# עבודה מסכמת מבני נתונים

Init: אתחול <<map<int, map<int, Tree*>>, המפה הראשונה מכילה ערך i ומפה שמכילה ערך j ופויינטר לעץ, כאמור ברגע האתחול המפות ריקות משמע שהאתחול ב(O(1.

PlantTree: הכנסה למפה של אינדקס i תתבצע ב((O(log(n של העצים בשורה i, והכנסה למפה של אינדקס j תתבצע ב((O(log(n של העצים בטור j, כלומר סהכ הכנסה של עץ חדש (tree) לmap (הממומשת ע"י עץ AVL כדי לשמור על התכונה של חיפוש והכנסה ב((O(Log(n) במיקום i,j תתבצע ב((O(Log(n כאשר n הוא מספר העצים בפרדס.

AddFruit: fruit הוא אובייקט המאוחסן כפויינטר ונמצא ברשימה מקושרת בתוך כל אובייקט tree הממויינת ע"פ מידת ripeness וע"פ ID כמיון משני, הכנסה של פרי חדש תתבצע ב(O(Log(n למציאת העץ הרלוונטי ועוד ב(O(k כאשר k הוא מספר הפירות על העץ, בנוסף אנו מחזיקים container מסוג unordered map (hash table) בשם m_fruitIdToCoor עם מפתח של fruitID, וערך מיקום העץ (i,j כpair, מה שיהפוך את i,j לאובייקט אחד) הכנסה של מפתח חדש תתבצע בממוצע ב(O(1 **וב(O(k במקרה הגרוע כאשר K הוא כל הפירות בפרדס (סבירות נמוכה שהסיבוכיות תעמוד על O(k) בהתחשב בכך שכל הערכים המוכנסים (fruit id) בהכרח שונים אחד מהשני). ניצור פויינטר fruit חדש ונכניס אותו לרשימה בעץ המתאים ב(O(k (k מספר הפירות על העץ) באופן ממוין ונכניס לU_map את המפתח ID של הfruit עם ערך המיקום של העץ (i,j).

PickFruit: ניגש לm_fruitIdToCoor עם הID של הfruit ונשלוף את המיקום של העץ (i,j) בסיבוכיות קבועה בממוצע (O(1**. עם (i,j) של העץ המתאים ניגש לעץ דרך pardes ב((O(Log(n, מתוך האובייקט tree ניגש לרשימה המקושרת של הfruit ונמחק אותו מהרשימה ב(O(k, (k מספר הפירות על העץ). בנוסף נמחק את הID של הפרי מm_fruitIdToCoor.

RateFruit: ניגש לm_fruitIdToCoor עם הID של הfruit ונשלוף את המיקום של העץ (i,j) בסיבוכיות קבועה (O(1**. עם (i,j) של העץ המתאים ניגש לעץ דרך pardes ב((O(Log(n, מתוך האובייקט tree ניגש לרשימה המקושרת של הfruit ונעדכן את השדה המתאים בfruit ע"פ הID בסיבוכיות של O(k), לאחר מכן נמחק את הפרי הרלוונטי ונכניס אותו שוב ב(O(k, כאמור הרשימה כתובה כך שכל ערך חדש יתמיין בהכנסה, **(k = מספר הפירות על העץ)**

GetBestFruit: עם (i,j) של העץ המתאים ניגש לעץ דרך הm_coorToTree ב(O(1 או דרך pardes ב((O(Log(n, ונשנה את הערך של הפוינטר שקיבלנו לערך הID של האיבר הראשון ברשימה של הfruits.

GetAllFruitsByRate: ע"פ ה(i,j) נמצא את המיקום של העץ דרך pardes ב((O(Log(n, לאחר מכן ניצור מערך בגודל K (פירות על העץ) של tree עם malloc, לאחר מכן נעבור על כל צומת ברשימה המקושרת של הfruits ונעתיק את הID בלבד לתוך כל תא במערך באופן סדרתי ב(O(k, ונעדכן את המצביעים הרלוונטים למערך ולמספר האיברים שבו.

UpdateRottenFruits: נעבור על כל העצים בPardes ב(O(n כאשר בכל עץ ניגש לרשימת הפירות שלו. תחילה ניצור שתי רשימות נוספות ריקות של פירות רקובים ופירות טובים. נעבור על כל הפירות ברשימה מההתחלה לסוף כאשר עבור כל פרי שנעבור נבצע את הבדיקה האם הוא רקוב או לא, בהתאם התוצאה נעביר אותו לרשימה הרלוונטית דרך PushBack. כאמור הרשימה המקורית כבר ממוינת ולכן כל הכפלה של ערכי הפירות בערך קבוע והכנסתם לרשימה חדשה ישמור על המיון שלהם. לאחר שעברנו את כל הרשימה המקורית והעברנו את הפירות לרשימות הרלוונטיות להם נמחק את הרשימה המקורית. לאחר מכן נעבור על שתי הרשימות שנוצרו ונשווה את האיברים שלהם אחד מול השני,

במידה ואחד קטן מהשני אז הוא ייכנס לטבלה המקורית בPushBack, כאמור כל הזמן הרשימות שומרות על המיון שלהם ע"י סדר ההכנסה, לאחר סיום המעבר על שתי הרשימות (במידה ואחת הסתיימה לפני השנייה נעביר את כל הפירות הנותרים מהרשימה הנוספת) והכנסה לרשימה המקורית נקבל את הרשימה המקורית ממויינת אחרי עדכון כל הפירות, וזאת בסיבוכיות קבועה של מספר הפירות על העץ, O(k).

Quit: נעבור על כל העצים בPardes ב(O(n ובכל צומת ניגש לעץ הרלוונטי, בעץ הרלוונטי נקרא לD'tor של העץ שבתורו יעבור על כל רשימת הפירות שנמצאים עליו ויקרא לD'tor של כל פרי עליו הוא עובר ב(O(k, כאשר בסוף נמחק את העץ עצמו ונעבור לעץ הבא ברשימה, לבסוף נמחק את הפרדס עצמו.