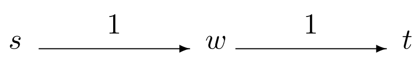
1. א. **נכון.** יש קבוע c כך ש $0.33^c = 0.01$: $c = \log_{0.33} 0.01$. על קלט x נשתמש באלגוריתם הבא:

```
for i = 1 to ⌈c⌉
    if A(x) = flase return false
return true
```

מכיוון שמריצים את האלגוריתם A מספר קבוע של פעמים, זמן הריצה של האלגוריתם הוא מסדר גודל של זמן הריצה של A. כלומר, האלגוריתם פולינומי.

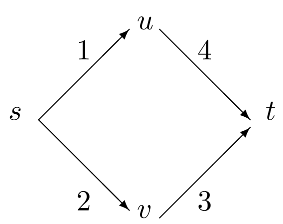
ב. **לא נכון.** נסתכל על החתך (S,T) כך ש S ־ הקודקודים הנגישים מ s ב $\mathcal{N}_f$. אזי, $C(S,T) = |f|$ וכל צלע שחוצה את החתך היא רוויה. אם נניח בשלילה ש e לא רוויה (ולכן לא חוצה את החתך), נקבל ש $|f| = C(S,T)$ זוגי ־ סתירה.

ג. **לא נכון.** נסתכל למשל על הרשת הבאה:



ישנם שני חתכים, שניהם מינימליים ושניהם מהווים דוגמה נגדית.

ד. **לא נכון.** נסתכל למשל על הגרף הבא:



2. א. עבור $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$ נגדיר:

$$OPT(i,j) := \begin{cases} A(n,n), & (i,j) = (n,n) \\ A(i,j) + OPT(i,j+1), & i = n, j < n \\ A(i,j) + OPT(i+1,j), & i < n, j = n \\ \min\{A(i,j) + OPT(i+1,j), A(i,j) + OPT(i,j+1)\}, & i < n, j < n \end{cases}$$

מיקום הפתרון: $OPT(1,1)$.

ב.

```
O(n,n) := A(n,n)
for k = n − 1 to 1 do {
    O(n,k) := A(n,k) + O(n,k+1)
    for l = n − 1 to k + 1 do {
        O(l,k) := min{A(l,k) + O(l+1,k), A(l,k) + O(l,k+1)} }
    O(k,n) := A(k,n) + O(k+1,n)
    for l = n − 1 to k + 1 do {
        O(k,l) := min{A(k,l) + O(k,l+1), A(k,l) + O(k+1,l)} }
    O(k,k) := min{A(k,k) + O(k+1,k), A(k,k) + O(k,k+1)} }
return O(1,1).
```

זמן ריצה: נסתכל על הלולאה הראשית בשורה 2: עבור $k = t$ כלשהו, מספר הפעולות שמתבצעות הוא לכל היותר $c \cdot (n - t)$ כאשר c קבוע מסוים (בצוע הלולאות בשורות 7,4 ועוד מספר קבוע של פעולות). סך הכל נקבל כי זמן הריצה הוא:

$$O(c \cdot ((n-(n-1)) + (n-(n-2)) + \cdots + (n-1))) = O(1+2+3+4+\cdots+n) = O(n^2)$$

מכיון שהקלט הוא מטריצה בגודל $n \times n$, האלגוריתם לינארי בגודל הקלט.

הסבר נוסף לזמן הריצה:

ישנן 3 לולאות באלגוריתם: לולאה ראשית בשורות 2-9, לולאה פנימית בשורות 4-5 ולולאה פנימית נוספת בשורות 7-8. בכל איטרציה של כל לולאה יש מספר קבוע של כתיבות למטריצה O ומספר קבוע של פעולות חישוב נוספות. למשל, בלולאה הראשית יש 3 כתיבות בשורות 3, 6 ו 9, ובלולאה הפנימית הראשונה יש כתיבה יחידה בשורה 5.

כעת, מכיוון שבכל איטרציה של כל לולאה מספר כתיבות קבוע ל O ומספר קבוע של פעולות חישוב, זמן הריצה הוא מסדר גודל של מספר הכתיבות למטריצה O. מכיוון שהלולאה הראשית רצה מ $k = n - 1$ עד $k = 1$, ניתן לראות שיש בדיוק כתיבה יחידה לכל תא במטריצה O (יחד עם התא $O(n,n)$ שהכתיבה אליו מתבצעת מחוץ ללולאות). מכיוון שיש n^2 תאים במטריצה O נקבל שזמן הריצה הוא $O(n^2)$.

ג. נשתמש בטבלה O שמחושבת על ידי האלגוריתם בסעיף ב' ונניח לשם הפשטות שכל גישה לתא מחוץ למטריצה מחזיר את הערך: ∞.

$x := 1$ $y := 1$
$print(x, y)$
while $(x, y) \neq (n, n)$ do {
    if $O(x+1, y) < O(x, y+1)$, $x := x + 1$
    else $y := y + 1$
    print (x, y) }

לא קשה לראות שהלולאה תסתיים לאחר $2n$ איטרציות, לכן זמן הריצה הוא $O(n)$, לינארי באורך הפלט - המסלול הנדרש.

3. א. **נכון.** נשתמש באלגוריתם האימות הבא המקבל כקלט מופע $G = (V, E), k$ ומחפש כעד תת קבוצה A המהווה קליקה בגודל k לפחות.

קלט: $G = (V, E), k$, A.

(א) בדוק אם $k \leq \sqrt{|V_G|}$.

(ב) אם כן, בדוק אם A קבוצת קודקודים שגודלה k לפחות.

(ג) אם כן, בדוק אם A קליקה ואם כן החזר "כן".

(ד) החזר "לא"

הבדיקות הן פולינומיות, ברור שנדחה מופע שאינה בשפה (כי אין קליקה בגודל הרצוי) ולכל מופע בשפה יש עד שגודלו $O(k)$ לכן גודל העד המינימלי פולינומי בגודל הקלט.

ב. **לא נכון** נשתמש ברדוקציה הבאה מ $clique$:

בהינתן מופע ל $clique$: $G = (V, E), k$, נבנה גרף חדש G' על ידי הוספת $|V|^2 - |V|$ קודקודים חדשים ל G. נחזיר G', k.

הרדוקציה פולינומית מפני שמוסיפים $O(V^2)$ קודקודים - פולינומי באורך הקלט. אם $G, k \in clique$ אז יש קליקה בגודל k ב G: A. ברור כי A גם קליקה ב G' ו $k = |A| \leq |V_G| = \sqrt{|V_{G'}|}$ לכן $G', k \in \sqrt{clique}$.

בכיוון השני, נניח כי $G', k \in \sqrt{clique}$ ותהא A קליקה בגודל k. נשים לב שאם $|A| = 1$ או $|A| = 0$ אז ברור ש $G, k \in clique$. אחרת, נסמן X־קבוצת הקודקודים של G ו Y־ קבוצת הקודקודים שהוספנו. מכיון שהדרגה של קודקוד ב Y היא 0, בהכרח $A \subseteq X$. מכאן שיש קליקה בגודל k ב G ולכן $G, k \in clique$ כנדרש.

4. א. נשתמש באלגוריתם הבא:

(א) סדר טופולוגית את V (ע"י הרצת DFS).

(ב) נסמן $U = (u_1, \ldots, u_k)$ הסדר המושרה מהמיון הטופולוגי של V.

(ג) עבור $i = 1$ עד $k - 1$ בדוק אם יש מסלול מ u_i ל u_{i+1}.

מימוש הבדיקה בשורה השלישית:

נסמן $u_i = v_l$, $u_{i+1} = v_m$. הרעיון הוא להחזיק לכל קודקוד בין v_l ל v_m שדה $d[v]$. $d[v] = T$ אם יש מסלול מ v_l ל v ואחרת $d[v] = F$. תחילה, רק $d[v_l] = T$. נסרוק את הקודקודים על פי הסדור הטופולוגי ובכל פעם שנגיע לקודקוד v כך ש $d[v] = T$, נכתוב T גם לשכניו (צלעות יוצאות).

(א) אתחול: לכל קודקוד v: $d[v] := F$. $d[v_l] := T$.

(ב) עבור $j = l$ עד $j = m - 1$ בצע:

(ג) אם $d[v_j] = T$, לכל שכן w של v_j בצע $d[w] := T$.

(ד) החזר $d[v_m]$

ראשית נוכיח את נכונות המימוש של הבדיקה:

<u>טענה:</u> הבדיקה מחזירה על קלט v_l, v_m: T אם"ם יש מסלול מ v_l ל v_m.

<u>הוכחה:</u> נובע מטענת העזר הבאה:

<u>טענת עזר 1.</u> לכל $l \leq j \leq m$, במהלך ריצת האלגוריתם, נכתב ל $d[v_j]$: T לפני האיטרציה ה j (מתחילים לספור מ $j = l$) אם"ם יש מסלול מ v_l ל v_j.

<u>הוכחה:</u> באינדוקציה על j.

<u>בסיס.</u> עבור $j = l$, נכתב T באתחול.

<u>מהלך:</u> עבור $j > l$. מכיוון שמדובר במיון טופולוגי, יש מסלול מ v_l ל v_j אם"ם יש $l \leq i < j$ כך שיש מסלול מ v_l ל v_i וצלע מ v_i ל v_j. על פי הנחת האינדוקציה: אם"ם באיטרציה ה i כבר כתוב T ב $d[v_i]$, ולפי שורה 3: אם"ם כותבים T ל $d[v_j]$ באיטרציה ה i.

כעת נוכיח את נכונות האלגוריתם. נשים לב שהאלגוריתם מחזיר "כן" אם"ם יש מסלול מכל u_i ל u_{i+1}. לכן נוכיח:

<u>טענת עזר 2.</u> יש מסלול שעובר דרך כל קודקודי U אם"ם לכל $i < k$ יש מסלול מ u_i ל u_{i+1}.

<u>הוכחה.</u> אם לכל $i < k$ יש מסלול מ u_i ל u_{i+1}, ברור שיש מסלול שעובר דרך קודקודי U. בכיוון השני: יהא P מסלול העובר דרך כל קודקודי U. נשים לב שאם $i < j \leq k$, אין מסלול מ u_j ל u_i (מיון טופולוגי). לכן, לכל $i < k$: u_{i+1} מופיע לאחר u_i ב P ומכאן נכונות טענת העזר.

זמן ריצה: DFS: $O(|V| + |E|)$. בסך כל ההרצות של שורה 3, כל צלע "נבדקת" פעם אחת: $O(|V| + |E|)$. סך הכל: $O(|V| + |E|)$.

פתרונות נוספים שראינו:

פתרון 1. לאחר המיון הטופולוגי נסמן $u_1 = v_l$ ו $u_k = v_m$. במקום לבדוק אם יש מסלול בין כל u_i ל u_{i+1} נבדוק אם יש מסלול מ u_1 ל u_k שעובר דרך k קודקודים מ U. השיטה ־ נשמור לכל קודקוד שדה $d[v]$ כך שבסוף הריצה, $d[v] = r$ אם r הוא מקסימלי כך שיש מסלול מ u_1 ל v שעובר דרך r קוקודים מ U. ניתן לעשות זאת בשתי צורות, האחת, בדומה לאלגוריתם הקודם, לסרוק לפי המיון הטופולוגי ובכל פעם לעדכן את ערכי השכנים של הקודקוד הנבדק. הצורה השניה היא דוקא לעדכן את הקודקוד הנבדק על פי ערכי הקודקודים בצלעות הנכנסות. להלן פסאודוקוד לגישה השניה:

(א) אתחול: לכל v, $d[v] := 0$, $d[v_l] := 1$.
(ב) עבור $j = l + 1$ עד $j = m$ בצע }
(ג) $n := \max\{d[w] : (w, v_j) \in E\}$
(ד) $d[v_j] := n$
(ה) אם $v_j \in U$, $d[v_j] := d[v_j] + 1$ {
(ו) החזר "כן" אם $d[v_m] = k$, אחרת החזר "לא".

פתרון 2. לאחר המיון הטופולוגי נקח את הקודקוד הראשון במיון (נניח ב.ה.כ. u_1) ואת האחרון (u_k). נוסיף צלע מ u_k ל u_1 ונריץ את Kosaraju-Sharir. נחזיר "כן" אם"ם כל קודקודי U באותו רכיב קשירות.

ב. נסמן $U = \{u_1, \dots, u_k\}$ ונשתמש באלגוריתם הבא:
(א) בנה את G' גרף הרכיבים הקשירים היטב של G.
(ב) $U' := \{[u_1], \dots, [u_k]\}$ (מחלקות השקילות של איברי U).
(ג) הרץ את האלגוריתם מסעיף קודם על G', U'.

נכונות האלגוריתם נובעת מהטענה הבאה:
<u>טענת עזר</u> יש מסלול שעובר דרך קודקודי U בG אם"ם יש מסלול שעובר דרך קודקודי U' ב G'.

ההוכחה פשוטה מכיון שבכל רכיב קשירות יש מסלול מכל קודקוד לכל קודקוד. ברור כי זמן הריצה הוא $O(|V| + |E|)$.