

מבני נתונים 234218 חורף תשפ"ב

גיליון רטוב מספר 2 – מעודכן לתאריך 25.12.2021
עמוד 1 מתוך 9



מתרגל ממונה על התרגיל:

תומר כהן, tomerc20@cs.technion.ac.il

תאריך ושעת הגשה:

15/01/2022 בשעה 23:55

אופן ההגשה:

בזוגות. יורד ציון לתרגילים שיוגשו ביחידים בלי אישור מהמתרגל הממונה על

התרגיל.

הנחיות כלליות:

- תשובות לשאלות המרכזיות אשר ישאלו יתפרסמו בחוץ ה FAQ באתר הקורס לטובת כלל הסטודנטים. שימו לב כי תוכן ה FAQ הוא מחייב וחובה לקרוא אותו, אם וכאשר הוא יתפרסם. לא יתקבלו דחיות או ערעורים עקב אי קריאת ה FAQ.
- לפני שאתם ניגשים לקודד את פתרוןכם, ודאו כי יש לכם פתרון העומד בכל דרישות הסיבוכיות התרגיל. תרגיל שאינו עומד בדרישות הסיבוכיות יחשב כפסול.
- **העתקת תרגילי בית רטובים תיבדק באמצעות תוכנת בדיקות אוטומטית, המזהה דמיון בין כל העבודות הקיימות במערכת, גם כאלו משנים קודמות.** לא ניתן לערער על החלטת התוכנה. התוכנה אינה מבדילה בין מקור להעתק! אנא הימנעו מהסתכלות בקוד שאינו שלכם.
- שאלות על התרגיל יש לפרסם **באתר הפיאצה של הקורס:**
piazza.com/technion.ac.il/fall2022/234218
- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת barakgahtan@cs.technion.ac.il.



הקדמה:

בתרגיל זה נדון שוב במשחקים מאתגרים בין סטודנטים על מנת לזכות במל"גים נחשקים ומומלצים, מזווית קצת שונה. הפעם אנו יודעים כי יש רק k קבוצות של שחקנים. ובתרגיל זה, נרצה לעזור לנשיא הטכניון לעקוב אחר המשחקים המתרחשים, כאשר כאן הנשיא מעוניין שלכל שחקן תהיה תוצאה, אותו הוא (הנשיא) יוכל לשנות כרצונו, תוצאה זו הינה בטווח 1 עד $scale$ (כאשר מתקיים $1 \leq scale \leq 200$, כלומר $scale$ קבוע מבחינת שיקולי חישובים). ובתור סטודנטים לקורס מבני נתונים, בחר בכך לעזור לו עם מערכת זו.



דרוש מבנה נתונים למימוש הפעולות הבאות:

`void* init(int k, int scale)`

מאתחל מבנה נתונים עם k קבוצות, כל קבוצה מוגדרת להיות ריקה בהתחלה.

פרמטרים: k מספר הקבוצות במשחק.

$scale$ התוצאה המקסימלית במשחק.

ערך החזרה: מצביע למבנה נתונים ריק או `NULL` במקרה של כישלון ($k \leq 0, scale > 200$).

נחשב ככישלון).

סיבוכיות זמן: $O(k)$ במקרה הגרוע.

`StatusType mergeGroups(void *DS, int GroupID1, int GroupID2)`

הפעולה מאחדת בין שתי הקבוצות עם המזהים `GroupID1`, `GroupID2`, כל השחקנים משתי הקבוצות עוברים להיות בקבוצה החדשה. לאחר איחוד שתי הקבוצות, שני המזהים `GroupID1`, `GroupID2` יתייחסו לקבוצה המאוחדת. לדוגמה, אם מאחדים את קבוצה מספר 3 עם קבוצה מספר 5, אז בעתיד כל התייחסות לקבוצה מספר 3 או 5 תהיה לקבוצה המאוחדת.

פרמטרים: `DS` מצביע למבנה הנתונים.

`GroupID1` מזהה הקבוצה הראשונה.

`GroupID2` מזהה הקבוצה השנייה.

ערך החזרה: `ALLOCATION_ERROR` במקרה של בעיה בהקצאת זכרון.

`INVALID_INPUT` אם $GroupID1 \leq 0, GroupID2 \leq 0$ או `DS == NULL`.

או $GroupID1 > k, GroupID2 > k$.

`SUCCESS` במקרה של הצלחה.

סיבוכיות זמן: $O(\log^*(k) + n)$ משוער, במוצע על הקלט, כאשר k הוא מספר הקבוצות ו- n מספר

השחקנים בשתי הקבוצות המאוחדות.

Union
Find

(אנאחור)
(3,5) סם 4,
וקהם (3,4)

מבני נתונים 234218 חורף תשפ"ב

גיליון רטוב מספר 2 – מעודכן לתאריך 25.12.2021
עמוד 3 מתוך 9



StatusType addPlayer(void *DS, int PlayerID, int GroupID, int score)

הוספת שחקן חדש שמתחיל את המשחק משלב Level=0 במשחק, ועם תוצאה התחלתית score, ומשתייך לקבוצה בעלת המזהה GroupID.

id
level
score

פרמטרים: DS מצביע למבנה הנתונים.
PlayerID מזהה השחקן.
GroupID מזהה הקבוצה.
score התוצאה ההתחלתית של השחקן.
ערך החזרה: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT אם $GroupID \leq 0, GroupID > k, DS == NULL$
או $score \leq 0, score > scale, PlayerID \leq 0$
FAILURE אם קיים כבר שחקן עם מזהה PlayerID.
SUCCESS במקרה של הצלחה.
סיבוכיות: $O(\log^*(k))$ משוערך, בממוצע על הקלט, כאשר k הוא מספר הקבוצות.

→
+ צריך
הוספת שחקן
להקבוצה
ב(1) α מוסדק
נני לא צוא אותו
אח"כ ה(1) α מוסדק
עם רעיון:
ש"אש מוסדק ב"א"מ
לקבוצת שחקנים

StatusType removePlayer(void *DS, int PlayerID)

השחקן בעל המזהה PlayerID "נפסל" מהמשחק, וניתן למחוק אותו מהמערכת.

פרמטרים: DS מצביע למבנה הנתונים.
PlayerID מזהה השחקן שיש להסיר מהמערכת.
ערך החזרה: INVALID_INPUT אם $PlayerID \leq 0$ או $DS == NULL$
FAILURE אם אין שחקן עם מזהה PlayerID.
SUCCESS במקרה של הצלחה.
סיבוכיות: $O(\log^*(k) + \log(n))$ משוערך, בממוצע על הקלט, כאשר k זה מספר הקבוצות, n זה מספר השחקנים הכולל במשחק כרגע.

StatusType increasePlayerIDLevel(void *DS, int PlayerID, int LevelIncrease)

הגדלת השלב במשחק של השחקן בעל המזהה PlayerID ב-LevelIncrease.

פרמטרים: DS מצביע למבנה הנתונים.
PlayerID מזהה השחקן שיש לעדכן.
LevelIncrease כמות שלבי המשחק שיש להוסיף לשחקן.
ערך החזרה: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT אם $LevelIncrease \leq 0$ או $PlayerID \leq 0, DS == NULL$
FAILURE אם אין שחקן עם מזהה PlayerID.
SUCCESS במקרה של הצלחה.
סיבוכיות: $O(\log^*(k) + \log(n))$ משוערך, בממוצע על הקלט, כאשר k זה מספר הקבוצות, n זה מספר השחקנים הכולל במשחק כרגע.

בנוסף (5 נק!): תארו בחלק היבש כיצד ניתן היה לממש את הפעולה הנ"ל בסיבוכיות של

$O(\log^*(k) + \log(n))$ משוערך (שימו לב שכאן הסיבוכיות אינה בממוצע על הקלט).

→
התפלגות
לכאן אח"כ לא יקרה?

Ⓢ האם השחקן
אמיתי? להקבוצה
על מספרו
זהו (אולי)
"אמיתי"

סיבוכי

הפעול
רנוח

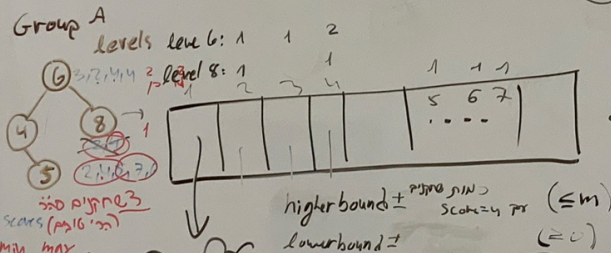
level ←

$$O(\underbrace{n}_{\text{size}} + \underbrace{k}_{\text{size}})$$

Union Find
(log*(n)) ≤ 1327 82

מבנה של $\log n$:
מבנה של n : $\log n$

חמור דא קאזא:



270N
Scores
1-Scale

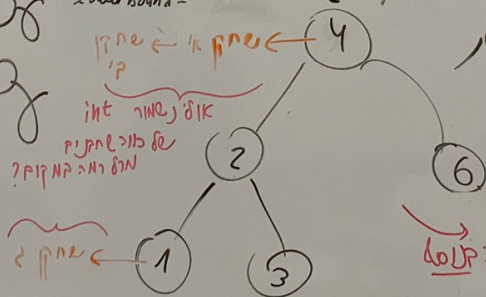
Score

$\frac{\text{no pignoz}}{\text{sears (p. 16 'n)}}$
 min max
 1 2 / ME3 PG
 BT PG

higher bound \pm ^{positive score} score = 4 per ($\leq m$)
lower bound \pm (≥ 0)

[illegible]

የሰነድ ጥያቄ
የሰነድ ጥያቄ



6) דולפן: כחית הים

StatusType averageHighestPlayerLevelByGroup(void *DS, int GroupID, int m, double * avgLevel)

הפעולה מחשבת את הרמה הממוצעת בה נמצאים m השחקנים ברמות הגבוהות ביותר בקבוצה עם המזהה $GroupID$, אם $GroupID == 0$ אז הפעולה מחזירה את סכום הרמות של m השחקנים ברמות הגבוהות ביותר מבין כל השחקנים במשחק.

לדוגמא, אם יש לנו את הקבוצה הבאה: 2 שחקנים ברמה 8, 4 שחקנים ברמה 6, 9 שחקנים ברמה 3 וקיבלנו להחזיר את הרמה הממוצעת של 9 השחקנים ברמות הגבוהות ביותר (כלומר קיבלנו $m = 9$), אזי $avgLevel$ יצביע לערך 5.444444, כי סכום הרמות של 9 השחקנים ברמות הגבוהות ביותר הינו 49 (2 שחקנים ברמה 8, 4 שחקנים ברמה 6, ו-3 שחקנים ברמה 3), ולכן נקבל $\frac{49}{9} = 5.444444$.

פרמטרים:	DS	מצביע למבנה הנתונים.
	GroupID	מזהה הקבוצה שעבורה נרצה לקבל את המידע.
	m	מספר השחקנים עבורם אנחנו רוצים לקבל את המידע.
	avgLevel	מצביע למשנה שבו תוחזר התוצאה.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם אחד המצביעים שווה ל-NULL או
		$GroupID < 0$ או $GroupID > k$ או $m < 0$.
	FAILURE	אם m גדול ממספר השחקנים בקבוצה עם המזהה $GroupID$
		או ממספר השחקנים במשחק אם $GroupID == 0$.
	SUCCESS	במקרה של הצלחה, כלומר כל מצב אחר.
סיבוכיות:	$O(\log^*(k) + \log n)$	משוערך, בממוצע על הקלט, כאשר k הוא מספר הקבוצות ו- n הוא מספר השחקנים הכולל במשחק כרגע.

בנוס (5 נקודות):

StatusType getPlayersBound(void *DS, int GroupID, int score, int m, int * LowerBoundPlayers, int * HigherBoundPlayers)

הפעולה מחשבת את הטווח של מספר השחקנים בעלי מספר נקודות השווה ל- $score$ מבין m השחקנים ברמות הגבוהות ביותר בקבוצה עם המזהה $GroupID$, אם $GroupID == 0$ אז הפעולה מחזירה את הטווח של מספר השחקנים בעלי מספר נקודות השווה ל- $score$ של m השחקנים ברמות הגבוהות ביותר מבין כל השחקנים במשחק. כלומר, בהינתן m , אם ברמה הגבוהה ביותר יש $a_1 \leq m$ אנשים, נבחר את כל האנשים האלה לבדיקה, אם ברמה השנייה הכי גבוהה יש $a_2 \leq m - a_1$ אנשים, נבחר את כל האנשים האלה לבדיקה, וכן הלאה, עד שנגיע לקבוצה שגודלה מקיים $a_k > m - \sum_{i=1}^{k-1} a_i$, ממנה נבחר $m - \sum_{i=1}^{k-1} a_i$ אנשים (כמות האנשים שנשאר לנו לבחור), בחירת אנשים אלה יכולה להשפיע על טווח התוצאות האפשריות. כלומר, בהינתן m , נגדיר קבוצה חוקית של m אנשים אם אין אדם שאינו בקבוצה ונמצא ברמה גדולה ממש מאדם שנמצא בקבוצה. ובפונקציה זו נרצה לבדוק מה הטווח של קבוצה עם m אנשים עם תוצאה ששווה ל- $score$ שמקיימת תנאי זה.

$$\forall a \in A \setminus B, level(a) \leq minLevelB$$

ה'תשס"ח
ביום ג' ט"ו שבט
ה'תשס"ח
יב"מלי.
יב"מלי.

השחקנים הכולל במשחק כרגע.

סיבוכיות:

סיבוכיות מקום - $O(n + k)$ במקרה הגרוע, כאשר n הוא מספר השחקנים ו- k הוא מספר הקבוצות.



הנחיות לגבי הסיבוכיות המשוערכת:

- הסיבוכיות המשוערכת הנוגעת לחלק שנובע ממספר הקבוצות (k) , היא משוערכת עבור כל הפעולות יחד. כלומר, לכל רצף של t פעולות מהקבוצה MergeGroups, NumPlayersAfterLevel, SumHighestLevelPlayers, ScoreChangePlayerID, IncreaseLevel, RemovePlayer, AddPlayer הסיבוכיות הכוללת תהיה $O(t \cdot \log^*(k) + t \cdot f(n, m))$ כאשר $f(n, m)$ היא פונקציה כלשהי שתלויה רק במספר השחקנים (n) , ומספר התוצאות האפשריות (m) .
- הסיבוכיות המשוערכת הנוגעת לחלק שנובע ממספר השחקנים (n) , היא משוערכת עבור כל הפעולות יחד. כלומר, לכל רצף של t_1 פעולות מהקבוצה IncreaseLevel, RemovePlayer, ו- t_2 פעולות מהקבוצה ScoreChangePlayerID הסיבוכיות הכוללת תהיה $O(m \cdot f(k) + (t_1 \cdot m + t_2) \cdot \log(n))$ כאשר $f(k)$ היא פונקציה כלשהי שתלויה רק במספר הקבוצות (k) .

ערכי החזרה של הפונקציות:

- בכל אחת מהפונקציות, ערך ההחזרה שיוחזר ייקבע לפי הכלל הבא:
- תחילה, יוחזר INVALID_INPUT אם הקלט אינו תקין.
- אם לא הוחזר INVALID_INPUT:
- בכל שלב בפונקציה, אם קרתה שגיאת הקצאה יש להחזיר ALLOCATION_ERROR.
- אם קרתה שגיאה אחרת, כפי שמצוין בכל פונקציה, יש להחזיר מיד FAILURE מבלי לשנות את מבנה הנתונים.
- אחרת יוחזר SUCCESS.



הנחיות:

חלק יבש:

- **הציון על החלק היבש הוא 50% מהציון של התרגיל.**
- לפני מימוש הפעולות בקוד יש לתכנן היטב את מבני הנתונים והאלגוריתמים ולוודא כי באפשרותכם לממש את הפעולות בדרישות הזמן והזיכרון שלעיל.
- הגשת החלק הרטוב מהווה תנאי הכרחי לקבלת ציון על החלק היבש, כלומר, הגשה בה יתקבל אך ורק חלק יבש תגרור ציון 0 על התרגיל כולו.
- יש להכין מסמך הכולל תיאור של מבני הנתונים והאלגוריתמים בהם השתמשתם בצירוף הוכחת סיבוכיות הזמן והמקום שלהם. חלק זה עומד בפני עצמו וצריך להיות מובן לקורא גם לפני העיון בקוד. אין צורך לתאר את הקוד ברמת המשתנים, הפונקציות והמחלקות, אלא ברמה העקרונית.
- ראשית הציגו את מבני הנתונים בהם השתמשתם. רצוי ומומלץ להיעזר בציור.
- לאחר מכן הסבירו כיצד מימשתם כל אחת מהפעולות הנדרשות. הוכיחו את דרישות סיבוכיות הזמן של כל פעולה תוך כדי התייחסות לשינויים שהפעולות גורמות במבני הנתונים.
- הוכיחו שמבנה הנתונים וכל הפעולות עומדים בדרישת סיבוכיות המקום.
- החסמים הנתונים בתרגיל הם לא בהכרח הדוקים ולכן יכול להיות שקיים פתרון בסיבוכיות טובה יותר. מספיק להוכיח את החסמים הדרושים בתרגיל.
- רמת פירוט: יש להסביר את כל הפרטים שאינם טריוויאליים ושחשובים לצורך מימוש הפעולות ועמידה בדרישות הסיבוכיות. אין לדון בפרטים טריוויאליים (הפעילו את שיקול דעתכם בקשר לזה, ושאלו את האחראי על התרגיל אם אינכם בטוחים). אין לצטט קטעים מהקוד כתחליף להסבר. אין צורך לפרט אלגוריתמים שנלמדו בכתה. כמו כן, אין צורך להוכיח תוצאות ידועות שנלמדו בכתה, אלא מספיק לציין בבירור לאיזו תוצאה אתם מתכוונים.
- **על חלק זה לא לחרוג מ-8 עמודים.**
- והכי חשוב **!keep it simple**

חלק רטוב:

- מומלץ לממש תחילה את מבני הנתונים בצורה הכללית ביותר ורק אז לממש את הפונקציות הנדרשות בתרגיל.
- אנו ממליצים בחום על מימוש **Object Oriented**, **C++**, מימוש כזה יאפשר לכם להגיע לפתרון פשוט וקצר יותר לפונקציות אותן עליכם לממש ויאפשר לכם להכליל בקלות את מבני הנתונים שלכם. על מנת לעשות זאת הגדירו מחלקה, נאמר `PlayersManager`, וממשו בה את דרישות התרגיל. אח"כ, על מנת לייצר התאמה לממשק ה C ב `library1.h`, ממשו את `library1.cpp` באופן הבא:

```
#include "library1.h"
```

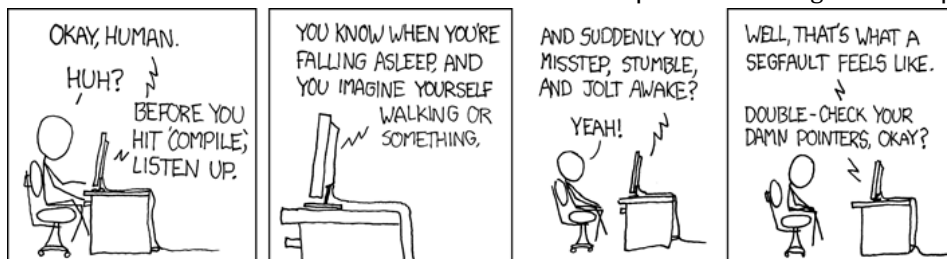
```
#include "PlayersManager.h"
```

```
void* Init() {
    PlayersManager *DS = new PlayersManager();
    return (void*)DS;
}
StatusType AddGroup(void *DS, int GroupID){
    return ((PlayersManager*)DS)-> AddGroup (GroupID);
}
```

- על הקוד להתקמפל על csl3 באופן הבא:

```
g++ -std=c++11 -DNDEBUG -Wall *.cpp
```

- עליכם מוטלת האחריות לוודא קומפילציה של התכנית ב- `g++`. אם בחרתם לעבוד בקומפיילר אחר, מומלץ לקמפל ב- `g++` מידי פעם במהלך העבודה.



מבני נתונים 234218 חורף תשפ"ב

גיליון רטוב מספר 2 – מעודכן לתאריך 25.12.2021

עמוד 9 מתוך 9



- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ `library1.h`
- קראו היטב את הקובץ הנ"ל, לפני תחילת העבודה.
- אין לשנות את הקבצים אשר סופקו כחלק מהתרגיל, ואין להגיש אותם.
- עליכם לממש בעצמכם את כל מבני הנתונים (למשל אין להשתמש במבנים של STL ואין להוריד מבני נתונים מהאינטרנט). **כחלק מתהליך הבדיקה אנו נבצע בדיקה ידנית של הקוד ונודא שאכן מימשתם את מבני הנתונים שבהם השתמשתם.**
- יש לתעד את הקוד בצורה נאותה וסבירה.
- מצורפים לתרגיל קבצי קלט ופלט לדוגמא.
- שימו לב: התוכנית שלכם תיבדק על קלטים שונים מקבצי הדוגמא הנ"ל, שיהיו ארוכים ויכללו מקרי קצה שונים. לכן, מומלץ מאוד לייצר בעצמכם קבצי קלט, לבדוק את התוכנית עליהם, ולודא שהיא מטפלת נכון בכל מקרה הקצה.

הגשה:

■ חלק יבש + חלק רטוב:

- הגשת התרגיל הנה **אך ורק** אלקטרונית דרך אתר הקורס.
- יש להגיש קובץ ZIP שמכיל את הדברים הבאים:
 - בתיקייה הראשית:
 - קבצי ה-Source Files שלכם (ללא הקבצים שפורסמו).
 - קובץ PDF אשר מכיל את הפתרון היבש עבור. מומלץ להקליד את החלק הזה אך ניתן להגיש קובץ PDF מבוסס על סריקה של פתרון כתוב בכתב יד. שימו לב כי במקרה של כתב לא קריא, כל החלק השני לא תיבדק.
 - קובץ `submissions.txt`, המכיל בשורה הראשונה את שם, תעודת הזהות וכתובת הדוא"ל של השותף הראשון ובשורה השנייה את שם, תעודת הזהות וכתובת הדוא"ל של השותף השני. לדוגמה:

Tomer Cohen 012345678 tomerc20@cs.technion.ac.il

Henry Taub 123456789 taub@cs.technion.ac.il

- **שימו לב כי אתם מגישים את כל שלושת החלקים הנ"ל.**
- אין להשתמש בפורמט כיווץ אחר (לדוגמה RAR), מאחר ומעריך הבדיקה האוטומטי אינו יודע לזהות פורמטים אחרים.
- **יש לוודא שכאשר נכנסים לקובץ הזיפ הקבצים מופיעים מיד בתוכו ולא בתוך תיקיה שבתוך קובץ הזיפ.** עבור הגשה שבה הקבצים יהיו בתוך תיקייה, הבדיקה האוטומטית לא תמצא את הקבצים ולא תוכל לקמפל ולהריץ את הקוד שלכם ולכן תיתן אוטומטית 0.
- לאחר שהגשתם, יש באפשרותכם לשנות את התוכנית ולהגיש שוב.
- ההגשה האחרונה היא הנחשבת.
- הגשה שלא תעמוד בקריטריונים הנ"ל תפסל ותקנס בנקודות!

דחיות ואיחורים בהגשה:

- דחיות בתרגיל הבית תינתנה אך ורק לפי תקנון הקורס.
- 5 נקודות יורדו על כל יום איחור בהגשה ללא אישור מראש. באפשרותכם להגיש תרגיל באיחור של עד 5 ימים ללא אישור. תרגיל שיוגש באיחור של יותר מ-5 ימים ללא אישור מראש יקבל 0.
- במקרה של איחור בהגשת התרגיל יש עדיין להגיש את התרגיל אלקטרונית דרך אתר הקורס.
- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת barakgahtan@cs.technion.ac.il.
- לאחר קבלת אישור במייל על הבקשה, מספר הימים שאושרו לכם נשמר אצלנו. לכן, אין צורך לצרף להגשת התרגיל אישורים נוספים או את שער ההגשה באיחור.

בהצלחה!