

פרויקט גמר – מבוא לתכנות

בפרויקט זה נרצה לנהל מאגר מידע של התלמוד הבבלי. נתחיל עם קובץ אחד bavli.txt המכיל את כל הטקסט הגולמי, ונמייין אותו לפי החיפוש שרצה לבצע בתוך המידע.

הדגשים לפרויקט:

1. השתמש בקלאסים. הקוד חייב להיות מחולק לקלאסים. כאשר כל פונקציה נמצאת בתוך הקלאס המתאים לה.
2. השתמש בפונקציות. אין לכתוב קטעי קוד ארוכים מידיי, בכל מצב בו ניתן להוציא קטע קוד לתוך פונקציה\מתודה בצע זאת.
- 3.

שלב 0:

פתח את קובץ bavli.txt עיין בתוכו ולמד את המבנה

שלב 0.5: כתיבת טסטים

שפות תכנות מאפשרות מנגנון לבדיקת הקוד ע"י שימוש במילה שמורה assert

דוגמא:

```
assert "hello" != "world"
assert "hello" == "hello"
assert 1 != 2
assert 1 == 1
assert true
```

בצעו קריאות למתודות שאתם כותבים וקראו לassert מול התוצאה הצפויה
לדוגמא

```
assert getDaf("ברכות", "ב", "א").startsWith("מאימתי קוראין")
```

שלב 1.

הדגמה לחיפוש בתוך מידע לא ממוין.

לפני שנמייין את המידע שלנו ונציג אותו באופן יעיל, ננסה לבצע חיפוש בתוך הטקסט הלא ממוין

הוסף לתוכנית את הפיצ'רים הבאים:

1. קבל כקלט מהמשתמש מסכתה ודף וחפש אותה בתוך הקובץ לאחר מכן הדפס אותה למסך.

יש לנהל מצבים בהם שם המסכת המוזן אינו שם תקין
יש לנהל מצבים בהם הדף שהוזן אינו תקין

2. חפש ציטוט מתוך התלמוד
במידה והציטוט לא נמצא, יש להדפיס הודעה מתאימה

שלב 2.

מיון הטקסט לפי דפים

כפי שניתן להבחין, חיפוש בתוך מאגר מידע גדול עם טקסטים לא ממוינים אינן יעיל ודורש זמן רב.

בשלב זה נרצה לחלק את תוכן הקובץ הגדול לקבצים המכילים כל עמוד בנפרד.

ניתן לשמור כל קובץ לפי שם המסכת, מספר הדף והעמוד

לדוגמא:

ברכות דף כ"ו עמוד א ישמר בנתיב `pages/brachot/26/alef.txt`

הוסף לתוכנית את הפיצ'רים הבאים:

1. צור קלאס המטפל בקריאה של תוכן הקובץ הגדול וכתובה לתוך קבצים קטנים לפי השם
 2. קבל כקלט מהמשתמש מסכתה ודף וחפש אותה בתוך הקובץ לאחר מכן הדפס אותה למסך.
- יש לנהל מצבים בהם שם המסכת המוזן אינו שם תקין
יש לנהל מצבים בהם הדף שהוזן אינו תקין

שלב 3.

טעינת רשימות ממוינות של שמות המסכתות והדפים לתוך הזיכרון

בשלב הנוכחי יש לנו דרך לבדוק האם מסכתה כל שהיא קיימת בתוך המערכת ע"י גישה למערכת קבצים. הבעיה עם כך היא שהגישה למערכת קבצים היא איטית ובמידה ומתבצעת לעיתים קרובות, יכולה לגזול משאבים מהמערכת.

במקום זאת, נרצה לטעון את כל המידע הזה אל תוך הזיכרון בזמן האתחול של התוכנית, ולהחזיק את המידע בזיכרון לאורך חיי התוכנית.

לפני שניגש לבדוק בדיסק אם קובץ כל שהוא קיים, נבדוק זאת קודם בתוך המבנה נתונים בזיכרון.

הוסף לתוכנית את הפיצ'רים הבאים:

1. צור מערך ממוין המחזיק את רשימת כל המסכתות הקיימות במערכת.
 2. עבור כל מסכתה החזק מערך ממוין של דפי הקיימים באותה מסכת.
 3. כתוב פונקציה המקבלת שם מסכת ודף והמבצעת חיפוש בינארי על רשימת המסכתות ועל רשימת הדפים.
 4. קבל כקלט מהמשתמש מסכתה ודף וחפש אותה בתוך הקובץ לאחר מכן הדפס אותה למסך.
- יש לנהל מצבים בהם שם המסכת המוזן אינו שם תקין
יש לנהל מצבים בהם הדף שהוזן אינו תקין

השתמש בקלאס כדי לייצג מסכתה ודפים של מסכתה ושומר את המידע בתוך ליסט.

שלב 4.

יצירת אינדקס עבור ביטויים

עד לשלב זה יצרנו אינדקסים לחיפוש מסכת ודף
כעת נרצה לאפשר חיפוש של ביטויים בתוך הדפים

4.1 מציאת כל המשניות של מסכת מבוקשת

אנו נרצה לקבל את כל המשניות של מסכת מבוקשת. כדי לבצע זאת נצטרך לעבור על כל הדפים של המסכת לקרוא את המידע ולהוציא החוצה את החלקים שהם משניות.

תהליך זה הוא יקר יחסית בהתחשב בעובדה שרוב הדפים אין בהם משניות

במקום זה, נרצה להוסיף לכל עמוד מידע האם קיים בתוכו משנה או לא ונקרא מהקבצים רק את הקבצים בהם יש משניות

בנוסף, נוסיף ערך המציין לאיזה פרק העמוד הזה משוייך

הוסף לתוכנית את הפיצ'רים הבאים:

1. סרוק את כל הדפים ועדכן את האינדקס עם פרטים לגבי העמודים בהם ישנם משניות

והפרק עליהם העמוד שייך

2. קבל מהמשתמש שם של מסכת כל שהיא ופרק והדפס למסך את תוכן כל הגמרא של אותו פרק.

3. קבל מהמשתמש שם מסכת ופרק והדפס את תוכן המשניות של אותו הפרק.

בנוסף. ניתן לציין בתוך האינדקס את מיקום תחילת המשנה וסופה וכך לבצע קריאה ישיר מהדיסק של תוכן המשנה.

שלב 5. יצירת cache

קריאת מידע מתוך דיסק איטית מאוד ביחס לקריאה מהזיכרון.

כדי לחסוך גישה לדיסק נהוג לשמור על שכבת cache בתוך הזיכרון עבור המידע שנקרא הכי הרבה.

ישנם אלגוריתמים שונים כיצד לנהל cache , אחד מהם נקרא LRU (Last Recently Used)

נרצה לממש שכבת cache בשביל אחסון דפי גמרא פופולריים.

נרצה להחזיק מערך דו ממדי המחזיק טקסט של דפים הפופולריים ביותר בכל מסכת ולנהל את את הדפים ב cache לפי הבקשות לצפייה בדף.