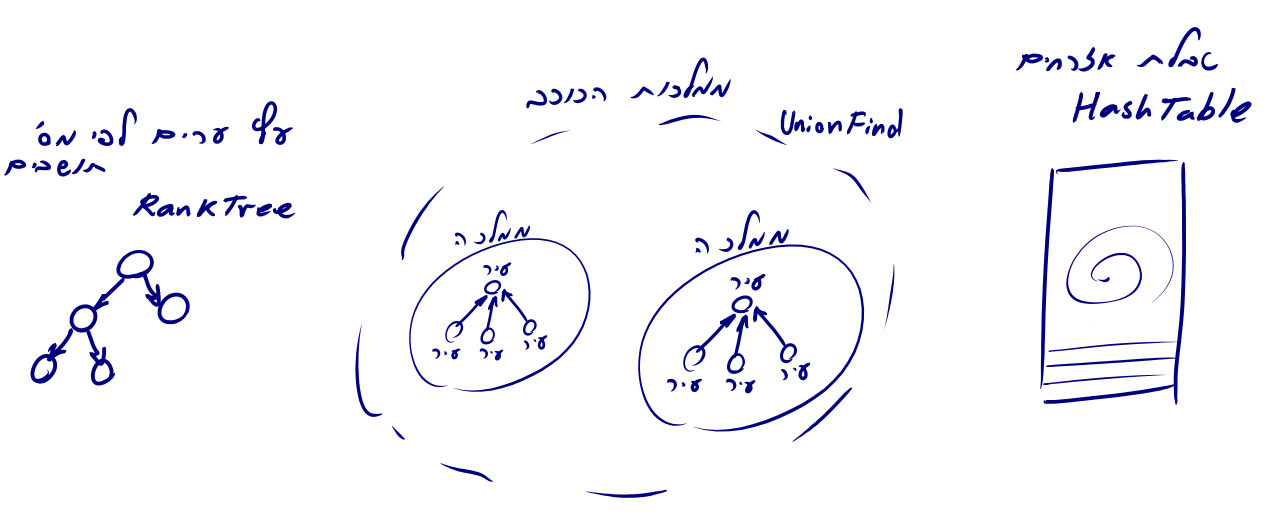
**מה מכיל:**

****

**CitizensTable** – HashTable – טבלה דינאמית, linear probing, אפשר רק להכניס אין מחיקה.

מאחסן את כל התושבים הרשומים על הכוכב כאשר המפתח הוא מספר תושב והמידע הוא מספר העיר לה הוא שייך או 0 (מספרי הערים ממסופרים 1 עדN )

**PlanetUnion** – UnionFind מבוסס עצים הפוכים, עם איחוד לפי גודל וכיווץ מסלולים.

כאשר הקבוצות הן הממלכות, בכל ממלכה רשומה עיר הבירה שלה. האיברים הם הערים מטיפוס City.

**CitiesTree** - עץ דרגות מבוסס AVL

כל הערים על הכוכב, ממוניות לפי מספר תושבים. בעץ יופיעו רק ערים עם מספר תושבים שונה מ-0.

City – מבנה פשוט שמייצג עיר ומכיל את מספרה ואת מספר התושבים בה.

**מימוש הפונקציות והסבר סיבוכיות:**

**void\* Init(int n)**

1. צור מבנה PlanetUnion עבור N ממלכות ובכל ממלכה תהיה עיר אחת עם 0 תושבים.
2. צור CitizenTable בתור רשימת תושבים ריקה.
3. צור CitysTree ריק.

**StatusType AddCitizen(void\* DS, int citizenID)**

1. חפש את המפתח עבור מספר התושב המבוקש O(1) בממוצע על הקלט
2. אם נמצא:
   1. החזר שגיאה (תושב כבר קיים במערכת)
3. אחרת:
   1. הוסף תושב חדש כאשר מספר העיר אילה הוא שייך הוא 0. O(1) בממוצע על הקלט, משוערך

הסבר סיבוכיות: טבלת הערבול דינאמית, הוכחת הסיבוכיות כמו מחסנית דינאמית שראינו בתרגול רק שההכנסה ב- O(1)ממוצע.

עבור הכנסה בלי הגדלה = O(1) ממוצע. עבור הכנסה עם הגדלה: מקצים מערך חדש עם אתחול יעיל O(1), מבצעים rehash לכל איבר בטבלה = Nפעמים O(1) ממוצע. מכיוון שהטבלה גדלה פי 2 בכל פעם, מובטח שנבצע N/2 פעולות הכנסה זולות לפני שצטרך להגדיל. לכן אם נפרוס את העלויות נקבל- 3\*O(1) בממוצע, משוערך = O(1) בממוצע, משוערך.

**StatusType MoveToCity(void\* DS, int citizenID, int city)**

1. חפש את התושב המבוקש ב CitizenTable.
2. אם התושב קיים או כבר משיוך לעיר
   1. החזר שגיאה.
3. עדכן את העיר אליה הוא משיוך
4. עדכן את מספר התושבים ב CitysTree O(logn)
5. עדכן את המפר התושבים ב PlanetUnion ועדכן את עץ הערים עבור הממלכה המדוברת כך שבסוף בראש העץ נקבל את העיר עם המספר המקסימלי של תושבים (תיקון ערמה) O(log\*n)

**StatusType JoinKingdoms(void\* DS, int city1, int city2)**

1. ערי בירה מיוצגות כעיר שהיא שורש בעץ ההפוך של ממלכה מסוימת במבנה ה UnionFind. בנוסף השורש יצביע על מבנה הGroup והGroup יחזיק מצביע לשורש העץ.
2. אם הערים לא עיר בירה (לא שורש) O(1)
   1. החזר שגיאה
3. אחרת
   1. בצע JOIN בין הממלכות לפי הממלכה עם הכי הרבה ערים O(1)
   2. אם הממלכה הקטנה יותר מכילה עיר בירה עם יותר תושבים (היא העיר בירה החדשה)
      1. עדכן שהיא תהיה שורש העץ והשורש הקודם יהפוך להיות בן של השורש החדש.

**StatusType GetCapital(void\* DS, int citizenID, int\* capital)**

1. חפש את התושב ב CitizenTable O(1)
2. אם התושב לא קיים או לא ממופה לשום עיר:
   1. החזר שגיאה
3. אחרת
   1. גש לעיר עם האינדקס המתאים לפי רשומת התושב במבנה PlanetUnion
   2. עלה לשורש עץ הממלכה וזו תהיה עיר הבירה O(log\*n)

**StatusType SelectCity(void\* DS, int k, int\* city)**

1. בצע פעולת Rank על עץ דרגות מאוזן CitysTree O(logn)

**StatusType GetCitiesBySize(void\* DS, int cities[])**

1. בצע מעבר Inorder על העץ CitysTree