# דף הסבר פרויקט Concordance - רן יונגר ואורי גרבי

את הפרויקט פיתחנו בשפת סי שארפ (#C) מכיוון שזאת שפת תכנות high-level נוחה וחזקה ביותר, והרגשנו שבה נוכל לממש את מבני הנתונים שבחרנו על הצד הטוב ביותר.

## הוראות שימוש:

כאשר התוכנית מורצת, יש להזין **כתובת מלאה (כולל שם הקובץ וסיומת)** של מיקום **קובץ txt** שרוצים לנתח. אחרי הזנת הכתובת, יש להקיש אנטר. ואז התוכנה תבנה את הקונקורדנציה לקובץ הנתון ותדפיס אותה.

במהלך הפיתוח מימשנו מספר מבני נתונים שיקלו עלינו את העבודה:

### Node

מחלקה זו מייצגת חוליה/צומת במבנה נתונים כלשהו המייצג אוסף. מימשנו אותה מחדש ובאופן גנרי כדי שלא נהיה מוגבלים לטיפוס מסוים. למחלקה זו אין מטודות.

### IntList

מחלקה זו הינה מימוש פרטני למחלקה List הגנרית, והיא מייצגת רשימה מקושרת של מספרי שורות עבור מילה מסוימת בקונקורדנציה.
במבנה נתונים זה מימשנו את המטודות הבאות:

- add - מקבלת מספר עמוד (int) ומוסיפה אותו לרשימה מקושרת.
  סיבוכיות: O(1)
- IndexOf - מקבלת מספר עמוד (int) ומחזירה את האינדקס שלו ברשימה, או את הערך 1- אם אינו ברשימה
  סיבוכיות: O(n)
- Contains - מקבלת מספר עמוד (int) ובודקת האם הוא ברשימה
  סיבוכיות: O(n)
- ToArray - מחזירה מערך מספרים המייצג את הרשימה
  סיבוכיות: O(n)
- ToString - מחזירה ייצוג טקסטואלי לרשימה
  סיבוכיות: O(n)

### Record

מחלקה זו מייצגת רשומה בטבלת גיבוב, כאשר רשומה מורכבת ממפתח ייחודי ומערך. במבנה נתונים זה מימשנו רק את מטודת ToString, המחזירה ייצוג טקסטואלי לרשומה (סיבוכיות: O(1)).

**RecordList**

מחלקה זו הינה מימוש פרטני נוסף למחלקה List הגנרית, והיא מייצגת רשימה מקושרת של רשומות, המרכיבה למעשה את טבלת הגיבוב.

במבנה נתונים זה מימשנו את המטודות הבאות:

- add - מקבלת רשומה ומוסיפה אותו לרשימה הקיימת
  סיבוכיות: O(1)
- add - מקבלת מפתח וערך ומוסיפה רשומה חדשה המורכבת מהמפתח והערך לרשימה הקיימת
  סיבוכיות: O(1)
- Contains - מקבלת רשומה ובודקת האם היא ברשימה
  סיבוכיות: O(n)
- ContainsKey - מקבלת מפתח string ובודקת האם הוא קיים באוסף המפתחות
  סיבוכיות: O(n)
- ContainsValue - מקבלת IntList המייצג ערך של רשומה ובודקת האם היא קיימת באוסף ערכי הרשומות
  סיבוכיות: O(n)
- IndexOf - מקבלת רשומה ומחזירה את האינדקס שלה ברשימה, או את הערך 1- אם אינה ברשימה
  סיבוכיות: O(n)
- IndexOfKey - מקבלת מפתח string ומחזירה את האינדקס שלו ברשימה, או את הערך 1- אם אינו ברשימה
  סיבוכיות: O(n)
- IndexOfValue - מקבלת IntList המייצגת ערך של רשומה ומחזירה את האינדקס שלה ברשימה, או את הערך 1- אם אינה ברשימה
  סיבוכיות: O(n)
- ToArray - מחזירה מערך רשומות המייצג את הרשימה
  סיבוכיות: O(n)
- ToKeyArray - מחזירה מערך string המייצג את המפתחות של כל הרשומות
  סיבוכיות: O(n)

- QuicksortString - שיטת מיון רקורסיבית המקבלת מערך string, אינדקס התחלה ואינדקס סיום, וממיינת את רשימת המפתחות בסדר מילוני (ובמקביל גם את הרשומות המיוצגות על ידי מפתחות אלו)
  סיבוכיות: O(nlogn)
  ToString המחזירה ייצוג טקסטואלי לרשימה
  סיבוכיות: O(n)

## Hashtable

בחרנו לממש טבלת גיבוב, אשר כל מילה היא מפתח לרשומה, והערך שלה הוא רשימת מספרי השורות בהן המילה מופיעה.
במבנה נתונים זה מימשנו את המטודות הבאות:

- add - מקבלת מילה ומספר שורה וקוראת לפונקצית add של מחלקת RecordList
  סיבוכיות: O(n)
- ContainsKey - מקבלת מילה וקוראת לפונקצית ContainsKey של מחלקת RecordList.
  סיבוכיות: O(n)
- ContainsValue - מקבלת רשימה מקושרת של מספרי שורות וקוראת לפונקצית ContainsValue של מחלקת RecordList.
- Sort - קוראת למטודה QuicksortString של מחלקת RecordList
  סיבוכיות: O(nlogn)
- ToString - מחזירה מחרוזת של הקונקורדנציה.
  סיבוכיות: O(n)

בחרנו לממש את הקונקורדנציה באמצעות טבלת גיבוב מכיוון שתכונת ייחודיות המפתח של טבלת הגיבוב סייעה לנו בטיפול בכפילויות של מילים.
בנוסף לכך, חיפוש והכנסה מתבצעים בסיבוכיות יחסית נמוכה (אצלנו סיבוכיות החיפוש היא O(n) מכיוון שאנו מריצים לולאה מאחורי הקלעים).