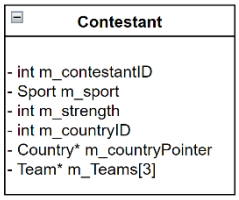
תרגיל רטוב 1 – מבני נתונים

איתי ברקוביץ 316088970

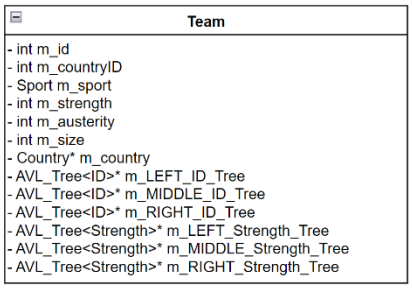
איל שטיין 208622142

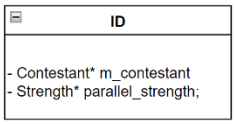
נסביר בקצרה על המחלקות במבנה מהקטן אל הגדול:

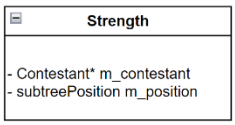
Contestant - המחלקה המכילה את כל הנתונים של מתחרה - מזהה שחקן, כוח, ספורט ומזהה מדינה שנתונים לנו ע"י המשתמש.
המחלקה מחזיקה פוינטר למדינת המתחרה ע"מ לעמוד בסיבוכיות בפונקציות מסוימות (יורחב בהמשך).
בנוסף, המחלקה מחזיקה מערך של מצביעים לקבוצות (Teams) בגודל 3 (כמספר הקבוצות שלמתחרה מותר להיות רשום בהן בו-זמנית), שוב על מנת לעמוד בסיבוכיות של הפונקציות הנדרשות.



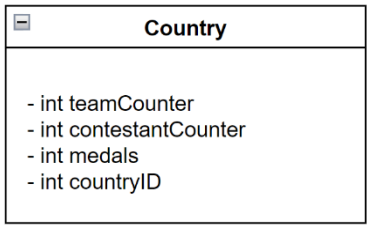
Team - המחלקה מכילה את כל הנתונים של הקבוצה - מזהה קבוצה, מזהה המדינה, ספורט שנתונים על ידי המשתמש. בנוסף, המחלקה מחזיקה בשדות את הערך של הכוח הנוכחי של הקבוצה על פי התנאים שנקבעו, את ערך ה-Austerity Measures הנוכחי שמחושב ע"י אלגוריתם שנרחיב עליו בהמשך, את גודל הקבוצה ולבסוף מחזיקה שישה עצי AVL המחולקים לקבוצות של שלושה תתי עצים: שלושה עצי AVL של המחלקה ID המחזיקים מצביע למתחרה ומצביע למחלקה המקבילה Strength (ע"מ לעמוד בסיבוכיות של פונקציות מסוימות, יורחב בהמשך), יחד עם שלושה עצי AVL של המחלקה Strength.
כך בפועל אנו מחזיקים את המתחרים באופן ממוין (בקריאת העצים לפי הסדר ב-InOrder) ומחולקים לשלושה חלקים כמעט שווים (הם שווים כאשר מספר המשתתפים בקבוצה מתחלק ב-3).



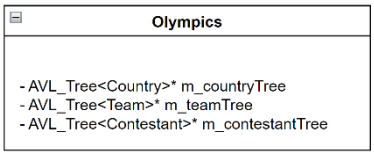




Country - המחלקה מכילה את כל הנתונים של מדינה: מזהה מדינה ומספר מדליות הנתונים לנו ע"י המשתמש.
בנוסף, המחלקה מכילה את מספר הקבוצות ששייכות למדינה ואת מספר השחקנים תחת אותה מדינה.



Olympics - מחלקה המכילה שלושה עצי AVL: עץ מדינות, עץ נבחרות ועץ שחקנים.



הערה/הארה על ניהול הזיכרון המערכת:

באופן כללי, עצי ה-AVL הטמפלייטים שמימשנו אחראים למחוק את ה-Node שלהם ואת הזיכרון שהוקצה עבור ה-Type T שהם מחזיקים. העצים שנמצאים במחלקה Olympics אחראיים על שחרור הזיכרון של ה-Country, Team ו-Contestant. שאר המחלקות המכילות מצביעים שונים למחלקות Country, Team או Contestant לא אחראיות על שחרור אותו זיכרון. באותו אופן, העצים של Team אחראיים למחוק את המחלקות ID ו-Strength אך לא את המתחרה שאותן מחלקות מצביעות עליו.
כך אנו מבטיחים את תקינות המערכת מבחינת זיכרון כי ישנה חלוקה ברורה באחריות על שחרור והקצאת הזיכרון בין העצים.

Olympics_t() - בנאי של המחלקה Olymics, יוצר שלושה עצים ריקים כפי שנלמד בהרצאה.

<u>סיבוכיות</u>: O(1)

~olympics_t() virtual - הורס של המחלקה Olympics. במקרה הגרוע ביותר נעבור רקורסיבית על כלל העצים במערכת. נרחיב על המעבר על העץ של הנבחרות: לכל נבחרת נעבור על שתת תתי-העצים של המחלקה ונמחק אותם, מספר המתחרים שבקבוצות הוא לכל היותר כמספר המתחרים שב-Olympic (עד כדי פקטור 3 בהינתן וכל המתחרים נמצאים בשלוש קבוצות שונות) ולכן הסיבוכיות של למחוק את כלל הקבוצות במקרה הגרוע ביותר הוא $O(n+m)$. מחיקת עצי המתחרים נעשית ב-$O(n)$ ומחיקת עצי המדינות נעשית ב-$O(k)$ ולכן הסיבוכיות הכוללת היא $O(n+k+m)$.

add_country(int countryId, int medals) StatusType - בודקים שמזהה המדינה הוא חוקי ושהמדינה לא קיימת כבר בעץ המדינות. מייצרים מדינה חדשה ומוסיפים אותה לעץ המדינות.
הבדיקה שהמדינה קיימת כבר בעץ נעשית ע"י פונקציית ()find של עץ ה-AVL שנעשית ב-$O(\log k)$.
יצירת אובייקט המדינה נעשה ב-$O(1)$ והכנסתו לעץ נעשית ע"י פונקציית ()Insert של עץ ה-AVL שנעשית ב-$O(\log k)$. ולכן הסיבוכיות הכוללת היא: $O(\log k)$

remove_country(int countryId) StatusType - בודקים שמזהה המדינה חוקי, לאחר מכן מוצאים את המדינה ובודקים אם מספר הנבחרות והמתחרים בה הוא אפס. אם כן, מסירים אותה מהעץ ומבצעים את הגלגולים

הנדרשים לעץ המדינות (בסיבוכיות O(1) כפי שנלמד בתרגול).
פעולת החיפוש וההוצאה נעשות בסיבוכיות של $O(\log k)$ ולכן הסיבוכיות הכוללת היא: $O(\log k)$

StatusType add_team(int teamId, int countryId, Sport sport) - בודקים שמזהי המדינה והקבוצה חוקיים. מבצעים חיפוש בעץ המדינות לברר שהמדינה אכן קיימת, חיפוש בעץ המדינות נעשה ב-$O(\log k)$, שומרים מצביע למדינה. בודקים בעץ הנבחרות שהנבחרת לא קיימת, חיפוש בעץ הנבחרות נעשה ב-$O(\log m)$, מאתחלים נבחרת חדשה ללא מתחרים ב-$O(1)$. מוסיפים את הנבחרת לעץ הנבחרות, פעולת ההוספה בעץ הנבחרות נעשית ב-$O(\log m)$ ושומרים בתוך הנבחרת את המצביע למדינה. מעלים ב-1 את מונה הנבחרות של המדינה.
הסיבוכיות הכוללת: $O(\log k + \log m)$

StatusType remove_team(int teamId) - בודקים שמזהה הנבחרת הוא חוקי. מחפשים אם הקבוצה בעץ ב-$O(\log m)$. אם הנבחרת נמצאה בעץ ואין לה מתחרים, ניגש למצביע של המדינה אליה היא שייכת ונעדכן את מונה הנבחרות שלה בהתאם. לבסוף נסיר אותה מעץ הנבחרות ב-O(log(m)), לכן הסיבוכיות הכוללת: O(log(m))

StatusType add_contestant(int contestantId,int countryId, Sport sport, int strength) - נבדוק תקינות של מזהה השחקן, מזהה המדינה ושל החוזק. אם הם תקינים נחפש את המדינה בעץ המדינות ב-O(log(k)), בהינתן ומצאנו את המדינה נשמור מצביע אליה ונחפש את המתחרה בעץ המתחרים ב- O(log(n)), במידה והמתחרה קיים נחזירFAILURE, אחרת נייצר מתחרה חדש ב-$O(1)$ עם הנתונים מהמשתמש והמצביע של המדינה ונוסיף אותו לעץ המתחרים ב-O(log(n)), נעלה את מונה המתחרים של המדינה ב-1.
הסיבוכיות הכוללת: O(log(k)+ log(n)).

StatusType remove_contestant(int contestantId) - נבדוק תקינות של מזהה המתחרה.
נחפש את המתחרה בעץ המתחרים ב-O(log(n)). נבדוק שהמערך שמתאר את שיוך המתחרה לקבוצות הוא מערך המצביע רק ל-nullptr (כלומר שהשחקן לא רשום באף קבוצה). נוריד את מספר המתחרים של המדינה שעליה מצביע המתחרה. ונסיר את המתחרה מעץ המתחרים ב- O(log(n)).
הסיבוכיות הכוללת: O(log(n)).

StatusType add_contestant_to_team(int teamId, int contestantId) - נבדוק תקינות של מזהה המתחרה ושל מזהה הקבוצה. נחפש את הקבוצה בעץ הקבוצות ב-O(log(m)) ואת המתחרה בעץ המתחרים ב-O(log(n)).

אם הם לא קיימים או שהמדינה והספורט שלהם לא זהים, או שהמתחרה לא פנוי להירשם לקבוצה נוספת או שהשחקן כבר רשום לקבוצה נחזיר FAILURE. אחרת, נוסיף את השחקן לאחד מתתי העצים של הקבוצה המתאים לו לפי מזהה המתחרה ב-O(log(n)).
אם צריך, נבצע איזון לתתי העצים, פעולה שהיא ב-O(1) כי היא תמיד כוללת הזזה של מספר קבוע של מתחרים מתת עץ לתת עץ.
ניצור למתחרה instance של Strength ונוסיף אותו לתת העץ המקביל (לדוגמה LEFT או MIDDLE) ב-$O(\log n)$. נעדכן את גודל הקבוצה, נחשב את חוזק הקבוצה ב-O(log(n)) ואת הausterity ב-O(log(n)).
לכן הסיבוכיות הכוללת של הוספת מתחרה לקבוצה היא: O(log(m)+ log(n))

StatusType remove_contestant_from_team(int teamId, int contestantId) - נבדוק שמזהה הקבוצה ומזהה המתחרה תקינים. נחפש את המתחרה בעץ המתחרים ב- O(log(n)) ואת הקבוצה בעץ הקבוצות ב-O(log(m)). אם הם לא קיימים או שהשחקן לא רשום בקבוצה נחזיר FAILURE. אחרת, נסיר את המתחרה בתת העץ הממוין ע"י מזהה המתחרה (ID) ב- O(log(n)). בנוסף, נסיר את המתחרה בתת העץ הממוין ע"י החוזק של המתחרה (Strength) ב- O(log(n)). אם צריך, נבצע איזון לתתי העצים פעולה שהיא ב-O(1) כי היא תמיד כוללת הזזה של מספר קבוע של מתחרים מעץ לעץ.
נעדכן את גודל הקבוצה, נחשב את חוזק הקבוצה ב-O(log(n)) ואת הausterity ב-O(log(n)).
לכן הסיבוכיות הכוללת של הוספת מתחרה לקבוצה היא: O(log(m)+ log(n)).

StatusType update_contestant_strength(int contestantId, int change) - נבדוק שמזהה המתחרה תקין, נחפש אם המתחרה קיים בעץ המתחרים ב-O(log(n)). נבדוק שהחוזק החדש אי-שלילי אחרת נחזיר FAILURE. נעבור על המערך של הקבוצות אליהן רשום המתחרה ונסיר אותו מכולן ב-O(log(n)) ונעדכן את גודל הקבוצות. לאחר מכן נעדכן את החוזק החדש של המתחרה ונכניס אותו מחדש לקבוצות אליהן היה רשום ב-O(log(n)) ונעדכן את גודל הקבוצות. נחשב ונעדכן את חוזק הקבוצה ב-O(log(n)) ואת הausterity ב-O(log(n)).
לכן הסיבוכיות הכוללת של הוספת מתחרה לקבוצה היא: O(log(n)).

output_t<int> get_strength(int contestantId) - נבדוק שמזהה המתחרה תקין, אם לא נחזיר INVALID_INPUT. לאחר מכן נחפש את המתחרה בעץ המתחרים ב-O(log(n)). אם המתחרה לא קיים נחזיר FAILURE. במידה והוא קיים נחזיר SUCCESS ואת החוזק של המתחרה.
הסיבוכיות הכוללת: O(log(n)).

output_t < int > get_medals(int countryId) - נבדוק שמזהה המדינה תקין, אם לא נחזיר INVALID_INPUT.
לאחר מכן נחפש את המדינה בעץ המדינות ב-O(log(k)). אם המדינה לא קיימת נחזיר FAILURE. במידה והיא

קיימת נחזיר SUCCESS ואת כמות המדליות של המדינה.
הסיבוכיות הכוללת: O(log(k)).

output_t < int > get_team_strength(int teamId) - נבדוק שמזהה הקבוצה תקין, אם לא נחזיר INVALID_INPUT.
לאחר מכן נחפש את הקבוצה בעץ הקבוצות ב- O(log(m)). אם הקבוצה לא קיימת נחזיק FAILURE. במידה והיא קיימת נחזיר SUCCESS ואת החוזק של הקבוצה שחושב מראש ושמור בשדה המתאים.
הסיבוכיות הכוללת: O(log(m)).

StatusType play_match(int teamId1, int teamId2) - נבדוק אם מזהי הקבוצות תקינים, אם לא נחזיר INVALID_INPUT. לאחר מכן נחפש את שתי הקבוצות ב-O(log(m)), אם לא מצאנו נחזיר FAILURE.
במידה והמדינות קיימות נבדוק שהספורט שלהן זהה, אם לא נחזיר FAILURE. אחרת, נחשב את הכח הכולל של שתי הנבחרות ע"י סכימת שדה החוזק ושדה המדליות של כל קבוצה. הקבוצה המנצחת היא זאת שהכח הכולל שלה גדול ממש מהשנייה, נוסיף לקבוצה המנצחת מדליה אחת ונחזיר SUCCESS. אם הכח הכולל של הקבוצות שווה אז יש תיקו ואף אחת לא תקבל מדליה ונחזיר SUCCESS.
סיבוכיות כוללת: O(log(m)).

StatusType unite_teams(int teamId1, int teamId2) - נבדוק אם מזהי הקבוצות תקינים, אם לא נחזיר INVALID_INPUT. לאחר מכן נחפש את שתי הקבוצות ב-O(log(m)), אם לא מצאנו נחזיר FAILURE.
במידה והמדינות קיימות נבדוק שהספורט שלהן והמדינות זהים, אם לא נחזיר FAILURE.
אחרת, נאחד את הקבוצות בעזרת הפונקציה StatusType Team::mergeTeams(Team* team).
הסיבוכיות הכוללת: O(log(m) + O(mergeTeams(Team* team)).
נראה שהסיבוכיות של O(mergeTeams(Team* team)) = O(n_team1 + n_team2)
ולכן הסיבוכיות לבסוף תהיה: O(log(m)+ n_team1+ n_team2) כנדרש.

StatusType Teams::mergeTeams(Team* team) - שתי הקבוצות מכילות את תתי העצים המרכיבים את סך השחקנים בקבוצה, כעת נעבור על עצי ה-ID בסדר inOrder של כל קבוצה וניצור מערך ממויין באורך כמות המתחרים של כל קבוצה כפי שנלמד בתרגול. פעולות אלו לוקחות O(n_team) כמספר השחקנים בכל קבוצה.
ממזגים את שלושת המערכים למערך אחד ממוין ב- O(n_team1+n_team2). עוברים על המערך הממוין כדי למצוא את כמות המתחרים הייחודיים, כלומר שלא נספור את אותו מתחרה פעמיים, ב-O(n_team1+n_team2).
באותו אופן יוצרים מערך אחד ממוין ללא חזרות מעצי ה-Strength ב-O(n_team1+n_team2).
חוצים את מערך ה-ID הממוין לשלושה חלקים כמעט שווים לפי אינדקסים, ומוסיפים לגדלי החלקים הללו את שארית החלוקה בשלוש על מנת להשלים לגודל הקבוצה החדש. נותנים לכל חלק שם המתאר אותו לפי תת העץ

שהוא עתיד להיות: {Right ,Middle ,Left}. כל זאת ב-O(1).
עוברים על מערך ה-ID ועבור כל איבר, ניגשים למקביל שלו במערך Strength ומסמנים את ה-Position:{Left, Middle, Right} בהתאם למיקום האינדקס של ה-ID שלו במערך ה-ID הממוין. הסיבוכיות היא ב-O(n_team1+n_team2).
כעת נחלק את שלושת חלקי מערך ה-ID שכבר סימנו לשלושה מערכים עפ"י חלוקת ה-Position. הסיבוכיות היא בO(n_team1+n_team2).
נבצע שלושה מעברים על מערך ה-Strength. במבער הראשון ניקח רק את האיברים המסומנים ב-LEFT ונכניס אותם למערך משלהם. נשים לב כי זה המערך הזה יהיה ממוין וייצג את כל ה-ID שנמצאים במערך ID השמאלי. באותו אופן נמשיל עד לקבלת לשלושה מערכי Strength ממוינים המחולקים למערכים לפי ה-Position שלהם בסיבוכיות של O(n_team1+n_team2).
כעת נותרנו עם שלושה מערכי ID ושלושה מערכי Strength ממוינים כאשר האיברים בכל מערך Strength מייצגים את החוזק של האיברים במערך ID שיש להם את אותו Position. נותר ליצור עצים ריקים המתאימים לכל מערך ואחר כך למלא את העצים ב-O(n_team1+n_team2) כפי שנראה בתרגול. עתה נעבור על העצים הישנים של הקבוצה, נסיר מהם את המצביעים שלהם ל-Data ע"י השמת nullptr ולאחר מכן נמחק את העצים ונבצע השמה לעצים החדשים. כל זאת ב-O(n_team1+n_team2).
נעדכן את גודל הקבוצה, את חוזקה ואת ה-austerity ב-O(log(n_team1+n_team2))
נמחק את הקבוצה השנייה בO(n_team1+n_team2). אפוא נותרנו רק עם Team1 המכילה את כלל השחקנים של שתי הקבוצות ללא חזרות על אותו שחקן ובנוסף החוזק של הקבוצה וה-austerity שלה כבר מחושבים.
סה"כ הסיבוכיות של הפונקציה הנ"ל: O(n_team1 + n_team2)

ולכן עפ"י ההסבר לעיל הסיבוכיות של StatusType unite_teams(int teamId1, int teamId2) הינה O(log(m) + n_team1 + n_team2).

output_t <int> austerity_measures(int teamId) - נבדוק שמזהה הקבוצה תקין, אם לא נחזיר INVALID_INPUT. לאחר מכן נחפש את הקבוצה בעץ הקבוצות בO(log(m)) . אם הקבוצה לא קיימת נחזיר FAILURE. במידה והיא קיימת נחזיר SUCCESS ואת הausterity של הקבוצה שחושב מראש ושמור בשדה המתאים.

<u>סיבוכיות:</u> נשים לב כי בהינתן קבוצה, אין צורך לחשב את המדד הזה מכיוון שהוא מחושב בכל הכנסה או הוצאה של שחקן מהקבוצה, באיחוד בין שתי קבוצות או עדכון חוזק של שחקן. כלומר עבור הפונקציה get_austerity_measures, עלינו לבצע חיפוש בעץ הנבחרות של המשחק ולשלוף את הערך בשדה m_austerity. לכן הסיבוכיות הכוללת:O(log(m)) .

הפונקציה שמחשבת את ב-austerity של קבוצה נקראת int Team::calculateAusterity(), ואנו משתמשים בה כאשר מוסיפים שחקן לקבוצה, מסירים שחקן מקבוצה, מעדכנים חוזק של שחקן או מאחדים קבוצות.

הפונקציה מחשבת את החוזק המקסימלי בהורדת שלושה מתחרים מהקבוצה. ישנם שנים עשר מקרים ובכל מקרה אנו מחשבים את החוזק המקסימלי האפשרי עבור המקרה הזה ולבסוף לוקחים את המקסימלי מבין כל המקרים.

12 מקרים של Austerity_measures:

נחלק את השחקנים לשלושה עצי ID בגדלים שווים, לפי left, middle, right בסדר עולה, כך שהשליש של השחקנים עם ה-ID הקטן ביותר נמצא בעץ שמאל, השליש האמצעי באמצע והשליש העליון בעץ הימני.

נחלק ל-12 מקרים זרים ומשלימים לפי מספר השחקנים שהורדנו מכל עץ ID.

ראשית נחלק למקרים זרים ומשלימים לפי כמות המתמודדים שהורדנו מהעץ השמאלי ובתוך כל מקרה כזה נחלק לפי כמות המתמודדים שנוריד משאר העצים כדי להשלים לשלוש הורדות:

- **אם הורדנו אפס שחקנים מהעץ השמאלי:**
  1. אם הורדנו אפס מהאמצע ושלושה בימין
  2. אם הורדנו שלושה באמצע ואפס מימין
  3. אם הורדנו באמצע איבר אחד, עם ה-id הכי גדול, והורדנו שניים מימין
  4. אם הורדנו באמצע איבר אחד והוא לא האיבר עם ה-id הכי גדול (קיימים לפחות שני איברים בעץ האמצעי כי לא מבצעים את כל החישוב בפונקציה אם אין לפחות שישה איברים) והורדנו שני איברים מימין.
  5. אם הורדנו אחד מימין ושניים מהאמצע.
- **אם הורדנו שחקן אחד מהעץ השמאלי:**
  6. אם הורדנו אחד באמצע ואחד מימין.
  7. אם הורדנו אפס באמצע ושניים מימין.
  8. אם הורדנו אפס מימין ושניים באמצע.
- **אם הורדנו שניים מהעץ השמאלי:**
  9. אם הורדנו מהאמצע את האיבר עם ה-id הכי קטן
  10. אם הורדנו איבר מהאמצע שהוא לא עם ה-id הכי קטן
  11. אם הורדנו איבר אחד מימין.
- **אם הורדנו שלושה מהעץ השמאלי**
  12. הורדנו אפס מהאמצע ואפס מימין.

בכל המקרים האלה, נבחר להוריד את האיברים עם הstrength הכי קטן, למעט כאשר אנו בוחרים להוריד מהעץ האמצעי את האיבר עם ה-id הכי קטן (בהורדה של שניים משמאל) או הכי גדול (בהורדה של שניים מימין).

דוגמה למקרים 3-4:

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
חוזק	23	2	5	9	30	31	3	2	2	12	2	2	50	1	5	4	3	2	10	20	10

מסומנים בצהוב האיברים אותם נבחר להוריד מהעץ הימני כי יש להם את הstrength הכי קטן.

בכחול מסומן האיבר אותו מורידים במקרה 3 ובאדום מסומן האיבר אותו אנו מורידים במקרה 4.

נשים לב כי המקרים היחידים בהם יש לנו אפשרות להוריד את האיבר שעתיד לעבור עץ (כלומר לשנות צבע) הם מקרים בהם אנו מורידים שניים מהעץ הימני (או השמאלי) ואחד מהעץ האמצעי. זאת מכיוון שמרגע שהורדנו שניים מהעץ הימני (בה"כ, המקרה השמאלי הוא סימטרי) נקבל בדוגמה שאיבר מספר 14 אמור לעבור לקבוצה הימנית ולשנות את צבעו ואיבר 7 עובר לקבוצה האמצעית.

אם נוריד את איבר מספר 14 בדוגמה, הרי שאיבר 13 חייב לעבור לקבוצה הירוקה במקומו.

לכן חילקנו למקרים זרים – האם הורדנו את 14 או לא הורדנו את 14. אם הורדנו את 14 אז סך הכול הורדנו שלושה איברים ולכן נשאר רק לחשב את החוזק של הקבוצה כרגע. אם לא הורדנו את 14 אז הוא עובר קבוצה ונשאר לנו לגבש מתוך האיברים 8-13 (שעתידים לקבל אליהם את איבר מספר 7 שיעבור ברגע שנוריד איבר אחד) קבוצה עם כוח גדול ככל האפשר. לכן מבין האיברים 8-13 נבחר את האיבר עם הstrength הכי קטן.

<u>נכונות:</u> חילקנו ל-12 מקרים זרים ומשלימים, ובכל אחד מהמקרים הללו האיברים שעוברים קבוצה נקבעים באופן יחיד ולכן בחרנו להוריד את האיברים שיביאו לחוזק המקסימלי. כלומר לקחנו את החוזק המקסימלי של כל מקרה. אם ניקח את החוזק המקסימלי מבין 12 המקרים נקבל את ה-austerity_measures הנדרש.

<u>סיבוכיות:</u> נשים לב כי ישנם 12 מקרים שבכל מקרה יש שלושה חיפושים והסרות של איברים מהעץ, כלומר מספר קבוע של פעולות הכנסה, הוצאה וחיפוש שכל אחת מהן היא בסיבוכיות של O(log(num_of_contestants_in_team)). ומתקיים תמיד שמספר השחקנים בקבוצה קטן או שווה למספר השחקנים במשחק כולו.