מבני נתונים תרגיל מעשי 2

מחלקת Graph:

הגדרנו 4 שדות למחלקה:

1. ערימת מקסימום – maxheap - ערימת מקסימום שאיבריה מכילים מפתח שהוא משקל הסביבה של צומת (סכום משקלו ומשקל שכניו) וערך שהוא הצומת עצמו.
2. טבלת האש – HashTable - טבלת האש שמומשה בשיטת ה chaining כך שפונקציית האש הינה universal hashing קרי, נלקחה ממשפחת פונקציות המודולו באקראי כאשר הראשוני הרלוונטי הוא p = 10^9 + 9 .
3. מספר הצמתים – numNodes – שדה שסופר את כמות הצמתים שבגרף.
4. מספר הקשתות – numEdges – שדה שסופר את כמות הקשתות שבגרף.

הסבר על הפונקציות:

1. Graph(Node [] nodes) – constructor של גרף, מקבל מערך של צמתים מאורך N (כאשר כל אוביקט צומת מכיל את המזהה שלו ואת המשקל שלו) ומאתחלת את מבנה הנתונים כגרף עם N צמתים אלו וללא קשתות. סיבוכיות הזמן הינה , זאת מכיוון שלכל צומת אנו יוצרים צמתים חדשים מהסוג שלו לערימה ולטבלת האש, ומכניסים אותו ב בתוחלת, ומכיוון שאנו זאת לכל צומת, נעשה זאת N פעמים ולכן בתוחלת כנדרש.
2. addEdge(int node1\_id, int node2\_id) – מקבלת שני מזהים של צמתים. אם שני הצמתים נמצאים בגרף, הפעולה מוסיפה קשת ביניהם. אחרת, היא איננה עושה דבר.   
   הפונקציה מחזירה true אם נוספה קשת ואחרת מחזירה false.

אם הצמתים נמצאים, מכניסים כל צומת לרשימת שכנויות של הצומת השני, ומעדכנים בערימת המקסימום את ערכי המשקלים ומחזירים true .

סיבוכיות הזמן הינה זאת מכיוון שפעולת ההכנסה לערימת המקסימום עלתה .

(פעולת מציאת הצמתים ועדכונם עולה בתוחלת. )

1. maxNeighborhoodWeight() - מחזירה צומת כך שמשקל הסביבה שלו מקסימלי. אם בגרף אין צמתים, מחזירה null. מימוש הפונקצייה היא פשוט שליפה מהירה משדה של MaxHeap ולכן .
2. getNeighborhoodWeight(int node\_id) – בהינתן מזהה קודקוד node\_id, מחזיר את משקל הסביבה של הקודקוד בעל מזהה זה אם הוא בגרף. אם אין קודקוד בעל מזהה זה בגרף, מחזירה . הפונקצייה קוראת לפונקציית הhash ומחזירה את הצומת המתקבל מהפונקציה אם הוא קיים. סיבוכיות הזמן היא: .
3. deleteNode(int node\_id) – מוחקת את הקודקוד בעל המזהה node\_id (ואת כל הקשתות המערבות אותו) מהגרף, אם הוא קיים (אחרת, הפעולה לא מבצעת דבר).   
   הפונקציה מחזירה true אם נמחק צומת, ואחרת false.

תחילה אנו מוחקים את הצומת הנתון מטבלת האש ב בתוחלת, לאחר מכן אנו מוחקים אותו מערימת המקסימום ב , לאחר מכן, אנו עוברים בלולאה על כל שכניו של הקודקוד הנתון ומורידים את המפתח שלהם בערימת המקסימום במשקלו של הקודקוד אותו אנו מוחקים, דבר אשר לוקח , ומוחקים אותו מרשימת השכנויות המקושרת שלהם. כמובן שמעדכנים שדות ופינטרים בזמן קבוע. ולכן בסה"כ , כאשר היא דרגתו של הצומת (כלומר, מספר שכניו בגרף כרגע).

1. getNumNodes() מחזירה את מספר הצמתים שנמצאים בגרף כרגע. שמרנו את מספר זה בשדה ועדכנו אותו לפי הצורך, ולכן זהו פשוט שליפה מהירה. מכאן כנדרש.
2. getNumEdges() מחזירה את מספר הקשתות שנמצאות בגרף כרגע. שמרנו את מספר זה בשדה ועדכנו אותו לפי הצורך, ולכן זהו פשוט שליפה מהירה. מכאן כנדרש.

מחלקת Node :

הגדרנו שני שדות למחלקה:

1. תעודת זהות – id - מספר מזהה לצמתים, מספר שלם ואי שלילי, לכל צומת יש תעודת זהות מיוחדת.
2. משקל- weight – המשקל של הצומת בגרף, מספר שלם ואי שלילי.

הסבר על הפונקציות:

1. Node(int id, int weight) – בנאי אשר יוצר צומת של גרף עם מזהה id ומשקל weight. .
2. getId()– מחזיר את המזהה של הצומת, .
3. getWeight ()– מחזיר את המשקל של הצומת, .

מחלקת ערימת המקסימום MaxHeap:

הגדרנו שלוש שדות למחלקה זאת:

1. heapNode [] arr - מערך של צמתים מסוג heapnode (פירוט המחלקה שלהם בהמשך).
2. lastIndex - מציין את המיקום של האיבר האחרון במערך ערימת המקסימום.

הסבר על הפונקציות:

1. MaxHeap(heapNode [] arr) - בנאי למחלקה, מקבל מערך של צמתים מסוג heapnode, מכניס אותו זמנית לשדה מערך הצמתים, וקורא לפונקציית העזר שמתקנת את המערך לערימת מקסימום ב כפי שראינו בכתה. בנוסף מעדכנים את שדה האינדקס להיות האינדקס התקין כרגע. סה"כ .
2. DeleteMax() - מוחקת את האיבר המקסימלי מהערימה, מחליפה אותו באינדקס האחרון של הרשימה, מורידה את אורך הרשימה ב -1, וקוראת לפונקציית העזר heapify\_down על האיבר המוחלף. שרצה ב .
3. Max() - מחזירה את האיבר המקסימלי במערך, .
4. decrease\_key(int i, int sum) - משנה את ערך המפתח של צומת בערימת המינימום, וקוראת לאחת מפונקציות העזר heapify\_down/ heapify\_up שרצות ב.
5. Delete(int i) - בהנתן אינדקס(משדה המערך) למחיקה מוחקת אותו מהמערך באופן הבא- מחליפה אותו באינדקס האחרון של הרשימה, מורידה את אורך הרשימה ב -1, וקוראת לפונקציית העזר heapify\_down/ heapify\_up על האיבר המוחלף. שרצות ב . [בנוסף יש קריאה לפונקציות העזר switchNodes ו- getParentIndex(i) שרצות בזמן קבוע.]
6. heapify\_up(int i) - פעפוע כלפי מעלה ע"י שימור ערימת המקסימום כפי שראינו בכתה, קוראת ל switchNodes, ורצה בסה"כ ב.
7. heapify\_down(int i)- פעפוע כלפי מטה ע"י שימור ערימת המקסימום כפי שראינו בכתה, קוראת ל switchNodes, ורצה בסה"כ ב.
8. getLeftIndex(int i) - מחזירה את האינדקס של הבן השמאלי של צומת נתון (ע"י האינדקס שלו במערך), רצה ב .
9. getRightIndex(int i) - מחזירה את האינדקס של הבן השמאלי של צומת נתון (ע"י האינדקס שלו במערך), רצה ב .
10. getParentIndex(int i) - מחזירה את האינדקס של ההורה של צומת נתון (ע"י האינדקס שלו במערך),כאשר יוצאים מנקודת הנחה שההורה של האיבר המקסימלי זה הוא עצמו. רצה ב .
11. switchNodes(int i, int parent) פונקצייה זו מחליפה בין זוג צמתים בערימת המקסימום ע"י עדכון מצביעים, רצה ב .
12. array\_to\_MaxHeap() - פונקצייה זו הופכת מערך של צמתים לערימת מקסימום ע"י אלגוריתם שראינו בכתה, היא רצה בלולאה על הצמתים אשר ברמה הלפני האחרונה (עד לאחרונה) ומתקנת אותם ע"י שימוש בפונקציית העזר heapify\_down שרצה ב ב, הראנו בכתה שפונקצייה זו רצה ב כש N מספר הצמתים שבאתחול הערימה.

#TODO