

מבני נתונים 234218 אביב תשפ"ד

גיליון רטוב מספר 2 – מעודכן לתאריך 23.07.2024
עמוד 1 מתוך 8



שחר כהן, shacharcohen@campus.technion.ac.il

מתרגל ממונה על התרגיל:

15/08/2024 בשעה 23:59

תאריך ושעת הגשה:

בזוגות. אין להגיש ביחידים. (אלא באישור מתרגל אחראי של הקורס)

אופן ההגשה:

הנחיות כלליות:

- שאלות על התרגיל יש לפרסם באתר הפיאצה של הקורס תחת לשונית "hw-wet-2":
 - האתר: piazza.com/technion.ac.il/spring2024/234218
 - נא לקרוא את השאלות של סטודנטים אחרים לפני שמפרסמים שאלה חדשה, למקרה שנשאלה כבר.
- נא לקרוא את המסמך "נהלי הקורס" באתר הקורס. בנוסף, נא לקרוא בעיון את כל ההנחיות בסוף מסמך זה.
- בפורום הפיאצה ינוהל FAQ ובמידת הצורך יועלו תיקונים כהודעות נעוצות (Pinned Notes). תיקונים אלו מחייבים.
- התרגיל מורכב משני חלקים: יבש ורטוב.
 - לאחר קריאת כלל הדרישות, מומלץ לתכנן תחילה את מבני הנתונים על נייר. דבר זה יכול לחסוך לכם זמן רב.
 - לפני שאתם ניגשים לקודד את פתרונכם, ודאו כי יש לכם פתרון העומד בכל דרישות הסיבוכיות בתרגיל. תרגיל שאינו עומד בדרישות הסיבוכיות יחשב כפסול.
 - את הפתרון שלכם מומלץ לחלק למחלקות שונות שאפשר לממש (ולבדוק!) בהדרגתיות.
 - המלצות לפתרון התרגיל נמצאות באתר הקורס תחת: "Programming Tips Session".
- המלצות לתכנות במסמך זה אינן מחייבות, אך מומלץ להיעזר בהן.
- 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.
- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת:
jonathan.gal@campus.technion.ac.il



הקדמה:

לאחר שעזרתם לממשלת אנגליה לעקוב אחר הפיראטים באיים הקריביים מממשלת אנגליה החליטה להקים ציי ספינות כדי להילחם בבעיית הפיראטים, אך מכיוון שאין לממשלת אנגליה תקציב הציים יורכבו מפיראטים שהיא תגייס על מנת להילחם בשאר הפיראטים. ידוע כי פיראטים הם בלתי צפויים ולכן היא רוצה שתבנו מערכת שתעזור לה לנהל את הציים שלה ולעקוב אחר הפיראטים שהיא מעסיקה. כל צי מורכב מספינות, יכול להעסיק פיראטים ויכול להתאחד עם ציים אחרים, הציים והפיראטים מיוצגים על ידי ID ייחודי, לא ניתן להסיר פיראטים מהמערכת או למחוק מידע על ציים שהתווספו למערכת. לכל אורך המסמך בהתייחסות לסיבוכיות זמן ריצה ומקום הסימון n הוא מספר הפיראטים ו- m הוא מספר הציים.

דרוש מבנה נתונים למימוש הפעולות הבאות:

`oceans_t()`

מאתחלת מבנה נתונים ריק. תחילה אין במערכת ספינות או פיראטים.

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

`virtual ~oceans_t()`

הפעולה משחררת את המבנה (כל הזיכרון אותו הקצאתם חייב להיות משוחרר).

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: $O(n + m)$ במקרה הגרוע.

`StatusType add_fleet(int fleetId)`

מוקם צי חדש בעל מזהה `fleetId`. בהתחלה לצי יש ספינה אחת ואין לו פיראטים.

פרמטרים:

`fleetId`

מזהה הצי שצריך להוסיף.

ערך החזרה:

ALLOCATION_ERROR

במקרה של בעיה בהקצאה/שחרור זיכרון.

INVALID_INPUT

`fleetId <= 0`

FAILURE

אם כבר התווסף למערכת צי עם מזהה `fleetId`.

SUCCESS

במקרה של הצלחה.

סיבוכיות זמן: $O(1)$ משוערך בממוצע על הקלט.



`StatusType add_pirate(int pirateId, int fleetId)`

פיראט בעל מזהה ייחודי `pirateId` מגויס לצי `fleetId`. הפיראט מתחיל בלי כסף ואם יש בצי `x` פיראטים הוא יקבל את הדרגה `x+1`.

פרמטרים:

| | |
|-----------------------|---------------------------|
| <code>pirateId</code> | מזהה הפיראט שצריך להוסיף. |
| <code>fleetId</code> | מזהה הצי של הפיראט. |

ערך החזרה:

| | |
|-------------------------------|--|
| <code>ALLOCATION_ERROR</code> | במקרה של בעיה בהקצאה/שחרור זיכרון. |
| <code>INVALID_INPUT</code> | אם <code>pirateId <= 0</code> או <code>fleetId <= 0</code> . |
| <code>FAILURE</code> | אם קיים כבר פיראט עם מזהה <code>pirateId</code> או שהצי עם המזהה <code>fleetId</code> לא קיים. |
| <code>SUCCESS</code> | במקרה של הצלחה. |

סיבוכיות זמן: $O(\log^* m)$ משוערך בממוצע על הקלט, כאשר m הוא מספר הציים במערכת.

`StatusType pay_pirate(int pirateId, int salary)`

ממשלת אנגליה רוצה לשלם לפיראט עם המזהה `pirateId` משכורת בסך `salary` מטבעות.

פרמטרים:

| | |
|-----------------------|---------------------------|
| <code>pirateId</code> | מזהה הפיראט שמקבל משכורת. |
| <code>salary</code> | מזהה הצי של הפיראט. |

ערך החזרה:

| | |
|-------------------------------|---|
| <code>ALLOCATION_ERROR</code> | במקרה של בעיה בהקצאה/שחרור זיכרון. |
| <code>INVALID_INPUT</code> | אם <code>pirateId <= 0</code> או <code>salary <= 0</code> . |
| <code>FAILURE</code> | אם לא קיים פיראט עם מזהה <code>pirateId</code> . |
| <code>SUCCESS</code> | במקרה של הצלחה. |

סיבוכיות זמן: $O(1)$ משוערך בממוצע על הקלט.

`output_t < int > num_ships_for_fleet(int fleetId)`

יש להחזיר את מספר הספינות שבצי `fleetId`.

פרמטרים:

| | |
|----------------------|-----------|
| <code>fleetId</code> | מזהה הצי. |
|----------------------|-----------|

ערך החזרה: מספר הספינות בצי, ובנוסף סטטוס:

| | |
|-------------------------------|---|
| <code>ALLOCATION_ERROR</code> | במקרה של בעיה בהקצאה/שחרור זיכרון. |
| <code>INVALID_INPUT</code> | אם <code>fleetId <= 0</code> . |
| <code>FAILURE</code> | אם הצי עם המזהה <code>fleetId</code> לא קיים. |
| <code>SUCCESS</code> | במקרה של הצלחה. |

סיבוכיות זמן: $O(\log^* m)$ משוערך בממוצע על הקלט, כאשר m הוא מספר הציים במערכת.

`output_t < int > get_pirate_money(int pirateId)`

ממשלת אנגליה רוצה לבדוק כמה מטבעות יש ברשות הפיראט שהמזהה שלו הוא `pirateId`.

פרמטרים:

| | |
|--|--|
| <code>pirateId</code> | מזהה הפיראט. |
| <u>ערך החזרה:</u> מספר המטבעות ברשות הפיראט, ובנוסף סטטוס: | |
| <code>ALLOCATION_ERROR</code> | במקרה של בעיה בהקצאה/שחרור זיכרון. |
| <code>INVALID_INPUT</code> | אם <code>pirateId ≤ 0</code> . |
| <code>FAILURE</code> | אם לא קיים פיראט עם מזהה <code>pirateId</code> . |
| <code>SUCCESS</code> | במקרה של הצלחה. |

סיבוכיות זמן: $O(1)$ בממוצע על הקלט.

`StatusType unite_fleets(int fleetId1, int fleetId2)`

הציים בעלי המזהים `fleetId1` ו `fleetId2` מתאחדים, והמזהה החדש שלהם הוא המזהה של הצי הגדול יותר מבחינת מספר הפיראטים מבין השניים (במקרה של שוויון המזהה החדש הוא `fleetId1`). מספר הספינות בצי המאוחד הוא סכום מספר הספינות בשני הציים, והדרגה של כל הפיראטים בצי הקטן יותר מבחינת מספר הפיראטים תגדל כמספר הפיראטים בצי הגדול יותר. לאחר האיחוד אם בה"כ מזהה הצי החדש הוא `fleetId1` אז לא קיים יותר צי במזהה `fleetId2` וגם לא ניתן להוסיף צי חדש במזהה זה. לדוגמא:

אם ישנם שני ציים 1 ו 2 כך שלצי 1 יש ספינה אחת ויש את הפיראטים 3 עם דרגה 1, 5 עם דרגה 2 ו 4 עם דרגה 3. ולצי 2 יש שתי ספינות ויש את הפיראטים 1 עם דרגה 1 ו 2 עם דרגה 2. לאחר איחוד הציים, הצי החדש יהיה עם המזהה 1, יהיו בו 3 ספינות ויהיו בו את הפיראטים 3 עם דרגה 1, 5 עם דרגה 2, 4 עם דרגה 3, 1 עם דרגה 4 ו 2 עם דרגה 5.

פרמטרים:

| | |
|-------------------------------|---|
| <code>fleetId1</code> | מזהה הצי הראשון. |
| <code>fleetId2</code> | מזהה הצי השני. |
| <u>ערך החזרה:</u> | |
| <code>ALLOCATION_ERROR</code> | במקרה של בעיה בהקצאה/שחרור זיכרון. |
| <code>INVALID_INPUT</code> | אם <code>fleetId1 ≤ 0</code> , <code>fleetId2 ≤ 0</code> , <code>fleetId1 == fleetId2</code> . |
| <code>FAILURE</code> | אם אין ציים במזהים <code>fleetId1</code> או <code>fleetId2</code> במערכת, או שאחד מהציים ריקים. |
| <code>SUCCESS</code> | במקרה של הצלחה. |

סיבוכיות זמן: $O(\log^* m)$ משוערך בממוצע על הקלט, כאשר m הוא מספר הציים במערכת.

`StatusType pirate_argument(int pirateId1, int pirateId2)`

הפיראטים בעלי המזהים `pirateId1` ו `pirateId2` נקלעו לוויכוח בעת עבודתם בצי, כתוצאה מכך, הפיראט שמספר דרגתו גבוה יותר יצטרך לשלם לפיראט שמספר דרגתו נמוך יותר, וסכום הכסף שהוא ישלם נקבע לפי ההפרש בין הדרגות שלהם. פיראט יכול להיות בעל סכום כסף שלילי לאחר שנדרש לשלם לפיראט אחר. לדוגמא:

אם ניקח את הצי מהדוגמא `unite_fleets`, והפיראטים בעלי המזהים 3 ו 2 מתווכחים, מכיוון שהדרגה של הפיראט בעל המזהה 3 היא 1 והדרגה של הפיראט בעל המזהה 2 היא 5, פיראט 2 יצטרך לשלם 4 מטבעות לפיראט 3.

פרמטרים:

| | |
|------------------------|---------------------|
| <code>pirateId1</code> | מזהה הפיראט הראשון. |
| <code>pirateId2</code> | מזהה הפיראט השני. |

ערך החזרה:

| | |
|--|------------------|
| במקרה של בעיה בהקצאה/שחרור זיכרון. | ALLOCATION_ERROR |
| אם $pirateId1 \leq 0$ או $pirateId2 \leq 0$ או $pirateId1 == pirateId2$. | INVALID_INPUT |
| אם אין פיראטים עם המזהים $pirateId1$ או $pirateId2$ או שהם לא באותו הצי. | FAILURE |
| במקרה של הצלחה. | SUCCESS |
| סיבוכיות זמן: $O(\log^* m)$ משוערך בממוצע על הקלט, כאשר m הוא מספר הציים במערכת. | |



סיבוכיות מקום:

סיבוכיות המקום הדרושה עבור מבנה הנתונים היא $O(n + m)$ במקרה הגרוע, כאשר n הוא מספר הפיראטים, ו- m הוא מספר הציים. כלומר בכל רגע בזמן הריצה, צריכת המקום של מבנה הנתונים תהיה לינארית בסכום מספרי הפיראטים והציים במערכת.

סיבוכיות המקום הנדרשת עבור כל פעולה (כלומר, זיכרון "העזר" שכל פעולה משתמשת בו) אינה מצוינת לכל פעולה לחוד, אך אסור לעבור את סיבוכיות המקום הדרושה שמוגדרת לכל המבנה.

ערכי החזרה של הפונקציות:

כל אחת מהפונקציות מחזירה ערך מטיפוס `StatusType` שייקבע לפי הכלל הבא:

- תחילה, יוחזר `INVALID_INPUT` אם הקלט אינו תקין.
 - אם לא הוחזר `INVALID_INPUT`:
 - בכל שלב בפונקציה, אם קרתה שגיאת הקצאה/שחרור יש להחזיר `ALLOCATION_ERROR`. מצב זה אינו צפוי אלא באחד משני מקרים (לרוב): באמת השתמשתם בקלט גדול מאוד ולכן המבנה ניצל את כל הזיכרון במערכת, או שיש זליגת זיכרון בקוד.
 - אם קרתה שגיאה אחרת, כפי שמצוין בכל פונקציה, יש להחזיר מיד `FAILURE` מבלי לשנות את מבנה הנתונים.
 - אחרת, יוחזר `SUCCESS`.
- חלק מהפונקציות צריכות להחזיר בנוסף עוד פרמטר (לרוב `int`), לכן הן מחזירות אובייקט מטיפוס `output_t<T>`. אובייקט זה מכיל שני שדות: הסטטוס (`__status`) ושדה נוסף (`__ans`) מסוג `T`. במקרה של הצלחה (`SUCCESS`), השדה הנוסף יכיל את ערך החזרה, והסטטוס יכיל את `SUCCESS`. בכל מקרה אחר, הסטטוס יכיל את סוג השגיאה והשדה הנוסף לא מעניין.
- שני הטיפוסים (`output_t<T>`, `StatusType`) ממומשים כבר בקובץ "`wet1util.h`" שניתן לכם כחלק מהתרגיל.

הנחיות: **חלק יבש:**

- החלק היבש הוא חלק מהציון על התרגיל כפי שמצוין בנהלי הקורס.
- לפני מימוש הפעולות בקוד יש לתכנן היטב את מבני הנתונים והאלגוריתמים ולוודא כי באפשרותכם לממש את הפעולות בדרישות הזמן והזיכרון שלעיל.
- החלק היבש חייב להיות מוקלד.
- 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. חלק יבש זה לא תיעוד קוד.
- 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. אין (וגם אין צורך) להשתמש בתוצאות של עצי דרגות והלאה.
- **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 (זכרו שיש תרגיל רטוב נוסף בהמשך הסמסטר).
- על הקוד להתקמפל על csl3 באופן הבא:

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

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



הערות נוספות:

- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ pirates24b2.h.
- קראו היטב את הקובץ הנ"ל, לפני תחילת העבודה.
- אין לשנות את הקבצים main24b2.cpp ו-wet2util.h אשר סופקו כחלק מהתרגיל, ואין להגיש אותם.
 - את שאר הקבצים ניתן לשנות.
 - תוכלו להוסיף קבצים נוספים כרצונכם, ולהגיש אותם.
 - העיקר הוא שהקוד שאתם מגישים יתקמפל עם הפקודה לעיל, כאשר מוסיפים לו את שני הקבצים wet2util.h ו-main24b2.cpp.
- 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 ונוודא שאכן מימשתם את מבני הנתונים שבהם השתמשתם.**
- בפרט, אסור להשתמש ב-std::vector, std::pair, או כל אלגוריתם של STL.

מבני נתונים 234218 אביב תשפ"ד

גיליון רטוב מספר 2 – מעודכן לתאריך 23.07.2024

עמוד 8 מתוך 8



- ניתן להשתמש במצביעים חכמים (Smart pointers כמו `shared_ptr`), בספריית `math` או בספריית `exception`.
- חשוב לוודא שאתם מקצים/משחררים זיכרון בצורה נכונה (מומלץ לוודא עם `valgrind`). לא חייבים לעבוד עם מצביעים חכמים, אך אם אתם מחליטים כן לעשות זאת, לוודא שאתם משתמשים בהם נכון. (תזכרו שהם לא פתרון קסם, למשל, כאשר יוצרים מעגל בהצבעות)
- שגיאות של `ALLOCATION_ERROR` בד"כ מעידות על זליגה בזיכרון.
- מצורפים לתרגיל קבצי קלט ופלט לדוגמא, ניתן להריץ את התוכנה על הקלט ולהשוות עם הפלט המצורף.
- **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, ולוודא 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. למעט הקבצים `main24b2.cpp` ו-`wet2util.h`, שאסור לשנות.
 - קובץ `PDF` בשם `dry.pdf` אשר מכיל את הפתרון היבש. החלק היבש חייב להיות מוקלד.
 - 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:

John Doe 012345678 doe@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!
 - אחרי שאתם מכינים את ההגשה בקובץ `zip` מומלץ מאוד לקחת אותה לשרת ולהריץ את הבדיקות שלכם עליה כדי לוודא שאתם מגישים את הקוד שהתכוונתם להגיש בדיוק (ושהוא מתקמפל).

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 jonathan.gal@campus.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!