שאלת חזרה בייצוג וברשימה

מפת העיר (גרסה נוספת)

מטרות

תרגול בייצוג של טיפוס נתונים מופשט

תרגול מערכים דו מימדיים

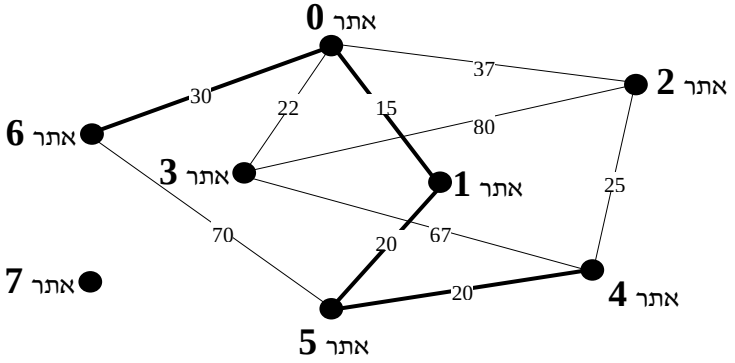
רמת השאלה

בינונית ומעלה.

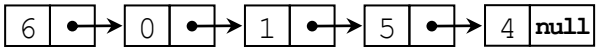
השאלה

במפה של עיר תיירותית ניתן למצוא מידע רב על אתרים שונים, מערכת הכבישים, תחנות אוטובוס ורכבות, מלונות, מקומות בילוי, מוסדות ומידע נוסף. בשאלה זו נתמקד ב**מפת-עיר** בה יש מידע על האתרים חשובים, הכבישים הישירים המחברים ביניהם ואורכו של כל כביש. מפת-העיר תוגדר על ידי המחלקה **CityMap** שתאפשר לבנות מפות ערים בהן יש לכל היותר 50 אתרים. כל אתר יקבל מספר סידורי החל ב-0 ועד 49, ואורכי הכבישים ימדדו בק״מ.

דוגמה: במפת העיר ״שיממון״ (ראו איור) יש 8 אתרים הממוספרים מ-0 ועד-7 (עיגולים שחורים), ניתן לראות את הכבישים המחברים ביניהם (הקווים בין העיגולים) ואורכו של כל כביש (מספר באמצע הקו). ניתן לראות, שאורך הכביש המחבר בין אתר מס׳ **6** לאתר מס׳ **5**, הוא 70 ק״מ. לעומת זאת, לא קיים כביש ישיר המחבר בין אתר מס׳ **6** לאתר מס׳ **4**, אך בכל זאת ניתן להגיע מהראשון לשני דרך כבישים אחרים במפה.



מסלול (path) במפה הוא כביש או רצף כבישים המחבר בין שני אתרים. את המסלול ניתן לתאר בעזרת רשימה של מספרי האתרים בהם עובר המסלול, כאשר האיבר הראשון (מספרו של האתר הראשון במסלול) והאחרון (מספרו של האתר האחרון במסלול) ברשימה, מציינים את תחילתו וסופו של המסלול בהתאמה. לדוגמה, המסלול המודגש במפה (המחבר בין אתר מס׳ 6 לאתר מס׳ 4) ועובר דרך האתרים שמספריהם: 0, 1, 5 יתואר באמצעות הרשימה הבאה:



הערה: יתכן מצב בו לא קיים מסלול בין אתר מסוים לאתר אחר – לדוגמה לא קיים במפה מסלול מאתר מס׳ 5 לאתר מס׳ 7. במקרה זה הרשימה שתתאר מצב זה () היא הרשימה הריקה.

© כל הזכויות שמורות למשרד החינוך ולאוניברסיטה העברית בירושלים

במפה קיימים מסלולים נוספים המחברים בין אתר מס׳ 6 לאתר מס׳ 4, אך המסלול המודגש במפה (והמתואר על ידי הרשימה) הוא המסלול הקצר ביותר - כי **אורך המסלול** (**legnth**), המחושב כסכום אורכי הכבישים במסלול, הוא: **85** = 20 + 20 + 15 + 30, והוא הקצר ביותר מבין כל המסלולים הנוספים.

המחלקה CityMap

CityMap (**int** numOfSites)	הפעולה בונה מפת-עיר בה יהיו לכל היותר numOfSites אתרים וללא כבישים. הנחה: numOfSites גדול מאפס
double getDistance **(int** s1**, int** s2)	הפעולה מחזירה את אורך הכביש הישיר בין אתר s1 לאתר s2. אם אין כביש ישיר בין s1 ל-s2 יוחזר הערך אפס. הנחה: s1 ו-s2 אתרים קיימים במפה
void setDistance (**int** s1, **int** s2, **double** dist)	הפעולה ״מחברת״ בין אתר s1 לאתר s2 בעזרת כביש ישיר שאורכו dist ק״מ. הנחות: s1 ו-s2 אתרים קיימים במפה, s1≠s2, ו- dist גדול מאפס
List<Integer> getShortestPath (**int** s1, **int** s2)	הפעולה מחזירה רשימת מספרים המתארת את המסלול הקצר ביותר המחבר בין אתר מס׳ s1 לאתר מס׳ s2. הנחות: s1 ו-s2 אתרים קיימים במפה ו- s1≠s2

א. כתבו תוכנית ראשית שתבנה את מפת העיר ״שיממון״ המתוארת באיור לעיל.

ב. כתבו את המחלקה CityMap, מבלי לממש את הפעולה getShortestPath(...).

ג. כתבו את הפעולה הבאה:

```
public static double getShortestPathDistance(CityMap map, int s1, int s2)
```

הפעולה מחזירה את אורך המסלול הקצר ביותר המחבר בין אתר s1 לאתר s2 במפת העיר map. הניחו שהאתרים s1 ו-s2 קיימים במפה.

הנחיות מיוחדות

- שאלה זו מופיעה בנוסח שונה מעט במאגר תחת השם: מפת ערים. בחנו מהו הנוסח המתאים לכם. תוכן השאלה כמעט זהה.
- קושי התרגיל הוא בייצוג המפה – מאחר במספר האתרים הוא לכל היותר 50, ובין כל אתר לאתר אחר ייתכן ויש כביש המחבר אותם – ניתן להשתמש במערך דו-מימדי ריבועי לייצוג המפה.
- האינדקסים של המערך מייצגים את מספרי האתרים. תאי המערך ייצגו את אורכי הכבישים בין אתר לאתר. במידה ולא קיים כביש, ערך התא יכול להיות 0 או מספר שלילי כלשהו.
- יש לשים לב לעובדה שבין שני אתרים המחוברים באופן ישיר יש רק כביש מחבר אחד. לכן המטריצה שלנו סימטרית ביחס לאלכסון הראשי. כלומר, הערך שנמצא במערך בתא בשורה 2 עמודה 3 (המייצג את אורך הכביש בין אתר 2 לאתר 3) זהה לערך שנמצא בתא במערך בשורה 3 עמודה 2. המשמעות היא שניתן לוותר על התאים במערך שנמצאים מעל האלכסון הראשי.

© כל הזכויות שמורות למשרד החינוך ולאוניברסיטה העברית בירושלים

- במידה ונרצה להרחיב את התרגיל ולהגיד שייתכן מצב בו יש שני כבישים שונים (אחד לכל כיוון) בין אתר לאתר, אז נצטרך להשתמש בכל תאי המטריצה.
- כעת, באמצעות ייצוג זה, מימוש הפעולות פשוט. ראו פתרון. שימו לב שאת הפעולה getShortestPath(...) אנו לא מבקשים לממש מהסיבה שהיא מאוד מורכבת מבחינה אלגוריתמית. אבל היא מעניינת ומתאימה לתלמידים חזקים.

פתרון

א. מפת העיר "שיממון" כפי שמתוארת באיור:

```
public class BuildCityMap
{
	public static void main(String[] args)
	{
		CityMap shimamon = new CityMap(8);

		shimamon.setDistance(0, 1, 15);
		shimamon.setDistance(0, 2, 37);
		shimamon.setDistance(0, 3, 22);
		shimamon.setDistance(0, 6, 30);

		shimamon.setDistance(1, 5, 20);

		shimamon.setDistance(2, 3, 80);
		shimamon.setDistance(2, 4, 25);

		shimamon.setDistance(3, 4, 67);

		shimamon.setDistance(4, 5, 20);

		shimamon.setDistance(5, 6, 70);
	}
}
```

ב. המחלקה CityMap:

```
public class CityMap
{
	private double[][] neighbors;

	public CityMap(int numOfSites)
	{
		this.neighbors = new double[numOfSites][numOfSites];
	}

	public void setDistance(int s1, int s2, double dist)
	{
		this.neighbors[s1][s2] = dist;
	}

	public double getDistance(int s1, int s2)
	{
		return this.neighbors[s1][s2];
	}
```


© כל הזכויות שמורות למשרד החינוך ולאוניברסיטה העברית בירושלים

```
	public List<Integer> getShortestPath(int s1, int s2)
	{
		...
	}
}
```

ג. מימוש הפעולה getShortestPathDistance:

```
public static double getShortestPathDistance(CityMap map, int s1, int s2)
{
	List<Integer> lst = map.getShortestPath(s1, s2);
	if(lst.isEmpty())
		return 0;

	double dist = 0;
	Node<Integer> p1 = lst.getFirst();
	Node<Integer> p2 = p1.getNext();

	while(p2 != null)
	{
		s1 = p1.getInfo();
		s2 = p2.getInfo();
		dist = dist + map.getDistance(s1, s2);
		p1 = p2;
		p2 = p2.getNext();
	}

	return dist;
}
```

שימו לב: הרשימה המוחזרת מהפעולה `getShortestPath(...)` יכולה להיות ריקה.

© כל הזכויות שמורות למשרד החינוך ולאוניברסיטה העברית בירושלים