

# תוכנה 1 – סתיו תשע"ד

## תרגיל מספר 8

### מבני נתונים מקושרים וגנריים

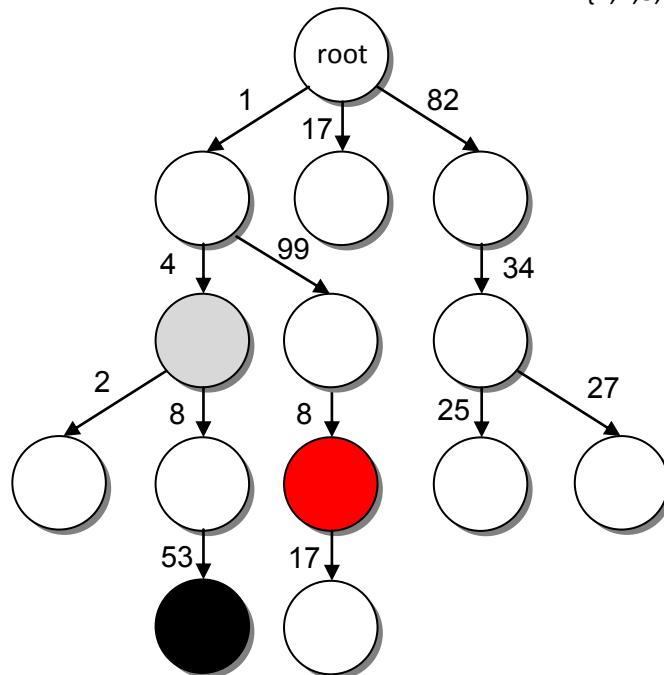
#### הנחיות כלליות:

קראו בעיון את קובץ נהלי הגשת התרגילים אשר נמצא באתר הקורס.

- הגשת התרגיל תעשה במערכת ה-moodle בלבד (<http://moodle.tau.ac.il/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש ומספר התרגיל (לדוגמא, עבור המשתמש aviv יקרא הקובץ aviv\_hw8.zip). קובץ ה-zip יכיל:
  - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז.
  - ב. קבצי ה-java של התוכניות אותם התבקשתם לממש, כולל תיקיות החבילה.

#### חלק א' - מבני נתונים מקושרים (50 נק')

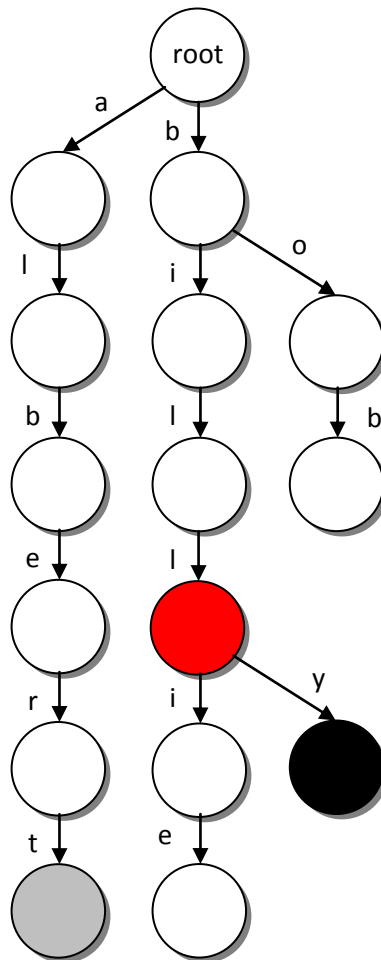
בחלק זה נתרגל עבודה עם מבני נתונים מקושרים, באמצעות מימוש עץ מסוג Trie. מבנה נתונים זה מאפשר לשמור ביעילות **רצפים** (אשר יכנו גם "מפתחות"), ולחפש אותם לפי התחיליות שלהם (prefix). כל רצף בעץ מיוצג ע"י מסלול מצומת השורש אל אחד הצמתים הפנימיים, כאשר כל קשת במסלול משוייכת לאיבר ברצף. לדוגמא, העץ בדוגמא הבאה מייצג Trie של רשימות של מספרים שלמים בין 0-99. המסלול לצומת הצבוע באדום מייצג את הרשימה {1,99,8}, והמסלול לצומת השחור מייצג את הרשימה {1,4,8,53}.



שימו לב, שמכל צומת יכולות לצאת לכל היותר 100 קשתות, עבור 100 המספרים בין 0-99. אם נרצה למצוא את כל הרצפים שמתחילים ב- {1,4}, ניקח את כל המסלולים העוברים דרך הצומת האפור.

בנוסף, מבנה ה-Trie שנממש יאפשר לנו לשמור, בצומת הקצה של מסלול שמייצג רצף, סט של ערכים המשויכים לרצף הזה. למשל, ניתן לשמור עבור כל רשימת מספרים את תאריכי ההוספה שלה. במקרה כזה, בצומת האדום נשמור את סט כל התאריכים בהם התבקשנו להוסיף לעץ את הרשימה {1,99,8} (מדובר בסט, כי ייתכן שזה קרה יותר מפעם אחת).

בתור דוגמה נוספת, נתבונן על Trie של מחרוזות. כאן כל איבר ברצף הוא תו במחרוזת. אם נניח שהמחרוזות שלנו מכילות רק את התווים a-z, מכל צומת ייצאו לכל היותר 26 קשתות. בדוגמה למטה, המסלול לצומת האדום מייצג את המחרוזת bill, המסלול לצומת השחור את המחרוזת billy, הערך שנשמור בצומת כל מחרוזת יכול להיות, למשל, מחרוזת אחרת שמייצגת את שם המשפחה. במקרה כזה, בצומת האדום נשמור את סט כל שמות המשפחה של אנשים בשם "bill".



באתר הקורס נתונים לכם שני מנשקים גנריים<sup>1</sup>:  $\text{TrieNode}\langle K, V \rangle$  ו- $\text{Trie}\langle K, V \rangle$  המייצגים צומת בעץ ואת העץ בהתאמה.  $K$  הוא טיפוס המפתחות הנשמרים בעץ, ו- $V$  הוא טיפוס הערכים הנשמרים עבור כל מפתח.

למשל, בדוגמה הראשונה למעלה,  $K$  יכול להיות  $\text{List}\langle \text{Integer} \rangle$  (רשימת שלמים) ו- $V$  יכול להיות  $\text{Date}$ , המייצג את תאריכי הוספת הרשימה לעץ. בדוגמה השנייה למעלה,  $K$  הוא  $\text{String}$  שמייצג שמות פרטיים, ו- $V$  הוא  $\text{String}$  שמייצג שמות משפחה.

<sup>1</sup> היזכרו במחלקות גנריות שנלמדו בשיעור 6 ובתרגול 8.

**TrieNode מגדיר את המתודות הבאות:**

**void** addSuffix(K suffix, V value)

- מקבלת סיומת של מפתח מטיפוס K וערך מטיפוס V, ומוסיפה אותם למבנה הנתונים תחת הצומת הנוכחי. לדוגמא, אם אנחנו בצומת האפור בעץ למעלה, וקיבלנו את הסיומת "a" ואת הערך "smith", נוסיף לצומת האפור קשת עם 'a' לצומת ילד חדש, ובצומת החדש נשמור את הערך "smith". כלומר, המסלול מהשורש לצומת החדש ייצג את השם "alberta", והערך שישמר עבורו הוא שם המשפחה "smith". בתור דוגמא נוספת, אם אנחנו בצומת האדום בעץ למעלה, וקיבלנו את הסיומת "ie" ואת הערך "jones", אין צורך להוסיף צמתים חדשים כי המסלול המתאים כבר קיים; נותר לנו רק להוסיף את הערך "jones" לצומת בקצה המסלול.

Set<V> getValues()

- מחזירה Set עם כל הערכים שנשמרו בצומת הנוכחי. למשל, אם הצומת הנוכחי הוא הצומת האדום מהדוגמא למעלה, נחזיר את כל שמות המשפחה שנשמרו עבור "bill".

TrieNode<K, V> getNext(K suffix)

- מחזירה את הצומת הבא על המסלול המתאים לסיומת הרצף הנתונה. למשל, הפעלת המתודה על הצומת האדום עם הסיומת "yabc" תחזיר את הצומת השחור. אם לא קיים צומת מתאים בעץ, יוחזר null.

List<TrieNode<K, V>> getNexts()

- מחזירה את רשימת כל הצמתים-ילדים של הצומת הנוכחי (כלומר, יש קשת מן הצומת הנוכחי אליהם).

**Trie מגדיר את המתודות הבאות:**

**void** addKey(K key, V value)

- מקבלת מפתח לעץ וערך המשויך אליו, ומוסיפה אותם לעץ.

Set<V> searchByPrefix(K prefix);

- מחפשת את כל המפתחות בעץ עם תחילית מסוימת, ומחזירה Set עם כל הערכים שנשמרו עבור המפתחות הללו. למשל, בדוגמא למעלה, בחיפוש התחילית "bill", המתודה תחזיר את כל שמות המשפחה שנשמרו עבור bill, billie, billy וכו'.

אנו נכתוב מימוש לשני המנשקים שבו K מוגדר להיות String, כלומר, נייצג Trie של מחרוזות.

1. באתר הקורס נתון לכם שלד המחלקה

**public class** StringTrieNode<V> **implements** TrieNode<String, V>

שימו לב - מכיוון שטיפוס המפתח מוגדר להיות String, למחלקה זו רק פרמטר גנרי אחד - V, טיפוס הערך הנשמר עבור כל מפתח.

לנוחיותכם, בשלד המחלקה כבר נתונים לכם שדות המחלקה, ובנאי.

השדה `nexts` - ישמור מצביעים לילדים בעץ. מכיוון שלכל צומת יש לכל היותר 26 ילדים, הבנאי הנתון מאתחל את רשימת הצמתים הבאים ב- 26 שומרי מקום לצמתים (null). באינדקס 0 יישמר מצביע לילד המחובר בקשת עם 'a', באינדקס 1 מצביע לילד המחובר ב-'b' וכו'. ניתן להוסיף ולגשת לילד באינדקס ה- i תוך שימוש במתודות `nexts.get(i)` ו- `nexts.set(i, child)`.

השלימו את מימוש המחלקה.

#### הערות:

- אנו נניח שכל הרצפים הנשמרים בעץ הם של אותיות a-z קטנות בלבד.
- אינכם צריכים לבדוק את תקינות הקלט למתודות - אנו נניח שבדיקת התקינות היא באחריות המחלקה `StringTrie`.
- אל תשתמשו ב- `Map` בפתרון הסעיף הנוכחי.
- מעבר לכך, אתם רשאים לשנות ולהוסיף למימוש הפנימי של `StringTrieNode` (השדות, הבנאי).
- אתם רשאים להוסיף מתודות ומחלקות עזר לפי הצורך, אך אל תשנו את המנשקים הנתונים.
- שימו לב שאינכם יודעים מראש מה יהיה הטיפוס הקונקרטי שנציב במקום `V` (`String` ?`Integer`), אך גם אין לכם צורך לדעת זאת - הפעולות היחידות שתצטרכו לבצע על ערכים מטיפוס `V` הן להוסיף אותם לאוספים שונים.

2. באתר הקורס נתון לכם שלד המחלקה

```
public class StringTrie<V> implements Trie<String, V>
```

השלימו את מימוש המחלקה.

#### הערות:

- נרצה שכל הרצפים הנשמרים בעץ הם של אותיות a-z קטנות בלבד. ב- `StringTrie` עליכם לבדוק את תקינות הקלט למתודות, ולוודא שהקלט שאתם מעבירים בקריאות למתודות של `StringTrieNode` תקין. אם הוא אינו תקין יש לזרוק שגיאה (ניתן להיעזר ב-`ErrorHelper` לשם כך).
- המפתח הריק "" תקין ובמקרה כזה יש לשמור את הערכים המשויכים אליו בשורש העץ. null אינו תקין.
- ניתן להעזר במתודה `addAll` של `Set` במימוש `searchByPrefix`.
- תוכלו לבדוק את עצמכם בעזרת התוכנית `TrieTest` הנתונה באתר הקורס. נתון לכם באתר גם הפלט המצופה.

## חלק ב' - Java collection framework (50 נק')

בחלק זה נתרגל עבודה עם אוספים (Collections) ע"י מימוש מנוע חיפוש פשוט. בין קבצי העזר של התרגיל תוכלו למצוא את המחלקות HTMLTokenizer, SearchEngine, Main ואת המנשק WordIndex שכבר נכתבו עבורכם.

מנוע החיפוש שלנו יטפל במספר מצומצם של דפי HTML אותם הוא יקרא מהרשת. דפים אלו מוגדרים מראש ורשימתם נמצאת במחלקה SearchEngine. אם תרצו תוכלו להוסיף או לשנות את הדפים בהם אתם משתמשים לצרכי בדיקה.

לאחר שהורדנו דף HTML מהרשת נתייחס רק לחלק הטקסט שבדף ונפרק חלק זה למילים בודדות. קוד זה כבר מומש עבורכם במחלקה HTMLTokenizer.

בנוסף, נתונה המחלקה SearchEngine שבה הקוד המפעיל את המערכת ומתקשר עם המשתמש. הקוד במחלקה זו קורא בתחילה למתודה index אשר תאכלס את אינדקס המילים. לאחר מכן, כגלות בקלט המשתמש יתבצע חיפוש של דפים לפי מילת שאילתה, או חיפוש של דפים לפי דמיון לכתובת דף נתון (במקרה זה יש להכניס את כתובת דף השאילתה לאחר הסימן ~, ראו דוגמא בהמשך).

### מה עליכם לעשות:

נרצה ליצור אינדקס של כל המילים שהופיעו בכל הדפים שהורדנו מהרשת. קיומו של אינדקס זה יאפשר לנו מאוחר יותר לבצע חיפושים עבור מילה מסוימת. עליכם לממש מחלקה בשם MyWordIndex המממשת את המנשק WordIndex. מחלקה זו שומרת את אינדקס המילים, מאפשרת הוספת מילים לאינדקס ולאחר מכן מאפשרת גם חיפוש באינדקס.

```
public interface WordIndex {

    /**
     * Add the words originating in the specifies URL.
     *
     * @param words - collection of words to add to the index
     * @param strURL - the URL of the page containing the words
     */
    void index(Collection<String> words, String strURL);

    /**
     * Search for pages containing a given word prefix in the index
     *
     * @param prefix - the word prefix to search
     * @return A list of pages containing the word. The pages are ordered
     *         according to the relative importance of the word within them.
     */
    List<String> search(String prefix);

    /**
     * Search for pages that are similar to a given URL in the index
     *
     * @param strURL - a URL, possibly not in the index
     * @return The page in the index that is the most similar to the input
     *         strURL.
     */
    String findSimilarPage(String strURL);
}
```

הערה: בקוד למעלה התיעוד כתוב בסגנון javadocs: התגית @param משמשת להסבר על פרמטר של המתודה, ו-@return משמשת להסבר על ערך ההחזרה. לא מדובר בחוזה!

**המתודות השונות:**• **המתודה index**

מתודה זו אחראית על אכלוס מבנה הנתונים שלכם. המתודה מקבלת אוסף של מילים (ייתכנו חזרות) ואת כתובת האינטרנט (URL) של הדף מהן הגיעו. עליכם לבחור את מבני הנתונים שבהם תשתמשו ולדאוג לשמור על הקשרים הבאים:

- לכל תחילית של מילה באילו דפים היא מופיעה וכמה פעמים בכל דף (למשל, בדף שמכיל רק את המילה java מופיעות התחיליות j, ja, j, java -i jav, פעם אחת כל אחת).
- לכל דף כמה מילים בסה"כ מופיעות בו (עם חזרות)
- לכל דף אילו מילים ייחודיות מופיעות בו (כלומר, ללא חזרות).

**שימו לב:**

- רשימת המילים בקלט היא כפי שהוחזרה ע"י HTMLTokenizer, אולם יש לשמור גרסת lowercase של המילים.
- אין צורך לנקות סימני פיסוק, רווחים וכו'.
- אין לשמור את המילה הריקה "" או null.

• **המתודה search**

מקבלת תחילית של מילה לחיפוש ומחזירה רשימה ממוינת של כתובות אינטרנט בהן המילה מופיעה. נמיינ את הרשימה כך שכלל שהמשקל היחסי של התחילית בדף גבוה יותר כך הכתובת תופיע במקום גבוה יותר ברשימה. המשקל היחסי של תחילית בדף יהיה מספר המילים שמתחילות בתחילית הנתונה (כולל מילים השוות בדיוק לתחילית) חלקי מספר המילים הכולל באותו דף (עם חזרות). לא נחזיר דפים בהם המילה אינה מופיעה כלל.

**שימו לב:** כדי להשוות טיפוסים פרימיטיביים ניתן להשתמש בשירות compare של הטיפוס העוטף. לדוגמא: Double.compare(double d1, double d2)

• **המתודה findSimilarPage**

מקבלת כתובת של דף אינטרנט (URL) ומחזירה את כתובת האתר באינדקס הדומה ביותר לאתר הנתון. לצורך חישוב רמת הדמיון בין שני דפי אינטרנט, נשתמש ב-Jaccard Index ([http://en.wikipedia.org/wiki/Jaccard\\_index](http://en.wikipedia.org/wiki/Jaccard_index)) המחושב כך: נחשב את קבוצת המילים הייחודיות בכל אחד מהדפים אותם נרצה להשוות, ואז נחלק את גודל חיתוך הקבוצות בגודל איחוד הקבוצות.

בנוסחא הבאה, A עשויה להיות קבוצת המילים הייחודיות של דף השאילתה, ו-B קבוצת המילים הייחודיות של דף אחר אליו נרצה לחשב רמת דמיון:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

**שימו לב:**

- בתוצאות החיפוש שיוחזרו ע"י המתודה יכללו רק דפים שחיתוך המילים המשותפות להם ולדף השאילתה מכיל לפחות 100 מילים (כדי לא להציג דפים שבהם יש חיתוך נמוך מאוד של מילים לדף השאילתה).
- אם האתר הנתון אינו מופיע באינדקס, אין להוסיף אותו לאינדקס!
- ניתן להיעזר במתודות retainAll ו-addAll המאפשרות חישוב חיתוך ואיחוד של אוספים (בהתאמה).

**הנחיות:**

- עליכם לכתוב ולהגיש את המחלקה MyWordIndex שממשת את המנשק WordIndex. ניתן להוסיף מחלקות ומתודות עזר.
- אין לשנות את הקבצים הנתונים באתר הקורס.
- לקבלת מלוא הנקודות על התרגיל יש להשתמש במבני הנתונים המתאימים לבעיה ובאופן ראוי (נכונות הפלט אינה הקריטריון היחיד). לדוגמא, כדי לשמור ערכים ללא חזרות יש להשתמש ב-Map או Set.
- את מיון הרשימה של כתובות האינטרנט בצעו בעזרת הפונקציה Collections.sort(...) שימו לב שה"מיון הטבעי" של הכתובות (String) הוא מיון לקסיקוגרפי ולא כפי שמתבקש בשאלה. כדי לשנות את שיטת המיון עליכם לכתוב מחלקת עזר המממשת את המנשק Comparator (מומלץ לחפש באינטרנט דוגמאות למימוש מנשק זה). מחלקה זו תיעזר בנתונים שנשמרו באינדקס לצורך מיון הכתובות. שימו לב שתוצאות ההשוואה תלויות במילת החיפוש הנוכחית.
- כדי לבדוק את התכנית שלכם השתמשו במחלקה Main המייצרת אובייקט חדש מטיפוס SearchEngine ומעבירה לו בבנאי מופע של המחלקה MyWordIndex שמימשתם. לאחר יצירת האובייקט נקרא השירות run.

**דוגמא (בהתבסס על רשימת הכתובות בקוד שניתן לכם):**

```
Indexing "http://en.wikipedia.org/wiki/Java_(programming_language)" ...
Indexing "http://en.wikipedia.org/wiki/Java" ...
Indexing
"http://docs.oracle.com/javase/7/docs/technotes/guides/collections/reference.html"
...
Indexing "http://www.gutenberg.org/files/1342/1342-h/1342-h.htm" ...
Indexing "http://www.gutenberg.org/files/1400/1400-h/1400-h.htm" ...
Indexing "http://www.gutenberg.org/files/76/76-h/76-h.htm" ...
Indexing "http://www.gutenberg.org/files/863/863-h/863-h.htm" ...
> hello
Searching for pages containing the prefix "hello"...
1. http://en.wikipedia.org/wiki/Java_(programming_language)
> world
Searching for pages containing the prefix "world"...
1. http://en.wikipedia.org/wiki/Java_(programming_language)
2. http://en.wikipedia.org/wiki/Java
3. http://www.gutenberg.org/files/1342/1342-h/1342-h.htm
4. http://www.gutenberg.org/files/76/76-h/76-h.htm
5. http://www.gutenberg.org/files/1400/1400-h/1400-h.htm
6. http://www.gutenberg.org/files/863/863-h/863-h.htm
> worl
Searching for pages containing the prefix "worl"...
1. http://en.wikipedia.org/wiki/Java_(programming_language)
2. http://en.wikipedia.org/wiki/Java
3. http://www.gutenberg.org/files/1342/1342-h/1342-h.htm
4. http://www.gutenberg.org/files/76/76-h/76-h.htm
5. http://www.gutenberg.org/files/1400/1400-h/1400-h.htm
6. http://www.gutenberg.org/files/863/863-h/863-h.htm
> zzzz
Searching for pages containing the prefix "zzzz"...
Unable to find relevant pages.
```

```
> ~http://www.gutenberg.org/files/161/161-h/161-h.htm
Searching for a page similar to URL "http://www.gutenberg.org/files/161/161-h/161-h.htm"...
  http://www.gutenberg.org/files/1342/1342-h/1342-h.htm
> ~http://fr.wikipedia.org/wiki/Java_%28langage%29
Searching for a page similar to URL
"http://fr.wikipedia.org/wiki/Java_%28langage%29"...
  http://en.wikipedia.org/wiki/Java_(programming_language)
> ~http://en.wikipedia.org/wiki/Java_(programming_language)
Searching for a page similar to URL
"http://en.wikipedia.org/wiki/Java_(programming_language)"...
  http://en.wikipedia.org/wiki/Java_(programming_language)
> (exit)
Bye!
```

שימו לב שהתוצאות עשויות להשתנות כתלות בהתעדכנות האתרים שנסרקו. אתם מוזמנים להשוות את הפלט עם חברים.

**בהצלחה !**