רז אגו 208139287
נאור סרף 305715781

JerryBoree

בעבודה השתמשנו בשני מבני נתונים עיקריים:

1. **HashTable JerryHash:**

במבנה זה השתמשנו בפונקציית גיבוב שעובדת על פי Jerry ID הפונקציה סוכמת את ערכי האסקי של התווים שמופיעם בשם של הג"רי שאותו אנו מוספים למבנה,
ואחר כך עושה: k = sum % hashnumber.

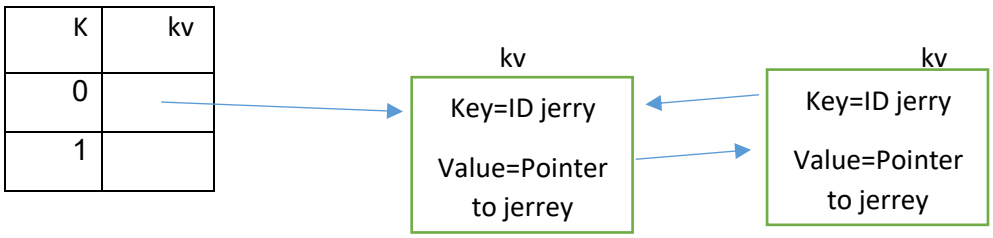
hashnumber קבענו כי יהיה מספר הג"ריים ההתחלתיים כפול 3, בהנחה כי במבנה הנתונים במהלך התוכנית יתבצעו מחיקות והוספה של ג"ריים. בחרנו להכפיל ב-3 את מספר הג"ריים ההתחלתיים כמספר קרוב אפשרי של מספרם בתוך מבנה הנתונים במהלך התוכנית, לא נבחר מספר גדול מדי בשביל לשמור על זיכרון.

הגורמים הנ"ל קובעים את מיקומו של הג"רי בטבלת ההאש.

כל מיקום מייצג את שרשרת ה keyvaluepair שאותה מימשנו באמצעות רשימה מקושרת דו כיוונית , וכל ערך מחזיק את הפוינטר לג"רי וכל מפתח הוא ID של ג'רי.
בחרנו ברשימה מכוונת דו כיוונית, מתוך ידיעה שימחקו ויתווספו ג'רים, וברשימה דו כיוונית כל הוספה היא בזמן ריצה של O(1) מתווסף ישירות לסוף , ומחיקה תלוי במיקום של הג'רי אוותו רוצים למחוק, רק ועידכון המצביעים הוא יותר מהיר.

התנגשויות של מיקומים שווים פתרנו באמצעות שרשור, כאשר נכניס ג"רי למיקום תפוס הוא יכנס במקום האחרון באותה שרשרת של מיקום זה.

Kv מחזיק בערכו את הפוינטר לגר"י, ובערך המפתח את ערכו של ה-ID של ג'רי.



רז אגו 208139287
נאור סרף 305715781

בדרך מימוש זאת גם אם ייתכן התנגשות פונקציית הגיבוב תביא להתפלגות אחידה של הג"ריים בערכי ה k, הרי גודל הטבלה תלוי במספר הגריים, לכן נוכל לשלוף ג"רי ב O(1) זמן ממוצע **כנדרש**, אפילו אם ייתכן התנגשות.

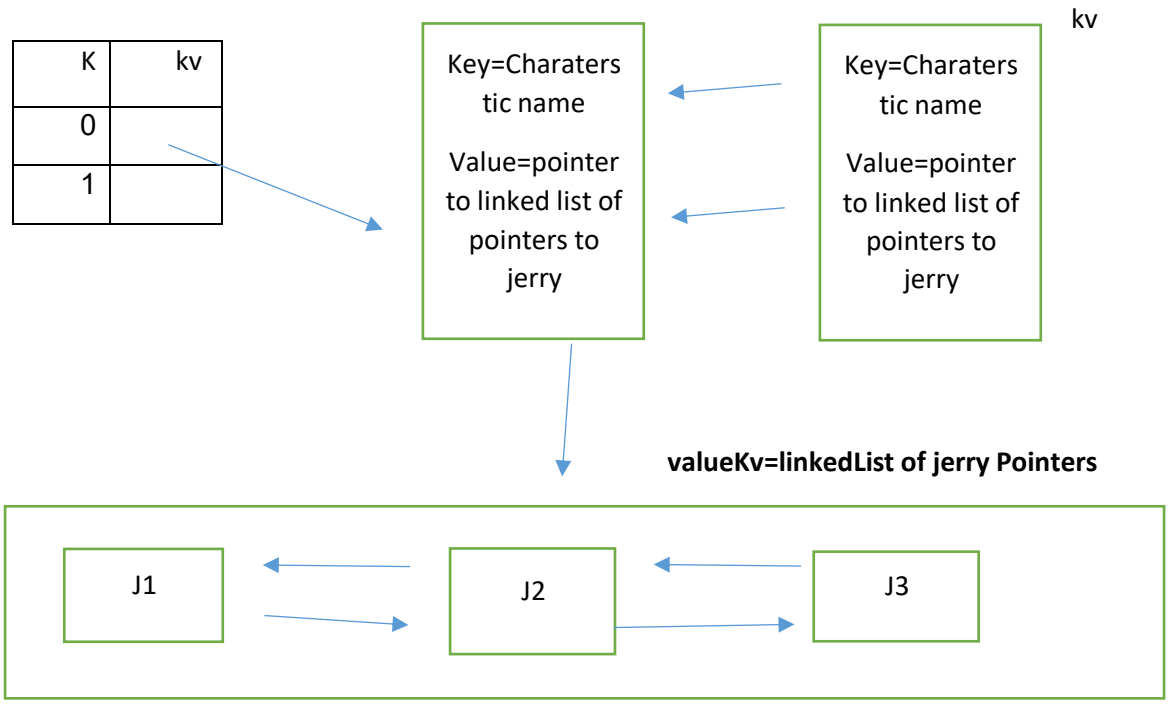
2. MsCp MultiValueHashTable:

במבנה זה השתמשנו בפונקציית גיבוב שעובדת על פי שם תכונה הפונקציה סוכמת את ערכי האסקי של התווים שמופיעם בשם של התכונה שאותה אנו מוספים למבנה, ואחר כך עושה: k = sum % hashnumber.

hashnumber קבענו כי יהיה מספר הג"ריים ההתחלתיים כפול 3, בהנחה כי במבנה הנתונים במהלך התוכנית יתבצעו מחיקות והוספה של ג"ריים. בחרנו להכפיל ב-3 את מספר הג"ריים ההתחלתיים כמספר קרוב אפשרי של מספרם בתוך מבנה הנתונים במהלך התוכנית. בנוסף קיימים ג"ריים עם יותר תכונות או כלל לא, לכן גם כאן בחרנו במספר זה, , לא נבחר מספר גדול מדי בשביל לשמור על זיכרון.

הגורמים הנ"ל קובעים את מיקומו של הג"רי בטבלת ההאש.

המימוש של מבנה זה דומה למימוש של ההאש טייבל השונה בניהם הוא שכל kv מחזיק בערכו רשימה דו כיוונית ,רשימת דו כיוונית של פוינטרים לג"ריים בעלי תכונה מסויימת, ובערך המפתח את שם התכונה.

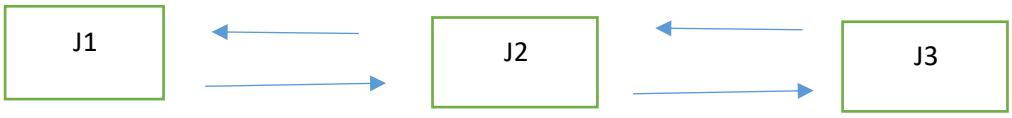


רז אגו 208139287
נאור סרף 305715781

מבנה זה נבחר, מתוך בקשה לשלוף ב O(1) אם קיימים ג'רים לתכונה מסויימת, ומי הם אותם ג'רים, שכן לתכונה אחת יכולים להיות כמה ג'רים שונים.
בצורה זאת פונקציית הגיבוב תביא להתפלגות אחידה של התכונות בערכי ה k, הרי גודל הטבלה תלוי במספר התכונות, לכן נוכל לשלוף רשימה של ג'רים ב O(1) זמן ממוצע כנדרש, ולהדפיס את כל הג'רים השייכים לתכונה מסויימת ב-
O(number of jerries with the physical characteristic)
שכן זה אורך הרשימה הקיימת בערך.
נבחרה רשימה דו כיוונית, על מנת לשמור על סדר כניסתם והוצאתם של הג'רים, ככה שיודפסו כל הג'רים השייכים לתכונה מסויימת, הם יודפסו על פי סדר כניסתם למערכת ובמקרה שימחק ג'רי אחד, הסדר לא ישפיע על הרשימה.
בנוסף לקח שתחזוקה של רשימה מקושרת דו כיוונית, כל הוספה של איבר עולה לנו ב O(1) וכל מחיקה תלוי במיקום האיבר, ותחזוקת מצביעים נוחה.

3.LinkedList JerryOrder

רשימה מקושרת דו כיוונית שמחזיקה בתוכה, על פי סדר כניסה לתוך המערכת פוינטרים של ג'רים
ככה שהראשון ברשימה, הוא הג'רי שנכנס הכי ראשון למערכת, והאחרון הוא הג'רי האחרון שנכנס למערכת
בחרנו במימוש מצורה הזאת, שכן ג'רים צריכים להיכנס ולצאת מהמערכת, ולבקשת ההדפסה שחייבת להיות על פי סדר ככניסתן, רשימה זאת מתוחזקת לארוך כל ריצת המערכת, בצורה כזאת שמי שנכנס נכנס אחרון, ומי שיוצא לא מפריע לסדר של הג'רים.



רז אגו 208139287
נאור סרף 305715781

בנוסף, המבנה היחיד שמבצע מחיקה עמוקה של ג'רי זה בJerryHash מפני שהתבקשנו לא להחזיק שיכפולים של ג'רים, שאר מבני התונים מחזיקים רק מצביעים לג'רים, ופונקציות המחיקה שלהם אינם מוחקות ג'רי מהמערכת, אלה רק את המבציע שלו מהמבנה עצמו.

כאשר אנחנו רוצים למחוק ג'רי מהמערכת נמחק את כל הקשרים שלו קודם במבנים השונים ואז נבצע מחיקה מתוך JerryHash, ככה דאגנו שלא יהיו העתקים של ג'רי בתוך המערכת שלנו מלבד במקום אחד.

במודול של ג'רי , הוספנו פונקציות אשר היו לנו חיוניות בכדי לגשת למידע בתוך ג'רי, שאין באפשרות מודול אחר לבצע.