ.
  - העצים ממוינים מיון ראשוני לפי רמות כח ומשני לפי הid(במקרה של שיוויון) והדרגה היא סכום הדרגות של תתי העצים שלו כולל הוא עצמו – כלומר סכום רמות הכח בתתי העץ (כולל עצמו). בנוסף כל צומת יחזיק את גודל תת העץ שלו.

- **Union-find**
  - מימוש בעזרת מערכים כפי שראינו בהרצאה
  - כל צוות ייוצג ע"י קבוצה זרה.
  - נחזיק בנוסף מערך של מס' נצחונות עבור כל צוות

- **Dynamic Hash-Table**
  - בטבלת הערבול נשמור את המוטנטים עצמם לפי המזהה שלהם.

תיאור הפעולות

Init: נאתחל את שלושת מבני הנתונים הקיימים-

- ניצור n עצי דרגות מסוג AVL ריקים O(n) .
- נאתחל טבלת ערבול, טבלה זו תאותחל להיות בגודל p ראשוני. כאשר פונצ' הערבול שלה היא h(id)=id(mod P), אתחול שכזה לוקח O(1) =O(P) פעולות.
- אתחול של מבנה הUnion-Find, מבנה זה ממוש ע"י מערך דו מימדי בגודל 3xn (כמוצג בהרצאה) אתחול מערך שכזה לוקח O(n) פעולות.

סה"כ סיבוכיות הזמן הינה O(n) במקרה הגרוע כנדרש.

AddStudent: נבצע הוספה של הסטודנט למערכת-

- ניצור אובייקט סטודנט O(1) ונוסיף את הסטודנט לטבלת הערבול על פי הStudentID שלו . O(1) משוערך בממוצע על הקלט.
- אם הטבלה התמלאה, כלומר כבר לא מתקיימת השמורה ש(k =O(m כאשר k הוא מס' הסטודנטים ו m הוא גודל הטבלה. נרחיב את הטבלה כמוצג בתרגול בעזרת threshold. ראינו כי פעולה זאת לוקחת O(1) משוערך בממוצע על הקלט.
- נוסיף את הסטודנט לעץ הדרגות הרלוונטי אליו במערך העצים לפי find(team) הספציפי שלו(פעולת find הינה בסיבוכיות O(log*n) משוערך).נשים לב שהצמתים בעץ מחזיקים את סכום דרגות תתי העץ שלהם. על מנת לתקן ולשמור על השמורה נדרש לעבור רק על מסלול החיפוש של הצומת שנוסף ולעדכן את הערכים. אורך מסלול חיפוש שכזה בעץ AVL הוא O(logk) ולכן סיבוכיות ההכנסה והעדכון הינה O(logk). ולכן בסה"כ פעולה זו תיקח O(logk+log*n) משוערך
- נבצע בדיקה ועדכון לmaxID של הצוות שאליו הוספנו את הסטודנט. O(1)

סה"כ סיבוכיות הזמן הינה O(1)+ O(log*n)+ O(logk)=O(logk+log*n) משוערך בממוצע על הקלט כנדרש.

RemoveStudent: נבצע הסרה של הסטודנט מן המערכת-

- נסיר את הסטודנט מטבלת הערבול על פי הStudentID שלו . O(1) משוערך בממוצע על הקלט.
- אם הטבלה "התרוקנה", כלומר עבור c קבוע מסויים מקיימת k≤c*m כאשר k הוא מס' הסטודנטים ו m הוא גודל הטבלה. אז נקטין את הטבלה כמוצג בתרגול בעזרת threshold. ראינו כי פעולה זאת לוקחת O(1) משוערך בממוצע על הקלט.
- נסיר את הסטודנט מעץ הדרגות הרלוונטי אליו במערך העצים לפי find(team) הספציפי שלו(פעולת find הינה בסיבוכיות O(log*n) משוערך).נשים לב שהצמתים בעץ מחזיקים את סכום דרגות תתי הצץ שלהם. על מנת לתקן ולשמור על השמורה נדרש לעבור רק על מסלול החיפוש של הצומת שהוסר ולעדכן את הערכים. אורך מסלול חיפוש שכזה בעץ AVL הוא O(logk) ולכן סיבוכיות ההסרה והעדכון הינה O(logk). ולכן בסה"כ פעולה זו תיקח O(logk+log*n) משוערך

- נבצע בדיקה ועדכון לmaxID של הצוות שממנו הסרנו את הסטודנט. O(logk) במקרה הגרוע (סיור "ימינה" עד לתחתית העץ, שם נמצא הסטודנט החזק ביותר בעץ – מתקיים עקב מיון העץ באופן שבו תיארנו לעיל).

סה"כ סיבוכיות הזמן הינה O(1)+ O(log*n)+ O(logk)=O(logk+log*n) משוערך בממוצע על הקלט כנדרש.

JoinTeams : נבצע איחוד לקבוצות באופן הבא-

- נבצע find על שתי הקבוצות שקיבלנו. פעולה זו לוקחת O(log*n) משוערך. (בזמן הפעולה מבצעים כיווץ של מסלולים)
- נבצע איחוד על הקבוצות T1 T2 שקיבלנו בעזרת פעולת Union. O(log*n) משוערך.
- נעדכן את האיברים במערך המייצג את הUF המייצגים את הקבוצה החדשה שקיבלנו. O(1)
- נאחד את עצי הצוותים מן התאים הרלוונטים במערך העצים(בעזרת אלגוריתם merge) ונשמור אותם כעץ שמייצג את הצוות החדש . $O(k_1+k_2)$

סה"כ סיבוכיות הזמן הינה $O(\log^*n+ k_1+k_2)$ משוערך כנדרש.

TeamFight : נבצע קרב בין 2 הקבוצות-

- תחילה נמצא את שני הצוותים בעזרת find על שתי הקבוצות. O(log*n) משוערך.
- ניגש לעצי הavl המתאימים O(1) ובעזרת אלגוריתם select ואלגוריתם loswestCommonAncestor על שני העצים נוציא את סכום n החזקים ביותר באופן הבא :
  - נמצא את האיבר האחרון והאיבר הb-n בגודלו בעץ בעזרת אלגוריתם select, כאשר b הוא גודל העץ ו n הוא מס' הלוחמים שיש לשלוח לקרב. ראינו בהרצאה כי אלגוריתם select פועל בסיבוכיות O(logb)=O(logk).
  - כעת, בעזרת אלגוריתם LCA הנ"ל, נמצא את הצומת המשותף הרחוק ביותר מן השורש עבור שני הצמתים שמצאנו בעזרת select.
  - מציאת צומת זה, דורשת מעבר על 2 מסלולי החיפוש של הצמתים שמצאנו O(2logk) בעץ AVL מאוזן בעל k צמתים, ומציאת הצומת האחרון במסלול החיפוש מהשורש אל הצומת, כך שהוא משותף לשני מסלולי החיפוש של הצמתים – הרי זהו צומת האב הקדמון המשותף שלהם.
  - לפי שמורת העץ (כל צומת שומר את סכום דרגות תתי העצים שלו) נוכל כעת לחשב את סכום n הלוחמים החזקים ביותר בצוות : אנו יודעים כי סכום זה הוא הסכום השמור בצומת האב הקדמון המשותף שמצאנו, פחות הסכום של תת העץ השמאלי שלא משתתף בקרב. נחשב סכום זה ונחזירו.
- נעדכן נצחונות של הצוות הרלוונטי במידה וקיימים O(1).

סה"כ סיבוכיות הזמן הינה O(log*n+ logk) משוערך כנדרש.

GetNumOfWins : נמצא את הצוות teamID בעזרת find בUF O(log*n) משוערך, ונחזיר את כמות נצחונות הצוות O(1).
סה"כ סיבוכיות הזמן הינה O(log*n) משוערך כנדרש.

GetStudentTeamLeader : נרצה להחזיר את המזהה של ראש הקבוצה-

- תחילה נחפש את הסטודנט על פי הid שלו בטבלת הערבול, ונשמור את הteamID שלו. O(1) משוערך בממוצע על הקלט.
- נחפש את הצוות(המעודכן) ב-Union-Find בעזרת המידע ששמרנו במערכת. O(log*n) משוערך.
- ניגש אל העץ בתא המערך המתאים לצוות שממצאנו ונוציא ממנו את maxID. O(1).

סה"כ סיבוכיות הזמן הינה O(log*n) משוערך בממוצע על הקלט כנדרש.

Quit : נמחק ונסיר את שלושת מבני הנתונים הקיימים-

- נסיר את כל העצים במערך הצוותים. O(n+k) במקרה הגרוע.
- נסיר את טבלת הערבול O(1) .
- נסיר את UF. O(n)

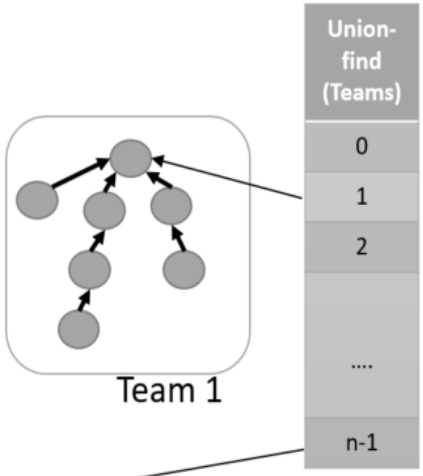
סה"כ סיבוכיות הזמן הינה O(n+k) במקרה הגרוע כנדרש.

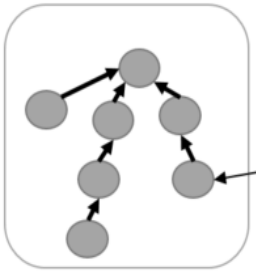
<u>סיבוכיות מקום</u>

בכל רגע במבנה הנתונים קיים מערך בגודל 3xn של הצוותים, מערך נוסף עבור העצים בגודל 2xn וטבלת הערבות בגודל m=O(k) כאשר k הוא מס' הסטודנטים במערכת.
כלומר בסך הכל סיבוכיות המקום של מבנה הXmen הוא O(n+k).

איור לדוגמה:

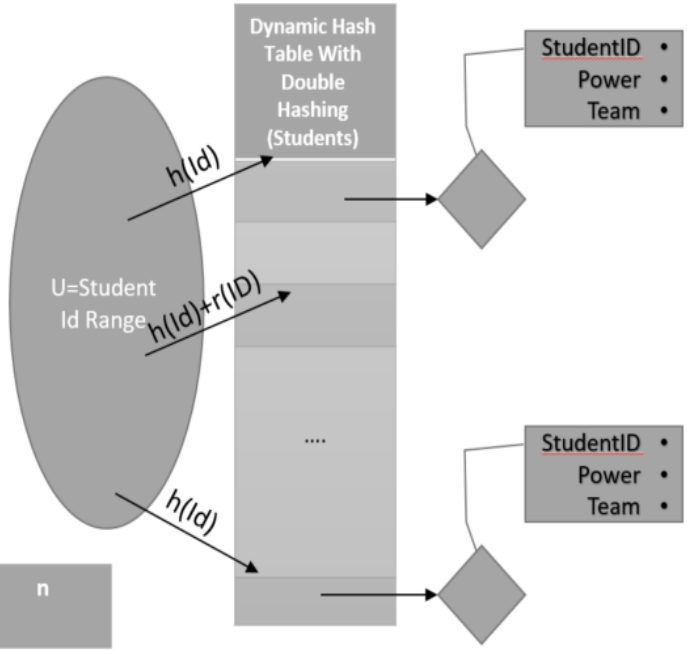
*נשים לב כי האילוסטרציה של עצים הפוכים בunion-find היא לשם המחשבה בלבד. המימוש האמיתי הוא בעזרת מערך כמוצג למטה



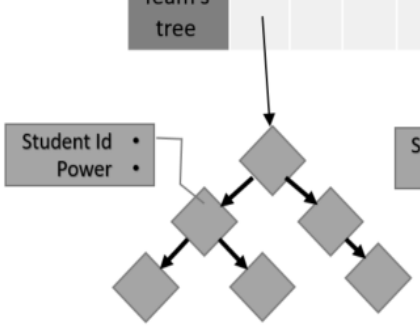


Team	1	2	3	...	n
Sets		4	-	...	3
Size	2	0	0		1
Wins	0	2	0	...	3

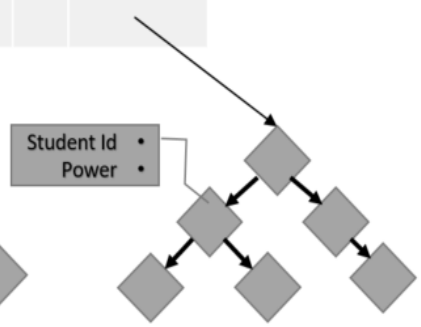
*הערכים במערך הם דוגמא בלבד



Teams Trees	1	2	3	...	n
MaxID	42				10
Team's tree					



Ranked AVL Tree(Students in Team 1)



Ranked AVL Tree(Students in Team n)