Wet HW 1 (dry part)

Graded

Group

טל כרמל

גיא פרידמן

View or edit group

Total Points

100 / 100 pts

Question 1

(no title)

100 / 100 pts

✔ **– 0 pts** Correct

Can Run Together function

- **– 15 pts** can_run_together function is missing
- **– 5 pts** There is no explanation of what happens if we get stuck in a circle
- **– 7 pts** can_run_together's space complexity is not as required (the implementation is wrong)
- **– 2 pts** You have not checked if there is only one horse that does not follow any other horse
- **– 4 pts** can_run_together's explanation is not detailed enough or you are using auxiliary functions without explaining them

---

Space complexity

- **– 10 pts** There is no analysis for space complexity
- **– 5 pts** Space complexity analysis is not detailed enough
- **– 7 pts** Space complexity analysis is not correct
- **– 2 pts** No analysis for the horses tree inside each Herd

---

Missing method

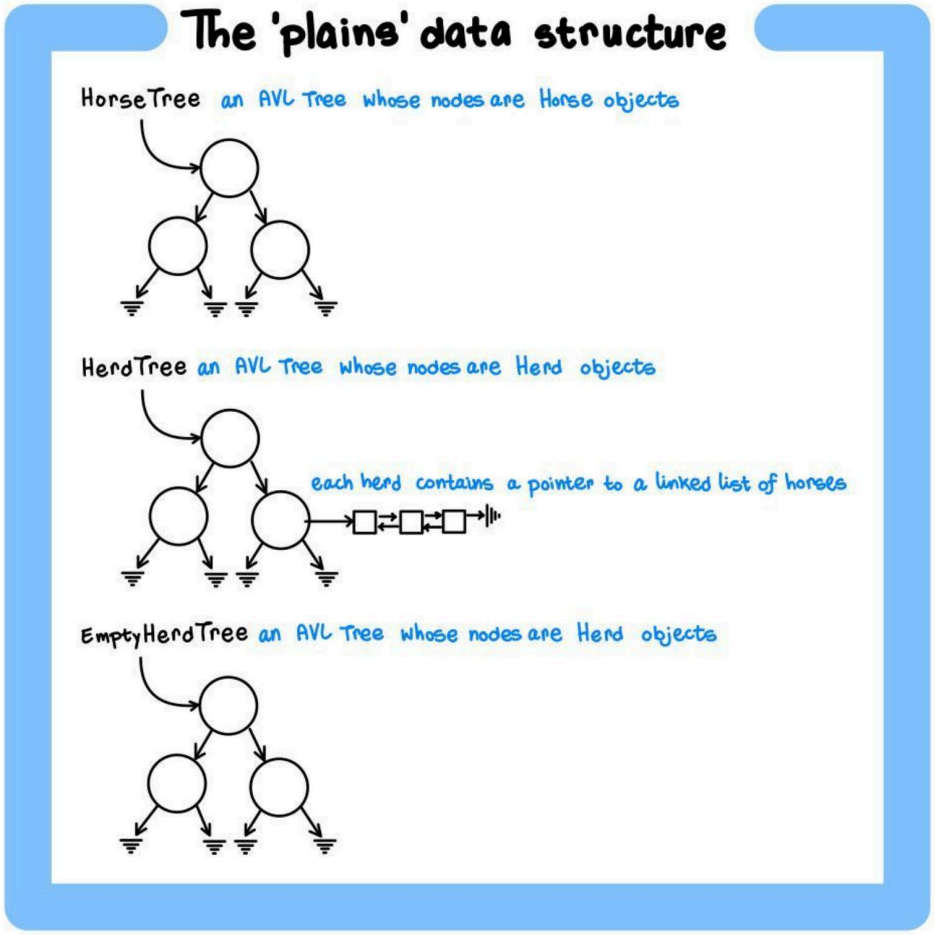
- **– 5 pts** Constructor is missing
- **– 5 pts** Destructor is missing
- **– 7.5 pts** add_herd is missing
- **– 7.5 pts** remove_herd is missing
- **– 7.5 pts** add_horse is missing
- **– 7.5 pts** join_herd is missing
- **– 10 pts** follow is missing
- **– 7.5 pts** leave_herd is missing
- **– 7.5 pts** get_speed is missing
- **– 10 pts** leads is missing

Question assigned to the following page: 1

תרגיל בית רטוב 1 – החלק היבש

מבנה הנתונים Plains מכיל:

1. HerdTree – עץ AVL בעל m צמתים המייצגים עדרים מלאים. עצם מסוג עדר (Herd) מכיל את מזהה העדר, כמות הסוסים הכוללת בעדר ומצביע לרשימה מקושרת של הסוסים שבעדר.
2. HorseTree – עץ AVL בעל n צמתים המייצגים את הסוסים במערכת. עצם מסוג סוס (Horse) מכיל את מזהה הסוס, מהירותו, מזהה העדר אליו שייך, מצביע לסוס אחריו הוא עוקב, שני משתנים מספריים ייחודיים ומשתנה בוליאני המאפשרים ניהול יחסי מעקב תקינים בין הסוסים.
3. EmptyHerdTree – עץ AVL בעל $m_\emptyset$ צמתים המייצגים עדרים ריקים.



No questions assigned to the following page.

מימושים לפונקציות:

1. plains_t()
   - נפעיל את הבנאים של עצי הAVL דרך רשימת אתחול, ללא כל פרמרטרים
   - הבנאי של עץ AVL שלא מקבל פרמטרים מאתחל את המצביע שלו ל-nullptr

   הפעולות שמבוצעות בבנאי אינן תלויות באף קלט, כלומר חסומות ע"י קבוע ולכן הסיבוכיות היא $O(1)$

2. ~plains_t()
   - הורס דיפולטיבי, שקורא להורסים של השדות (עצי AVL)
   - ההורס של עץ AVL קורא להורס של חוליית הראש בעץ
   - ההורס של חוליה בעץ הוא דיפולטיבי, ומפעיל את ההורסים של הערך השמור בחוליה (סוס או עדר), ואת ההורס עבור החוליות הבנות שלה.
   - ההורס של עצם הסוס מאתחל את המצביעים שלו בnullptr (אין לסוס שדות שהוקצו דינמית). ההורס של עצם העדר הוא דיפולטיבי, ומפעיל את ההורס של הרשימה המקושרת שהוא מכיל.
   - ההורס של הרשימה המקושרת הוא דיפולטיבי, ומפעיל את ההורסים של החוליות בו.
   - הורס החוליות של הרשימה המקושרת מאתחל את המצביעים השמורים בו לערך ולחוליה הוקדמת כ-nullptr, ומפעיל את ההורס של החוליה הבאה.

   סיבוכיות הפעולה:
   בהריסת עצי הAVL נדרשת הריסת כל החוליות בעץ. בעץ הסוסים יש n חוליות, בעץ העדרים המלאים יש m חוליות ובעץ העדרים הריקים יש $m_\emptyset$ חוליות. סה"כ הסיבוכיות היא $O(n+m+m_\emptyset)$. חשוב לציין: על אף שמחיקת עצם מסוג עדר מפעילה הורס של רשימה מקושרת שבמקרה הגרוע ביותר הסיבוכיות של היא $O(n)$, לא ייתכן שסיבוכיות מחיקת עץ העדרים הוא $O(nm)$ מכיוון שלא ייתכן שאותו הסוס יהיו ביותר מעדר אחד. לכן, לכל היותר מחיקת הרשימה המקושרת תוסיף n לסיבוכיות.

3. add_herd(int herdId)
   - נחפש האם המזהה קיים כבר בעץ HerdTree
   - נחפש האם המזהה קיים כבר בעץ EmptyHerdTree
   - אם המזהה לא נמצא באף עץ, נוסיף את העדר כצומת לעץ EmptyHerdTree

   ראינו כי סיבוכיות החיפוש וההכנסה לעץ AVL היא לינארית בגובה העץ, כלומר $O(\log m + \log m_\emptyset)$ לפי גובה העץ HerdTree וגובה העץ EmptyHerdTree.

4. remove_herd(int herdId)
   - נחפש האם המזהה קיים בעץ EmptyHerdTree
   - אם כן, נמחק הצומת המתאים מהעץ
   - במידת הצורך נאזן את העץ מחדש

   סיבוכיות חיפוש והוצאה, סיבוכיות מציאת צמתים בהם מופר האיזון וסיבוכיות ביצוע גלגולים (לכל היותר אחד ברמה) כולן לינאריות בגובה העץ. לכן $O(\log m_\emptyset)$

5. add_horse(int horseId, int speed)
   - נחפש האם המזהה קיים כבר בעץ HorseTree
   - אם לא, נוסיף אותו כצומת לעץ
   - במידת הצורך נאזן את העץ מחדש

No questions assigned to the following page.

סיבוכיות חיפוש והכנסה, סיבוכיות מציאת צמתים בהם מופר האיזון וסיבוכיות ביצוע גלגולים (לכל היותר אחד ברמה) כולן לינאריות בגובה העץ. לכן $O(\log n)$

6. join_herd(int horseId, int herdId)
- נחפש את מזהה הסוס בעץ HorseTree $O(\log n)$ ונוודא שאינו בעדר כלשהו.
- אם נמצא שם, נחפש את המזהה של העדר בעץ EmptyHerdTree $O(\log m_{\emptyset})$
- אם אכן קיים, נשייך את הסוס לעדר המתאים $O(1)$ (הוספה לתחילת רשימה מקושרת).
- נוציא את העדר מהעץ EmptyHerdTree $O(\log m_{\emptyset})$
- ונכניס אותו לעץ HerdTree $O(\log m)$
- אם המזהה של העדר לא נמצא ב- EmptyHerdTree, נחפש אותו ב- HerdTree $O(\log m)$
- אם אכן קיים, נשייך את הסוס לעדר המתאים $O(1)$

פעולות חיפוש, הוצאה וכנסה לינאריות בגובה העץ, לכן לפי הפירוט לעיל, נקבל כי סה"כ הסיבוכיות היא $O(\log m + \log n + \log m_{\emptyset})$

7. follow(int horseId, int horseToFollowId)
- נחפש את מזהי הסוסים בעץ HorseTree
- אם אכן קיימים ומשויכים לאותו העדר, נגדיר את המעקב בין הסוסים על ידי עדכון השדות המיועדים.

סיבוכיות חיפוש לינארית בגובה העץ ולכן $O(\log n)$

8. leave_herd(int horseId)
- נחפש את מזהה הסוס בעץ HorseTree
- אם הסוס משוייך לעדר כלשהו, נעדכן את השדות שלו (את העדר ואת הסוס אחריו הוא עוקב ל- null)
- נסיר את החולייה של הסוס מרשימת הסוסים השייכים לעדר המתאים.
- הסרת החולייה מהרשימה כוללת את עדכון כל המצביעים כדי לשמור על רציפות הרשימה, ולבסוך הפעלת ההורס של החולייה שנמחקה (מכיוון שנותקה מהרשימה מחיקתה למעשה רק מאתחלת את המצביע לערך בה ל- nullptr)

סיבוכיות הפעולה מורכבת מסיבוכיות חיפוש הסוס בעץ AVL, ויתר הפעולות חסומות על ידי קבוע. סך הכל סיבוכיות $O(\log n)$.

9. get_speed(int horseId)
- נחפש את מזהה הסוס בעץ HorseTree ונחזיר את מהירותו

סיבוכיות חיפוש לינארית בגובה העץ ולכן $O(\log n)$

10. leads(int horseId, int otherHorseId)
- נחפש את מזהי הסוסים בעץ HorseTree
- נעבור על רשימת הסוסים של העדר אליו שייכים שני הסוסים, נזהה ונאחסן מצביעים לסוס העוקב ולסוס המוביל על ידי בדיקות מיועדות.

סיבוכיות הסיבוכיות מתחשבת בחיפוש הסוסים בעץ, כמו גם מעבר על רשימת הסוסים שבעדר, ולכן הסיבוכיות הכוללת היא: $O(\log\ n\ + n_{herdId})$

11. can_run_together(int herdId)
- נחפש את מזהה העדר בעץ העדרים הלא ריקים $O(\log m)$

No questions assigned to the following page.

- נעבור על רשימת חברי העדר ונסכום את כמות הסוסים שאינם עוקבים אחר אף סוס (בעלי פוטנציאל להוביל את העדר). אם הערך שונה מ-1 נחזיר failure. $O(n_{herdId})$
- נעבור פעם נוספת על רשימת חברי העדר ונבדוק הצבעה מעגלית[1]. לכל סוס נסמן האם הגענו אליו או לא ובכל בדיקת מעגליות לא נצטרך לבדוק שוב סוסים שכבר סומנו. $O(n_{herdId})$.

סה"כ נקבל סיבוכיות $O(\log m + n_{herdId})$

<u>סיבוכיות מקום:</u>

הזיכרון הנדרש עבור מבנה הנתונים ומימוש הפעולות לפי תכנון זה הוא כמפורט להלן:

- עץ כל הסוסים דורש n מקום כדי לאחסן את כל הסוסים.
- עץ העדרים הלא ריקים דורש m מקום כדי לאחסן את כל העדרים הפעילים.
- עץ העצים הריקים דורש $m_{\emptyset}$ מקום כדי לאחסן את כל העדרים הלא פעילים.
- כל עדר מכיל מצביע לרשימה מקושרת המכילה את חברי העדר. סכום כל החוליות בכל הרשימות המקושרות הוא n (כמספר הסוסים במערכת).

לא נדרשת הקצאה נוספת של זיכרון במהלך המימוש שלנו, לכן סך הכל סיבוכיות המקום עבור מבנה נתונים זה היא: $O(n + m + m_{\emptyset})$ כנדרש.

---

[1] הפונקציה לבדיקת הפניה מעגלית- ׳שיטת שני הפויינטרים׳: כאשר אנו נמצאים במבנה נתונים מסוג גרף, ניעזר בשני פויינטרים על מנת לדעת אם אנחנו נמצאים בהפניה מעגלית או לא-
בקריאה הראשונית נגדיר ׳פויינטר 1׳ ו- ׳פויינטר 2׳ שיצביעו על החוליה הראשונה, ובכל קריאה לחוליה הבאה נקדם את ׳פויינטר 1׳ חוליה אחת (שיצביע על החולייה הבאה אליה נקרא), ונקדם את ׳פויינטר 2׳ 2 חוליות.
כאשר ׳פויינטר 2׳ יגיע לסוף המבנה נדע כי אנחנו לא נמצאים בהפניה מעגלית.
במידה ואנחנו כן נמצאים בהפניה מעגלית, נדע זאת כאשר ׳פויינטר 1׳ ו- ׳פויינטר 2׳ יצביעו על אותו הערך\ חוליה.
סיבוכיות הפעולה: זוהי פונקציה רקורסיבית שכל הפעולות בה הן O(1). עומק הרקורסיה הוא המינימלי מבין אורך שרשרת המעקב ופרמטר ההתקדמות. לכן סיבוכיות הפעולה היא המינמלי מבין n (כמות הסוסים המקסימלית שיכולה להיות ברשימת מעקב) לבין פרמטר ההתקדמות, שבהתחלה שווה לכמות הסוסים שבעדר (לכל היותר n). נקבל שהסיבוכיות היא O(n).