

תרגיל בית תכנותי
להגשה עד 27.01.22 בשעה 23:50
בהצלחה!

תרגיל זה מנוסח בלשון זכר מטעמי נוחות בלבד והוא מיועד לכל המגדרים.
מתרגל אחראי על התרגיל: שמעון

הוראות:

1. יש להגיש קובץ zip **יחיד** בעל השם EXP_ID1_ID2 כאשר ID1 ו ID2 הם מספרי תעודות הזהות של שני בני הזוג. קובץ ה zip יכיל תיקייה בודדת בשם src ובה כל קבצי ה Java שיצרתם, ללא תיקיות נוספות ותתי תיקיות. אין צורך להגיש קבצים שסופקו ע"י צוות הקורס.
2. ההגשה תתבצע רק ע"י אחד מבני הזוג למקום הייעודי באתר הקורס במודל.
3. **עליכם לוודא לפני ההגשה במודל כי הקוד שלכם מתקמפל ורץ בשרת Microsoft Azure שהוקצה לכם** (הוראות מצורפות בקובץ נפרד).
4. זוג שהתרגיל שלו לא יתקמפל בשרת שהוקצה או יעוף בזמן ריצה **ציונו בתרגיל יהיה 0.**
5. יש לכתוב קוד קריא ומסודר עם שמות משמעותיים למשתנים, למתודות ולמחלקות.
6. יש להקפיד למלא את כל דרישות התרגיל (שימוש בייצוג הנכון, סיבוכיות זמן וכו') **אי עמידה בדרישות התרגיל תגרור ציון 0.**

בתרגיל בית זה אתם מתבקשים לממש בשפת Java מבנה נתונים דינמי המאפשר לבצע פעולות על גרף מכון פשוט בשם DynamicGraph.

הוראות התרגיל:

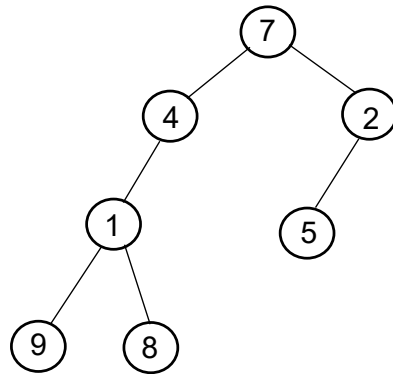
הגדרות:

1. לכל אורך התרגיל נסמן את הגרף המכוון הנוכחי המיוצג באמצעות המחלקה `DynamicGraph` באמצעות הסימון הסטנדרטי לגרף בקורס $G = (V, E)$. את מספר הצמתים ב G נסמן ב n ואת מספר הקשתות ב m .
2. בתרגיל בית זה לכל צומת בגרף (בין אם בגרף G או בעץ מושרש שתדרשו לממש) יש מזהה ייחודי המיוצג באמצעות מספר שלם חיובי.

שלב ראשון של המימוש:

1. ליצור קובץ בשם `GraphNode.java` ובו יש מחלקה **פומבית** בשם `GraphNode`. מחלקה זו מייצגת צומת $v \in V$ ולה המתודות **הפומביות** הבאות:
 1. `public int getOutDegree()` - מתודה המחזירה את דרגת היציאה של צומת v כלומר, מספר הקשתות היוצאות מצומת v . מילוי המטודות $O(\deg_{out}(v))$. סיבוכיות נדרשת $O(\deg_{out}(v))$.
 2. `public int getInDegree()` - מתודה המחזירה את דרגת הכניסה של צומת v כלומר, את מספר הקשתות הנכנסות לצומת v . סיבוכיות נדרשת $O(\deg_{in}(v))$.
 3. `public int getKey()` - המתודה מחזירה את המזהה של הצומת. סיבוכיות זמן נדרשת $O(1)$.
2. ליצור קובץ בשם `GraphEdge.java` ובו יש מחלקה **פומבית** בשם `GraphEdge`. מחלקה זו מייצגת קשת $e \in E$.
3. ליצור קובץ בשם `RootedTree.java` ובו יש מחלקה **פומבית** בשם `RootedTree`. מחלקה זו מייצגת עץ מושרש ולה המתודות **הפומביות** הבאות:
 1. `public RootedTree()` - בונה ברירת מחדל (default constructor) למחלקה `RootedTree` היוצר עץ מושרש ריק, ללא צמתים. סיבוכיות נדרשת $O(1)$.
 2. `public void printByLayer(DataOutputStream out)` - המתודה מקבלת רפרנס לאובייקט מסוג `DataOutputStream` ומדפיסה ל `stream` את המזהים הייחודיים של הצמתים תחת הדרישות הבאות:
 1. מזהים של צמתים השייכים לרמה i בעץ (צמתים בעומק i) יודפסו בשורה ה $i+1$ של `stream`.
 2. הדפסות של מזהים באותה הרמה יופרדו רק באמצעות פסיקים.
 3. הדפסה של מזהים של צמתים באותה הרמה מתבצעת משמאל לימין. פורמלית, לכל שני צמתים u ו v עם הורה משותף כך ש u הוא אח שמאלי (לאו דווקא ישיר) של v מתקיים: המזהה של u יודפס לפני המזהה של v . אם u' ו v' הם צאצאים של u ו v , בהתאמה ונמצאים באותה רמה בעץ, אז המזהה של u' יודפס לפני המזהה של v' .

סיבוכיות זמן נדרשת - $O(k)$, כאשר k מייצג את מספר הצמתים בעץ.
לדוגמא, הפעלת המתודה הנ"ל על הייצוג של העץ המושרש



צריכה להדפיס:

7
4,2
1,5
9,8

שימו לב כי אין פסיק בסוף השורה. ניתן לראות דוגמאות נוספות בפלט החוקי שקיבלתם.

3. **public void preorderPrint(DataOutputStream out)** -

המתודה מקבלת רפרנס לאובייקט מסוג `DataOutputStream` ומדפיסה ל `stream` את המזהים של צמתי העץ בסדר `preorder` (כפי שנלמד בתרגול 3) תחת הדרישה שהדפסת תתי העצים של צומת מתבצעת מהילד השמאלי ביותר לימני ביותר. ההדפסה מתבצעת בשורה אחת כאשר בין מזהים מפריד רק פסיק (אין פסיק בסוף השורה).

שלב שני של המימוש:

כעת אנו יכולים להגדיר את המחלקה `DynamicGraph`.

עליכם ליצור קובץ בשם `DynamicGraph.java` ובו יש מחלקה **פומבית** בשם `DynamicGraph`. המחלקה `DynamicGraph` משתמשת במחלקות `GraphNode`, `GraphEdge` ו-`RootedTree` על מנת לממש את המתודות הפומביות הבאות:

1. **`public DynamicGraph()`** - בונה ברירת מחדל (default constructor) למחלקה `DynamicGraph` היוצר גרף דינמי חדש ריק, ללא קשתות וללא צמתים. סיבוכיות זמן נדרשת - $O(1)$.
2. **`public GraphNode insertNode(int nodeKey)`** – מתודה זו מכניסה צומת חדש לגרף עם מזהה ייחודי `nodeKey` ומחזירה רפרנס לאובייקט מסוג `GraphNode` המייצג את הצומת שהוכנס. הצומת החדש מתווסף לגרף ללא קשתות נכנסות או יוצאות. סיבוכיות זמן נדרשת - $O(1)$.
3. **`public void deleteNode(GraphNode node)`** – המתודה מקבלת רפרנס בשם `node` לצומת שנמצא בגרף. אם לצומת `node` יש קשתות יוצאות או נכנסות המתודה לא עושה כלום. אחרת, המתודה מוחקת את הצומת `node` מהגרף. סיבוכיות זמן נדרשת - $O(1)$.
4. **`public GraphEdge insertEdge(GraphNode from, GraphNode to)`** – המתודה מקבלת שני רפרנסים לצמתים בגרף `from` ו-`to`. המתודה מוסיפה לגרף קשת מהצומת `from` לצומת `to` ומחזירה רפרנס לאובייקט המייצג את הקשת שהוכנסה. סיבוכיות זמן נדרשת - $O(1)$.
5. **`public void deleteEdge(GraphEdge edge)`** – המתודה מקבלת רפרנס לקשת בגרף בשם `edge`. המתודה מוחקת את הקשת `edge` מהגרף. סיבוכיות זמן נדרשת - $O(1)$.
6. **`public RootedTree scc()`** – המתודה מחשבת רכיבים קשירים היטב בגרף $G = (V, E)$. המתודה מחזירה רפרנס לאובייקט מסוג `RootedTree`. מבנה ה-`RootedTree` הוא כדלקמן:
השורש הוא צומת וירטואלי (שלא קיים בגרף) עם מזהה ייחודי 0. קבוצת הצאצאים של כל ילד של השורש מהווה רכיב קשיר היטב בגרף G . סיבוכיות זמן נדרשת - $O(n + m)$.
(שימו לב לדרישות הנוספות המצורפות למטה).
7. **`public RootedTree bfs(GraphNode source)`** – המתודה מקבלת רפרנס לצומת בגרף `source`, ומחזירה עץ מסלולים קצרים מהצומת `source`. סיבוכיות זמן נדרשת - $O(n + m)$.
(שימו לב לדרישות הנוספות המצורפות למטה).

עליכם לחשוב איך מחלקה תשתמש במחלקה אחרת, ואולי להגדיר מחלקות, משתנים ומתודות נוספות כרצונכם.

דרישות:

שימו לב, אי עמידה בדרישות אלו תגרור ציון 0.

1. שמות המחלקות והחתימות של המתודות הפומביות צריכים להופיע בקוד שלכם בדיוק כפי שמופיעים בקובץ זה.
2. **אין להשתמש בשום import בקוד שלכם מלבד**
 - `import java.io.DataOutputStream`
 - `import java.io.IOException`

בקובץ `RootedTree.java`.
3. אין להשתמש במחלקה `System` (אין לרשום בקוד שאתם יוצרים `System`) מלבד `System.lineSeparator()`.
4. העץ המושרש אותו עליכם לממש באמצעות המחלקה `Rooted_Tree` משתמש בייצוג `left child right sibling` כפי שנלמד בתרגול 3.
5. על מנת להקל על תהליך הבדיקה (והוידוא שהפלט תקין) עליכם לממש את המתודות הפומביות:
 - **`public RootedTree scc()`**
 - **`public RootedTree bfs(GraphNode source)`**

של המחלקה `DynamicGraph` באמצעות האלגוריתמים המתאימים שנלמדו בהרצאה.

כחלק מדרישה זו נוסיף את הדרישות הבאות:

1. אלגוריתמים `BFS` ו `DFS` מבצעים סריקה של רשימת השכנויות של צמתים (שורה 4 באלגוריתם `BFS` ושורה 4 בפרוצדורה `DFS_VISIT`). בהקשר זה נדרוש שסריקה של קשתות תתבצע מהקשת החדשה ביותר שנכנסה לגרף לקשת הישנה ביותר. בנוסף, אלגוריתם `DFS` מבצע גם סריקה של צמתים (שורה 5 באלגוריתם `DFS`). נדרוש שהסריקה תתבצע מהצומת החדש ביותר לצומת הישן ביותר.
דרישה זו מקבעת את סדר הגילוי של הצמתים ונקרא לסדר זה הסדר המושרש (כלומר, אם צומת v נמצא לפני צומת u בסדר המושרש אז `BFS` או `DFS` גילו את צומת v לפני צומת u).
2. העץ המושרש המוחזר ע"י המתודות הנ"ל צריך לקיים את התכונה הבאה: לכל שני צמתים u ו v בעץ שלהם הורה משותף, אם צומת v מופיע לפני צומת u בסדר המושרש אז צומת v הוא אח שמאלי (לאו דווקא ישיר) של צומת u .

הסבר על הקבצים שקיבלתם:

1. `Test.java` – דוגמת הרצה.
2. `test_output.txt` – פלט לאחר הרצה של `Test` בשרת `Microsoft Azure` שהוקצה לכם.

הנחות:

- ניתן להניח כי לא תוכנס לגרף קשת שהיא לולאה עצמית.
- ניתן להניח כי לא תוכנס לגרף אותה הקשת יותר מפעם אחת.
- ניתן להניח כי בעת הפעלת המתודה

`public GraphNode insertNode(int nodeKey)`

של המחלקה `DynamicGraph` הערך `nodeKey` הוא ייחודי כלומר, לא קיים צומת ב G עם המזהה `nodeKey`.

הדרכה:

אתם נדרשים לממש גרף דינמי המאפשר הכנסה ומחיקה של צמתים בסיבוכיות זמן $O(1)$. חשבו האם הייצוגים של גרף שנלמדו מאפשרים זאת. במידה ולא, אילו שינויים ניתן לבצע בייצוג על מנת לתמוך בפעולות אלו בסיבוכיות הזמן הדרושה.

הסבר על תהליך הבדיקה האוטומטית:

אנחנו נריץ את הקבצים שלכם עם מחלקת `Test` שונה מזאת שקיבלתם עם פרסום התרגיל. ב `Test` הבדיקה ייתכן ויתווספו אובייקטים, הפעולות וסדר הפעולות ישתנה, גודל הקלט ישתנה וכו'.

במהלך הבדיקה יקומפלו כל הקבצים שהגשתם בתוספת `Test` הבדיקה בשרת `Microsoft Azure`. **חשוב מאוד שתגישו את כל קבצי ה `java` שיצרתם ותוודאו שהקוד מתקמפל בשרת.**

בהנחה והקוד מתקפל, הקוד יורץ והפלט של התוכנית ישווה לפלט חוקי.

המלצות:

1. אל תשאירו את הבדיקה בשרת לרגע האחרון. ייתכן והקוד לא יתקמפל בשרת ותצטרכו לתקנו לפני ההגשה.
2. התחילו מבניית פעולות ההכנסה והמחיקה. רק לאחר שבדקתם שפעולות אלו עובדות נכון המשיכו במימוש שאר הפעולות.

בהצלחה!