תכנון וניתוח אלגוריתמים
תרגיל 6

מטריצת מסלולים - סגור טרנזיטיבי

שאלה 1

- א.

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- ב.

$$A^2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$A^3 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$A^4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$P = A + A^2 + A^3 + A^4 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

שאלה 2

- $P_K[i][j]$= True **אם ורק אם** קיים מסלול מ-i אל j שאינו עובר דרך קודקוד שמספרו גדול מ-K.
- כלומר המסלול עובר דרך הקודקודים ששייכים לתחום 1..K.
- א. אם $P_K[i][j]$=True אזי גם $P_{K+1}[i][j]$=True מכיוון שנתון שקיים מסלול מ-i אל j שעובר דרך הקודקודים ששייכים לתחום 1..K.

- אפילו אם לא קיים מסלול מ-i אל-j העובד דרך הקודקוד K+1 זה בסדר, מכיוון ש-$P_{K+1}[i][j]=True$ אם קיים מסלול מ-i אל j שאינו עובר דרך קודקוד שמספרו גדול מ-K+1.
- כלומר, אם $P_K[i][j]$ בעל ערך True, זה מבטיח שיש מסלול כזה.

- ב. המצב היחיד שבו $P_{K+1}[i][j]=True$, בעוד $P_K[i][j]=False$, הוא כאשר קיים מסלול מ-i אל j העובר דרך הקודקוד K+1 אך לא קיים מסלול מ-i אל j העובר רק דרך הקודקודים השייכים לתחום 1..K.
- פירוש הדבר, במקרה כזה, שחייבים להיות :

- מסלול מ-i אל K+1 העובר דרך הקודקודים ששייכים לתחום 1..K
- ובנוסף מסלול מ-K+1 ל-j העובר גם דרך הקודקודים ששייכים לתחום 1..K.
- סופית:
- $P_{K+1}[i][j]=P_K[i][j]$ or $(P_K[i][K+1]$and$P_K[K+1][j])$

- ג. בעבור A, שהינה מטריצת סמיכות, המייצגת גרף נתון מתקיים $A=P_O[i][j]$, מכיוון שמטריצה זו הינה מטריצת מסלולים באורך 1 (בעצם קשתות) ומסלולים אלו אינם עוברים דרך קודקודים שמספרם גדול מאפס.
- כאן אני מניחים שאם בגרף יש n קודקודים אזי הם ממוספרים אקראית מ-1 עד n.

- הערה: $P_n[i][j] = P[i][j]$ מאחר שקיים מסלול מ-i ל-j אם ורק אם קיים מסלול כזה העובר רק דרך הקודקודים ששייכים לתחום 1..n.
- לא יכול להיות מסלול שיעבור דרך קודקוד כלשהו שמספרו גדול מ-n.

- ד. לאור האמור לעיל, להלן אלגוריתם לקבלת מטריצה $P_K[i][j]$:

- for i=1 to n do
  - for j=1 to n do
    - $P_K[i][j] \leftarrow P_{K-1}[i][j]$ or $(P_{K-1}[i][k]$ and $P_{K-1}[k][j])$

- ניתן לפשט את האלגוריתם הזה כך:
- לכל i,j:

- $P_K \leftarrow P_{K-1}$
- for i=1 to n do
  - if $P_{K-1}[i][k]$ then
    - for j=1 to n do
      - $P_K[i][j] \leftarrow P_{K-1}[i][j]$ or $P_{K-1}[k][j]$

- ה. להלן שגרה בשפת C המחשבת בגישה זו את הסגור הטרנזיטיבי.

```
void maslul(adjmatrix adj, adjmatrix P)
/* מייצגת מטריצת סמיכות adj */
}
   int i,j,k;
   p←adj; /* put adj in p */
   for(k=1; k<=n; k++)
       for(i=1; i<=n; i++)
            if(P[i][k])
                 for(j=1; j<=n; j++)
                      P[i][j] = P[i][j]||P[k][j];
}
```

- קל לראות כי זמן הריצה הוא $O(N^3)$, כאשר $n=|V|$.

שאלה 3

- א. מאחר שהגרף G אינו מכיל מעגל, חייב להיות לפחות קודקוד אחד ב-G שאין לו קודקוד קודם.
- כדי להשתכנע בכך: נניח שלכל קודקוד יש קודקוד קודם לו. נבחר אקראית קודקוד a שיש לו קודם b. b שונה מ-a, וכי אחרת היה בגרף מעגל מ-a לעצמו.
- מאחר שלכל קודקוד יש קודקוד קודם לו אז גם ל-b יש קודקוד קודם לו, נניח קודקוד c השונה מ-a ו-b (מאותו נימוק).

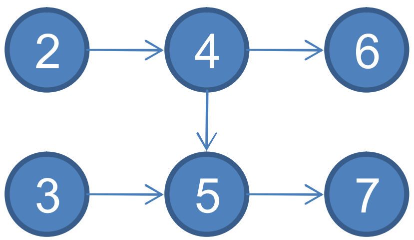
- אם נמשיך כך נקבל סדרה של קודקודים שונים:

  a b c x y …

- מכיוון שבגרף יש מס' סופי של קודקודים, הרי שבסופו של דבר יהיו בסדרה זו שני קודקודים שווים ואולם אז יהיה מעגל בסתירה להנחה הראשונית שבגרף אין מעגל.

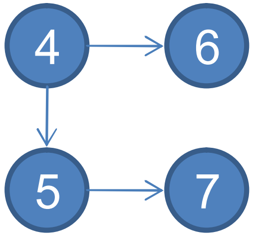
- מכאן נסיק כי <u>ההנחה</u>, שלכל קודקוד יש קודקוד קודם, היתה <u>מוטעית</u> .
- על כן יש <u>לפחות קודקוד אחד</u> שאין לו קודקוד קודם.

- ב. בתרשים הנתון יש קודקוד שמספרו 1 שאין לו קודקוד קודם.
- ג. ביחידת זמן ראשונה ניתן לבצע את המטלה '1', שהינה קודקוד 1, כי אין לו מקדימים.
- ד. נסיר מהגרף את הקודקוד 1 ואת הקשתות הנוגעות בו. אז נקבל:

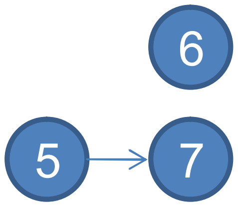


- ה. עתה לקודקודים 2 ו-3 אין קודקוד קודם.
- לכן ביחידת זמן שניה ניתן לבצע את התת מטלות שהם מייצגים: מטלות '2' ו-'3', בעת ובעונה אחת בלא שיהיה צורך להמתין להשלמת תת מטלה אחרת כלשהיא.
- כל תת מטלה אחרת חייבת להמתין עד שלפחות '2' או '3' יסתיימו.

- ו. עתה נסיר מהגרף את הקודקודים 2 ו-3 ונקבל:
- (כולל הקשתות הנוגעות בהם)



- לכן ביחידת זמן שלישית ניתן לבצע רק את 4.
- ז. עתה נסיר מהגרף את 4 ונקבל:



- לכן ביחידת זמן רביעית ניתן לבצע את 5 ו-6 בעת ובעונה אחת.

- ח. עתה נסיר מהגרף את 5 ו-6 אז נקבל:

- לכן ביחידת זמן חמישית ניתן לבצע רק את 7.
- לסיכום להלן אלגוריתם לטיפול בבעיה הנתונה:

- ט. <u>אלגוריתם</u>
- ***<u>צעד 1</u>*** קבע את סדר הקדימויות של תת-המטלות ובנה את הגרף המתאים.
- ***<u>צעד 2</u>*** כל עוד הגרף הוא לא ריק בצע:
  - ***2.1*** אתר את כל הקודקודים שאין להם קודקוד קודם (כלומר דרגת כניסתם שווה לאפס).
  - ***2.2*** בעבור קבוצה כזו של קודקודים הדפס שהם ניתנים לביצוע בעת ובעונה אחת.

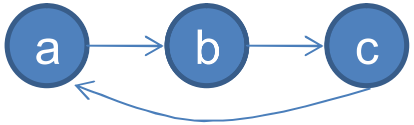
– ***2.3*** בעבור כל קודקוד w ששייך לקבוצה כזו שמצאנו בצעד 2.1 בצע:

- ***2.3.1*** להסיר את כל הקשתות הנוגעות לקודקוד w.
- ***2.3.2*** הסר את הקודקוד w מהגרף.

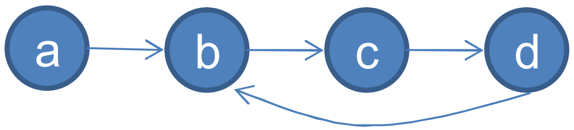
- י. כל שורת קלט מכילה שמות של שתי מטלות שהראשונה צריכה להיתבצע לפני השניה.
- יא. 1. אם מס' הקודקודים לא ידוע בגרף, אז נייצג את הגרף באמצעות מבנה רב מקושר.
- יא. 2. איך ניתן לדעת שלקודקוד אין קודקוד קודם? ניתן לשמור בכל קודקוד שדה Count שבו ירשם מספר הקודקודים הקודמים לו.

- איננו מעוניינים לדעת אילו קודקודים קודמים לקודקוד נתון, אלא רק כמה כאלה יש.
- 3. אם שדה Count של קודקוד כלשהו שווה ל-0, פירוש הדבר שלקודקוד זה אין קודקוד קודם וניתן לבצעו ביחידת זמן הנתונה.
- 4. כל פעם שקודקוד x עובר לפלט (המטלה מתבצעת), יש לעבור על רשימת הסמיכות של הקשתות שלו ולהפחית את שדה ה-Count של כל קודקוד הסמוך ל-x ב-1.

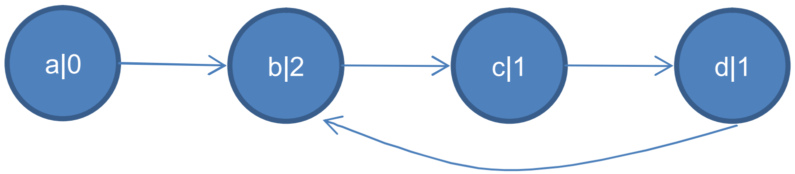
- 5. בכל פעם שבודקים את התת-מטלות בפרק זמן מסויים, יש לעבור על רשימת הקודקודים שנותרו בגרף כדי לאתר את אלה ששדה ה-Count שלהם הוא 0.
- אלה יצאו לרשימת הפלט כתת-מטלות שניתנות לביצוע בעת ובעונה אחת – בפרק זמן זה.
- יב. גרף אשר מתאר את סדר הקדימויות, אסור שיכיל מעגל, מכיוון שאם בגרף יש מעגל כגון:



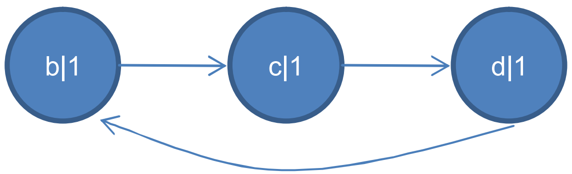
- a קודם ל-b, b קודם ל-c ו-c קודם ל-a, פירושו a קודם ל-a (לעצמו) שזה לא יתכן.
- יג. אם ביחידת זמן מסויימת אין אף קודקוד שה-Count שלהם שווה לאפס, פירוש הדבר שבגרף קיים מעגל.
- כך למשל בגרף הבא:



- נציין בעבור כל קודקוד גם את ה-Count שלו:



- ביחידת זמן אחת נבצע את a, כי ה-Count שלו שווה לאפס. עתה מסירים אותו מהגרף ואת הקשתות היוצאות מ-a. כמו כן מפחיתים את שדה ה-Count של b ב-1 ואז נקבל:



- עתה ביחידת זמן שניה אין אף קודקוד שה-Count שלו שווה ל-0, פירוש הדבר שבגרף יש מעגל.
- יד. <u>אלגוריתם (ולא תכנית)</u>

- ***Step 1*** period=0
- ***Step 2*** while not is_empty(graph) do
  - ***Step 2.1*** period=period+1
  - ***Step 2.2*** outp=emptylist()
  - ***Step 2.3*** p=graph

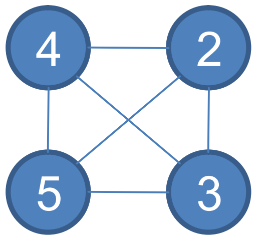
- ***Step 2.4*** while not is_empty(p)
  - ***Step 2.4.1*** if count(p) = 0 then
    - הסר את קודקוד p מהגרף וצרף אותו לרשימה outp
    - להפחית את שדה count של כל קודקוד הסמוך ל-p ב-1
  - ***Step 2.4.2*** קדם את p לקודקוד הבא.

- ***<u>Step 2.5</u>*** if is_empty(outp)
  - הודעה שיש מעגל ועצור.
- הדפס את כל הקודקודים השייכים לרשימת outp

- אם הגרף מיוצג באמצעות רשימת סמיכות אז סיבוכיות זמן הריצה של האלגוריתם היא $O(|E|+|V|)$, מכיוון שסורקים את רשימת הסמיכות רק פעם אחת וכידוע סכום אורכי רשימות הסמיכות שווה למספר הקשתות בגרף המכוון.

שאלה 4

- א. בגרף הנתון "תת הגרף המלא" הוא בגודל 4 והוא {2,3,4,5}



- ב. גרסה 1

- for i=1 to |V| do
  - for j=1 to |V| do
    - If $i \in A \ and \ \ j \in A$
      - if not adjacent(G,A[i],A[j])
        - » return FALSE
- return true

- <u>גרסה 2</u>
- <u>האם A "תת גרף מלא" ב-G</u>
- 1. "אמת" ← S {הנח כי Vקבוצה A "תת גרף מלא"}
- 2.כל עוד הרשימה A אינה ריקה ו-S הוא "אמת" בצע
  - 2.1 הוצא קודקוד מרשימה A והצב אותו במשתנה l.
  - 2.2 k ←1

– 2.3 כל עוד K=!|V| וגם S הוא "אמת" בצע:

– 2.3.1 אם A $\in$ k

- אם M(I,K)=0 אז בצע:

  S ← "שקר"

- 2.3.2 קדם k ב-1

- החזר את S.

- ג.

- for i=1 to n do
  - for j=1 to n do
    - for k=1 to n do
      - if M[i,j]=1 and M[i,k]=1 and
        - » M[j,i]=1 and M[j,k]=1 and
        - » M[k,i]=1 and M[k,j]=1
      - then
        - » return TRUE
- return FALSE

- <u>תוספות</u>: קבוצה A המכילה n קודקודים, מהווה "תת גרף ריק" של הגרף G אם לכל זוג קודקודים (i,j) הנמצאים בקבוצה A, הקשת (i,j) אינה נמצאת בגרף.
- לדוגמא בגרף שבסעיף א' הקבוצה {1,3,6} היא "תת גרף ריק" בגודל 3.
- המטרה להציע אלגוריתם אשר מקבל כקלט גרף G ובודק אם G מכיל "תת גרף ריק" בגודל 3.

- האלגוריתם המבוקש זהה לאלגוריתם שבסעיף ג' פרט לשינוי אחד. במשפט if בתנאים במקום לשאול האם 1=, נשאל האם 0=.