חידה לחימום

נתונה התכנית הבאה, שאמורה להדפיס 5 כוכביות, אולם היא אינה מבצעת את משימתה:

```c
#include <stdio.h>

int main()

{

 int i, n = 5;

 for (i = 0; i + n; i--)

     printf ("*");

 return 0;

}
```

מצאו שלוש דרכים שונות לתקן את התכנית כך שתבצע את הנדרש. תיקון התכנית פירושו – הוספת תו אחד או שינוי של תו בודד.

מבני נתונים ויעילות אלגוריתמים

(16.2.2015)

1. אלגוריתם בלמן-פורד (Bellman-Ford) למציאת מסלול קצר ביותר

אלגוריתם בלמן-פורד מקבל כקלט גרף מכוון וממושקל G, וקודקוד מקור s, ופותר את בעיית המסלול הקצר ביותר מ-s לכל אחד מקודקודי הגרף. סיבוכיותו גרועה מזו של דייקסטרה, אלא שלהבדיל מדייקסטרה, אלגוריתם בלמן-פורד מתאים לשימוש גם במקרה ובו המשקלים על הקשתות עשויים להיות מספרים שליליים.

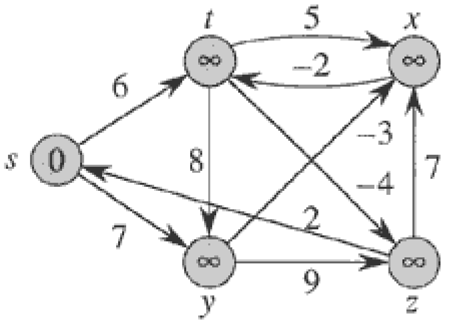
אם הגרף מכיל מעגל שלילי (כלומר, מעגל מכוון שסכום המשקלים על קשתותיו הוא מספר שלילי) אז אין פתרון לבעיית המסלול הקצר ביותר, שכן תמיד ניתן יהיה להמשיך ולהסתובב במעגל הזה, ולקבל מסלולים בעלי משקל שלילי קטן יותר ויותר.

במידה והאלגוריתם מופעל על גרף מכוון G שיש בו מעגל שלילי הניתן להגעה מקודקוד המקור s, אלגוריתם בלמן-פורד יזהה זאת ויחזיר FALSE. במידה ואין מעגל כזה, האלגוריתם יחשב עבור כל קודקוד את המסלול הקצר ביותר מהמקור s ועד אליו, ויחזיר TRUE.

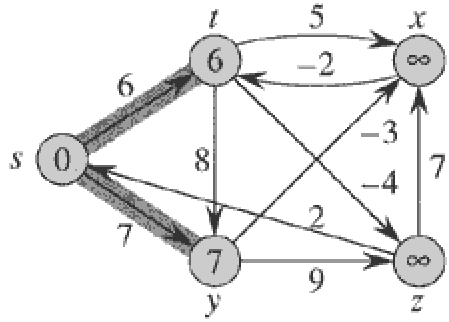
```
BELLMAN-FORD(G,s)
1     for each vertex v∈V
2         do v.distance ← ∞
3            v.previous ← NULL
4     s.distance ← 0
5     for i ← 1 to |V|-1
6         do for each edge (u,v)∈E
7                do if v.distance > u.distance + w(u,v)
8                      then v.distance ← u.distance + w(u,v)
9                           v.previous ← u
10    for each edge (u,v)∈E
11        do if v.distance > u.distance + w(u,v)
12              then return FALSE
13    return TRUE
```

סיבוכיות זמן הריצה של האיתחול בשורות 1-4 היא $\Theta(|V|)$, הלולאה החיצונית בשורה 5 מבצעת $|V|-1$ איטרציות, והלולאה הפנימית בשורה 6 מבצעת $|E|$ איטרציות. בסך הכול, צמד הלולאות הזה מבצע $\Theta(|V||E|)$ פעולות. הלולאה בשורה 10 מבצעת $|E|$ איטרציות, אך זה מתגמד לעומת $\Theta(|V||E|)$, שזו סיבוכיות האלגוריתם.

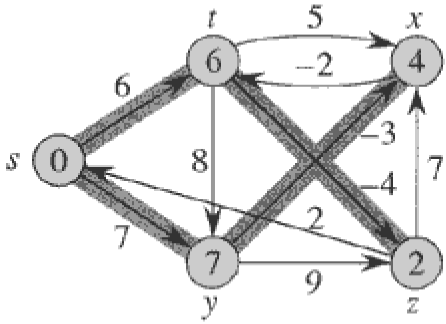
להלן תרשימים המתארים את פעולתו של בלמן-פורד על גרף מכוון וממושקל. נניח כי הקשתות נסרקות לפי הסדר הבא (משמאל לימין): (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)


t
5
x
∞
−2
∞
6
s
0
8
−3
−4
7
2
7
∞
9
∞
y
z

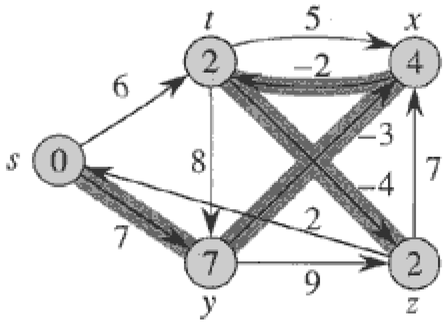
(a)


t
5
x
6
−2
∞
6
s
0
8
−3
−4
7
2
7
7
9
∞
y
z

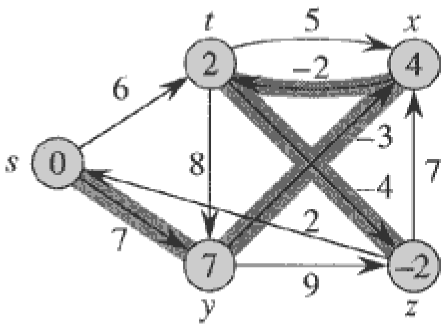
(b)


t
5
x
6
−2
4
6
s
0
8
−3
−4
7
2
7
7
9
2
y
z

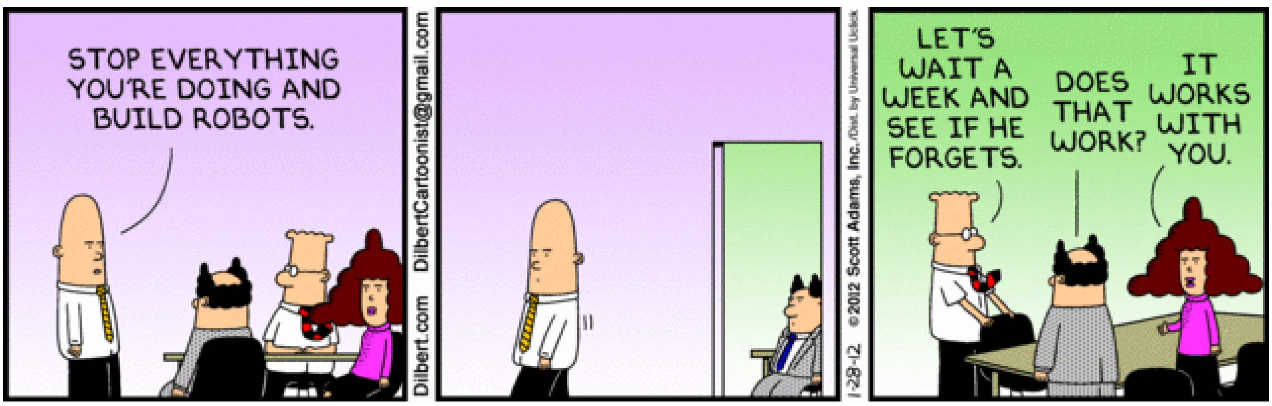
(c)


t
5
x
2
−2
4
6
s
0
8
−3
−4
7
2
7
7
9
2
y
z

(d)


t
5
x
2
−2
4
6
s
0
8
−3
−4
7
2
7
7
9
−2
y
z

(e)


STOP EVERYTHING YOU'RE DOING AND BUILD ROBOTS.
Dilbert.com DilbertCartoonist@gmail.com
1-28-12 © 2012 Scott Adams, Inc./Dist. by Universal Uclick
LET'S WAIT A WEEK AND SEE IF HE FORGETS.
DOES THAT WORK?
IT WORKS WITH YOU.

2. מיון בסיס (Radix Sort)

מיון בסיס הוא דוגמא נוספת לאלגוריתם מיון שאינו מבוסס השוואות. הוא עובד על-ידי כך שהוא ממיין את המספרים ספרה-ספרה: תחילה הוא ממיין על-פי הספרה הימנית ביותר (הכי פחות משמעותית), לאחר מכן לפי הספרה שלשמאלה, לאחר מכן – לפי הספרה שלשמאלה, עד שלבסוף הוא ממיין לפי הספרה השמאלית ביותר (הכי משמעותית).

דוגמא לפעולת מיון בסיס:

329		720		720		329
457		355		329		355
657		436		436		436
839	→	457	→	839	→	457
436		657		355		657
720		329		457		720
355		839		657		839

נשים לב שכשממיינים לפי ספרה מסוימת, אם יש שני מספרים שהספרה הזו זהה אצלם – אז הסדר היחסי שלהם נשמר. כלומר, כאשר אנחנו ממיינים ספרות, יש להשתמש באלגוריתם מיון שהוא יציב (stable).

האלגוריתם מקבל כקלט מערך A, המכיל n איברים, שכל אחד מהם הוא מספר בן d ספרות:

```
RADIX-SORT (A,d)
1     for i ← 1 to d
2         do use a stable sort to sort array A on digit i
```

בשורה 2 – כדאי להשתמש במיון מנייה (כי טווח הספרות הוא סופי, אינו גדול וידוע מראש). אם נניח כי טווח הספרות גודלו k, אז כל זימון של מנייה ספירה (Counting Sort) בשורה 2 יארך זמן של $\Theta(n+k)$. מכיוון שמזמינים שורה זו d פעמים, הסיבוכיות הכוללת של מיון בסיס היא: $\Theta(d\cdot(n+k))$. אם d קבוע ו-$k=O(n)$, אז מיון בסיס רץ בזמן לינארי.