מגישים:

אופיר קרנגל 302621305

רומן אברמזון 306359001

# שאלה 1

1. מחלקת Graph – מחלקה היוצרת גרף המורכב מצמתים בעלי טיפוס כלשהו. מכיוון שהגרף לא יודע (ולא צריך לדעת לצורך מימושו) מהו סוג הצומת, סוג הצומת מתקבל ע"י generic בשם Node. כל צומת יכול להופיע רק פעם אחת בגרף ולכן השדה הפנימי לתיאור רשימת הצמתים הוא מסוג HashSet. בנוסף ,הגרף מחזיק תיאור של קשתות בין צמתים שונים וגם כאן אותה קשת לא יכולה להופיע פעמיים ולכן נחבר מימוש זהה לתיאור רשימת הקשתות מסוג HashSet. לצורך תיאור קשת יצרנו מחלקה בשם Edge שמקבלת את צומת האב וצומת הילד.
2. סיבוכיות המתודות השונות:

מכיוון שרשימת הצמתים ורשימת הקשתות מומשו באמצעות HashSet אז הסיבוכיות על הוספה (add) או האם האובייקט מוכל (contained) היא O(1) והיא הנמוכה מבין המימושים. בפונקציה getChildren השתמשנו ב PriorityQueue בסדר יורד מכיוון שאלגוריתם DFS משתמש בה בעיקר ולשם פעולתו הוא צריך לקבל את הילדים של צומת בסדר יורד. הסיבוכיות עבור כל אחד מ containers מפורטת בטבלה הבאה:

|  |
| --- |
| List | Add | Remove | Get | Contains | Next | Data Structure  ------------------------------------------------------------------------------ |
| ArrayList | O(1) | O(n) | O(1) | O(n) | O(1) | Array |
| LinkedList | O(1) | O(1) | O(n) | O(n) | O(1) | Linked List |
|  |
|  |
|  |
|  |
| Set | Add | Remove | Contains | Next | Size | Data Structure |
| ----------------------|----------|----------|----------|----------|------|------------------------- |
| HashSet | O(1) | O(1) | O(1) | O(h/n) | O(1) | Hash Table |
| LinkedHashSet | O(1) | O(1) | O(1) | O(1) | O(1) | Hash Table + Linked List |
| EnumSet | O(1) | O(1) | O(1) | O(1) | O(1) | Bit Vector |
| TreeSet | O(log n) | O(log n) | O(log n) | O(log n) | O(1) | Red-black tree |
|  |
|  |
|  |
|  |
|  |
| Queue | Offer | Peak | Poll | Remove | Size | Data Structure |
| ------------------------|----------|------|----------|--------|------|--------------- |
| PriorityQueue | O(log n) | O(1) | O(log n) | O(n) | O(1) | Priority Heap |
| LinkedList | O(1) | O(1) | O(1) | O(1) | O(1) | Array |
| ArrayDequeue | O(1) | O(1) | O(1) | O(n) | O(1) | Linked List |
|  |
| Map | Get | ContainsKey | Next | Data Structure |
| ----------------------|----------|-------------|----------|------------------------- |
| HashMap | O(1) | O(1) | O(h / n) | Hash Table |
| LinkedHashMap | O(1) | O(1) | O(1) | Hash Table + Linked List |
| EnumMap | O(1) | O(1) | O(1) | Array |
| TreeMap | O(log n) | O(log n) | O(log n) | Red-black tree |

# שאלה 2

1. DFS – Depth-first search – הינו אלגוריתם לסריקת גרף. האלגוריתם מתחיל מצומת התחלה, בוחן את הילדים של הצומת כאשר הוא בוחר בכל פעם את הילד בעל המחיר הגבוה ביותר (ואם יש באותו מחיר אז בסדר אלפבתי נתוך יותר) וממשיך לסרוק ממנו. כאשר האלגוריתם מגיע לצומת ללא ילדים, הוא חוזר לצומת ממנו הגיע לילד וממשיך לילד הבא שאותו לא ביקר. האלגוריתם עוצר כאשר הוא מגיע לצומת סוף או כאשר הוא סיים לעבור על כל הצמתים בגרף.

נדגים את פעולתו על הגרף הבא כאשר נסרוק את המסלול מ A ל E:



לפני הרצת האלגוריתם מרוקנים את visited וצובעים את כל הצמתים ללבן.

שלב א - צובעים את A לאפור ומוסיפים אותו ל visited. מחלצים שהבנים של A שהם B, G, D ובוחרים להמשיך עם B.

שלב ב – צובעים את B באפור ומוסיפים אותו ל visited. מחלצים שהבנים הם A ו E וממשיכים ל E (כי ב A כבר היינו).

שלב ג – צובעים את E באפור ומוסיפים אותו ל visited. מכיוון ש E היא הצומת סוף אז מחזירים true. משם הערך true מחלחל האחורה לאב B ולאב A.

בסוף האלגוריתם צובעים את הצמתים חזרה ללבן.

בסוף התהליך המסלול המוחזר הוא visited = A, B, E.

הרצת האלגוריתם עבור A-> D:

לפני הרצת האלגוריתם מרוקנים את visited וצובעים את כל הצמתים ללבן.

שלב א - צובעים את A לאפור ומוסיפים אותו ל visited. מחלצים שהבנים של A שהם B, G, D ובוחרים להמשיך עם B.

שלב ב – צובעים את B באפור ומוסיפים אותו ל visited. מחלצים שהבנים הם A ו E וממשיכים ל E (כי ב A כבר היינו).

שלב ג – צובעים את E באפור ומוסיפים אותו ל visited. מכיוון ש E אין בנים אז צובעים לשחור ומחזירים false.

שלב ד – צובעים את B בשחור ומחזירים false.

שלב ה – עוברים ל G (הבן הבא של A). צובעים באפור ומוסיפים ל visited ומלחצים את הבנים R, D.

שלב ו – צובעים את R באפור ומוסיפים אותו ל visited. מכיוון ש R אין בנים אז צובעים לשחור ומחזירים false.

שלב ז – צובעים את D באפור ומוסיפים ל visited. מכיוון שזהו צומת הסוף, מחזירים true. משם הערך true מחלחל האחורה לאב G ולאב A.

בסוף האלגוריתם צובעים את הצמתים חזרה ללבן.

בסוף התהליך המסלול המוחזר הוא visited = A, B, E, G, R, D.

הרצת האלגוריתם עבור B:

לפני הרצת האלגוריתם מרוקנים את visited וצובעים את כל הצמתים ללבן.

שלב א - צובעים את B לאפור ומוסיפים אותו ל visited. מחלצים שהבנים של B שהם E, A ובוחרים להמשיך עם E.

שלב ג – צובעים את E באפור ומוסיפים אותו ל visited. מכיוון ש E אין בנים אז צובעים לשחור ומחזירים false.

שלב ד – עוברים ל A (הבן הבא של B). צובעים לאפור ומוסיפים ל visited. מחלצים שהבנים של A הם B, G, D. ממשיכים ל G (כי ב B כבר היינו).

שלב ה –צובעים את G באפור ומוסיפים ל visited ומלחצים את הבנים R, D.

שלב ו – צובעים את R באפור ומוסיפים אותו ל visited. מכיוון ש R אין בנים אז צובעים לשחור ומחזירים false.

שלב ז – צובעים את D באפור ומוסיפים ל visited. מכיוון ש D אין בנים אז צובעים לשחור ומחזירים false.

שלב ח – צובעים את G בשחור ומחזירים false.

שלב ט – עוברים לילד הבא של A שהוא D אך מכיוון שכבר היינו בו אז צובעים את A בשחור ומחזירים false.

בסוף האלגוריתם צובעים את הצמתים חזרה ללבן.

בסוף התהליך המסלול המוחזר הוא visited = B, E, A, G, R, D.

ג. כדי לבדוק האם יש קשתות אחוריות בגרף, נסרוק בתחילת הפעלת המתודה DFS את צבעי הילדים. כל ילד בצבע אפור הוא סימן לאב קדמון (כי צומת בצבע אפור הוא צומת שממנו הגיעו לילדים). לכל צומת נוסיף שדה שמציין כמה קשתות אחוריות יש ונתחשב בזה ב NodeCountingPath.