מבנה הנתונים

* Union-find של חברות כך שכל node מכיל בנוסף לשדות של חברה רגילה גם שדה נוסף שידאג לאזן את ערך החברה כך שנקבל שערכה יתקבל כתוצאה של אלגוריתם find (בדומה לשאלה של המגדלים של הקוביות בתרגול של union-find).
* עץ דרגות מאוזן של מצביעים לעובדים שמכיל בכל צומת גם את סכום הדרגות בתת העץ של הצומת ואת מספר העובדים בתת העץ של הצומת, שיכיל את כל העובדים שהשכר שלהם גדול מ-0.
* טבלת ערבול דינמית שתכיל את כל העובדים.
* מספר העובדים עם משכורת 0.
* סכום הדרגות של העובדים עם משכורת 0.
* Bump\_grade\_interns - משתנה שסופר כמה דרגות הוספנו עבור העובדים עם משכורת 0.

חברה

* ID
* Value
* עץ דרגות מאוזן של מצביעים לעובדים בחברה שמכיל בכל צומת גם את סכום הדרגות בתת העץ של הצומת ואת מספר העובדים בתת העץ של הצומת ובשביל הבונוס יהיה שדה שיכיל כמה להעלות/להוריד בדרגות את אלו שמתחתיו, שיכיל את כל העובדים שהשכר שלהם גדול מ-0.
* מספר העובדים עם משכורת 0.
* סכום הדרגות של העובדים עם משכורת 0.

עובד

* Id
* Salary
* Grade
* Companyid
* Bump\_grade\_balance – ברגע שנכניס עובד לחברה כלשהי נקבע את משתנה זה להיות מינוס Bump\_grade\_interns.

\*מבני הנתונים לא יהיו גנריים

התייחסות לפעולות ספציפיות:

sumOfBumpGradeBetweenTopWorkersByGroup – שימוש בעץ דרגות של עובדים כך שכל צומת יכיל את מספר העובדים בתת העץ שלו ואת סכום הדרגות בתת העץ שלו, כדי למצוא את סכום הדרגות של העובדים המרוויחים ביותר ב-.

averageBumpGradeBetweenSalaryByGroup – נשמור את סך כל הדרגות שיש בשורש ואת מספר העובדים הכולל שיש בשורש. נעבור על עץ דרגות באופן הבא:  
אם הצומת הנוכחי מחוץ לתחום אז נרד בהתאם ונוריד מסך כל העובדים והדרגות את מה שמופיע בצומת שלא ירדנו אליו.  
אם אנחנו בתחום נבצע חיפוש של המינימום והמקסימום כך שתוך כדי מסלולי החיפוש אנו עושים את אותו תהליך כמו בהתחלה.

Acquire – מיזוג עצים: נבצע תחילה סיור preorder על שני העצים על מנת לאפס את הדרגות, נאחד את העצים רגיל כמו בAVL, ולבסוף נבצע postorder שיעדכן את כל השדות הנוספים.