תיאור כללי של המבנה:

המבנה מכיל מבנים שונים, נסביר עליהם בקצרה: (מלבד דור להמחשה)

AVLTree<int,ship> tree_ships

המפתחות הם המספר המזהה של הספינות

המידע מוכל ship

AVLTree<int, pirate> tree_pirates

המפתחות הם המספר המזהה של ה pirate

המידע מוכל pirate.

נסביר על הclasses שהשתמשנו בהם:

class ship שמהווה את הספינה ומכיל את הנתונים הבאים:

m_shipId: המספר המזהה של הספינה.

m_cannons: מספר התותחים של הספינה.

richest_pirate: מחזיק את מצבה הנוכחי בעל מי המלבוזות הרבה ביותר בספינה.

m_bonusTreasure: כך גם המלבוזות ש pirate בספינה הרוויח/הפסיד בbattle (מגבר).

num_order: האינדקס של ה pirate באחרון שנכנס לספינה.

m_shippirates: עץ AVL שמכיל את כל pirates של הספינה ממוינים לפי id.

m_shipOrderedpirates: עץ AVL שמכיל את כל pirates של הספינה ממוינים לפי סדר הכניסה.

m_ship_richpirates: עץ AVL שמכיל את כל pirates של הספינה ממוינים לפי מספר המלבוזות שיש להם.

class pirate שמכיל את הנתונים הבאים:

m_pirate: המספר המזהה של ה pirate.

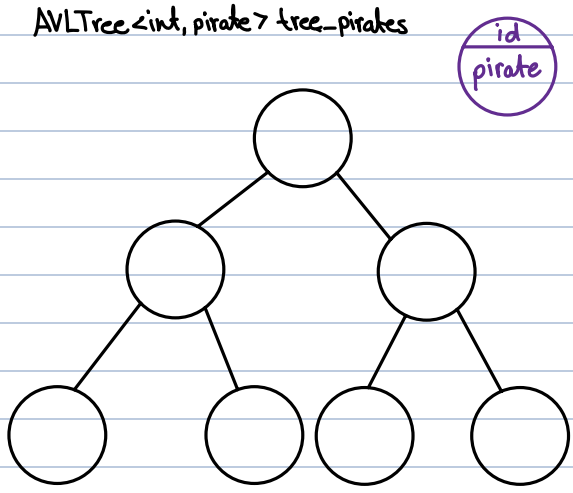
belong_shipId: המספר המזהה של הספינה שה pirate שייך אליה.

treasure: כמות המלבוזות של ה pirate.

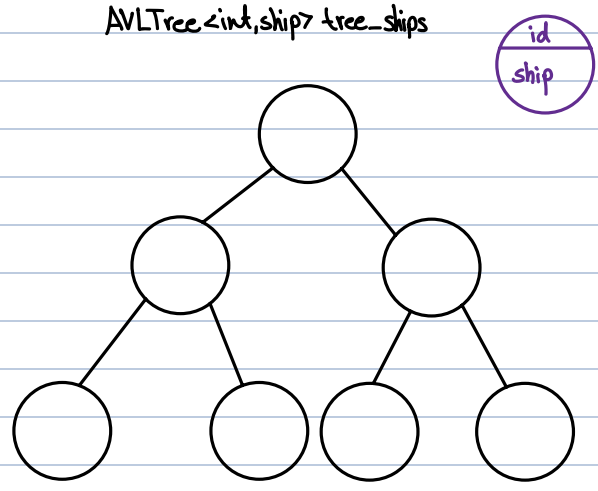
belongToShip: מצביע לספינה שה pirate שייך אליה.

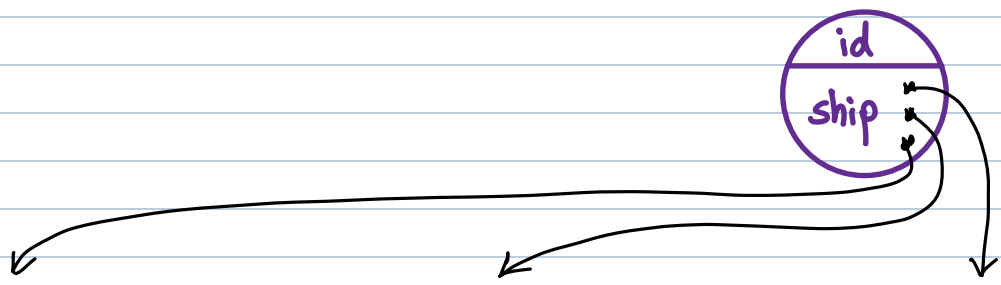
m_time: אינדקס ה pirate בספינה (לפי סדר הכניסה).

AVLTree<int, pirate> tree_pirates

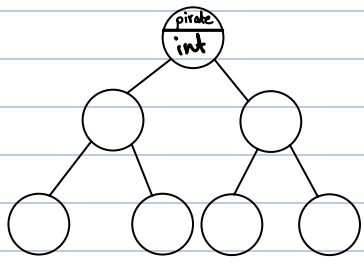


AVLTree<int,ship> tree_ships

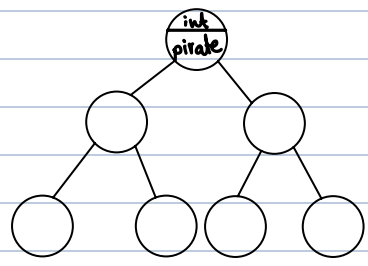




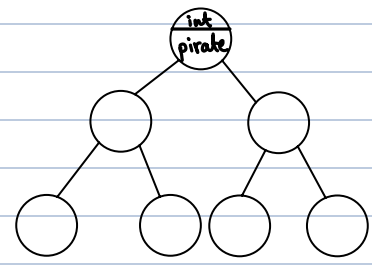
AVLTree<pirate,int> m_richPirates



AVLTree<int,pirate> m_orderedPirates



AVLTree<int,pirate> m_shipPirates



oceans_t()

מאתחלים מבנה חדש ריק באופן הבא
נאתחל עץ AVL ריק הממוין לפי מספר מכבה על ships ב O(1) ועץ AVL ממוין לפי מספר מכבה על הpirates
וגם מתבצע ב O(1) כפי שלמדנו בהרצאות.
סיבוכיות זמן: O(1) במקרה הגרוע.

virtual ~oceans_t()

משחרר את המבנים שיש במבנה שלנו ומשווה אותם לNULL.
עץ הships משתחרר ב O(m)
עץ ה pirates משתחרר ב O(n)
סיבוכיות זמן O(n+m) במקרה הגרוע.
משחררים את העצים ריקורסיבית בסדר inorder.

StatusType add_ship(int shipId, int cannons)

נבדוק את תקינות הקלט ונחכיר שגיאה במידת הצורך ב O(1).
נבדוק אם קיימת סכינה עם אותו מכבה על ידי חיפוש בעץ ships
החיפוש הזה מתבצע ב O(logm) במקרה גרוע כפי שלמדנו בהרצאה.
אחרת, נאתחל אובייקט חדש מסוג ship ונשים m_cannons מקבל cannons ונוסיף אותו לעץ ה ships.
אתחול האובייקט ב O(1) והוספתו לעץ ב O(logm) כאשר m הינו
מספר הסכינות במערכת.
סיבוכיות זמן: O(1)+O(logm)=O(logm) במקרה הגרוע, m הינו מס' הסכינות.

StatusType remove_ship(int shipId)

נבדוק את תקינות הקלט ונחכיר שגיאה במידת הצורך ב O(1).
נבדוק אם קיימת סכינה עם אותו מכבה על ידי חיפוש בעץ ships
החיפוש הזה מתבצע ב O(logm) במקרה גרוע כפי שלמדנו בהרצאה.
אחרת נמחק את ה node שבמכבה שלו הינו shipId מעץ הships ב O(logm)
סיבוכיות זמן: O(logm), כאשר m הינו מספר הסכינות.

StatusType add_pirate(int pirateId , int shipId, int treasure)

נבדוק את תקינות הקלט ונחכיר שנאלה מתאלמה בגודל הדורך ב $O(1)$.
נבדוק אם קיים ship עם מספר shipId בחיפוש בעץ ships ב $O(\log m)$
כך ש m היא מספר הספינות.
נבדוק אם קיים pirate עם מספר pirateId ב $O(\log(n))$ ע"י חיפוש בעץ ה pirates
כך ש n היא מספר ה pirates
אחרת, מאתחלים pirate חדש ב $O(1)$ ושבה treasure שבו להיות treasure (אבל נחסר את ה bonusTreasure בעכחי
מה treasure כי חישוב treasure ב get_treasure כולל הוספת הבונוס אך ה pirate החדש לא מושפע מכה.
ונאתחל belong to ship ל shipId.
אם כן נכניס את ה pirate לעץ של rich pirates, לעץ ordered pirates
ועץ ship pirates ב $O(3\log n)=O(\log n)$ כך ש n מס' ה pirates
סיבוכיות כמן: $O(\log n)+O(\log m)$, m מס' הספינות, n מס' ה pirates

StatusType remove_pirate(int pirateId)

נבדוק את תקינות הקלט ב $O(1)$
נבדוק אם קיים pirate עם מספר pirateId
על ידי חיפוש בעץ ב tree_pirates שכולל את ה pirates בערכת ובה מתבצע ב $O(\log(n))$
בשביל רה pirate קיים, היא מכיל שדה שהוא belongToShip מצביע לספינה שה pirate עליה.
ב ship קיימים 3 עצים שמכילים את ה pirate:
נמחק מ m_shipsPirates את הnode שמספרו pirateId ב $O(\log(n))$.
נמחק מ m_ship_orderedPirates את ה node שמפתחו m_time שה pirate ב $O(\log(n))$.
נמחק מ m_richPirates את ה node שמפתחו treasure שה pirate ב $O(\log(n))$.
ונמחק את pirate מ tree_pirates בסיבוכיות $O(\log(n))$
סיבוכיות כמן: $O(\log(n))$.

StatusType treason(int sourceShipId, int destShipId)

נבדוק את תקינות הקלט ב $O(1)$

נבדוק אם קיימות שתי ספינות בtree_ships שבמספר המזהה שלהן sourceShipId

ו destShipId על ידי חיפוש ב treeships (פעמיים) בסיבוכיות $O(2\log(m))$

אם קיימות שתי הספינות, נמצא את האדר בעל המזהה הכי קטן בעל m_shipOrderpirates

(שהיא האיבר השמאלי ביותר) שבספינת המקור, אחרי שנמצא את ה id של הpirate הכה

ומספר המטבעות שיש לpirate, נמחק אותו משלושת העצים של ספינת המקור

בסיבוכיות $O(3\log(n))=O(\log(n))$

ונכניס את אותו pirate לשלושת העצים בספינת היעד בסיבוכיות $O(3\log(n))=O(\log(n))$

סך הכל בסיבוכיות: $O(\log(m))+O(\log(n))=O(\log m+\log n)$

StatusType update_pirate_treasure(int pirateId , int change)

נבדוק את תקינות הקלט ב $O(1)$

נחפש את הpirate בעל המזהה pirateId בtree_pirates ב $O(\log(n))$

אם לא קיים נחזיר את הטיפול המתאים, אחרת, ונעדכן את treasure ב $O(1)$.

אחרי העדכון נתקן את מקום ה pirate בעל m_ship_richpirates

שם ship שה pirate שייך אליה (יש שדה מצביע לship בpirate) על ידי מחיקת

הpirate לפני העדכון והכנסתו אחר כך ב $O(2\log(n))=O(\log(n))$

סך הכל הסיבוכיות: $O(\log(n))$

output_t < int > get_treasure(int pirateId)

בדוק תקינות קלט ב-O(1).
נחפש ב-tree_pirates אם קיים pirate שבמבנה שלו הוא pirateId ב-O(log(n))
אם לא קיים נחזיר שגיאה בהתאם.
אם קיים, נחזיר את שדה ה-treasure שבpirate פרט תוספת ה-bonusTreasure של הסניף
שבו נמצא אליה, יש שדה ב-pirate מצביע לסניף כדי לכן כל זה מתבצע ב-O(1).
סיבוכיות כמן: סך הכל O(log(n)).

output_t < int > get_cannons(int shipId)

בדוק תקינות קלט ב-O(1).
נחפש ב-tree_ships אם קיים ship שבמבנה שלו הוא shipId ב-O(log(m))
אם לא קיים נחזיר שגיאה בהתאם.
אם קיים, נחזיר את שדה ה-cannons שב-ship ב-O(1).
סיבוכיות כמן: סך הכל O(log(m)).

output_t < int > get_richest_pirate(int shipId)

בדוק תקינות קלט ב-O(1).
נחפש ב-tree_ships אם קיים ship שבמבנה שלו הוא shipId ב-O(log(m))
אם לא קיים נחזיר שגיאה בהתאם.
אם קיים, נחזיר את השדה richest_pirate שב-ship
שמחזיק את המספר במבנה של ה-pirate שיש לו מספר
מטבעות מקסימלי.
סך הכל הסיבוכיות היא O(1)+O(log(m))=O(log(m)) (לפי הניפוש).

הסבר לשדרוג ותקינות Richest pirate:
בזמן הכנסת pirate חדש לסניף נבדוק השוואה בין מס' המטבעות
שלו לעל המקסימלי הנוכחי, אם הוא גדול מהנוכחי, נעדכן richest_pirate
כך שיחזיק את ה-id של ה-pirate החדש.
השוואה כזאת תתבצע ב-O(log(n)) בפונקציית add pirate.
לכן richest_pirate תמיד מעודכן להחזיק את id ה-pirate הכי עשיר.

StatusType ships_battle(int shipId1, int shipId2)

נבדוק תקינות קלט ב-O(1).

נבדוק אם קיימות שתי ספינות בtree_ships שהמספר המזהה שלהן shipId1

ו shipId2 על ידי חיפוש ב treeships (בגמ"ר) בסיבוכיות O(2log(m))

נבצע את ההשוואות ב-O(1).

נוסיף ל m_bonusTreasure בship ונחסיר ל m_bonusTreasure של ship2.

חישוב הש"ט של מס' המטבעות אצל ה pirates שבספינות מוסבר ב add_pirate

ו get_treasure

סך הכל סיבוכיות: O(log(m)) (חיפוש).

סיבוכיות מקום של מבנה הנתונים:

יש לנו עץ של pirates מכיל לכל היותר n, כל צומת מכיל pirate

תופס סיבוכיות O(1).

עץ הships מכיל לכל היותר m תאים מכיל ship.

כל צומת מכיל ship מכיל 3 מצביעים כל תא מכיל int ו pirates

מס' התאים ב 3 המצביעים האלה של כל ה ships הוא 3n

כיון שכל pirate שייך לכל היותר ל ship אחת בלבד.

סך הכל הסיבוכיות $O(n+m+3n)=O(4n+m)=O(m+n)$.