מבני נתונים 234218 - גיליון רטוב מספר 1
חלק יבש

אמיר פוסטילניק 316397843
יהונתן יוסף 203304480

9 בדצמבר 2019

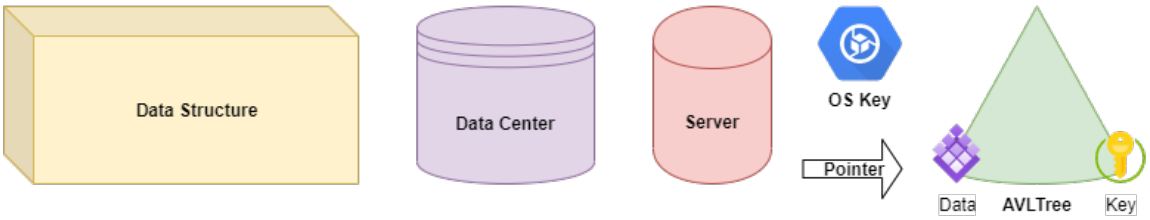
1 מבט על מבני הנתונים

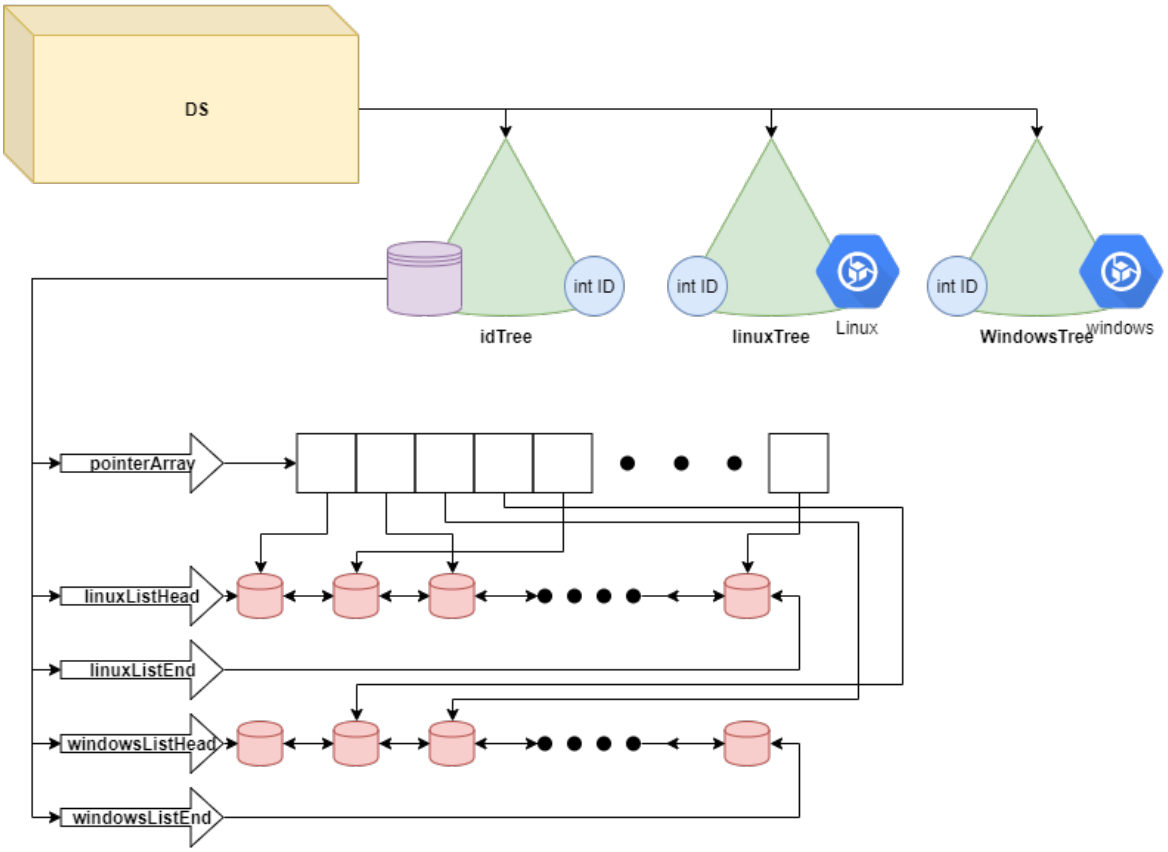
מבני הנתונים שלנו כולל (ראה דיאגרמה בעמוד הבא):

1. Data Structure
2. AVL Tree
3. AVL Node
4. Data Center
5. Server
6. OS Key
7. Data Stracture Exceptions

נסביר על כל אחת מהמחלקות בעמודים הבאים

Index:


Data Structure
Data Center
Server
OS Key
Pointer
Data
AVLTree
Key


DS
int ID
idTree
int ID
linuxTree
Linux
int ID
WindowsTree
windows
pointerArray
linuxListHead
linuxListEnd
windowsListHead
windowsListEnd

2 המחלקות ומבנה הקוד

1. $DataStracture$:
מחלקה האחראית על ניהול המידע וסנכרון הפעולות. כל פעולות הספרייה המסופקת ($library1$) יעברו דרך מחלקה זו והיא תנהל את מבני הנתונים ותמיר חריגות בהחזרת ערכי שגיאה ($Exceptions- > statusType$). המחלקה מכילה שלושה עצי AVL המכילים מפתח, ומידע. העץ הראשון, שומר $DataCenter$ וממיין אותם לפי ה־ID. זהו העץ הראשי ורב הפעולות יעברו דרכו.
שני העצים האחרים שומרים ID וממויינים לפי $OSKey$. עצים אלו משמשים לשמירה על סדר לפי מערכות ההפעלה השונות.

2. $AVLTree$:
מבנה נתונים הנלמד בהרצאות. העץ ממומש ע"י Key אשר מסופק בעת הכנסת איבר ($Data$) לעץ. העץ אחראי על יצירת העתקים וניהול הזיכרון. העץ ממומש בעזרת $AVLNode$. מטעמי נוחות, השורש של העץ הוא איבר דמה - והשורש האמיתי של להעץ הוא הבן השמאלי של איבר הדמה.

3. $AVLNode$:
חוליית עזר המשמשת לבנייה ותפעול עץ AVL. החולייה מכילה פוייטרים לילדים ולאבא שלה, גובה ו־BF, ושומרת את המפתח ואת ה$Data$.

4. $DataCenter$:
מבנה נתונים אשר שומר ומנהל את הנתונים לחווה. המבנה כולל פויינטרים לראש ולזנב של שתי רשימת שרתים, ומערך מצביעים לשרתים.

5. $Server$:
מחלקת קצה המייצגת שרת ומשמשת כחולייה ברשימת שרתים. המחלקה מכילה שדות לשמירת המידע (מערכת ההפעלה, id וזמינות) בנוסף למצביע אחד לשרת הקודם מצביע לשרת הבא.

6. $OSKey$:
מחלקת עזר אשר נשלחת כמפתח לעצי AVL הממויינים לפי מערכת ההפעלה. המחלקה מחזיקה את מספר השרתים אשר מותקנת בהם מערכת ההפעלה המתאימה (למיון ראשוני) ו־id (למיון משני). בנוסף כמפתח לעץ AVL קיימים לה אופרטורים הכרחיים כגון $<, ==$.

7. $DataStractureExceptions$:
מחלקת עזר אבסטרקטית לניהול חריגות. הפונקציה האבסטרקטית היא פונקציה המחזירה $StatusType$ מתאים (בהתאם למחלקה הדורסת).

3 המטודות וחישובי הסיבוכיות

פונקציית $Init()$

נאתחל 3 עצי AVL ריקים בסיבוכיות זמן של $3 \cdot O(1)$ כפי שהוכח בתרגול.

פונקציית $AddDataCenter()$

- נחפש בעץ ה־AVL הממוין לפי ה־ ID האם ה־ $DataCenter$ קיים. סיבוכיות זמן של $log(n)$ כפי שנלמד בתרגול.
- במידה וה־ $DataCenter$ קיים נזרוק חריגה מתאימה.
- ניצור את ה־ $DataCenter$:
  - נאתחל 2 רשימות שרתים, אחת ריקה ואחת מאורך m. כך שהרשימות ייצגו את השרתים הפנויים מכל מערכת הפעלה.
    הרשימה הריקה תשמש את מערכת ההפעלה של הווינדוס מכיוון שלא קיימים שרתים עם מערכת הפעלה זו.
    הרשימה מאורך m תייצג את השרתים הפנויים בעלי מערכת ההפעלה של לינוקס אשר מסודרים תחילה בסדר עולה.
  - איתחול שרת מתבצע ב־ $O(1)$ ואיתחול הרשימות מתבצע בסיבוכיות זמן של $O(m) + O(1)$ וסיבוכיות מקום של $O(m)$.
  - נאתחל מערך מאורך m כך שיצביע על כל אחד מהשרתים בסיבוכיות זמן ומקום של $O(m)$.
  - סה"כ קיבלנו כי יצירת $DataCenter$ מתבצע בסיבוכיות זמן ומקום של $O(m)$.
- ניצור 2 $OSKey$ בסיבוכיות זמן ומקום של $O(1)$.
- לבסוף נכניס בעזרת פעולת $insert$ של עצי ה־ AVL שהיא מתבצעת בסיבוכיות זמן של $3 \cdot log(n)$ כפי שהוראה בתרגול, במקרה של הכנסה בעייתית נבצע "גלגולים" המתבצעים ב $O(1)$.
- סה"כ קיבלנו כי סיבוכיות הזמן של פונקציה זו היא : $O(log(n) + m)$, סיבוכיות המקום הנוסף שהוספנו היא $O(m)$ כנדרש.

פונקציית $RemoveDataCenter()$

- נחפש בעץ ה־AVL האם ה־ $DataCenter$ קיים בסיבוכיות זמן של $log(n)$ כפי שנלמד בתרגול.
- במידה וה־ $DataCenter$ לא קיים נזרוק חריגה מתאימה.
- אחרת ניצור 2 $OSKey$ בסיבוכיות זמן ומקום של $O(1)$.
- נוציא את ה־ $DataCenter$ המבוקש משלושת העצים כפי שהוראה בתרגול $3 \cdot O(log(n))$ במקרה שההוצאה מפרה את הסדר של העץ נבצע "גלגולים" ב־ $O(1) \cdot O(log(n))$.

- לאחר הוצאת ה־ $DataCenter$ מהעץ נהרוס את 2 הרשימות ב $O(m)$ ואת המערך ב־ $O(1)$.
- סה"כ קיבלנו כי סיבוכיות הזמן של הפונקציה הזו היא $O(log(n)+m)$ כנדרש.

פנונקציית $RequestServer()$

- נחפש בעץ ה־AVL האם ה־ $DataCenter$ קיים בסיבוכיות זמן של $log(n)$ כפי שנלמד בתרגול.
- במידה וה־ $DataCenter$ לא קיים נזרוק חריגה מתאימה.
- אחרת ניצור 2 $OSKey$ בסיבוכיות זמן ומקום של $O(1)$.
- ניגש דרך רשימת הפויינטרים לשרת הרצוי $O(1)$:
  - במקרה והוא פנוי נסמן אותו כתפוס (נשנה את מערכת ההפעלה במידת הצורך) ונוציא אותו מהרשימה אליה הוא שייך $O(1)$.
  - במקרה והוא תפוס ניגש אל תחילת הרשימה $O(1)$ לפי סדר העדיפויות ברשימה ו"נתפוס" שרת (במידת הצורך נחליף מערכת הפעלה) $O(1)$.
- במקרה שמערכת ההפעלה שונתה , עצי ה־ AVL יעודכנו בהתאם על ידי הוצאה והכנסה של ה־ $Data$ בעצי ה־ AVL המתאימים. סיבוכיות זמן $4 \cdot O(log(n))$
- סה"כ פעולה זו דורשת סיבוכיות זמן של $O(log(n))$ כנדרש.

פונקציית $FreeServer()$

- נחפש בעץ ה־AVL האם ה־ $DataCenter$ קיים בסיבוכיות זמן של $log(n)$ כפי שנלמד בתרגול.
- במידה וה־ $DataCenter$ לא קיים נזרוק חריגה מתאימה.
- אחרת ניגש אל ה־ $DataCenter$ הרצוי,
  - דרך מערך הפויינטרים ניגש אל השרת המבוקש $O(1)$ (אם לא קיים תיזרק חריגה) נשחרר אותו ונשים אותו בסוף הרשימה המתאימה למערכת ההפעלה שלו $O(1)$.
- סה"כ פעולה זו דורשת סיבוכיות זמן של $O(log(n))$ כנדרש.

פונקציית $GetDataCentersByOS()$

- נקצה מערך בגודל מספר החוות הקיימות פעולה זו מתבצעת בסיבוכיות זמן של $O(1)$ ומקום של $O(n)$.
- נעבור על עצי ה־ AVL של מערכות ההפעלה בסיור $inorder$ ונכניס לתוך המערך המוקצה את המזהים הדרושים. היות והעצים ממויינים כנדרש בפונקציה , סיור זה מתבצע ב־ $O(n)$.
- סה"כ פעולה זו מתבצעת בסיבוכיות זמן של $O(n)$ ומקום של $O(n)$ כנדרש.

פונקציית $Quit()$

- הפעולה הבאה מתבצעת על שלושת העצים באופן זהה:
  - נעבור בסיור $postoreder$ על עץ ה־AVL ונמחק את הצמתים בעץ. סיבוכיות הסיור היא $O(n)$.
- עץ ה־ID מוחק גם את החוות וכפי שהראינו לעיל מחיקה של חווה מתבצעת בסיבוכיות של $O(m)$.
- לכן סה"כ פעולה זו מתבצעת בסיבוכיות זמן של $O(m+n)$.