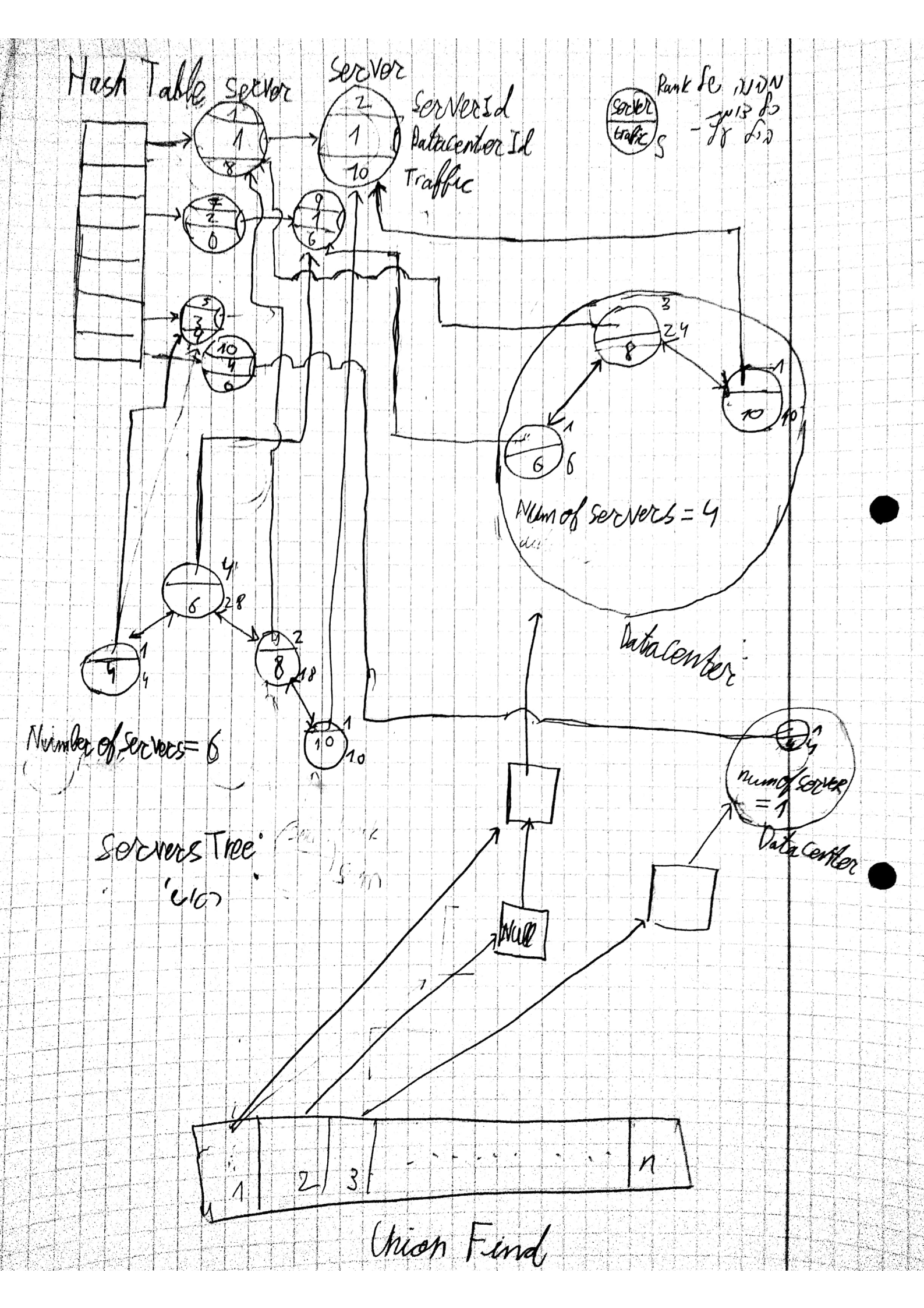
Hash Table Server
Server
ServerId
DatacenterId
Traffic
Rank לפי מספר
הנתונים
Server
traffic
Num of servers = 4
Datacenter
Number of servers = 6
Servers Tree
num of server = 1
Datacenter
Union Find

תיאור מבנה הנתונים:

שרת - Server

מייצג שרת. מכיל את ה ID של השרת שהוא מייצג, את ה ID של חוות השרתים שמכילה אותו ואת התעבורה של השרת.

חוות שרתים – Data Center

מייצג חוות שרתים. כל חווה תכיל את השרתים ששייכים אליה. היא תחזיק את הסרברים שייכים אליה

בעזרת עץ AVL שהמפתח יכיל מצביע לשרת שהצומת מייצג. ה- DATA בעץ יכיל את התעבורה של השרת.

העץ יהיה ממוין לפי התעבורה של השרת (במקרה של תעבורה שווה העץ ימיין לפי מזהה השרתים). בנוסף לכך בכל צומת נשמור מידע נוסף שהוא מספר הצמתים בתת העץ ושדה שמכיל את סכום התעבורות בתת עץ שנסמנו S. שדות אלה ניתנים לתחזוקה כפי שלמדנו בהרצאה מבלי לפגוע בסיבוכיות העץ. בנוסף לכך כל חוות שרתים תחזיק משתנה בשם numOfServers שיציין את מספר השרתים שיש בחווה.

Hash Table

טבלת האש טבל תכיל מערך שכל שדה יחזיק רשימה מקושרת שתכיל מצביעים לסרברים שקיימים בחוות השרתים. כאשר פקטור העומס יעלה על 0.75 נקצה מערך גדול פי 2 ונעתיק את ערכי המערך הישן אל החדש ונשחרר את המערך הישן. כאשר פקטור העומס יהיה קטן מ 0.3 נקצה מערך חדש שקטן פי 2 מהישן ונעתיק את ערכי המערך הישן אל המערך החדש ונשחרר את המערך הישן.

Union Find

ממומש באמצעות עצים הפוכים כפי שראינו בהרצאה. כל קבוצה תייצג את אוסף חוות השרתים שהיא מכילה. כאשר נאחד קבוצות – ניצור חוות שרתים חדשה שתכיל את כל השרתים של כל חוות השרתים שבקבוצה וחווה זו תהיה שמורה בשורש של העץ בתור DATA. נמחק את כל חוות השרתים האחרות.

Manager

מבנה הנתונים הראשי שמכיל את כל המערכת. יכיל את HashTable , UnionFind . בנוסף יכיל עץ AVL ראשי שיכיל את כל השרתים שקיימים במערכת ממוינים לפי התעבורה שלהם. בכל צומת בעץ נחזיק מידע נוסף שהוא סכום התעבורה בתת עץ ומספר הצמתים בתת עץ.

הסבר על סיבוכיות המקום של המערכת:

המבנה מכיל:

- עץ AVL בגודל m(כאשר m הוא סה"כ כל הסרברים) – O(m) מקום.

- מערך בגודל n שכל אינדקס מכיל כתובת לקבוצה שאליה שייכת חוות השרתים. סכום השרתים הוא m ולכן סה"כ סיבוכיות מקום O(n + m).

- טבלת ערבול דינמית שאת גודלה נסמן ב s. כאשר m הוא מספר השרתים הכולל. הטבלה גדלה

פי 2 כאשר עומס הפקטור נהיה גדול מ 0.75 וקטנה פי 2 כאשר עומס הפקטור נהיה קטן מ 0.3.

לכן 0.3m<=s<=0.75m כלומר s=O(m).

- לכן סה"כ סיבוכיות מקום היא O(n+m)

הסבר לסיבוכיות משוערכת של UnionFind וטבלת העירבול:

כאשר נחפש איבר ב UNION FIND נבצע כיווץ מסלולים כפי שלמדנו בהרצאה.

כאשר נאחד בין עצים הפוכים (זה קורה בפונקציית MergeDataServers) נאחד את העץ הקטן לגדול כפי שלמדנו בהרצאה. לכן כל פעולת חיפוש ואיחוד תארך באופן משוערך (O(log*(n)) זמן כאשר n הוא מספר חוות השרתים במבנה.

נכניס ונוציא איברים מטבלת הערבול כפי שלמדנו בכיתה. גם שינוי גודל הטבלה יתבצע כפי שלמדנו בכיתה, כך שסיבוכיות המשוערכת של הוספה והוצאה מטבלת הערבול תהיה O(1) בממוצע על הקלט.

void * Init(int n)

מאתחל מבנה נתונים עם n חוות שרתים. התעבורה בכל השרתים בהתחלה מוגדרת להיות 0.

פרמטרים:	n	מספר חוות השרתים במערכת.
ערך החזרה:	מצביע למבנה נתונים ריק או $NULL$ במקרה של כישלון.	
סיבוכיות זמן:	$O(n)$ במקרה הגרוע.	

מאתחלים עץ AVL ריק במבנה Manager ו n חוות שרתים ריקות בUnionFind. כל חוות שרתים תאתחל את העץ AVL שלה להיות ריק ואת המשתנה numOfServers להיות 0.

איתחול חוות שרתים לוקח O(1) זמן, ולכן איתחול n חוות שרתים לוקח O(n) זמן ריצה.

נאתחל את טבלת הערבול הדינאמית שתהיה בגודל קבוע. זמן ריצה O(1).

נחזיר מצביע למבנה הנתונים במידה והכל הצליח. אם היה כישלון נחזיר NULL.

סה"כ O(n) זמן ריצה.

StatusType MergeDataCenters(void *DS, int dataCenter1, int dataCenter2)

הפעולה מאחדת את חוות השרתים עם המזהים dataCenter1 ו-dataCenter2, כל השרתים משתי החוות עוברים להיות בחווה החדשה. לאחר איחוד שתי חוות שני המזהים dataCenter1 ו-dataCenter2 יתייחסו לחווה המאוחדת. לדוגמה, אם מאחדים את חווה מספר 1 עם חווה מספר 2 אז בעתיד כל התייחסות לחווה מספר 1 או 2 תהווה התייחסות לחווה המאוחדת.

פרמטרים:	DS	מצביע למבנה הנתונים.
	dataCenter1	מזהה חוות השרתים הראשונה.
	dataCenter2	מזהה חוות השרתים השניה.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, dataCenter1<=0, dataCenter1 > n, או dataCenter2<=0, dataCenter2 > n.
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(log^*(n)+m)$ משוערך, בממוצע על הקלט, כאשר n הוא מספר חוות השרתים במערכת ו m הוא מספר השרתים בשתי החוות המאוחדות.	

נבדוק אם הקלט תקין. אם הוא לא תקין נחזיר INPUT_INVALID. אחרת, הקלט תקין.

נחפש את חוות השרתים dataCenter1 ו dataCenter2. כל חיפוש כזה יקח O(log*(n)) משוערך.

נאחד את חוות השרתים – נאחד את העצים ההפוכים לפי גודל כפי שלמדנו בהרצאה וניצור DataCenter חדש שיהיה איחוד של dataCenter כפי שלמדנו בהרצאה. אם יצירת חוות שרתים המאוחדת נכשלה נחזיר ERROR_ALLOCATION. איחוד 2 חוות שרתים מתבצע בכך שמאחדים את עצי ה AVL של כל אחת מהן.

איחוד שני העצים מתבצע באופן הבא:

1. לכל עץ נבצע את הפעולות הבאות:
   a. נמצא את מספר האיברים בעץ לפי ע"י סיור Inorder. נסמן את האורך ב-mi. סיבוכיות הזמן היא O(mi).
   b. נבנה שני מערכים בגודל העץ- מערך מצביעי מפתחות ומערך מצביעי מידע. סיבוכיות זמן ומקום של O(mi).
   c. נעבור על העץ בסיור Inorder ונמלא את המערכים בערכים המתאימים (לא נעתיק את האלמנטים עצמם אלא רק את המצביעים לאלמנטים). מכיון שסיור Inorder עובר בסדר של ערכי המפתחות נקבל שני מערכים שממויינים לפי סדר המפתחות. סיבוכיות זמן של O(mi).
   d. נהרוס את העץ. סיבוכיות זמן של O(mi).
2. נסמן את סכום גודלי העצים ב-m. צעד 1 ארך בסופו של דבר O(m) זמן ו-O(m) מקום. ניצור מערך מפתחות ומערך מידע בגודל m כל אחד. נסמן מערכים אלה ב-mergedKeys ו- mergedData בהתאמה. סיבוכיות מקום של O(m).
3. נמזג את מערכי המידע של העצים לתוך mergedData (שוב, לא נעתיק את האלמנטים עצמם אלא רק את המצביעים לאלמנטים). לפי מבנה מערכי המידע ומערכי המפתחות של שני העצים, המפתח של המידע שנמצא במערך המידע מקום ה-i הוא המפתח שנמצא במערך המפתחות במקום ה-i. לכן, במהלך המיזוג, בהשוואת האיבר במקום ה- i נשתמש בערך המפתח במקום ה-i כדי לודא שהמערך הסופי יהיה ממוין. קיבלנו מערך שמכיל את כל איברי המידע של שני העצים ממוינים לפי ערכי המפתחות שלהם. סיבוכיות הזמן של המיזוג היא O(m).
4. נבצע מיזוג של שני מערכי המפתחות למערך mergedKeys (שוב, לא נעתיק את האלמנטים עצמם אלא רק את המצביעים לאלמנטים). סיבוכיות הזמן היא O(m). לפי מבנה הפעולות שבצענו, המפתח של האיבר ה-i ב-mergedData הוא האיבר ה- i במערך mergedKeys.
5. כעת נהרוס את המערכים המכילים את המפתחות והערכים של כל עץ. סיבוכיות זמן O(m).
6. כעת נבנה עץ ריק לא שלם בגודל m. נבנה את השורש, ואז נבצע את הפעולה הבאה עם הפרמטר size, כאשר ערכו ההתחלתי הוא m-1.
   a. אם size = 0 לא נבצע דבר.
   b. אם size = 1 נבנה צומת שמאלי ונעצור.
   c. אם size = 2 נבנה צומת שמאלי וצומת ימני ונעצור.
   d. אחרת, size > 2. נבנה צומת שמאלי וצומת ימני ונוריד 2 ל- size (כי בנינו כבר שני צמתים). אם size זוגי נבצע את הפעולה על העץ הימני ולאחר מכן על העץ השמאלי, לכל אחד עם פרמטר size/2. אם size אי זוגי נבצע את הפעולה על העץ השמאלי עם פרמטר של ערך עליון של size/2 ונבצע את הפעולה על העץ הימני עם פרמטר של ערך תחתון של size/2.

לפי מבנה הפעולה, ניצור עץ כמעט שלם ריק עם m צמתים. לפי התכונות של עץ כמעט שלם הוא עץ AVL. במהלך הפעולה בנינו m צמתים עם פעולות בגודל קבוע בכל צומת בעץ, לכן סיבוכיות הזמן והמקום היא O(m).

7. נתחיל מהשורש ונבצע סיור Inorder דרך הפעולה הבאה. נתחיל עם פרמטרים שמצביעים לאיבר הראשון בmergedKeys ו- mergedData בשמות keyptr ו- dataptr בהתאמה.
   a. אם העץ לא קיים (nullptr) נעצור.
   b. נבצע את הפעולה בבן השמאלי.

c. נכניס את האיבר ש-keyptr ו- dataptr מצביעים אליו להיות המפתח והערך של הצומת הנוכחית בהתאמה.
d. נקדם את keyptr ו- dataptr ב-1.
e. נבצע את הפעולה בבן הימני.
f. נחשב את ה-Rank של הצומת הנוכחית להיות סכום ה-Rank-ים של שני הבנים (בן שלא קיים הוא בעל Rank 0).
g. נחשב את S להיות סכום המידע של שני הבנים (בן שלא קיים הוא בעל מידע 0)

8. לפי מבנה Inorder נתחיל להכניס את הערכים מהצומת הקטן ביותר. נכניס אליו את המפתח והמידע הראשונים במערכים. בצומת הבא בסיור נכניס את המפתח והמידע השניים במערכים. בצומת הבא בסיור נכניס את המפתח והמידע השלישיים במערכים וכן הלאה. בסופו של דבר נעבור על כל העץ ונמלא אותו בכל האיברים של mergedKeys ו- mergedData. מכיון ש-mergedKeys ו- mergedData ממויינים בסדר עולה, והמעבר על העץ הוא בסדר עולה, העץ יתמלא באופן כזה שיהיה ממויין לפי ערכי המפתחות. בנוסף, לאחר שהפעולה התבצעה בבנים של כל צומת, ניתן לחשב את ערכי ה-Rank ו- S שלו לפי הערכים בבנים שלו (שבשלב זה כבר מכילים את הערכים והמפתחות המתאימים להם). עבור כל צומת אנחנו מבצעים פעולות בזמן קבוע, לכן סיבוכיות הפעולה היא O(m).
9. נהרוס את mergedKeys ו- mergedData.

מספר השלבים בתהליך המיזוג הוא קבוע, וכל שלב הוא בעל סיבוכיות זמן של O(m) במקרה הגרוע, לכן סיבוכיות הזמן הכוללת של פעולת המיזוג היא O(m). אם במהלך המיזוג נתקלנו בבעיית הקצאה נחזיר מייד ALLOCATION_ERROR. במהלך המיזוג כל מבני העזר והעצים הקודמים של שני חוות השרתים היו בגודל m, לכן סיבוכיות המקום הכוללת של המיזוג היא O(m) .

נחזיר SUCCESS.

את האיחוד נבצע בזמן O(m) ולכן סה"כ זמן הריצה של הפעולה הוא O(log*(n) + m) משוערך.

StatusType AddServer(void *DS, int dataCenterID, int serverID)

הפעולה מוסיפה שרת עם המזהה serverID לחוות השרתים עם המזהה dataCenterID התעבורה של השרת החדש היא 0.

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	dataCenterID	מזהה חוות השרתים.
	serverID	מזהה השרת החדש.
<u>ערך החזרה</u>:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, dataCenterID<=0, dataCenterID > n, או serverID<=0.
	FAILURE	אם כבר קיים שרת עם המזהה הנתון בחוות שרתים **כלשהי** במערכת.
	SUCCESS	במקרה של הצלחה.
<u>סיבוכיות זמן</u>:	$O(log^*(n))$ משוערך, בממוצע על הקלט, כאשר n הוא מספר חוות השרתים במערכת.	

נבדוק אם הקלט תקין. אם הוא לא תקין נחזיר INPUT_INVALID. אחרת, הקלט תקין.

נחפש את השרת בעל המזהה serverID בטבלת הערבול. אם הוא קיים אז נחזיר FAILURE.

אחרת, הוא לא קיים. סיבוכיות זמן של חיפוש זה הוא O(1) בממוצע על הקלט.

נקצה אובייקט מטיפוס Server שיהיה בעל מזהה serverID ותעבורה 0 ומזהה חוות השרתים יהיה dataCenterID. במידה וההקצאה נכשלה נחזיר ALLOCATION_ERROR. אחרת, ההקצאה הצליחה.

נוסיף את השרת לטבלת העירבול. במידה ופקטור העומס עולה על 0.75 נקצה מערך חדש ונעתיק את התוכן של המערך הישן אל החדש ונהרוס את המערך הישן. הקצאת שרת והוספתו לטבלת עירבול תארך O(1) זמן ריצה משוערך, בממוצע על הקלט. ניגש לחוות השרתים עם המזהה dataCenterID ונגדיל את ה numOfServers ב-1.

גישה לחוות השרתים והגדלת numOfServers תיקח O(log*(n)) זמן משוערך. נחזיר SUCCESS.

סה"כ זמן ריצה הוא O(log*(n)) משוערך, בממוצע על הקלט.

```
StatusType RemoveServer(void *DS, int serverID)
```

הפעולה מסירה את השרת עם המזהה serverID.

פרמטרים:	DS	מצביע למבנה הנתונים.
	serverID	מזהה השרת להסרה.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, או serverID<=0.
	FAILURE	אם לא קיים שרת עם המזהה הנתון.
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(log^*(n) + log(m))$ משוערך, בממוצע על הקלט, כאשר n הוא מספר חוות השרתים	

במערכת ו-m הוא מספר השרתים.

נבדוק את תקינות הקלט. אם הוא לא תקין נחזיר INPUT_INVALID. אחרת, הקלט תקין.

נחפש את השרת בעל המזהה serverID בטבלת ערבול. אם הוא לא קיים נחזיר FAILURE. אחרת, הוא קיים. חיפוש זה יתבצע בזמן O(1) בממוצע על הקלט.

נוציא את האיבר מטבלת הערבול. אם לאחר ההסרה פקטור העומס יהיה קטן מ 0.3 נקטין את המערך פי 2 ונעתיק את תוכן מערך הישן אל החדש, ונמחק את המערך הישן. זה יקח O(1) זמן ריצה משוערך, בממוצע על הקלט.

נמצא את חוות השרתים שמכילה את השרת בעל המזהה serverID. חיפוש זה יקח O(log*(n)) זמן ריצה משוערך.

נחפש את השרת בעץ השרתים של החווה. אם הוא קיים בעץ נוציא אותו ונעדכן את השדה numOfServers להיות קטן ב-1. זה יקח O(log k) זמן ריצה כאשר K הוא מספר השרתים בחווה זו ו m הוא מספר השרתים הכולל ולכן k<m. נחפש את השרת בעץ הראשי. אם הוא קיים בעץ נוציא אותו גם מהעץ הראשי של כל השרתים – זה יקח O(log m) זמן ריצה. נחזיר SUCCESS.

לכן סה"כ O(log*(n) + log m) זמן ריצה משוערך, בממוצע על הקלט.

לא השתמשנו בהקצאות בפעולה זו, לכן לא תתכן שגיאת הקצאה.

```
StatusType SetTraffic(void *DS, int serverID,  int traffic)
```

הפעולה קובעת את התעבודה בשרת עם המזהה serverID להיות traffic.

פרמטרים:	DS	מצביע למבנה הנתונים.
	serverID	מזהה השרת.
	traffic	כמות התעבורה בשרת.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם serverID<=0 או traffic<0.
	FAILURE	אם לא קיים שרת עם המזהה serverID במערכת.
	SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(log^*(n) + log(m))$ משוערך, בממוצע על הקלט, כאשר n הוא מספר חוות השרתים במערכת ו-m הוא מספר השרתים במערכת כולה.

נבדוק את תקינות הקלט. אם הקלט לא תקין נחזיר INPUT_INVALID. אחרת, הקלט תקין.

נחפש את השרת בעל המזהה serverID בטבלת הערבול. זה יקח O(1) זמן בממוצע על הקלט. אם לא מצאנו את השרת אז נעצור ונחזיר FAILURE כי השרת לא קיים. אחרת השרת קיים ולכן ניגש לשרת ונבדוק מה מזהה החווה שהוא שייך אליה. נמצא את החווה הזו בזמן O(log*(n)) משוערך.

נחפש את השרת בעץ AVL שבחווה , וגם בעץ AVL הראשי שמכיל את כל השרתים. אם הוא נמצא בעצים, נוציא אותו מהם ונכניס בחזרה. אם הוא לא נמצא בהם (במקרה שהתעבורה הקודמת שלו הייתה 0), נוסיף אותו לשני העצים. בכל מקרה נשנה את התעבורה של השרת לערך traffic החדש לפני ההכנסה לשני העצים.

במידה והקצאה כלשהי נכשלה נחזיר ALLOCATION_ERROR. אחרת, ההקצאה הצליחה.

הוצאה והכנסה אורכות 2O(log k) זמן לעץ שבחוות השרתים (כאשר k מספר השרתים בחווה זו ולכן k<m כי m הוא מספר השרתים הכולל), והוספה והוצאה מהעץ שמכיל את כל הסרברים לוקח 2O(log m) זמן.

נחזיר SUCCESS.

לכן סה"כ זמן ריצה הוא O(log*(n) + log m) משוערך, בממוצע על הקלט.

בונוס (5 נק'): תארו בחלק היבש כיצד ניתן היה לממש את הפעולה הנ"ל בסיבוכיות של $O(log^*(n) + log(m))$ משוערך.

במקום טבלת ערבול דינמית בה כל תא יכיל רשימת מקושרת, נבנה טבלת ערבול דינמית שכל תא יכיל עץ AVL. במקרה זה חיפוש השרת בעל המזהה serverID בטבלת הערבול יארך O(log m) במקרה הגרוע (אם כל השרתים במערכת נמצאים באותו התא בטבלת הערבול ואז העץ בגודל m) ושאר האלגוריתם יתבצע כמו שתארנו בפירוט האלגוריתם של הפעולה.

לכן בסופו של דבר סה"כ זמן הריצה יהיה O(log*(n) + log m) משוערך בלי דרישת ממוצע על הקלט.

```
StatusType SumHighestTrafficServers(void *DS, int dataCenterID,  int k, int *traffic)
```

הפעולה מחשבת את סך כל התעבורה בk השרתים עם התעבורה הגבוהה ביותר בחווה עם המזהה dataCenterID ומחזירה את התוצאה. אם dataCenterID==0 הפעולה מחזירה את סך כל התעבורה בk השרתים עם התעבורה הגבוהה ביותר מבין **כל** השרתים במערכת.

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	dataCenterID	מזהה חוות השרתים.
	traffic	מצביע למשתנה שבו תוחזר התוצאה.
<u>ערך החזרה</u>:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, dataCenterID<0, dataCenterID > n, k<0 או traffic == NULL.
	SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(log^*(n) + log(m))$ משוערך, כאשר n הוא מספר חוות השרתים במערכת ו-m הוא מספר השרתים במערכת כולה.

אם הקלט לא תקין נחזיר INPUT_INVALID. אחרת, הקלט תקין.

אם מזהה חוות השרתים שווה ל 0 – נפנה לעץ AVL הראשי שמכיל את כל הסרברים, אם הוא שונה מ 0 אז

נמצא את הקבוצה שאליה שייכת חוות השרתים זו במבנה Union find וניגש לעץ ששייך לקבוצה. זה יקח O(log*(n)) זמן משוערך.

כאשר מצאנו את העץ הרלוונטי נבצע את האלגוריתם הבא:

נלך לצומת הכי גדול בעץ לפי מפתח (שהוא בעל התעבורה הכי גדולה) – נעשה זאת בכך שנלך לבן הימני מהשורש כמה שאפשר. נסמן צומת זה בv זה יקח O(log m) זמן ריצה. לאחר שמצאנו את הצומת נריץ את אלגוריתם Rank(v) כפי שלמדנו בתרגול. סיבוכיות זמן ריצה שלו היא O(log m). לאחר ביצוע אלגוריתם RANK נדע מה האינדקס של הצומת זה. נסמנו ב r.

יש לציין שמכיון שהשרתים שאינם נמצאים בעץ בהכרח בעלי תעבורה 0, הם לא יתרמו לסכום הסופי שנדרש לחשב, לכן ניתן לחשב רק את סכום התעבורות של K השרתים בעלי התעבורה הגבוהה ביותר הנמצאים בעץ.

אם r-k קטן מ-1 אז בעץ נמצאים פחות מ-K שרתים. במקרה זה נרצה לחשב את סכום של כל השרתים בעץ, מכיון ששאר השרתים של החווה (אם קיימים) הם בעלי תעבורה 0 ולא יתרמו לסכום הסופי.

נריץ את אלגוריתם Select(r-k) שראינו בתרגול. זמן הריצה שלו הוא O(log m). נסמן את הצומת שיוחזר ב q.

(במקרה של r-k קטן מ-1 נסמן את הצומת עם התעבורה המינימלית ב-q).

נמצא את סכום התעבורות של כל השרתים עם אינדקס קטן מצומת הכי גדול(כולל). לאחר מכן נמצא את סכום כל התעבורות של כל הצמתים בעלי אינדקס קטן מאינדקס של q (וכולל תעבורת הצומת עצמו) . הפרש שני הסכומים הוא סכום כל התעבורה ב K השרתים עם התעבורה הגבוהה ביותר (במקרה של r-k קטן מ-1 לא נצטרך לחשב את הסכום השני כי הוא 0). את הסכומים נחשב באופן הבא:

נניח שברצוננו למצוא את סכום התעבורות של כל הצמתים עם אינדקס קטן מאינדקס צומת L

(כולל תעבורת L).

נשמור משתנה בשם SUM שישמור את סכום התעבורה. נאתחל אותו ל 0. נלך מהשורש אל הצומת L כך – - - אם הצעד הבא הוא ללכת לבן השמאלי אז נלך לבן השמאלי.

- אם הצעד הבא הוא ללכת לבן הימני אז נוסיף ל SUM את S של הבן השמאלי וגם את התעבורה של הצומת הנוכחי.

נמשיך כך באופן רקורסיבי. כאשר נגיע לצומת המבוקש L נוסיף את התעבורה שלו ל SUM ונוסיף את S של הבן השמאלי של L (אם קיים). כך נחשב את סכום התעבורה של כל התעבורות שקטנות או שוות לתעבורה של הצומת L. זמן הריצה הוא O(log m). לכן בסך הכל חישוב הסכום הנדרש יארך O(log m).

לכן סה"כ זמן ריצה של הפעולה הוא O(log*(n) + log(m)) משוערך.

void Quit(void **DS)

הפעולה משחררת את המבנה. בסוף השחרור יש להציב ערך NULL ב-DS, אף פעולה לא תקרא לאחר מכן

<u>פרמטרים</u>: DS מצביע למבנה הנתונים.

<u>ערך החזרה</u>: אין.

<u>סיבוכיות זמן</u>: $O(n+m)$ במקרה הגרוע, כאשר n הוא מספר חוות השרתים ו-m הוא מספר השרתים הכולל בכל החוות.

נעבור על כל החוות ובכל חווה נהרוס את העץ שלה, נהרוס את החווה עצמה ואת המבנה Union find.

נהרוס את העץ הראשי שמכיל את השרתים, ואת טבלת הערבול.

סה"כ זמן ריצה O(n + m)