מבני נתונים 1

גליון (רטוב) 1 (החלק היבש)

מגישים:

שם	ת.ז	מייל
אילון הלוי	328137831	eilon.halevy@campus.technion.ac.il
מקסים גורביץ	322207671	maximgurwitz@campus.technion.ac.il

דרישות מבנה הנתונים:

$Ocean(\,)$	יצירת אובייקט חדש שמתאר רשימה ריקה.	$O(1)$
$\sim Ocean(\,)$	מחיקת מבנה הנתונים (פינוי כל הזיכרון)	$O(n+m)$
$StatusType\ add\_ship(int\ shipId, int\ cannons)$	הפעולה מוסיפה ספינה חדשה למבנה הנתונים (במידה וספינה אם אותו מזהה לא קיימת)	$O(\log m)$
StatusType remove_ship(int shipId)	הפעולה מסירה את הספינה בעלת המזהה shipId מתוך מבנה הנתונים, זאת במידה וקיימת ספינה בעלת המזהה הנ"ל וגם ספינה זו ריקה מפיראטים.	$O(\log m)$
StatusType add_pirate(int pirateId, int shipId, int treasure)	במידה ולא קיים פיראט בעל המזהה pirateId על הספינה, וכן קיימת ספינה בעלת המזהה shipId מוסיפים את הפיראט לספינה.	$O(\log m + \log n)$
StatusType remove_pirate(int pirateId)	אם במבנה הנתונים קיים הפיראט בעל המזהה pirateId, מוחקים אותו ממבנה הנתונים.	$O(\log n)$
StatusType treason(int sourceShipId, int destShipId)	במידה וקיימות ספינות במבנה הנתונים עם המזהים sourceShipId, destShipId, ובנוסף הספינה בעלת המזהה sourceShipId אינה ריקה מפיראטים, מעבירים את הפיראט הותיק ביותר על הספינה בעלת המזהה sourceShipId לספינה בעלת המזהה destShipId.	$O(\log m + \log n)$
StatusType update_pirate_treasure(int pirateId, int change)	במידה וקיים פיראט בעל המזהה pirateId במבנה הנתונים, לשנות את ערך האוצר שלו ב-change מטבעות.	$O(\log n)$
output_t < int > get_treasure(int pirateId)	מחזיר את הערך של אוצר הפיראט בעל המזהה pirateId במידה וזה קיים במבנה הנתונים.	$O(\log n)$
output_t < int > get_cannons(int shipId)	במידה וקיימת ספינה בעלת המזהה shipId במבנה הנתונים, מחזירים את מספר התותחים של הספינה.	$O(\log m)$
output_t < int > get_richest_pirate(int shipId)	במידה וקיימת הספינה בעלת המזהה shipId במבנה הנתונים, ובנוסף ספינה זו אינה ריקה מפיראטים, מחזירים את הפיראט העשיר ביותר בספינה.	$O(\log m)$
StatusType ships_battle(int shipId1, int shipId2)	במידה וקיימות הספינות עם המזהים shipId1, shipId2 במבנה הנתונים, בודקים איזו ספינה מנצחת לפי כמות התותחים המאויישים, כל פיראט מהספינה המפסידה נותן לכל הפיראטים מספינה המנצחת מטבע מהאוצר שלו.	$O(\log m)$

בנוסף, ממבנה הנתונים נדרש לעמוד בהגבלה של סיבוכיות זיכרון $O(n+m)$ במבנה עצמו, ובכל הפעולות.

מימוש:

בחרנו לממש את מבנה הנתונים בעזרת עץ AVL של ספינות שבו כל ספינה מחזיקה עץ AVL של (מצביעים של) פיראטים ורשימה מקושרת דו-כיוונית של (מצביעים של) פיראטים.

את העץ הראשי (עץ הספינות) נמיין לפי ה-ID של הספינות. עבור כל ספינה עץ הפיראטים שלה ימוין על פי כמות המטבעות של כל פיראט (במקרה של כמות זהה של מטבעות נמיין לפי ה-ID), ואילו הרשימה המקושרת תמוין על פי סדר הכנסתם לספינה (בדומה למבנה הנתונים- תור).

כמו כן, נחזיק בנפרד עץ AVL של כלל הפיראטים הממוין לפי ה-ID כדי לתמוך בגישה לפיראטים לפי המזהה.

נוחות מימוש:

<u>שדות הפיראט</u>:

- מזהה הפיראט
- מצביע לספינה עליה הוא נמצא
- המצביע ל-$Node$ של הרשימה הדו-כיוונית של הספינה בה נשמר המצביע לפיראט זה
- גודל האוצר של אותו פיראט, ללא החלק שהוא מפקיד בספינה

<u>שדות הספינה:</u>

- מזהה הספינה
- מספר התותחים של הספינה
- המאזן הכלכלי של הספינה (כל הפיראטים מפקידים בספינה אותה כמות מטבעות)
- רשימה מקושרת דו-כיוונית של מצביעי הפיראטים שנמצאים על אותה הספינה
- עץ AVL של מצביעי הפיראטים שנמצאים על אותה הספינה שממויינים לפי איזה פיראט עשיר יותר. (פרטי מימוש- יצרנו אובייקט PirateRank שמחזיק במצביע ועושה אובר-לואוד לאופרטורים $=,<$)
- מצביע לפיראט העשיר ביותר

הערה: כל פיראט יימצא על ספינה אחת בלבד! תמיד (עד שמוסר ממבנה הנתונים)

<u>שדות האוקיינוס (מבנה הנתונים):</u>

- העץ הראשי (עץ AVL של ספינות, ממויין לפי ID)
- עץ כלל הפיראטים (ממויין לפי ID)

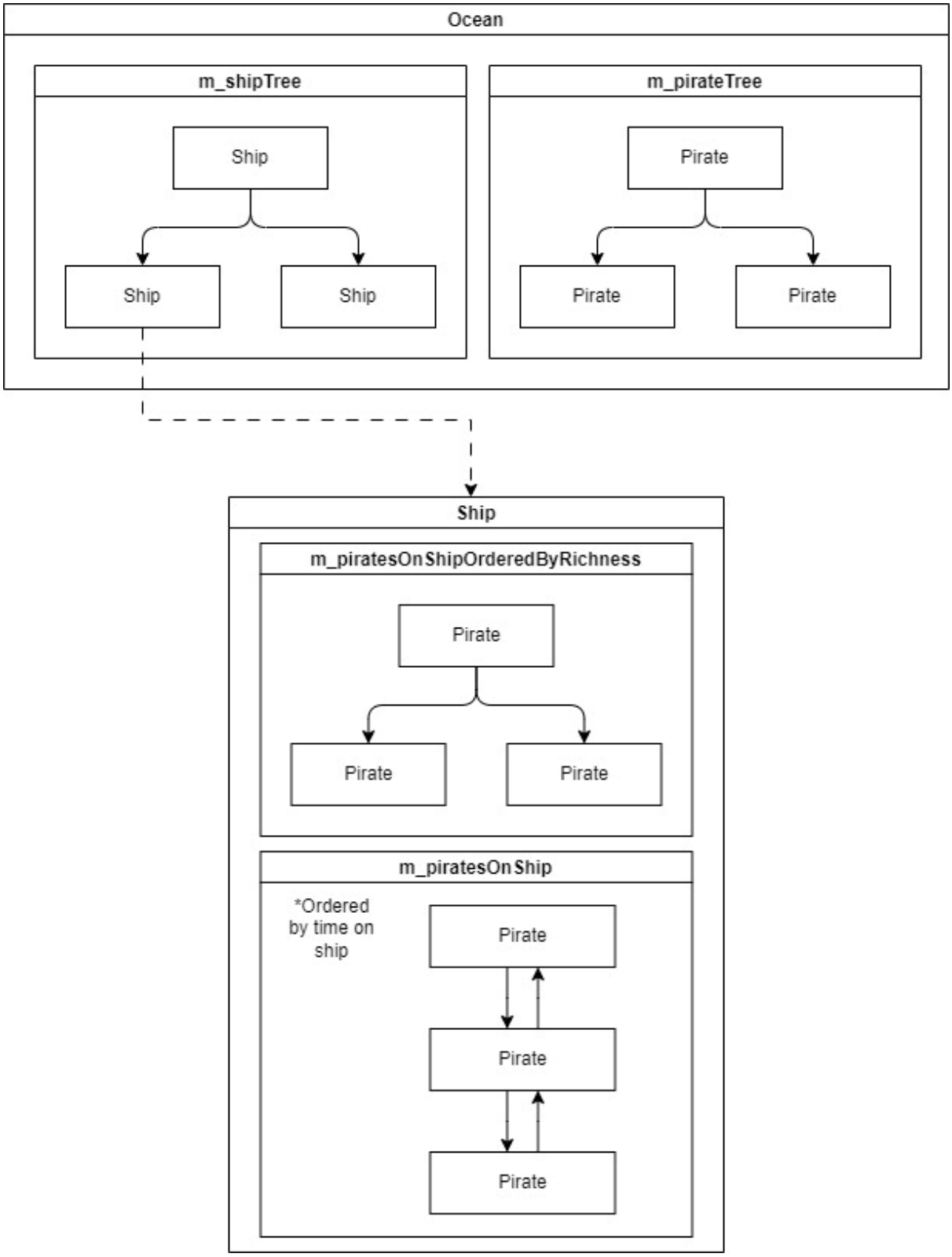
סיבוכיות הזיכרון של מבנה הנתונים:

- גודל העץ הראשי הוא כמספר הספינות
  - בכל ספינה, גודל עץ הפיראטים על אותה ספינה הוא מספר הפיראטים באותה ספינה
  - בכל ספינה, גודל הרשימה המקושרת הדו-כיוונית באותה ספינה
- גודל עץ כלל הפיראטים הוא כמספר הפיראטים הכולל במבנה הנתונים (באוקיינוס)

אנו יודעים שסיבוכיות המקום של רשימה מקושרת הן של עץ AVL היא $O(n)$, כאשר n זהו גודל מבנה הנתונים (מספר האיברים במבנה הנתונים). בנוסף, כל פיראט נמצא בספינה אחת בדיוק, ולכן סכום גדלי כל הרשימות המקושרות של כל הספינות, כמו סכום גדלי כל עצי AVL של כל הספינות הוא מספר הפיראטים.

לכן, סיבוכיות המקום למבנה הנתונים סה"כ הוא כנדרש: $O(m) + 2 * O(n) + O(n) = O(n + m)$

שרטוט מבנה הנתונים:



נוכיח כעת את סיבוכיות זמן הריצה של כל הפעולות

הפעולה add_ship:

הוספה של איבר חדש לעץ AVL היא בזמן ריצה $O(\log m)$, כאשר m הוא גודל עץ (מספר הספינות).

הפעולה נכשלת אם קיים בעץ איבר עם אותו מזהה. לכן, סה"כ סיבוכיות הזמן של הפעולה היא $O(\log m)$, כנדרש.

הפעולה remove_ship:

הסרה של איבר מעץ AVL היא בזמן ריצה $O(\log m)$, כאשר m הוא גודל עץ (מספר הספינות).

הפעולה נכשלת אם לא קיים בעץ איבר עם אותו מזהה, או כאשר על הספינה יש פיראט/ים. לכן, סה"כ סיבוכיות הזמן של הפעולה היא $O(\log m)$, כנדרש.

הפעולה add_pirate:

בדיקה האם קיים פיראט בעל אותו מזהה במבנה הנתונים- חיפוש בעץ הפיראטים, זוהי פעולה בזמן ריצה $O(\log n)$, כאשר n הוא גודל עץ (מספר הפיראטים).

חיפוש הספינה בעלת מזהה במבנה הנתונים- חיפוש בעץ הראשי, זוהי פעולה בזמן ריצה $O(\log m)$, כאשר m הוא גודל עץ (מספר הספינות).

אם הפעולות צלחו (לא נמצא פיראט ונמצאה הספינה), נוסיף את הפיראט לספינה:

- הכנסה לרשימה מקושרת (לתחילתה) ושמירת המצביע באובייקט הפיראט, $O(1)$.
- הכנסה של הפיראט לעץ הספינה $O(\log n)$. (גודל עץ הספינה חסום ע"י גודל עץ כלל הפיראטים)
- עדכון הפיראט העשיר ביותר (העלה השמאלי ביותר בעץ הפיראטים של הספינה). זמן ריצה $O(\log n)$, חסם לגובה העץ, כידוע מספר האיברים חסום ע"י מספר הפיראטים הכולל, ועומק העץ חסום ע"י $O(\log n)$, כאשר n הוא גודל העץ.

סה"כ: $O(\log n) + O(\log m) + O(1) + O(\log n) + O(\log n) = O(\log m + \log n)$, כנדרש.

הפעולה remove_pirate:

בדיקה האם קיים פיראט בעל אותו מזהה במבנה הנתונים- חיפוש בעץ הפיראטים, זוהי פעולה בזמן ריצה $O(\log n)$, כאשר n הוא גודל עץ (מספר הפיראטים). אם נמצא הפיראט- נסיר את הפיראט מהספינה:

- גישה לספינה באמצעות המצביע שבשדה הפיראט, $O(1)$.
- הסרת ה-$Node$ מהרשימה המקושרת הדו-כיוונית של אותה הספינה, $O(1)$.
(באמצעות המצביע שבשדה הפיראט לאותו ה-$Node$)
- הסרת הפיראט מעץ הפיראטים של הספינה, $O(\log n)$. (גודלו חסום ע"י גודל עץ כלל הפיראטים)
- עדכון הפיראט העשיר ביותר (השורש השמאלי ביותר בעץ הספינה), זמן ריצה $O(\log n)$, חסם לגובה העץ.
- לבסוף, נסיר את הפיראט מעץ כלל הפיראטים, $O(\log n)$.

סה"כ: $O(\log n) + O(1) + O(\log n) + O(\log n) + O(\log n) = O(\log n)$, כנדרש.

הפעולה treason:

ראשית נחפש כל ספינה בעץ הספינות, זמן כל חיפוש $O(\log m)$.

אם הספינות קיימות וספינת המקור אינה ריקה:

- נסיר את הפיראט הותיק ביותר מהרשימה המקושרת - ממוינת לפי הותק לכן $O(1)$.
- נסיר את אותו פיראט מעץ הפיראטים של הספינה המקורית - חסום על ידי סך כל הפיראטים לכן $O(\log n)$.
- נוסיף לפיראט את המאזן הכלכלי של הספינה אותה עזב, $O(1)$ (כרגע הוא יחזיק בכל האוצר שלו).
- נכניס את הפיראט לרשימה המקושרת של הספינה החדשה (לתחילת הרשימה) - $O(1)$.
- נכניס את הפיראט לעץ הפיראטים של הספינה החדשה - $O(\log n)$.
- נחסיר מהפיראט את המאזן הכלכלי של הספינה אליה נכנס, $O(1)$. (תכונה של מבנה הנתונים).
- בהסרה ובהכנסה של אותו הפיראט, מעדכנים את המצביע לפיראט העשיר ביותר על אותה ספינה, כפי שתואר, בסיבוכיות זמן $O(\log n)$.

סה"כ:

$$O(\log m) + O(1) + O(\log n) + O(1) + O(1) + O(\log n) + O(1) + O(\log n) = O(\log n + \log m)$$

כנדרש.

הפעולה update_pirate_treasure:

חיפוש הפיראט בעץ הפיראטים (לפי מזהה) - $O(\log n)$. אם הוא קיים בעץ:

- ניגש לספינה עליה הוא נמצא, $O(1)$, כי מחזיק מצביע אליה.
- נסיר אותו מעץ הפיראטים של הספינה עליה הוא נמצא, $O(\log n)$.
- נעדכן את ערך האוצר שלו - $O(1)$ כי הוא שמור כשדה של כל פיראט.
- נוסיף אותו חזרה לעץ הפיראטים של הספינה עליה הוא נמצא (על מנת לשמר את סדר המיון בעץ). סיבוכיות זמן $O(\log n)$.
- נעדכן את הפיראט העשיר ביותר בספינה, $O(\log n)$.

לכן סה"כ $O(\log n)$ כנדרש.

הפעולה get_treasure:

חיפוש הפיראט בעץ הפיראטים $O(\log n)$. אם הוא קיים בעץ נחזיר את ערך האוצר שיש עליו בנוסף למאזן הכלכלי של הספינה בה הוא נמצא (זאת לפי תכונת מבנה הנתונים)- $O(1)$ כי הוא שמור כשדה של כל פיראט, וגם המצביע לספינה עליה נמצא הפיראט שמור כשדה שלו. לספינה יש שדה של המאזן הכלכלי ולכן ניתן להחזירו בסיבוכיות זמן $O(1)$.

לכן סה"כ $O(\log n)$ כנדרש.

הפעולה get_cannons:

חיפוש הספינה בעץ הספינות $O(\log m)$. אם היא קיימת בעץ נחזיר את מספר התותחים שלה - $O(1)$ כי הוא שמור כשדה של כל ספינה.

לכן סה"כ $O(\log m)$ כנדרש.

הפעולה get_richest_pirate:

נחפש את הספינה בעץ הספינות $O(\log m)$.

אם הספינה קיימת ואינה ריקה נחזיר את הפיראט העשיר ביותר - $O(1)$ כי הספינה מחזיקה מצביע אליו.

לכן סה"כ $O(\log m)$ כנדרש.

הפעולה ships_battle:

נחפש את שתי הספינות בעץ הספינות, זמן כל חיפוש $O(\log m)$.

אם שתי הספינות קיימות נשווה בין כמות התותחים המאויישים שלהן. זמן ההשוואה הוא $O(1)$ כי לכל ספינה יש שדה של כמות התותחים, ולעץ הפיראטים שלה יש שדה ששומר את הגודל שלו כי כך בנינו את העץ הגנרי.

נעדכן את המאזן הכלכלי של שתי הספינות בהתאם למספר הפיראטים בכל ספינה - $O(1)$.

הסבר-

כל פיראט מהספינה המנצחת מקבל מטבע מכל אחד מהספינה המפסידה- במילים אחרות, המאזן הכלכלי של הספינה המנצחת עולה במספר הפיראטים שנמצאים בספינה המפסידה.

כל פיראט מהספינה המפסידה נותן מטבע לכל אחד מהספינה המנצחת- במילים אחרות, המאזן הכלכלי של הספינה המפסידה יורד במספר הפיראטים של הספינה המנצחת.

סה"כ: $O(\log m) + O(\log m) + O(1) + O(1) = O(\log m)$, כנדרש.