שמות המגישים	מספר זהות
דביר ברטוב	209028620
רון דבידובסקי	209502863

חלק יבש – תרגיל בית רטוב 2

נתאר את מבנה הנתונים בו אנחנו משתמשים עבור World Cup:

- עץ AVL המורכב מקבוצות אשר נמצאות במערכת, לא כולל קבוצות מודחות, המסודר לפי TeamId.
- עץ דרגות הממומש באמצעות עץ AVL של קבוצות אשר נמצאות במערכת, לא כולל קבוצות מודחות, המסודר לפי דרישת יכולת הקבוצה ללא הנקודות.
- מבנה נתונים Union Find אשר ממומש באמצעות:
  - Hash Table המכיל מצבעים על Node-ים שממומש באמצעות:
    - מערך דינמי.
    - Chain Hashing באמצעות עצי AVL.
    - פונקציית מודולו אשר משתנה בהתאם לגודל המערך, ומשמשת ל-Hashing.
    - לשם גישה לאיברים ב-Hash Table ב-$O(1)$ נסביר כיצד אנו שומרים על פקטור העומס כ-$O(1)$:
      - במקרה בו מספר האיברים ב-Table משתווה לגודל המערך, נגדיל את המערך פי 2.
      - גישה לאיבר תתבצע עדיין ב-$O(1)$ בממוצע על הקלט כאשר a פקטור העומס, ולכן נקבל כי פעולת הגישה במקרה הזה תתבצע ב-$O(1)$ בממוצע על הקלט, כפי שראינו בהרצאה.
      - פעולת הכנסה אם כן, כפי שראינו בתרגול על מערכים דינמיים, תתבצע בסיבוכיות $O(1)$ משוערך, שכן פעולת הגדלת המערך מתרחשת לעיתים רחוקות מספיק.
  - Node הוא אובייקט שמכיל שחקנים, פוינטרים לקבוצות מעץ הקבוצות, מספר משחקים ופרמוטציות בעלי תכונות אשר יפורטו בפירוט הפונקציות עצמן. האובייקט מייצג את מיקום השחקנים בעץ ההפוך, וכולל פוינטר ל-Node אב.
- אובייקט Player המכיל את כלל הנתונים הדרושים עבור שחקן (יכולת, מזהה וכו')
- אובייקט Team המכיל את כלל הנתונים הדרושים עבור קבוצה, וכן פוינטר ל-Node השורש (נקרא לו head) המייצג את הקבוצה בעץ ההפוך ב-Union Find.

סה"כ סיבוכיות המקום עבור מבנה הנתונים – $O(n+k)$, כאשר n מספר השחקנים במערכת ו-k מספר הקבוצות הפעילות במערכת, כנדרש.

כעת נתאר את פונקציות World Cup והוכחת סיבוכיות זמן ריצה ומקום לפי הדרישה:

World_cup_t()

הפונקציה מאתחלת את כלל שדות מבנה הנתונים world_cup. מבני הנתונים מאותחלים כריקים ולכן כלל האתחולים מתבצעים בסיבוכיות $O(1)$, מספר קבוע של פעמים, לכן סה"כ נקבל לסיכום כי:

<u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(1)$.</u>

<u>הפונקציה בעלת סיבוכיות מקום של $O(1)$.</u>

~World_cup_t()

- הפונקציה פולטת למערך את עץ הקבוצות ומוחקת את הקבוצות אחת אחת, לאחר מכן מוחקת את עץ ה-AVL ועץ הדרגות, באמצעות סיור postorder. סיבוכיות הזמן של פעולה זו היא $O(k)$ וסיבוכיות המקום היא $O(\log k)$.
- לאחר מכן נקרא ההורס של UnionFind. הפונקציה פולטת למערך את תוכן ה-HashTable באמצעות עצי ה-AVL, ואז עוברת על המערך ומוחקת את השחקנים. סיבוכיות הזמן של פעולה זו היא $O(n)$ במקרה הגרוע וסיבוכיות המקום הנוסף היא $O(n)$ במקרה הגרוע כי אנחנו מקצים מערך עזר למחיקת השחקנים.
- לבסוף נקרא ההורס של ה-HashTable. ההורס מוחק את המערך המכיל עצי AVL (אובייקטים ולא מצביעים) ולכן נקראים ההורסים של עצים אלו. מחיקת העצים מתבצעת כאמור באמצעות סיור postorder. המקרה הגרוע הוא המקרה בו כל השחקנים נמצאים באותו עץ וסיבוכיות הזמן תהיה $O(n)$ וסיבוכיות המקום היא $O(\log n)$.

סה"כ נקבל לסיכום כי:

<u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(n+k)$, כאשר n הוא מספר השחקנים במבנה הנתונים ו-k הוא מספר הקבוצות הפעילות.</u>

<u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(n+k)$, כאשר n הוא מספר השחקנים במבנה הנתונים ו-k הוא מספר הקבוצות הפעילות.</u>

add_team

פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- הכנסת קבוצה לעץ הקבוצות ולעץ הדרגות של הקבוצות – פעולת הכנסה בעץ AVL, מספר קבוע של פעמים, עבור עצמים אשר חסומים מלמעלה על ידי מספר הקבוצות – סיבוכיות זמן ריצה ומקום נוסף של $O(\log k)$ במקרה הגרוע כפי שראינו בהרצאה.

סה"כ נקבל:

<u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(\log k)$ במקרה הגרוע, כאשר k הוא מספר הקבוצות הפעילות.</u>

<u>הפונקציה בעלת סיבוכיות מקום נוסף של $O(\log k)$, כאשר k הוא מספר הקבוצות הפעילות.</u>

remove_team

פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- הסרת קבוצה מעץ הקבוצות ועץ הדרגות של הקבוצות – פעולת הוצאה מעץ AVL, מספר קבוע של פעמים, עבור עצמים אשר חסומים מלמעלה על ידי מספר הקבוצות – סיבוכיות זמן ריצה ומקום נוסף של $O(\log k)$ במקרה הגרוע כפי שראינו בהרצאה.

סה"כ נקבל:

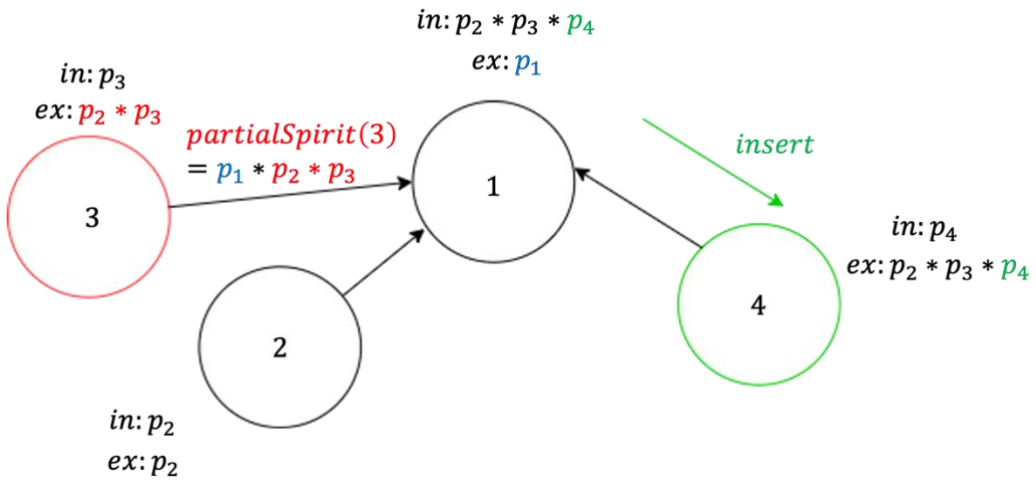
<u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(\log k)$ במקרה הגרוע, כאשר k הוא מספר הקבוצות הפעילות.</u>

<u>הפונקציה בעלת סיבוכיות מקום נוסף של $O(\log k)$, כאשר k הוא מספר הקבוצות הפעילות.</u>

add_player

פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- מציאת הקבוצה המבוקשת בעץ הקבוצות – סיבוכיות הזמן הריצה של פעולה זו היא $O(\log k)$. סיבוכיות המקום היא $O(\log k)$ במקרה הגרוע.
- הפונקציה יוצרת Node חדש בעץ ההפוך של UnionFind:
  - ה-Node החדש יצביע על head (צומת השורש בעץ ההפוך) של הקבוצה שמצאנו.
  - **שמירת מספר המשחקים בצומת** תתבצע על ידי $gamesPlayed - head \rightarrow games$.
  - **שמירת התמורות בצומת (מוסיפים צומת $n+1$):**
    - נכפול משמאל את התמורה שבשדה $insertSpirit$ של $head$ ($p_2 * \dots * p_n$) בתמורה של השחקן החדש (p_{n+1}). כלומר נקבל $insertSpirit = (p_2 * \dots * p_n * p_{n+1})$.
    - ביצירת ה-Node החדש, נגדיר את השדה $extractSpirit = p_2 * \dots * p_n * p_{n+1}$.
    - **הערה:** השדה $insertSpirit$ הוא השדה שמכפילים אותו בתמורה בנתיב ההכנסה, השדה $extractSpirit$ הוא השדה שמכפילים אותו בתמורה בנתיב היציאה (בחישוב $partial\ spirit$).



  - הכנסת הצומת לטבלת הערבול של הצמתים – סיבוכיות זמן $O(1)$ משוערך בממוצע על הקלט, כפי שהראינו בתיאור המבנה.

אנחנו מבצעים קריאה לבנאי שמאתחל שדות, מעדכנים מצביעים ומבצעים חישובים קבועים, לכן סיבוכיות הזמן של חלק זה היא $O(1)$. סיבוכיות המקום היא $O(\log n)$ במקרה הגרוע בו כל השחקנים נמצאים בעץ AVL אחד של טבלת הערבול.

סה"כ נקבל, כי עבור סדרת פעולות בגודל m בפונקציה זו, לעולם נבצע חיפוש בסיבוכיות $O(\log k)$ ופעולות שמשוערכות ב-$O(1)$ בממוצע על הקלט לכן נקבל לסיכום כי סיבוכיות סדרת הפעולות תהיה:

$O(m \cdot \log k + m) = O(m \cdot \log k)$ בממוצע על הקלט, ולכן מהגדרת סיבוכיות משוערכת:

- <u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(\log k)$ משוערך, בממוצע על הקלט, כאשר k הוא מספר הקבוצות הפעילות.</u>
- <u>הפונקציה בעלת סיבוכיות מקום נוסף של $O(\log k + \log n)$, כאשר k הוא מספר הקבוצות הפעילות.</u>

play_match

פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- חיפוש הקבוצות בעץ הקבוצות – חיפוש בעץ AVL החסום על ידי מספר הקבוצות הפעילות – סיבוכיות זמן ריצה ומקום נוסף של $O(\log k)$ במקרה הגרוע.
- בדיקת חוקיות הקבוצות – סיבוכיות זמן ריצה ומקום נוסף של $O(1)$
- חישוב teamSpirit מתבצע ע"י הכפלת $extractSpirit$ של צומת ה-head של הקבוצה ב-$UnionFind$ ב-$insertSpirit$ של הצומת. לאחר מכן קריאה לפונקציה $strength$ הנתונה של התמורה המתקבלת. כלל הפעולות הן סיבוכיות זמן ומקום של $O(1)$.
- ביצוע המשחק והוספת הנקודות בהתאם לקבוצה המנצחת (או במקרה של תיקו) – סיבוכיות זמן ריצה ומקום נוסף של $O(1)$
- הגדלת כמות כוללת של משחקים שקבוצה ב-1 ועדכון פרמטר כמות המשחקים של ה-Node המתאים לקבוצות ב-Union Find. עדכון פרמטר זה יתרום לנו בחישוב מספר המשחקים ששיחק כל שחקן, כפי שנפרט בהמשך – סיבוכיות זמן ריצה ומקום נוסף של $O(1)$

סה"כ נקבל:

<u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(\log k)$ במקרה הגרוע, כאשר k הוא מספר הקבוצות הפעילות.</u>

<u>הפונקציה בעלת סיבוכיות מקום נוסף של $O(\log k)$, כאשר k הוא מספר הקבוצות הפעילות.</u>

num_played_games_for_player

פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- תחילה נמצא את השחקן הדרוש ב-HashTable של ה-UnionFind – סיבוכיות זמן $O(1)$ בממוצע על הקלט.
- נשתמש באלגוריתם find של מבנה הנתונים UnionFind. כפי שתיארנו בפונקציית הכנסת השחקן, השדה השומר את מספר המשחקים נשמר באופן הבא:

  $gamesPlayed - (root \rightarrow games)$.

באופן זה, עדכון מספר המשחקים בשורש העץ ההפוך בלבד תאפשר להשיג את מספר המשחקים של השחקן המבוקש ע"י מציאת הצומת שלו בעץ, ועלייה במעלה העץ תוך סכימת ערך שדה מספר המשחקים עד השורש (כולל), בדומה ל"בעיית הארגזים" אותה ראינו בתרגול.

- לאחר מכן, בפונקציה זו אנחנו גם **מבצעים דחיסת מסלולים** של העץ ההפוך על מנת לעמוד בדרישות הסיבוכיות, לכן נתחזק את שדה המשחקים כפי שראינו בתרגול בזמן דחיסת מסלולים ב"בעיית הארגזים".

  הפעולה גם מתחזקת את $spirit$, על כך נפרט בפונקציה $get\_partial\_spirit$.

  פעולה זו משוערכת יחד עם סדרת פעולות find ו-unite אשר מתבצעות בפונקציות האחרות ומבצעות פעולות כיווץ מסלולים ואיחוד לפי גודל, כך שסדרת פעולות אלה יתנו כפי שראינו בהרצאה סיבוכיות זמן משוערכת של $O(\log^* n)$. סיבוכיות המקום הנוסף תהיה $O(\log n)$ במקרה הגרוע בו כלל האיברים ב-Hash Table נמצאים באינדקס יחיד על עץ AVL אחד.

סה"כ נקבל:

<u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(\log^* n)$ משוערך, בממוצע על הקלט, כאשר n מספר השחקנים במערכת (כולל שחקני עבר)</u>

<u>הפונקציה בעלת סיבוכיות מקום נוסף של $O(\log n)$, כאשר n מספר השחקנים במערכת (כולל שחקני עבר)</u>

add_player_cards

פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- מציאת השחקן הדרוש ב-Hash Table של ה-Union Find – חיפוש ב-Hash Table – סיבוכיות זמן $O(1)$ בממוצע על הקלט.
- חיפוש הקבוצה אליה משתייך השחקן ובודקת האם הקבוצה פעילה או לא (אם לא פעילה, פוינטר הקבוצה הוא nullptr). החיפוש מתבצע באמצעות פעולת find של מבנה הנתונים Union Find –כפי שראינו בפונקציה הקודמת הסיבוכיות של פעולה זו היא $O(\log^* n)$ משוערך. סיבוכיות המקום הנוסף תהיה $O(\log n)$ במקרה הגרוע בו כלל האיברים ב-Hash Table נמצאים באינדקס יחיד על עץ AVL אחד.
- עדכון מספר הכרטיסים לשחקן הדרוש – סיבוכיות זמן ומקום $O(1)$ במקרה הגרוע.

סה"כ נקבל:

<u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(\log^* n)$ משוערך, בממוצע על הקלט, כאשר n מספר השחקנים במערכת (כולל שחקני עבר)</u>

<u>הפונקציה בעלת סיבוכיות מקום נוסף של $O(\log n)$, כאשר n מספר השחקנים במערכת (כולל שחקני עבר)</u>

get_player_cards

פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- מציאת השחקן הדרוש ב-Hash Table של ה-Union Find – חיפוש ב-Hash Table – סיבוכיות זמן של $O(1)$ בממוצע על הקלט. סיבוכיות המקום הנוסף תהיה $O(\log n)$ במקרה הגרוע בו כלל האיברים ב-Hash Table נמצאים באינדקס יחיד על עץ AVL אחד, כאשר n מספר השחקנים במערכת

סה"כ נקבל:

<u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(1)$, בממוצע על הקלט.</u>

<u>הפונקציה בעלת סיבוכיות מקום נוסף של $O(\log n)$, כאשר n מספר השחקנים במערכת (כולל שחקני עבר)</u>

get_team_points

פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- מציאת הקבוצה הרלוונטית בעת הקבוצות – חיפוש בעץ AVL – סיבוכיות זמן ריצה ומקום $O(\log k)$ במקרה הגרוע
- החזרת שדה הנקודות השמור בקבוצה – $O(1)$

סה"כ נקבל:

<u>הפונקציה בעלת סיבוכיות זמן ריצה של $O(\log k)$ במקרה הגרוע, כאשר k הוא מספר הקבוצות הפעילות.</u>

<u>הפונקציה בעלת סיבוכיות מקום נוסף של $O(\log k)$, כאשר k הוא מספר הקבוצות הפעילות.</u>

get_ith_pointless_ability

פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- ביצוע פעולת חיפוש אינדקס, Select בעץ הדרגות של הקבוצות הממוין לפי יכולות הקבוצה ללא הנקודות שלה – החיפוש מתבצע באמצעות פרמטר weight המתוחזק כפי שראינו בהרצאה במהלך בניית העץ, כך שפעולת Select מבצעת את אלגוריתם החיפוש תוך כדי חיפוש האינדקס i, כך שנקבל פעולת Select סטנדרטית – סיבוכיות מקום וזמן נוסף של $O(\log k)$ כפי שראינו בהרצאה ובתרגול.

סה"כ נקבל:

הפונקציה בעלת סיבוכיות זמן ריצה של $O(\log k)$ במקרה הגרוע, כאשר k הוא מספר הקבוצות הפעילות.

הפונקציה בעלת סיבוכיות מקום נוסף של $O(\log k)$, כאשר k הוא מספר הקבוצות הפעילות.

get_partial_spirit

פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- מציאת השחקן הדרוש ב-Hash Table של ה-Union Find – חיפוש ב-Hash Table– סיבוכיות זמן $O(1)$ בממוצע על הקלט.
- באופן דומה למציאת מספר המשחקים של כל שחקן, נשתמש באלגוריתם find ונבצע כפל תמורות מהצומת של השחקן עד השורש (כפל התמורות בשדות $extractSpirit$). הפעולה דומה ל"בעיית הארגזים" רק שאנחנו מבצעים כפל תמורות (הרכבה) במקום חיבור. נשים לב שבכל צומת אנחנו כופלים משמאל את התמורה של צומת האב בתמורה של הצומת הנוכחי. כלומר אם המסלול הוא $v_3 \to v_2 \to v_1$ אז נקבל $p(v_1) * p(v_2) * p(v_3)$.
- כמו בפונקציות דומות, נבצע **דחיסת מסלולים** על מנת לשמור על הסיבוכיות הנדרשת. במהלך הדחיסה נתחזק את השדה ה-$spirit$ באופן הבא:
  - נעלה במעלה העץ ונבצע כפל של התמורות בשדות $extractSpirit$ מהצומת המבוקש עד לשורש (לא כולל).
  - נבצע שוב סריקה מהצומת המבוקש עד לשורש ובכל צומת נכפיל מימין את התמורה ההופכית של הצומת הקודם בערך שצברנו ונשים את הערך החדש בשדה $extractSpirit$ של אותו צומת. כך למשל, עבור מסלול $v_3 \to v_2 \to v_1 \to r$ נקבל בצומת v_2 את הערך:
    $$p(v_1) * p(v_2) * p(v_3) * p(v_3)^{-1} = p(v_1) * p(v_2)$$
- לבסוף נעדכן את המצביע של הצומת להצביע על השורש.

סה"כ נקבל כי בדומה לפונקציות קודמות, פונקציה זו משוערכת עם פעולות find ו-unite המבצעות כיווץ מסלולים ואיחוד לפי גודל ולכן:

הפונקציה בעלת סיבוכיות זמן ריצה של $O(\log^* n)$ משוערך, בממוצע על הקלט, כאשר n מספר השחקנים במערכת (כולל שחקני עבר)

הפונקציה בעלת סיבוכיות מקום נוסף של $O(\log n)$, כאשר n מספר השחקנים במערכת (כולל שחקני עבר)

buy_team

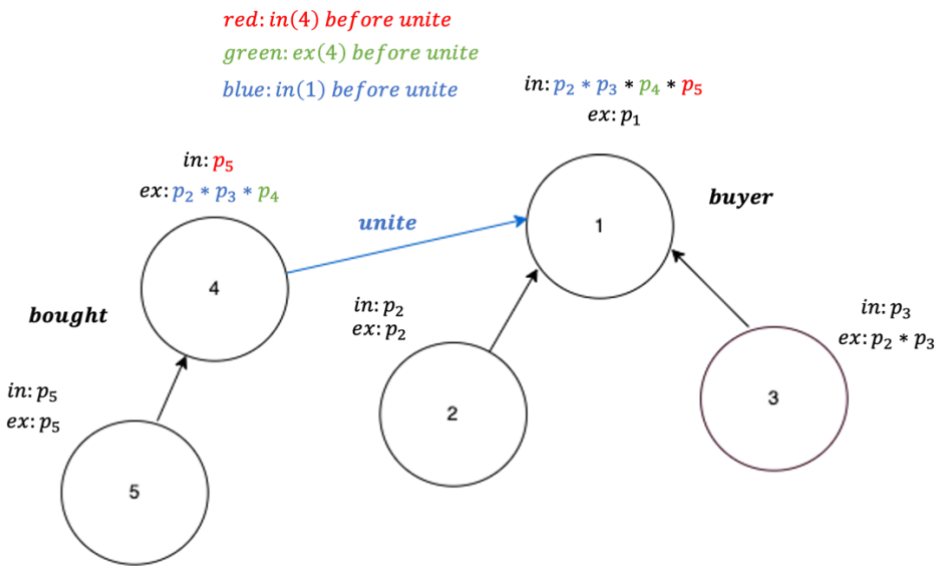
פעולות הפונקציה (בהנחה שהפרמטרים חוקיים):

- מחיקת הקבוצה הנקנית מעץ הקבוצות ומחיקת 2 הקבוצות טרם הקנייה מעץ הדרגות ושמירת מצביעים אליהם. מחיקה מעץ הדרגות והכנסה מחדש תשמור על סדר הדרגות – מספר קבוע של מחיקות מעץ AVL – $O(\log k)$ במקרה הגרוע.
- ביצוע פעולת Union במבנה ה-Union Find לפי גודל הקבוצות, לשני ה-Node-ים תוך כדי עדכון הפרמטרים הרלוונטיים למספר המשחקים ששחקן שיחק ול-Partial Spirit. נפרט את אופן עדכון השדות:
  - **עדכון מספר המשחקים** יתבצע ע"י חיסור מספר המשחקים של השורש שלאחר האיחוד (נסמנו r_1) ממספר המשחקים של השורש בקבוצות המתאחדת (נסמנו v_2). כלומר $games(v_2) - games(v_1)$.

בשל קומוטטיביות של פעולת החיבור אין הבדל באופן העדכון מספר המשחקים אם הקבוצה הקונה היא הקטנה או הגדולה.

- **עדכון $spirit$ כאשר $|bought| \geq |buyer|$.** נסמן את שורש הקבוצה הקונה ב-r_{buyer} ואת שורש הקבוצה הנקנית ב-r_{bought}. נעדכן את השדה $insertSpirit$ של r_{buyer} בכך שנכפיל אותו מימין שדה ה-$extractSpirit$ ואז את השדה ה-$insertSpirit$, כך כל צומת חדש שיוכנס יכיל בשדה ה-$extractSpirit$ את כל התמורות של שתי הקבוצות לפי הסדר (לא כולל תמורת השורש) באופן כזה שהתמורות של הקבוצה הקונה יהיו לפני אלו של הנקנית לפי הדרישה. בנוסף נכפול משמאל את $insertSpirit$ של r_{buyer} (לפני העדכון שתיארנו) ב-$extractSpirit$ של r_{bought}. כך כל תמורה בעץ ההפוך של הקבוצה הנקנית יוכפל גם בתמורות של העץ ההפוך של הקבוצה הקונה (שהן תמורות של שחקנים שנמצאים לפני בסדר השחקנים).

תרשים לדוגמה:



- **עדכון $spirit$ כאשר $|bought| < |buyer|$. נעדכן את השדה $extractSpirit$ של r_{bought}:** $r_{buyer}(exSpirit) * r_{buyer}(inSpirit) * r_{bought}(extractSpirit)$. כך כל השחקנים בעץ הנקנה יוכפלו בתמורות של השחקנים בעץ הקונה (שהן תמורות של שחקנים שנמצאים לפני בסדר השחקנים).

לאחר מכן **נעדכן את השדה $extractSpirit$ של r_{buyer}:**

$$\left(r_{buyer}(exSpirit) * r_{buyer}(inSpirit) * r_{bought}(extractSpirit)\right)^{-1} * r_{buyer}(exSpirit)$$

נסביר זאת באמצעות דוגמה: נסמן את התמורות בקבוצה הקונה ב: p_1, p_2 ואז התמורות בקבוצה הנקנית ב-p_3, p_4, p_5. לכן $r_{bought}(exSpirit) = p_1 * p_2 * p_3$ ו-$r_{buyer}(inSpirit) = (p_1 * p_2 * p_3)^{-1} * p_1$

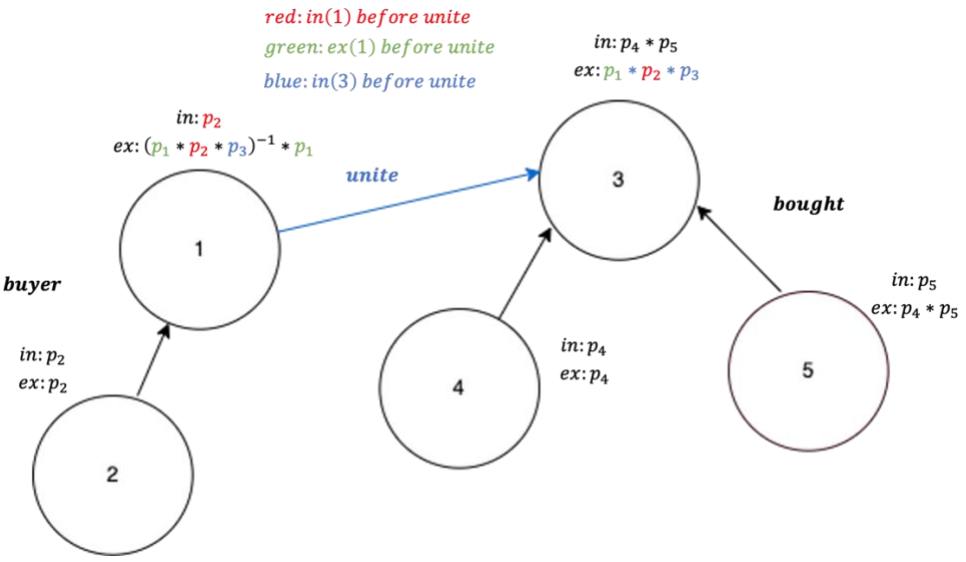
$permutation\_t$ מתארת את חבורת התמורות S_5, לכן לפי משפט מתורת החבורות:

$$r_{buyer}(inSpirit) = p_3^{-1} * p_2^{-1} * p_1^{-1} * p_1 = p_3^{-1} * p_2^{-1}$$

כך למשל עבור מציאת $partial\ spirit$ של p_2 בקבוצה הקונה, נקבל:

$$p_1 * p_2 * p_3 * p_3^{-1} * p_2^{-1} * p_2 = p_1 * p_2$$

תרשים לדוגמה:



סיבוכיות פעולה זו משוערכת עם הפונקציות האחרות במערכת, אשר משתמשות בנוסף בפעולת find וכיווץ מסלולים, לכן פעולה זו משוערכת יחד עם הפונקציות האחרות בסיבוכיות זמן ריצה של $O(\log^* n)$.

- עדכון הנקודות לקבוצה המאוחדת והוספתה מחדש לעץ הדרגות – הכנסה לעץ דרגות הממומש באמצעות עץ AVL – $O(\log k)$ במקרה הגרוע.

סה"כ נקבל כי בכל מצב הפונקציה תבצע חיפוש, הכנסה והוצאה מספר פעמים קבוע ב-$O(\log k)$ ולכן עבור כל סדרת m פעולות של פונקציה והפונקציות המשוערכות האחרות נקבל:

$O(m(\log k + \log^* n)$ ולכן מהגדרה נקבל לסיכום:

<u>הפונקציה בעלת סיבוכיות זמן ריצה $O(\log k + \log^* n)$ משוערך, כאשר k מספר הקבוצות ו-n מספר השחקנים במערכת (כולל שחקני עבר).</u>

<u>הפונקציה בעלת סיבוכיות מקום נוסף $O(\log k)$, כאשר k מספר הקבוצות.</u>