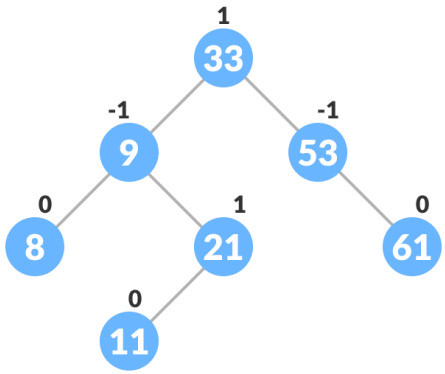
מגישים:
מיתר בסון 208252452
אריאל ארגס 327018552

<u>למימוש הפעולות נציע את מבנה הנתונים הבא:</u>

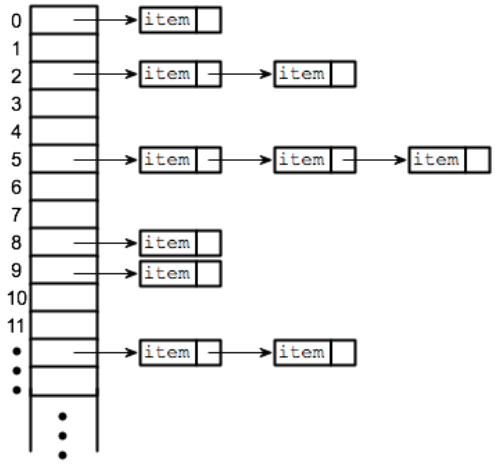
- $m\_teams, m\_teamsAbility$: שני עצי AVL לסידור ושמירת הקבוצות בטורניר, כך שכל עץ ממוין באופן שונה, האחד לפי $team\_id$ והוא עץ AVL בדומה לתרגיל הקודם. העץ השני נשמר לפי היכולת, $ability$ והוא עץ דרגות הנעזר בפונקציית $select$ שנלמדה בתרגולים.
- $m\_playersHash$: טבלת ערבול לשחקנים – מבנה דינאמי של $hashmap$ שאחראי לשמור על השחקנים בטורניר ולאפשר גישה והכנסה בסיבוכיות שנלמדה בכיתה. (המבנה אחראי לגדול לבד באמצעות מערך דינאמי כשצריך ולכן מאותחל ב$O(1)$ זיכרון לגודל קבוע).
- $m\_playersNodes$: מבנה $Union - Find$ לשחקנים הממופים על ידי טבלת הערבול. המבנה יממש חלוקה של השחקנים לקבוצות זרות (כפי שקורה בטורניר) כאשר ניתן לאחד קבוצות ולמצוא את ראש הקבוצה בסיבוכיות שנלמדה בכיתה. (עם מימש כיווץ מסלולים בכל find ואיחוד לפי גודל).

<u>תרשימים להמחשה:</u>

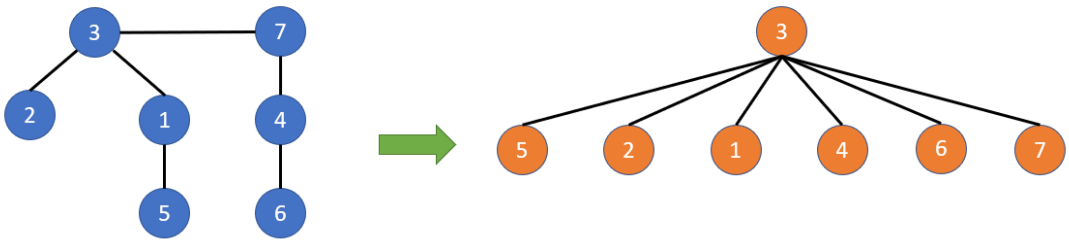
עצי **AVL** עבור הקבוצות:



טבלת הערבול לשחקנים:



מבנה ה-**UF** לשחקנים, כולל איחוד לפי גודל וכיווץ מסלולים:



World_cup_t()

נאתחל את 4 מבני הנתונים לעיל כמבנים ריקים, כנלמד. לכן סה"כ סיבוכיות זמן ומקום $O(1)$.

---

virtual ~world_cup_t()

נשחרר את כל הזיכרון ששמרנו במבנה – כלומר את כלל השחקנים שנוספו ואת כלל הקבוצות שנוספו. כיוון ששמרנו מספר סופי וקבוע (4) של מבני נתונים ובכל אחד מהם כמות הזיכרון לינארית למספר השחקנים והקבוצות, סה"כ נקבל סיבוכיות זמן: $O(n+K)$ במקרה הגרוע, כאשר n הוא מספר השחקנים הכולל ו-K הוא מספר הקבוצות מאז תחילת המערכת.

StatusType add_team(int teamId)

לאחר בדיקה שלא קיימת קבוצה עם המזהה הנ"ל, נוסיף למבני הנתונים של הקבוצות $m\_teamsAbility$, m_teams את הקבוצה בעלת המזהה teamId. כיוון שאלו פשוט שתי הוספות לעצי AVL (כולל גלגולים) בצורה שנלמדה, נקבל כי סה"כ סיבוכיות הפעולה היא כ-$O(\log\ k)$. סיבוכיות המקום לינארית לכמות הקבוצות ולכן נותרת $O(k)$.

StatusType remove_team(int teamId)

לאחר בדיקה שאכן קיימת קבוצה עם המזהה הנ"ל, נמחק אותה ממבני הנתונים של הקבוצות $m\_teamsAbility$, m_teams. נדאג לעדכן את ראש הקבוצה שקבוצתו כבר אינה פעילה (זהו שדה של הקבוצה שמחזיק מצביע לראש הקבוצה הזו במבנה ה-union-find שלנו, לכן בסיבוכיות $O(1)$). סה"כ כמחיקה מעץ AVL נקבל סיבוכיות כוללת של $O(\log k)$. כיוון שזכרון אובייקט הקבוצה משוחרר סיבוכיות המקום כוללת נותרת לינארית ל-n + k כלומר $O(n+k)$.

StatusType add_player(int playerId, int teamId, const permutation_t &spirit, int gamesPlayed, int ability, int cards, bool goalKeeper)

לאחר בדיקת תקינות כלל הפרמטרים, ניצור אובייקט שחקן חדש במערכת ונשמור אותו בשני מבני הנתונים של השחקנים: $m\_playersNodes, m\_playersHash$. נמצא את הקבוצה הרלוונטית בעץ הקבוצות ב-$O(\log k)$. כעת דרך שדה ראש הקבוצה של הקבוצה נעדכן במבנה ה-union-find שלנו שהשחקן החדש יצביע על ראש הקבוצה, זהו רק שינוי מצביעים לכן מתרחש ב-$O(1)$. בעת הכנסה לטבלת הערבול הסיבוכיות היא $O(1)$ בממוצע על הקלט, וכיוון שטבלה זו גדלה באמצעות מערך דינאמי כמו שלמדנו בתרגול, סיבוכיות ההכנסה אליה הינה בסך הכל $O(1)$ משוערכת בממוצע על הקלט. לכן סה"כ נקבל

סיבוכיות זמן של $O(\log k)$ משוערך, בממוצע על הקלט וסיבוכיות מקום לינארית למספר השחקנים במערכת ולכן $O(n)$.

---

output_t<int> play_match(int teamId1, int teamId2)

לאחר בדיקת תקינות הפרמטרים, נמצא את שתי הקבוצות בעץ הקבוצות לפי id בסיבוכיות $O(\log k)$. נוודא שהקבוצות אכן חוקיות (בעלות שוער) ונחשב ב-$O(1)$ זמן את היכולת של כל קבוצה לנצח (ואת הכוח הרוחני), באמצעות שדות שאנו מעדכנים בעת הוספת כל שחקן לקבוצה. נבדוק מי הקבוצה המנצחת ונעדכן לכל קבוצה את שדה הניקוד שלה בהתאם. סה"כ נקבל סיבוכיות זמן של $O(\log k)$ במקרה הגרוע.

---

output_t<int> num_played_games_for_player(int playerId)

לאחר בדיקת תקינות הפרמטרים, נמצא את השחקן עם המזהה הנתון בטבלת הערבול בסיבוכיות $O(1)$ משוערכת בממוצע על הקלט (כנלמד). שחקן זה יחזיק שדה של מספר המשחקים היחסי שלו ביחס להורה שלו בעץ ההפוך שב-UF, וזו שמורה שנקפיד לשמור עליה לאורך הקוד. נתחיל לטפס במסלול ה-find במבנה זה ובכל צומת שנעלה אליה נוסיף את שדה המשחקים היחסי של הצומת לסכום. לבסוף נקבל את מספר המשחקים הכולל של השחקן. סיבוכיות מסלול ה- find במבנה union-find היא $O(\log^* n)$ לכן סה"כ סיבוכיות הפעולה היא $O(\log^* n)$ משוערך בממוצע על הקלט.

StatusType add_player_cards(int playerId, int cards)

לאחר בדיקת תקינות הפרמטרים נחפש את השחקן בטבלת הערבול. לאחר מכן נבדוק במבנה ה--union find האם קבוצתו של השחקן פעילה (מוחזק כפרמטר אצל ראש הקבוצה). אם כן, נעדכן את מספר הכרטיסים של השחקן בכמות שהתווספה. סה"כ מחלצים איבר מטבלת הערבול ומבצעים פעולת find במבנה של Union-find, כל אלו כנלמד בהרצאות ובתרגולים ולכן סה"כ הסיבוכיות של הפעולה היא $O(\log^* n)$ משוערך, בממוצע על הקלט.

---

output_t<int> get_player_cards(int playerId)

לאחר בדיקת תקינות הפרמטרים, נמצא את השחקן בטבלת הערבול של השחקנים ונחזיר את השדה ששומר את המספר הכרטיסים של השחקן. סה"כ מבצעים גישה לאיבר בטבלת ערבול ולכן הסיבוכיות $O(1)$ בממוצע על הקלט.

---

output_t<int> get_team_points(int teamId)

לאחר בדיקת תקינות הפרמטרים נחפש את הקבוצה הנתונה בעץ הקבוצות לפי id. במידה ולא קיים נחזיר ערך החזרה מתאים ואם הקבוצה קיימת נחזיר את השדה ששומר על ניקוד הקבוצה (שדה זה מתעדכן בעת הוספת שחקן ולכן תמיד מעודכן). מבצעים כפי שלמדנו חיפוש בעץ AVL ולכן סה"כ הסיבוכיות של הפעולה היא $O(\log k)$ במקרה הגרוע.

---

output_t<int> get_ith_pointless_ability(int i)

לאחר בדיקת תקינות הפרמטרים נחפש את הקבוצה במקום ה-i בעץ הקבוצות המסודר לפי ability. זהו עץ דרגות הממש את הפונקציה select שנלמדה בתרגול, ונשתמש בפונקציה זו בדיוק בשביל למצוא את הקבוצה שמיקומה הוא i מבחינת ה-ability שלה ביחס לשאר הקבוצות. סה"כ מבצעים חיפוש בעץ AVL שהוא גם עץ דרגות לכן סיבוכיות הפעולה היא $O(\log k)$ במקרה הגרוע, כמו שראינו בתרגול.

---

output_t<permutation_t> get_partial_spirit(int playerId)

לאחר בדיקת תקינות הפרמטרים, נמצא את השחקן (אם קיים) עם המזהה הנתון בטבלת הערבול בסיבוכיות $O(1)$ משוערכת בממוצע על הקלט. שחקן זה יחזיק שדה של סך רוחות השחקנים 👻 שנכנסו לטורניר לפניו, כולל אותו, ביחס להורה שלו (ראש הקבוצה). לאחר מכן נסייר במעלה מסלול ה-find של חיפוש ראש קבוצת השחקן במבנה ה-union-find שלנו, ובכל צומת שנגיע אליה (שהיא ראש קבוצתו) נכפיל משמאל את הרוח שלה ברוח המצטברת שאנו מחשבים. (תקינות שדות אלו במעלה מסלול החיפוש מעודכנות בעת כיווצי המסלולים ולכן שמורה זו נשמרת בקוד). לבסוף נקבל את החלק היחסי של רוח השחקן בקבוצה (רוחו שלו ורוחות כל השחקנים הקודמים לו). סה"כ מחלצים איבר מטבלת הערבול ומבצעים פעולת find במבנה של Union-find, כל אלו כנלמד בהרצאות ובתרגולים ולכן סה"כ הסיבוכיות של הפעולה היא $O(\log^* n)$ משוערך, בממוצע על הקלט.

StatusType buy_team(int buyerId, int boughtId)

לאחר בדיקת תקינות הפרמטרים נמצא את שתי הקבוצות בעצי הקבוצות. בנוסף נמצא גם את ראשי הקבוצות במבנה ה-union-find. נבצע פעולת union לשתי הקבוצות לפי גודל, על מנת לבצע את השיפור לסיבוכיות כפי שנלמד בהרצאה. בכל אחד מהמקרים נתחשב בעדכון השדות של מס' המשחקים הכולל של כל תת קבוצה והרוח המצטברת של כל תת קבוצה, כל זה בשמירה על השמורות שהתייחסנו אליהן בפונקציות get_partial_spirit, num_played_games_for_player. כעת לאחר העברת הבעלות על כל

השחקנים לקבוצה הקונה, אנחנו נוודא למחוק את הקבוצה שנקנתה מעצי הקבוצות. נעדכן גם את ניקוד הקבוצה הקונה ואת שאר הפרמטרים הרלוונטיים לה (שוערים, יכולת). סה"כ אנו מבצעים חיפוש בעצי AVL ואיחוד שתי קבוצות במבנה union-find ולכן סיבוכיות הזמן היא: $O(\log k + \log^* n)$ משוערך. (וסיבוכיות המקום נותרת לינארית למספר הקבוצות + מספר השחקנים).

---

*בכל הפונקציות אנו דואגים להחזיר את ערכי ההחזרה כנדרש, בהתאם לכל מקרה ותרחיש אליו נדרשנו בקובץ התרגיל, קרי ...SUCCESS, FAILURE.