

מבנה הנתונים יכול:

world_cup_t:

- **Avl<Team> all_teams** • עץ חיפוש ביןארי AMAZON-LVL שמכיל את כל הקבוצות במערכת
- **Avl<Team> good_teams** • עץ AVL שמכיל רק את הקבוצות הקשורות
- **Avl<Player> players_by_id** • עץ AVL שמכיל את כל השחקנים במערכת כאשר המפתחות בעץ הם המזהה של כל שחקן.
- **Avl<Player> players_by_goals** • עץ AVL שמכיל את כל השחקנים במערכת כאשר האופרטור בעץ זה מוגדר על ידי הסדר שМОגדר בפונקציה `get_all_players`. **כלומר:** שחקן₁ > שחקן₂ אם ו ורק אם השערים שהבקיע שחקן₁ קטן ממס' השערים שהבקיע שחקן₂. אם מס' השערים זהה, אז אם מס' הcartesisms שהוביל שחקן₁ גדול ממס' הcartesisms שהוביל שחקן₂, אם מס' הcartesisms גם זהה אז אם המזהה של שחקן₁ קטן מהמזהה של שחקן₂.
- **good_teams num** • מספר הקבוצות הקשורות - גם מספר הוצאותים בעץ `good_teams`.
- הבחנה חשובה:** מספר זה תמיד קטן ממספר השחקנים במערכת ובפרט מספר הקבוצות במערכת.
- **num_of_players** • מספר השחקנים במערכת.
- **Player* top_scorer** • השחקן שמדדorg הכى גבוי במערכת לפי מספר שערים (אם שווה אז לפי האופרטור שהוסבר לעללה).

:Team

- מזהה הקבוצה **teamId** •
- **Avl<Player> players_by_id** • עץ השחקנים של הקבוצה ממון לפי מזהה השחקנים.
- **Avl<Player> players_by_goals** • עץ השחקנים של הקבוצה ממון לפי שערים בהתאם להסביר בעץ של המערכת.
- **num_of_players** • מספר השחקנים בקבוצה.
- **points** • מספר הנקודות של הקבוצה.
- **goals** • סכום השערים של כל השחקנים בקבוצה.
- **cards** • סכום הcartesisms שהובילו השחקנים בקבוצה.
- **num_of_matches** • מספר המשחקים ששיחקה הקבוצה מאז שנכנסה למערכת.
- **Player* top_scorer** • השחקן שמדדorg הכى גבוי בקבוצה.
- **num_of_goal_keepers** • מספר השוערים בקבוצה. - שניתן יהיה לעקוב بكلות מתי קבוצה הופכת לכשירה ומתי כבר לא.
- **Team* team_in_all_teams** • מצביע של קבוצה בעץ הקבוצות הקשורות למועד המתואימה שלה בעץ הקבוצות של כל המערכת. (ה מצביע הוא אלה בקבוצות בעץ כל המערכת)

:Player

- **playerId** - מזהה השחקן
- **teamId** - מזהה הקבוצה של השחקן
- **Team* player_team** - מצביע לטיפוס שמייצג את קבוצת השחקן. - כך נוכל לקבל את הפרטים של הקבוצה בגישה ישירה.
- **goals** - מספר השערים שהבקיע השחקן. - מתעדכן בהתאם.
- **cards** - מספר הרטיסים שקיבל השחקן. - מתעדכן בהתאם.
- **games_played** - מספר המשחקים ששיחק השחקן כשהציגו למערכת.
- **games_of_team_without_player** - מספר המשחקים ששיחקה קבוצת השחקן לפני שהציגו אליה. אם שחקן עבר קבוצה באמצעותו, הערך מתעדכן בהתאם.
- **bool goalKeeper** - אם השחקן הוא שוער יקבל true, אחרת false.
- **Player* closest_player** - מצביע לשחקן הקרוב ביותר של השחקן. מתעדכן תוך כדי הפונקציות `add`, `remove`, `update_player_stats` וזה מתעדכן אצל השחקן בעז שמנוי לפי `id` בקבוצתו כדי שהסיבוכיות של הפונקציה `get_closest` לא תיפגע ותהייה $O(\log k + \log_{teamId} n)$ כאשר n קבוצת השחקן שיש להחזיר עבורו את השחקן הקרוב ביותר אליו.
- **bool checking_by_goals** - אם נרצה את השחקן בעז לפי `id`, קיבל את הערך false, אחרת קיבל true.
- אופרטור של שחקן יהיה מוגדר לפי הדגל הזה. אם הדגל דולק האופרטור יהיה רק לפי `id` של השחקן, אחרת לפי מה שהוגדר למעלה עם השערים, כרטיסים ועוד מזהה.
- **Player* himself_in_players_by_id** - מצביע לשחקן בעז השחקנים לפי `id` של כל המערכת. המצביע קיים מפני שבשינוי קבוצה של שחקן علينا לשנות את המידע הזה עבורו בצורת שמייצג אותו בעז כל המערכת כי אנו משתמשים בו לצורך מציאת קבוצת השחקן.

הערה: ראיינו בהרצאה שהכנה, הוצאה, חיפוש בעץ חיפוש מאוזן (AVL) מתקיימים בסיבוכיות של $O(\log n)$ במקרה הגרוע כאשר n זה מספר הצמתים בעץ. לכן כל פעולה של הכנסה, הוצאה או חיפוש צומת בעץ בפונקציות יהיו בסיבוכיות זו.

סיבוכיות מקום: מבנה הנתונים מכיל עצים בגודל χ שחקנים לכל היותר ועצים בגודל k קבוצות לכל היותר, בנוסף מספר שדות קבוע לכל שחקן ולכל קבוצה. ומספר שדות סופי בתחום $cup world$ לכט $O(k+h)$ מקום.

add_player

אם מגיע ערך לא חוקי יוחזר `invalid input`.

מוצאים את הקבוצה של השחקן בעץ הקבוצות(חיפוש בעץ בעאָג). אם לא מצאנו תיזרק חריגה ונחזיר `FAILURE`.

יצרים שחקן עם הנתונים המתאימים שהוא `goals_by` ואחד עם אותם נתונים שהוא `id_by`.

מוצאים את השחקן הקרוב ביותר אליו בצורה הבאה:

מוצאים את השחקן העוקב לו בדירוג השחקנים(על ידי חיפוש בעץ `players_by_goals` בסיבוכיות `(logn)O`)

מוצאים את השחקן שלפניו בדירוג השחקנים (על ידי חיפוש בעץ `players_by_goals` בסיבוכיות `(logn)O`)

משווים מי יותר קרוב ב-(1)O ומעדכנים אותו אצל השחקן שעתיד להיכנס גם למערכת אז זה יעודכן גם בצדמת שלו בקבוצתו.

מכניסים את השחקן שהוא `player_by_goals` בסיבוכיות `(logn)O` (אם השחקן כבר קיים תיזרק חריגה ונחזיר `FAILURE`). ואת השחקן שהוא לא `by_id` מעדכנים בסיבוכיות `(logn)O`.

טענה: השחקנים היחידים שיכל להשתנות להם `player_by_closest` הם שני השחקנים הקרובים ביותר. כי השחקן שנוסף יכול להיות הקרוב ביותר אליו המודרך.

כי אם קיים שחקן נוסף יותר זו סטירה בדרך בחירה של אותם שחקנים.

מוצאים את השחקנים הקרובים (אלו שנמצאו קודם) בסיבוכיות `(logn)O`

ומעדכנים אותם את `player_by_closest` באותה צורה שעדכנו את השדרה בשחקן שייצרנו.

מעדכנים את `top_scorer` על ידי השמרת הערך המקסימלי בעץ `players_by_goals` בסיבוכיות `(logn)O`.

מעדכנים את מספר השחקנים במערכת.(+1)

מוצאים את השחקן שהכנסנו בתוך העץ `players_by_id` בסיבוכיות `(logn)O` ומעדכנים לשחקן שייצרנו את המצביע לשם.

בודקים אם לפני הוספת השחקן הקבוצה כשרה:

אם כן מוסיףם את השחקן לקבוצה, מוסיףם את `goals` וה`cards` שלו לקבוצה ומעדכנים את `top_scorer` שלו כדי שמעדכנים את `top_scorer` של המערכת. מחזירים `success`

אם לא, מוסיףם את השחקן לקבוצה, מוסיףם את `goals` וה`cards` שלו לקבוצה ומעדכנים את `top_scorer` שלו כדי שמעדכנים את `top_scorer` של המערכת.

בודקים האם אחרי הוספת השחקן היא כשרה

אם כן מוסיףם אותה לעץ הקבוצות הקשריות, מעדכנים שם מצביע לקבוצה בעץ כל הקבוצות ומוסיפים 1 למספר הקבוצות הקשריות.

מחזירים `success`.

[remove_player\(int teamId\)](#)

אם מגע ערך לא חוקי יוחזר input invalid.

יצרים שחקן זמני עם הת.ז של השחקן שורצים למחוק ומוצאים את השחקן בפז_by_players. (אם לא מצאנו תירק חריגה וויחזר failure).

יצרים עותק של השחקן ובוותק משנים את goals_by_to奕 לאמת וכך מוצאים את השחקן בטאלס_goals. שומרים פינטרא לקבוצה שלו בעזרת השדה team שבו. מוחקים את השחקן מהקבוצה (מורידים goals_cards, num_of_goal_keepers בהתאם) ומורידים את מספר השחקנים בקבוצה ב-1.

בודקים אם הקבוצה הייתה כשרה ולאחר ההסרה כבר לא, אז מוחקים אותה מds_teams_good ומוחסרים 1 ממספר הקבוצות הקשורות. (סיבוכיות של חיפוש בטאלס_teams_good היא פחות מlogn כי בכל קבוצה כשרה יש לפחות 11 שחקנים ולכן יש players_by_goals מדויק ומשחקן אחריו כמו בפז_by_players. מוחקים את השחקן מds_players_by_to奕 ומשחקן closest_player בשחקן שלפניו ומשחקן אחריו כמו בפז_player_top. מעדכנים את הערך top_scorer על ידי השמת הערך המקסימלי בעז players_by_goals בפז_top_scorer. מוחסרים 1 ממספר השחקנים וויחזר success. סיבוכיות: חיפוש והסרה לעצם השחקנים - O(logn) במקרה הגורע.

[update_player_stats](#)

אם מגע ערך לא חוקי יוחזר input invalid.

יצרים שחקן זמני עם הת.ז של השחקן שורצים למחוק ומוצאים את השחקן בפז_by_players. שומרים פינטרא לקבוצה שלו בעזרת השדה team שבו. מעתיקים את השחקן לשחקן לשחקן copy כדי לשמור את הנתונים הנוכחיים.

מסירים את השחקן מהמערכת ומקבוצתו. ויצרים שחקן חדש עם הערכים המעודכנים בהתאם למקרים הבאים:
אם 0==games_played מכניסים את השחקן למערכת מחדש עם הנתונים החדשים.
אם 0>games_played מכניסים את השחקן למערכת עם הוספה הנתונים החדשים אלה הקיימים.
ומוחזרים success.

סיבוכיות: חיפוש, הסרה והכנה לעצם השחקנים - O(logn) במקרה הגורע.

[add_team\(int teamId, int points\)](#)

אם אחד הערכים לפחות לא חוקי יוחזר INVALID INPUT בהתאם.

אחרת, יוצרים קבוצה Team עם המזהה שלו ומספר הנקודות ההתחלתית, מתחילה את עצי השחקנים בה להיות ריקים, ואת שאר הערכים לאפסים. ומכניסים את הקבוצה לעץ הקבוצות של המערכת. אם הקבוצה כבר קיימת תירק שגיאה ונחזיר FAILURE, אחרת, הקבוצה תיקנס בהצלחה וויחזר SUCCESS.

סיבוכיות: הכנסה לעץ הקבוצות של המערכת, שהוא עצ חיפוש מואزن, בסיבוכיות O(logk) במקרה הגורע כאשר אזה מספר הקבוצות במערכת וזהו סך הסיבוכיות של הפונקציה במקרה המקורי.

remove_team(int teamId)

עבור ערך לא חוקי יוחזר INVALID INPUT .
יצרים קבוצה זמנית עם המזהה זהה כדי למצוא את הקבוצה שנרצה להסרה. אם הקבוצה לא קיימת תיזרק חריגה מתאימה ונחזיר FAILURE . אחרת, נבדק אם יש בקבוצה שחקן אחד לפחות - נחזיר FAILURE . אחרת, נסיר את הקבוצה מעץ הקבוצות וסימנו.

סיבוכיות: הסירה מהעץ מתקיימת ב($O(\log k)$) במקורה הגרוע, וזה אכן סיבוכיות של הפונקציה.

play_match(int teamId1, int teamId2)

עבור ערך לא חוקי יוחזר INVALID INPUT .
מוצאים את הקבוצות בעץ הקבוצות של המערכת ב - $O(\log k)$ (אם אחת הקבוצות לא קיימת תיזרק שגיאה ויהזיר FAILURE).

בודקים ששתי הקבוצות כשירות (11 שחקנים ושוער אחד לפחות), אם אחת מהן לא, נחזיר FAILURE .
בודקים מי הקבוצה המנצחת על ידי הנוסחה שמחשבת את התוצאה (עם points, goals, cards). הקבוצה המנצחת מקבלתוספה 3 נק לערך points שלה, ובמקורה של תיקו נוסיף 1 לכל קבוצה.
בנוסף, מוסיפים 1 לערך games_played_of_matches של כל קבוצה כדי לעדכן בהתאם את games_played של כל השחקנים בקבוצות.

סיבוכיות: מציאת הקבוצות בעץ הקבוצות ב - $O(\log k)$ במקורה הגרוע, בדיקות ועדכון ב - $O(1)$, וכן סה"כ סיבוכיות הפונקציה ($O(\log k)$) במקורה הגרוע כנדרש.

get_num_played_games(int playerId)

עבור ערך לא חוקי יוחזר INVALID INPUT .
מוצאים את השחקן בעץ השחקנים של המערכת לפי id על ידי שחקן זמני שיוצרים עם המזהה שהתקבל. (אם השחקן לא קיים תיזרק שגיאה ונחזיר FAILURE).
מחשבים ומוחזרים את הערך הבא:

games_played + num_of_matches - games_of_team_without_player

כאשר num_of_matches זה מספר המשחקים ששיחקה קבוצת השחקן. ככה, יובטח שהערך שיתקבל הוא אכן מספר המשחקים שהשחקן שיחק.

סיבוכיות: מציאת השחקן ב - $O(\log n)$ במקורה הגרוע והחזרת הערך מהשחקן שמצאנו. סה"כ ($O(\log n)$).

get_team_points(int teamId)

עבור ערך לא חוקי יוחזר INVALID INPUT .
מוצאים את הקבוצה בעלת המזהה teamId בעץ הקבוצות של המערכת. אם לא מוצאים נחזיר FAILURE .
אחרת, מצאנו את הקבוצה. נחזיר את ערך points של הקבוצה.

סיבוכיות: מציאת הקבוצה בעץ הקבוצות של המערכת ב-($O(\log k)$) במקורה הגרוע והחזרה של ערך הנקודות ב-($O(1)$). סה"כ ($O(\log k)$) במקורה הגרוע כנדרש.

unite_teams(int teamId1, int teamId2, newTeamId)

עבור ערך לא חוקי כלשהו יוחזר INPUT INVALID.

מצאים את הקבוצות בעץ הקבוצות של המערכת. (אם לא קיימים נחזיר FAILURE). מתקיים ב- $O(\log k)$ במקורה הגרוע. עבור כל קבוצה:

- מכניסים את השחקנים מהעץ players_by_id בסדר inorder לטור מערך.
- מכניסים את השחקנים מהעץ players_by_goals בסדר inorder לטור מערך.

מציגים את מערך השחקנים לפי pi בצורה ממונת לטור מערך גדול שמכיל את כל השחקנים של שתי הקבוצות ממונינים לפי pi .

באוטו אופן בדיק, מציגים את השחקנים לפי goals למערך אחד. מתקיים ב- $O(n_{teamId1} + n_{teamId2})$.

מחלקים למקרים:

newTeamId == teamId1

* שומרים דגל שאומר האם קבוצה 1 (הקבוצה בעלת המזהה teamId1) כשרה לפני האיחוד או לא.

* עוברים על השחקנים מקבוצה 2 (הקבוצה בעלת המזהה teamId2) ונעדקן לכל שחקן את:

- המצביע והמזהה של הקבוצה לה שיר - כתעת לקבוצה 1.

- מוסיפים ל- $\text{player_games_of_team_withput_player}$ שלו את num_of_matches של קבוצה 1.

* לחברים את כל הערכים של קבוצה 2 עם קבוצה 1: $\text{points}, \text{goals}, \text{cards}, \text{num_of_players}$, ...

. $O(n_{teamId2})$ - זה מתקיים ב-

* אם קבוצה 2 הייתה קבוצה כשרה, מוחקם אותה מעץ הקבוצות הקשריות.

* מוחקם את קבוצה 2 מעץ הקבוצות של המערכת. - $(\log k)O$ במקורה הגרוע.

* יוצרים עץ כמעט שלם ריק שמכיל את מספר הזרים המתאים שהוא סכום השחקנים בשתי הקבוצות לפי מה שראינו בתרגול.

* מכניסים לקבוצה 1 את השחקנים מהמערך הממווג לעץ כמעט שלם שבנו. (סדר inorder).

* מעדכנים top_scorer חדש לקבוצה המאוחדרת. (מציאת המקסימלי ב- $O(\log k)$). כי הולכים ימינה עד שאי אפשר יותר).

* אם קבוצה 1 לא הייתה קבוצה כשרה לפני וعصיו כן כשרה, נכניס אותה לעץ הקבוצות הקשריות עם מצביע לקבוצה בעץ הקבוצות.

newTeamId == teamId2

זהה למקרה 1 רק שהכל הפוך בין קבוצה 1 וקבוצה 2. (הכל סימטרי לחłówטן).

newTeamId משני המזהאים

* יוצרים קבוצה עם המזהה החדש ומכניסים אותה עם המזהה newTeamId והערכים של קבוצה 1 (לא כולל עצי השחקנים) לעץ הקבוצות של המערכת. (אם קיימת קבוצה עם המזהה זהה תיזרק חריגה ויוחזר FAILURE).

* מוצאים את הקבוצה בעץ הקבוצות כדי שנוכל להכניס את כל השחקנים ולעדקן את כל המידע בהתאם.

* עוברים על השחקנים במערכות הממווגים (לפי pi ולפי goals) ונבצע עבור כל שחקן את הדברים הבאים:

- אם השחקן מכבוצה 1, מוסיפים `games_of_team_without_player` שלו את `num_of_matches` - של קבוצה 2.
- אם השחקן מכבוצה 2, מוסיפים `games_of_team_without_player` שלו את `num_of_matches` - של קבוצה 1.
- מעדכנים את המצביע והמזהה לקבוצה אצל השחקן להיות הקבוצה עם המזהה החדש.
- מוחברים את כל הערכיהם של קבוצה 2 לקבוצה כי היא מכילה את המידע של קבוצה 1. (,(),.cards, num_of_players, num_of_matches, num_of_goalkeepers
 - מכנים את השחקנים ב- `inorder` מהמערכות המומוגים לעצם המתאימים להם בקבוצה. (id, goals)
 - בהתאם מהמערך לעצם. $O(n_{teamId_1} + n_{teamId_2})$.
 - מעדכנים את `top_scorer` החדש. $O(\log(n_{teamId_1} + n_{teamId_2}))$.

* אם הקבוצות היו קבוצות כשרות, מסירים אותן מעץ הקבוצות הכשרות.

מסירים את הקבוצות מעץ הקבוצות של המערכת. $O(\log k)$.

* אם הקבוצה החדשה המאווחדת היא קבוצה כשרה, מכנים אותה לעץ הקבוצות הכשרות, כמובן עם מצביע לקבוצה בעץ הקבוצות.

סיבוכיות: מציאות הקבוצות והכנסות לעצם הקבוצות מתבצעים ב- $O(\log k)$ במקרה הגרוע, מעברים על המערכת, ועדכנים אצל השחקנים, הכנסה לעצם בגודל של כמות השחקנים בשתי הקבוצות יחד - $O(n_{teamId_1} + n_{teamId_2})$ אז סה"כ $O(\log k + n_{teamId_1} + n_{teamId_2})$

get_top_scorer(int teamId)

אם מגיע ערך לא חוקי יוחזר `invalid`.

אם `teamId < 0` ומספר השחקנים במערכת הוא 0 יוחזר `failure`. אחרת, מחזירים את המזהה של `top_scorer` של המערכת ב- $O(1)$ מהצביע שמתוחזק.

אם `teamId < 0` מוצאים את הקבוצה בעץ הקבוצות ב- $O(\log k)$. אם אין שחקנים בקבוצה מחזירים `failure`. אחרת, מחזירים `success` ואת המזהה של `top_scorer` שנמצא בקבוצה.

סיבוכיות: אם `teamId < 0` זה מתקיים ב- $O(1)$, אחרת, $O(\log k)$.

get_all_players_count(int teamId)

אם מגיע ערך לא חוקי יוחזר `invalid`.

אם `teamId < 0` יוחזר `players_of_team` של המערך. אחרת, מוצאים את הקבוצה ב- `teams_all` ב- $O(\log k)$ ומוחזרים את מספר השחקנים בקבוצה.(אם הקבוצה לא קיימת בעץ יוחזר `failure`).

סיבוכיות: אם `teamId < 0` זה מתקיים ב- $O(1)$, אחרת, $O(\log k)$.

get_all_players(int teamId)

אם `teamId < 0` ניכnis למערך בגודל מס' השחקנים במערכת את השחקנים מהעץ `inorder` ב- `players_by_goals` אחרית, מוצאים את הקבוצה ומוכנים לערוך בגודל מס' השחקנים בקבוצה את השחקנים מהעץ `inorder` `players_by_goals` של הקבוצה ב- `zerod`.

סיבוכיות: אם $0 < \text{teamId} < n$, אחרת ב- $O(\log k + \log n_{\text{teamId}})$.

get_closest_player(int playerId, int teamId)

אם הקלט לא יהיה תקין יוחזר `invalid input`. מוצאים את הקבוצה (אם לא נמצאה יוחזר `failure`), מוצאים את השחקן בקבוצה ובו יהיה `playerId` שלו כי דאגנו לעדכן גם שם בפונקציות בהם השתנה.

סיבוכיות: מציאת הקבוצה וצאת השחקן בקבוצה - $O(\log k + \log n_{\text{teamId}})$ במקרה הגורע.

knockout_winner(min teamId, max teamId)

אם הקלט לא יהיה תקין יוחזר `invalid input`.
יצור מערך של קבוצות ונוסיף אליו את כל הקבוצות הקשורות שהמזהה שלהם נמצא בטוחה הנתון. - נניח שמשתמש הקבעות כנ"ל הוא z .

נעשה זאת באופן הבא: בעץ `good_teams` נמצא את הערך המינימלי שבתוך, המקסימלי והאב המשותף הכי גבוהה שלהם. מכיוון שהערך המינימלי בטוחה אז גם כל התת עץ הימני שלו אשר בו ערכים גדולים ממנו, מוצאים בטוחה. לכן, נוסיף אותו ואז אוטם לפ' הסדר במערך, נעה מהלך בעץ ונעשה אותו דבר לכל ערך בעץ עד שעולים לאב המשותף. האב המשותף בטוחה, לכן נוסיף אותו ומשם נרד' ימינה בעץ, אם הערך בטוחה אז כל התת עץ השמאלי שלו נמצא בטוחה גם כן כי הוא גדול מהאב המשותף אך קטן מהערך המקסימלי בטוחה. ממשיר לרדת ימינה בכל העץ עד שנסויים אותו. כמובן $k = \min\{n, \text{num_of_good_teams}\}$ וגם $\text{num_of_good_teams} \leq n$.
ולכן החיפוש בעץ `good_teams` הוא $O(\log(\min\{n, k\}) + r)$.

נכnis את המזהים של הקבוצות למערך `scores` בגודל z ואת ערכי הנוסחא של הניקוד של הקבוצות לתוך מערך `scores`. נתאר את האלגוריתם שמצוין את `z` לאחר שכל הערכים הדרושים נמצאים במערכות:
אם z אי זוגי, נשמר את הקבוצה الأخيرة שבמערך בצד.

כל עוד z עד לא 1:

רצים מ0 עד -1 ומשווים כל זוג קבוצות: כל קבוצה שמנצחת שומרים את ה- dp שלה ואת הניקוד שהוא סכום הניקודים פלוס 3 במערכות המתאימים בתא המתאים.
מחלקים את z בשתיים. אם z אי זוגי כעת יש קבוצה בצד, נכnis אותה למערך.
אם הוא אי זוגי ואין קבוצה בצד, נשמר את הקבוצה الأخيرة בצד.
לבסוף, אם הגענו ל- $-1 = z$ ואין קבוצה בצד סימנו ויש לנו את המזהה של הקבוצה המנצחת.

סיבוכיות: תהליך הכנסת הקבוצות המתאימות מתקיים בסיבוכיות $O(\log(\min\{n, k\}) + r)$ במקרה הגורע,

האלגוריתם למציאת הקבוצה המנצחת רץ: $(r) = (r/2 + r/4 + \dots + 1) = r(1 + 1/2 + \dots + 1/r) = O(\log(\min\{n, k\}) + r)$.
לכן, סה"כ הסיבוכיות במקרה הגורע של הפונקציה היא $O(\log(\min\{n, k\}) + r)$.