קוד כבוד

1. אין להעתיק את התשובות מכל מקור.

2. אל תשתמש בכלי AI גנרטיביים ליצירת הקוד שלך.

3. אינך רשאי לקבל או לחלוק חלק כלשהו מהקוד שלך עם אף אחד מלבד השותף שלך.

4. סטודנטים בוגדים יעמדו בפני ועדת המשמעת.

פרטי המשימה

במטלה זו תטמיעו מספר אלגוריתמי מיון שראיתם בכיתה. אנו מספקים אתה עם "טסטרים" שמפעילים את הפונקציות שלך על מערכים אקראיים ומציירים גרפים של זמני ריצה. גרפים אלה יאפשרו לך לקבל תחושה של ביצועי הפונקציות שלך, והן מצבים שבהם אלגוריתם אחד מתאים יותר מאחר. הגרפים שנוצרו יהיו חלק מההגשה שלך.

שלד הפרויקט שסופק מכיל מספר קבצים המיישמים את התווים/הגרפים. אתה חייב להוסיף קובץ חדש Sorter.java. הקובץ מכיל את סדרן הכיתה שבו עליך ליישם הפונקציות שלהלן.

Listing 1: Sorter’s public functions

public static <T extends Comparable <T>> void quickSort (T[] array );

public static <T extends Comparable <T>> void mergeSortNoRecursion (T[] array );

public static void radixSort ( Long [] array , int bitsPerDigit );

1. quickSort

עליך לבחור את הpivot על ידי בחירת אלמנט אקראי מהמערך. תחשוב על איך אתה יכול לשנות את הפסאודוקוד שתמיד משתמש באלמנט הימני ביותר של התת-מערך בתור ה-pivot.

הערה: פונקציה זו ממיינת מערך של אובייקטים T אשר מיישמים את ממשק Comparable (בדומה לשיעורי הבית של Heap).

2. mergeSortNoRecursion

למימוש זה שתבצע אסור להשתמש בקריאות רקורסיביות, הוא צריך להיות איטרטיבי בלבד (לולאות). דנו בזה בקצרה בכיתה. אתה יכול גם להיזכר בהדגמה של מיזוג רקורסיבי mergeSort מהתרגול. באיזה סדר אוחדו תתי-המערכים? תחשוב איך ליישם את אותה ההתנהגות ללא רקורסיה.

הערה: פונקציה זו ממיינת מערך של אובייקטים T אשר מיישמים את ממשק Comparable (בדומה לשיעורי הבית של Heap).

3. radixSort

פונקציה זו צריכה ליישם את אלגוריתם מיון ה-radix שראית בכיתה. יישום זה יהיה במיוחד למקרה כאשר הבסיס הוא חזקת שתיים כדלקמן. אתה יכול להניח שהבסיס הוא חזקת שתיים בין ל-. אתה יכול גם להניח שכל המספרים אינם שליליים (הקוד שלך לא צריך לאמת את ההנחות הללו).

Listing 2: Radix sort’s auxiliary function

static void countingSort ( Long [] arr , int bitsPerDigit , int digitIndex );

static int extractDigit ( Long key , int bitsPerDigit , int digitIndex );

עליך ליישם שיטת עזר פרטית בשם extractDigit, אשר לוקחת כ- input מספר Long כ- key .

ומחזירה את הספרה ה-digitIndex של ה- key כאשר הוא "מטופל" כמספר בבסיס .

לדוגמה, אם bitsPerDigit הוא 4, אז הבסיס צריך להיות בשימוש הוא 16 = , וארבעת "ביטי-המפתח" הפחות משמעותיות מייצגות את הספרה באינדקס 0 (כלומר, הספרה הכי פחות משמעותית) בייצוג בסיס 16 (הקסה-דצימלי) של key. 4 הביטים הבאים מייצגים את הספרה באינדקס 1 וכן הלאה.

יש להחזיר את ספרת הפלט למשתמש בתור int. נתקלת בפעולות סיביות בקורסים אחרים, למשל, תכנות מערכות ב-C. עליך ליישם extractDigit באמצעות מניפולציה של סיביות (למשל הסטה, מיסוך, וכו') אל תשתמש בחלוקה, מודולו וכו' במימוש שלך של extractDigit.

לאחר מכן, עליך ליישם את שיטת העזר הפרטית countingSort הממיין את מערך של Longs. במימוש זה של countingSort, המפתח של כל איבר ב- Array אינו כל המספר , אלא הספרה ה-digitIndex של כאשר רואים אותה כמספר בבסיס . ניתן לחלץ ספרה זו על ידי קריאה ל-extractDigit.

לבסוף, עליך לממש את radixSort שמתייחס ל- Longs במערך הקלט כאל מספרים בבסיס על ידי מציאת הערך המקסימלי במערך הקלט, עליך להסיק את מספר הספרות שיש להשתמש במיון הרדיקס. מימוש מיון רדיקס עם המספר של ספרות שחישבת באמצעות פונקציית העזר countingSort כדי לבצע את אלגוריתם המיון יציב. לפי כל ספרה.

בעת מימוש radixSort, מומלץ מאוד:

(א) לקרוא ולהבין היטב את הפסאודוקוד של radixSort ו-countingSort

(ב) לכתוב (ובדוק) extractDigit.

(ג) לכתוב (ולבחון) countingSort

(ד) לבסוף, כתוב (ובדוק) radixSort.

הנחיות ועצות

עליך **לבדוק כל אחת מפונקציות המיון שלך ביסודיות.** זה לא מספיק להפעיל כל אחד על כמה מערכים לדוגמה, עליך לחשוב על כל מקרי קצה האפשרים - הם מגוונים ורחבים (מאוד). לשנות את אורך המערך, לשנות את התפלגות הערכים, למיין מערכים שכבר ממוינים, מיון מערכים קצרים מאוד ומערכים ארוכים במיוחד (כדי לציין כמה מקרים). אם זה נשמע כמו משהו מוגזם, שים לב להטמעות ספריות סטנדרטיות של פונקציות מיון מתישהו יתגלו באגים שנים לאחר היישום כי לא היו מקרי קצה קטנים שנלקחו בחשבון על ידי המפתחים. **פונקציות מיון שנכשלות עבור סוגים מסוימים של מערכים יגרום לאובדן גדול של נקודות בציון.**

כתוב קוד לביצוע בדיקה זו עבורך, בדיקה ידנית לעולם לא תמצא את כל הבאגים. אֲפִילוּ למרות שאינך צריך לשלוח את הקוד שבודק את הפונקציות שלך, זה חיוני לבצע בדיקה כזו מסגרת אם ברצונך לקבל ציון טוב במשימה זו.

הערה: אנו מספקים בדיקה בסיסית וכמה כלי עזר שימושיים בקובץ בדיקה. אתה יכול להפעיל את הקובץ באמצעות

Maven/ה-IDE שלך בדיוק כמו ב-HW02. אתה יכול להשתמש במבחן זה כתבנית לבדיקות נוספות (כל הבדיקות יכולות להיכנס לקובץ הזה). הקוד שלך ייבדק מול בדיקות רבות נוספות אשר אנחנו לא מספקים. הקפד לכתוב טסטרים משלך.

לפונקציות צריכות להיות מאפייני הביצוע שתוארו בכיתה. למשל, אם mergeSort שלך הוא לא O(n · log(n)) במקרה הגרוע, משהו מאוד לא בסדר ואתה תפסיד נקודות. ברגע שאתה מרוצה שפונקציה נכונה, אתה יכול להתחיל לייעל אותה. אותו פסאודוקוד ניתן ליישם בדרכים רבות, חלקם יהיו מהירים יותר מאחרים. האופטימיזציות האלה לא ישפיעו על המורכבות האסימפטוטית של הפונקציות, אבל הם יכולים לעשות את הפונקציות שלך מהר יותר על ידי גורם קבוע.

ברגע שאתה מרוצה שהפונקציות שלך נכונות, עליך להפעיל את השיטה הראשית ב- SortAnalyser.java. פעולה זו תיצור פלט תיקיה ותציג גרפים של הפונקציות שלך, זמני ריצה. הפעלת הקוד עשויה להימשך מספר דקות בהתאם לביצועי המחשב שלך.

הקפד לבדוק כל אחד מהגרפים כדי לראות אם הביצועים היחסיים של המיון שלך מתפקדים ושזה מתיישב עם ההבנה שלך. קבצי PNG שנוצרו בפלט התיקיה צריכים להיות כלול ב-PDF שאתה שולח. ה-PDF חייב לקבל את השם graphExplanation.pdf. ב -PDF, עליך לכלול הסבר קצר על הניסויים בכל אחד מהגרָפים. ההסבר הכולל לא צריך להיות מעל עמוד (פחות זה בסדר). הסבר (אם אתה יכול) מדוע בניסויים מסוימים אלגוריתם x מהיר יותר מאלגוריתם y, אבל בניסויים אחרים y הוא מהיר יותר מ-x. הכותרות והתוויות של גרפים אלה צריכות להיות מובנות מאליהן, אבל אם יש לך כל ספק אתה יכול לקרוא את הקוד ב-SortAnalyser.java כדי לראות מה באמת כל ניסוי עושה.

מומלץ (אך אופציונלי): צור פרוייקט ב- Intellij IDEA

ניתן להוריד את Intellij IDEA בחינם (כמעט) בכל פלטפורמה, אנו ממליצים בחום להשתמש בה עבור שיעורי הבית האלה אבל זה בסדר גמור להשתמש בכל IDE אחר שאתה אוהב. ההוראות הבאות אמורות להכין את הפרויקט שלך ב-IDEA, אך הפרטים הספציפיים יכולים להשתנות לפי פלטפורמה/גרסה. אם אתה בגוגל "Intellij IDEA ייבוא פרויקט Maven קיים", אתה יכול למצוא מדריכים עבורך מערכת ההפעלה הספציפית שלך.

1. חלץ את HW5.zip לתיקיה בשם HW5.

2. ב-IDEA, צור "פרויקט ממקורות קיימים" חדש ובחר בתיקייה hw05 (זה ניתן למצוא תיקייה בתוך HW5).

3. בחר באפשרות "ייבוא פרויקט מדגם חיצוני". בחר "Maven" מהרשימה.

4. לחץ על "צור".

יש למקם את קובץ ה-Java שאתה כותב בתיקייה src/main/java (לצד קבצי Java שסופקו לך כמו SortAnalyser.java).

כל הקוד שלך צריך להיות ב- Sorter.java. מכיוון שלגרפים שיווצרו מהקוד שלך שאנו מספקים לך יש תלות של צד שלישי, עליך לוודא שייבאת את פרוייקט נכון כפרויקט Maven. אם אתה לא יודע איך לעשות את זה, זה דובר בשעות הקבלה של ג'ון במהלך HW02. הנה קישור לחלק הרלוונטי של ההקלטה.

<https://runi.cloud.panopto.eu/Panopto/Pages/Auth/Login.aspx?instance=MoodleProd2024&Auth=SessionView&panoptoState=c41a06cb-ab57-48d8-b8e4-b1760134b033>

אם נכשלת בייבוא נכון, זה ימנע ממך להפעיל את קוד הגרפים שאנו מספקים, מה שמקשה הרבה יותר לדעת אם הקוד שלך מתפקד כפי שאתה מצפה ויוצר. אי אפשר להגיש את הגרפים שנוצרו על ידי SortAnalyser.java.

הנחיות שונות

• כמו ב-HW02, תצטרך להבין את הגנריות ואת הממשק Comparable, ועוד כמה נושאים אחרים של Java. גנריות וממשקים משמשים (כמעט) בכל בסיס קוד לא טריוויאלי של Java, אז **כדאי מאוד** להבין אותם היטב. אם אתה מרגיש בנוח עם HW02, אתה מוכן כבר.

• אתה מורשה ל:

- להוסיף פונקציות פרטיות חדשות שיעזרו לך (וכדי לשמור על הקוד נקי וקריא).

• אינך רשאי:

- להסיר או לשנות את החתימה של פונקציות ציבוריות בכל מחלקה.

- לשנות שמות של מחלקות או שדות ציבוריים.

- לייבא כל דבר מיותר שעלול להביס את מטרת יישום המיון

פונקציות, לפתרון הרשמי יש רק ייבוא אחד ב- Sorter.java (רמז: זהו java.util.Random).

**הוראות הגשה**

**ניתן להגיש את המטלה בזוגות, זה לא חובה אבל מומלץ.**

**קח את הזמן כדי לעבור על הקוד שלך ולוודא שהוא מובן, לא ארוך מדי, וכן מעוצב היטב. עם קצת מחשבה, ניתן לשמור על יישומים קצרים למדי. סטיות מהנחיות אלו יגרור עונש בניקוד .**

**ודא שהקוד שלך יתבצע קומפילציה ופועל. קוד שלא יתקמפל לא יקבל ציון.**

**שלח קובץ zip בודד המכיל רק את הקבצים הבאים (ללא קבצים נוספים, תיקיות וכו'):**

**• Sorter.java**

**• graphExplanation.pdf**

**יש לתת שם ל-zip לפי הפורמט הבא.**

**אם שולחים לבד: ID FIRSTNAME-LASTNAME.zip**

**למשל: 1234 John-Doe.zip**

**אם שולחים כזוג: ID1 FIRSTNAME1-LASTNAME1 ID2 FIRSTNAME2-LASTNAME2.zip**

**לדוגמה: 1234 John-Doe 5678 Jane-Doe.zip**

**קובץ הגשת jar check.jar כלול במטלה זו. אתה יכול לדעת אם ההגשה שלך (קובץ zip) נקרא ומעוצב כהלכה על ידי הפעלת הפריטים הבאים בטרמינל/מעטפת:**

**java -jar submission check.jar example/path/123456 Jon-Simon.zip**

**מועד אחרון**

**המועד האחרון הוא ה-31 במאי. אתה עשוי לקבל הארכה אוטומטית עד ה-4 ביוני.**