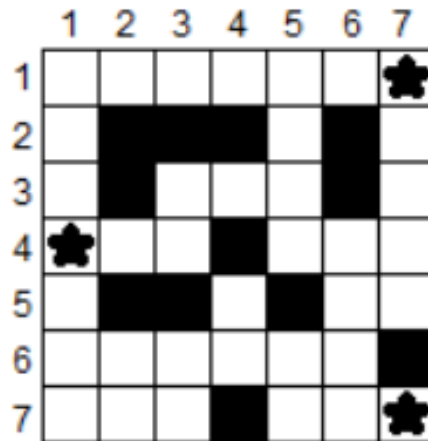


חידה לחימום

שלושה רובוטים נמצאים על משבצות שונות של לוח משבצות, כמתואר באיור הבא:



המיקום ההתחלתי של כל אחד מהרובוטים מסומן על הלוח באמצעות כוכב.

מעוניינים להפגיש את שלושת הרובוטים באותה המשבצת, באמצעות סדרה של צעדים. צעד של כל אחד משלושת הרובוטים הוא הזזתו למשבצת סמוכה, אופקית או אנכית (כלומר – מעלה, מטה, ימינה או שמאלה).

אסור להזיז רובוט למשבצת שחורה, או להוציאו מהלוח.

מותר שעל משבצת אחת יהיה יותר מרובוט אחד ברגע נתון.

המטרה היא להפגיש את שלושת הרובוטים באותה המשבצת, תוך מספר כולל קטן ככל האפשר של צעדים.

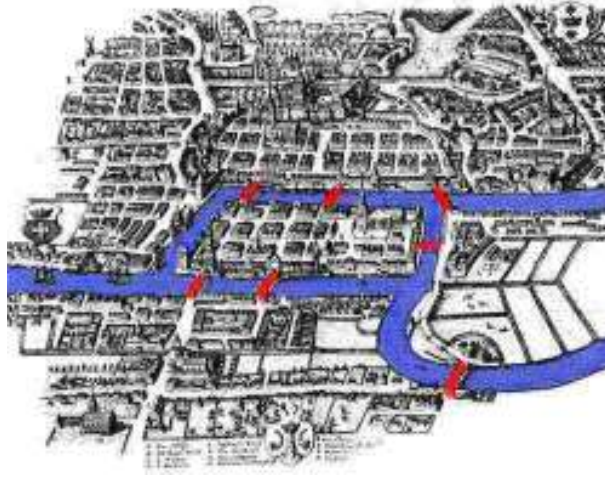
מהו מספר הצעדים הכולל הקטן ביותר הדרוש כדי להפגיש את הרובוטים במשבצת אחת, בציוור שלעיל? ומהן המשבצות בהן ניתן להפגיש את הרובוטים תוך מספר זה של צעדים?

מבני נתונים ויעילות אלגוריתמים

(28.1.2015)

1. מסלולי אוילר ומעגלי אוילר

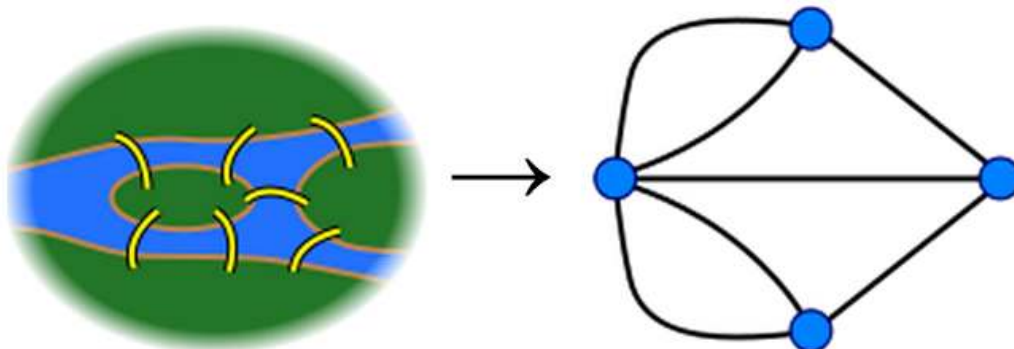
בעיר קניגסברג (Konigsberg) בפרוסיה המזרחית (אשר כיון כיום נקראת קלינינגרד, ושוכנת בתחומי רוסיה) זורם נהר הפרגל (Pregel). על גבי הנהר נבנו שבעה גשרים:



לתושבי העיר הציקה שאלה: האם ניתן לטייל בעיר, ולחצות את כל שבעת הגשרים, כך שלא נעבור באותו הגשר פעמיים? תושבי העיר ניסו לתת מענה לשאלה זו, אך לא הצליחו. הם פנו למתמטיקאי ליאונרד אוילר (Euler) אשר הוכיח ב-1735 שלא ניתן לעשות זאת.

אוילר (1707-1783) נולד בשוויץ, וחי בגרמניה וברוסיה. הוא פירסם יותר מאמרים וספרים בימי חייו מכל מתמטיקאי אחר, ונחשב למתמטיקאי הפורה בהיסטוריה.

איך פתר אוילר את בעיית הגשרים של קניגסברג? באמצעות הפשטה: הוא ייצג את מפת העיר באמצעות גרף, שקודקודיו הם ארבעת חלקי העיר, וקשתותיו הן שבעת הגשרים:



בגרף הזה – דרגתו של כל קודקוד היא אי-זוגית. אבל כדי שיהיה טיול העובר דרך כל גשר פעם אחת בדיוק, צריך שבכל פעם שניכנס לקודקוד – נוכל גם לצאת ממנו. כלומר – צריך שדרגת כל קודקוד תהיה זוגית.

למעשה, מותר שיהיו שני קודקודים בלבד בעלי דרגה אי-זוגית: קודקוד ההתחלה וקודקוד הסיום. אבל בגרף הנתון, יש יותר משני קודקודים בעלי דרגה אי-זוגית, ולכן – אין טיול מהסוג המבוקש.

הגדרה: יהי $G = (V, E)$ גרף קשיר ולא מכוון. מסלול אוילר (Euler path) הוא מסלול המבקר בכל קשת בדיוק פעם אחת. אם קודקוד ההתחלה של המסלול זהה לקודקוד הסיום, אז מדובר במעגל אוילר (Euler circuit).

משפט: יהי $G = (V, E)$ גרף קשיר ולא מכוון.

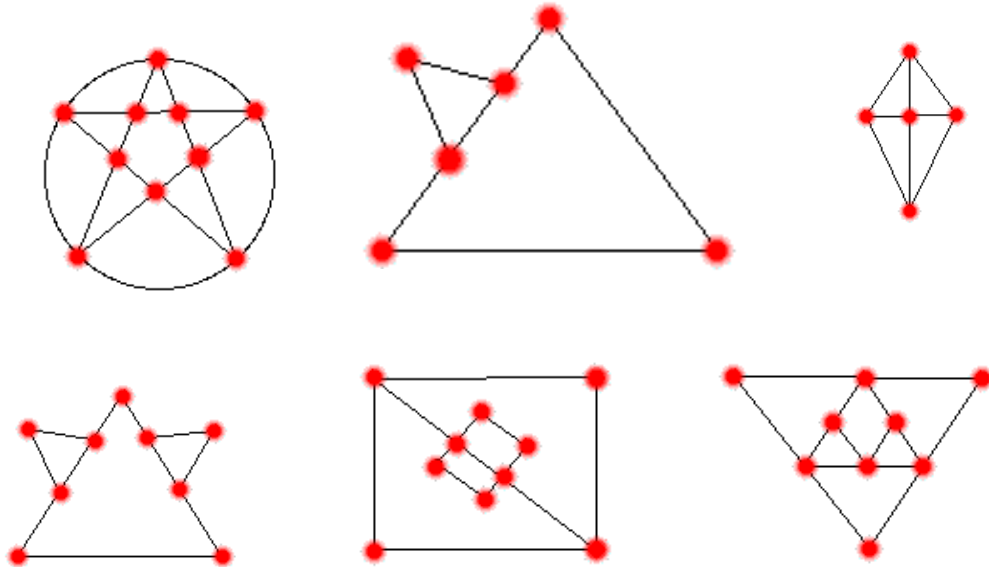
- אם דרגות כל הקודקודים בגרף הן זוגיות – בגרף יש מעגל אוילר.
- אם דרגות כול הקודקודים בגרף הן זוגיות, פרט לשני קודקודים שדרגתם אי-זוגית – בגרף יש מסלול אוילר, אשר מתחיל ומסתיים באותם שני קודקודים.

הגדרה: גרף ייקרא גרף אוילריאני (Eulerian graph) אם יש בו מסלול אוילר או מעגל אוילר.

תרגיל:

איזה מהגרפים הבאים הוא גרף אוילריאני?

עבור גרף שהוא אוילריאני – קבעו האם יש בו מעגל אוילר או מסלול אוילר, ובמידה ויש בו מסלול אוילר – קבעו מי הם קודקודי ההתחלה והסיום של המסלול.



משפט: יהי $G = (V, E)$ גרף מכוון, אשר גרף התשתית שלו הוא קשיר.

- אם מתקיים $d_{in}(v) = d_{out}(v)$ עבור כל קודקוד $v \in V$, אז בגרף יש מעגל אוילר.
- אם קיים קודקוד $a \in V$ עבורו מתקיים $d_{out}(a) = d_{in}(a) + 1$, וקיים קודקוד $b \in V$ שעבורו מתקיים $d_{in}(b) = d_{out}(b) + 1$, וכמו כן מתקיים $d_{in}(v) = d_{out}(v)$ לכל קודקוד $v \in V - \{a, b\}$, אז בגרף יש מסלול אוילר המתחיל ב- a ומסתיים ב- b .

2. חיפוש לעומק תחילה (Depth-First Search)

האסטרטגיה הנקטת בחיפוש לעומק-תחילה היא, כפי שמרמז שמו, לחפש "עמוק יותר" בגרף, ככל שהדבר אפשרי. בחיפוש לעומק נבדקות הקשתות היוצאות מן הקודקוד שהוא אחרון הקודקודים שהתגלו, שעדיין יש לו קשתות היוצאות ממנו וטרם נבדקו. לאחר שנבדקו כל הקשתות היוצאות מקודקוד v , החיפוש "נסוג" וממשיך בבדיקת הקשתות היוצאות מהקודקוד שממנו התגלה v .

תהליך זה נמשך עד שמתגלים כל הקודקודים שניתן להגיע אליהם מקודקוד המקור. אם נותרו קודקודים שטרם התגלו, למשל במקרה ובו הגרף לא קשיר, אז אחד מהם ייבחר כמקור חדש, והחיפוש ימשיך ממקור זה. תהליך זה יחזור על עצמו עד שיתגלו כל הקודקודים. במהלך הסריקה האלגוריתם בונה יער (forest) המורכב מעצי עומק (depth trees).

כמו בחיפוש לרוחב, גם בחיפוש לעומק נצבע קודקודים במהלך החיפוש כדי לציין את מצבם. כל קודקוד צבוע בתחילה בלבן. הוא נצבע באפור כאשר הוא מתגלה במהלך החיפוש, והוא נבצע בשחור כאשר הטיפול בו מסתיים, כלומר – כאשר כל השכנים שלו נבדקו.

האלגוריתם לחיפוש לעומק שומר בכל קודקוד חותמות זמן (timestamps). לכל קודקוד v יש שתי חותמות זמן: הראשונה, v .discovered, מציינת מתי התגלה v לראשונה (ונצבע באפור). השנייה, v .finished, מציינת מתי סיים החיפוש לבחון את כל שכניו של v (וצבע אותו בשחור). חותמות זמן אלו הם ערכים שלמים בין 1 ל- $|V|$, שכן כל אחד מהקודקודים ב- V מתגלה פעם אחת, והטיפול בו מסתיים פעם אחת. בנוסף, עבור כל קודקוד v מתקיים: v .discovered < v .finished. קודקוד v הוא לבן לפני v .discovered, הוא אפור בין v .discovered לבין v .finished, והוא שחור אחרי v .finished.

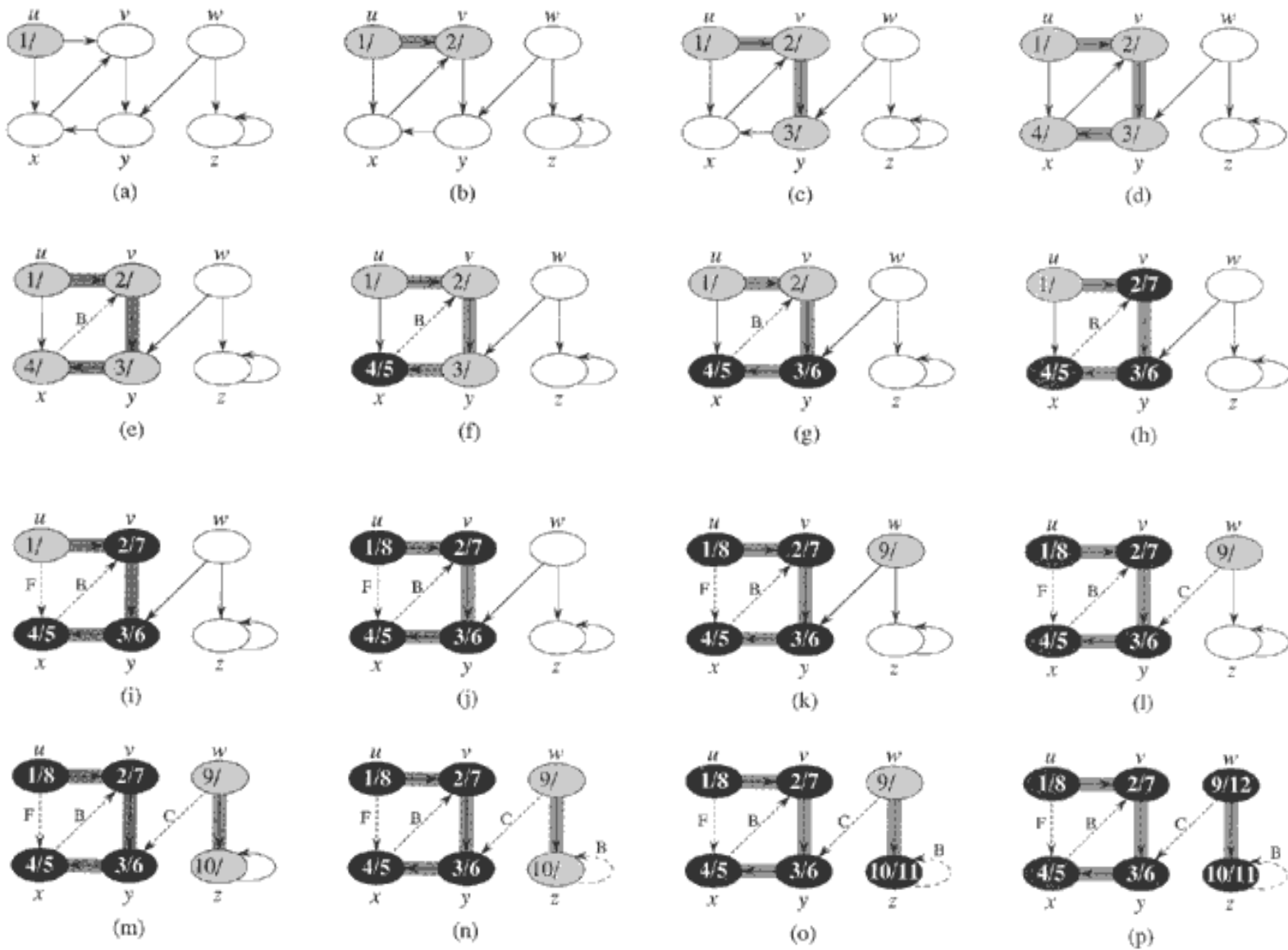
בדומה ל-BFS, גם האלגוריתם DFS, שומר עבור כל קודקוד $u \in V$ את צבעו (u .color) ואת הקודקוד הקודם אליו (u .previous). אם לקודקוד u אין קודקוד קודם (למשל, אם $u = s$ או אם הקודקוד u טרם התגלה), אזי u .previous = NULL.

```
DFS(G)
1   for each vertex  $u \in V$ 
2       do  $u$ .color  $\leftarrow$  WHITE
3            $u$ .previous  $\leftarrow$  NULL
4   time  $\leftarrow$  0
5   for each vertex  $u \in V$ 
6       do if  $u$ .color = WHITE
7           then DFS-VISIT( $G, u$ )
```

```

DFS-VISIT(G, u)
1   u.color ← GRAY
2   time ← time + 1
3   u.discovered ← time
4   for each vertex v adjacent to u
5       do if v.color = WHITE
6           then v.previous ← u
7              DFS-VISIT(G, v)
8   u.color ← BLACK
9   time ← time + 1
10  u.finished ← time
    
```

נתאר דוגמה לריצת האלגוריתם לחיפוש לעומק DFS:



השגרה DFS פועלת באופן הבא: שורות 1-3 צובעות את כל הקודקודים בלבן ומאתחלות את השדה previous שלהם לערך NULL. שורה 4 מאפסת את מונה הזמן (שהוא משתנה גלובלי). שורות 5-7 בודקות כל קודקוד ב-V, וכאשר נמצא קודקוד לבן, נערוך בו "ביקור" באמצעות DFS-VISIT. בכל פעם שמתבצעת בשורה 7 קריאה ל-DFS-VISIT(G,u) אז הקודקוד u הופך לשורש של עץ חדש ביער העומק.

בכל קריאה ל-DFS-VISIT(G,u), הקודקוד u הוא בהתחלה לבן. שורה 1 של DFS-VISIT צובעת את u באפור, ושורות 2-3 מחשבות את מועד הגילוי של u, על-ידי הוספת 1 למשתנה הגלובלי time, והצבתו ב-u.discovered. שורות 4-7 בוחנות כל קודקוד v הסמוך ל-u, ואם הוא לבן, עורכים בו ביקור באופן רקורסיבי. לאחר שנבדקו כל הקשתות היוצאות מ-u, שורות 8-10 צובעות את u בשחור, ומציבות ב-u.finished את מועד הסיום.

את הקשתות בגרף $G = (V, E)$ ניתן לסווג לארבעה סוגים:

1. קשתות עץ (tree edges) – הן קשתות השייכות ליער העומק (אוסף עצי העומק הנוצרים על-ידי ריצת האלגוריתם DFS). קשת (u, v) היא קשת עץ אם v התגלה לראשונה על-ידי בדיקת הקשת (u, v) .
2. קשתות אחורה (back edges) – הן אותן קשתות (u, v) המחברות קודקוד u לאב קדמון שלו v . גם לולאות עצמיות נחשבות לקשתות אחורה.
3. קשתות קדימה (forward edges) – הן אותן קשתות (u, v) שאינן קשתות עץ, אשר מחברות קודקוד u לצאצא v בעץ עומק.
4. קשתות חוצות (cross edges) – הן כל הקשתות האחרות. הן עשויות לקשר קודקודים באותו עץ עומק, כל עוד קודקוד אחד אינו צאצא של האחר, או לקשר קודקודים בעצי עומק שונים.

כל קשת (u, v) ניתן לסווג על פי צבעו של הקודקוד v שמגיעים אליו מ- u כאשר הקשת נבדקת לראשונה במהלך ריצת DFS: אם הקודקוד לבן אז מדובר בקשת עץ, אם הקודקוד אפור אז מדובר בקשת אחורה, ואם הקודקוד שחור אז זוהי קשת קדימה (אם מתקיים $u.discovered < v.discovered$) או קשת חוצה (אם מתקיים $u.discovered > v.discovered$).

מהו זמן הריצה DFS? הלולאות שבשורות 1-3 ובשורות 5-7 של DFS מתבצעות בזמן $\Theta(V)$, לא כולל הזמן הדרוש לביצוע הקריאות ל-DFS-VISIT. השגרה DFS-VISIT נקראת בדיוק פעם אחת עבור כל קודקוד $v \in V$, שכן היא נקראת רק על קודקודים לבנים, ופעולתה הראשונה היא לצבוע את הקודקוד באפור. במהלך ביצועה של DFS-VISIT, הלולאה שבשורות 4-7 מתבצעת מס' פעמים השווה לדרגת הקודקוד v . מאחר שידוע לנו (על פי משפט לחיצות הידיים) כי סכום דרגות הקודקודים בגרף שווה לפעמיים מספר הקשתות בגרף, הרי שהעלות הכוללת לביצוע שורות 4-7 של DFS-VISIT היא $\Theta(E)$.

זמן הריצה של DFS, בסיכומו של דבר, הוא $\Theta(V+E)$, בדומה לזמן הריצה של BFS.

3. האלגוריתם של קרוסקל (Kruskal) למציאת עץ פורש מינימלי

כזכור, בהינתן גרף פשוט $G = (V, E)$ נגדיר עץ פורש (spanning tree) $T = (V, E')$ בתור עץ שקבוצת קודקודיו מורכבת מ- V (מקודקודי הגרף המקורי G), וקבוצת קשתותיו E' היא תת-קבוצה של קבוצת הקשתות E בגרף המקורי G . כלומר: $E' \subseteq E$.

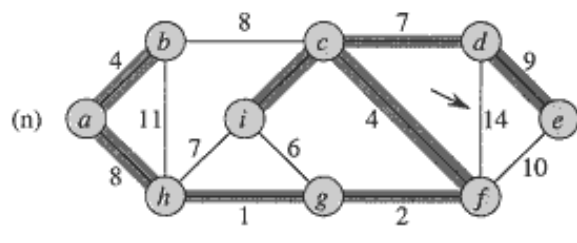
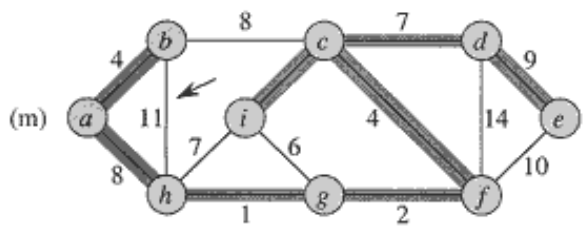
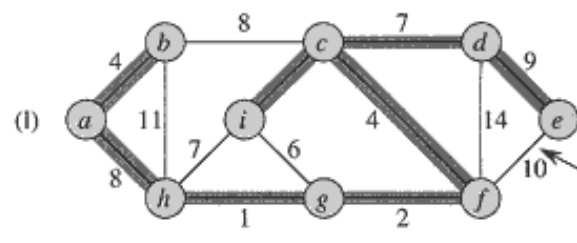
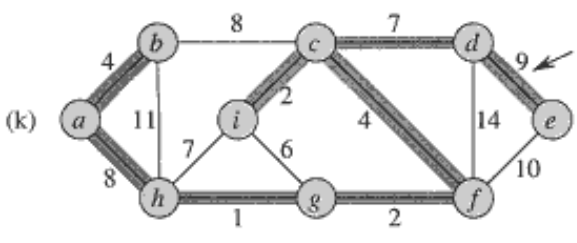
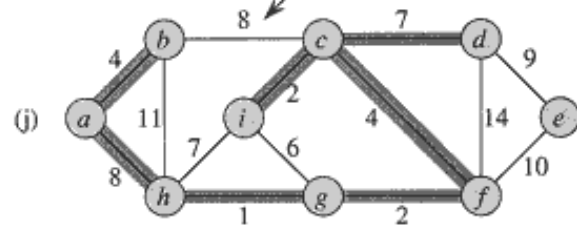
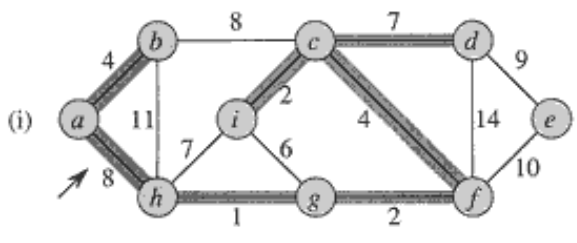
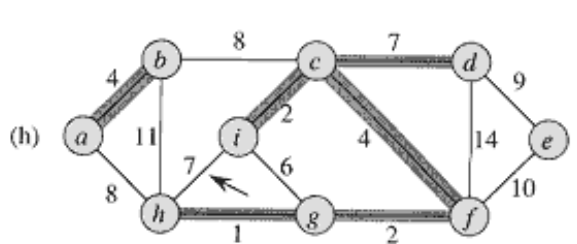
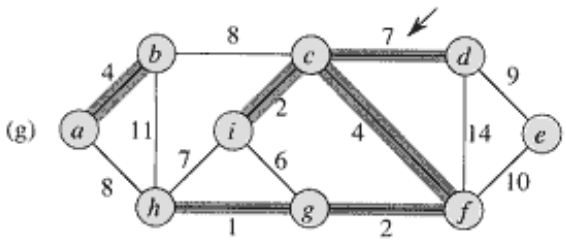
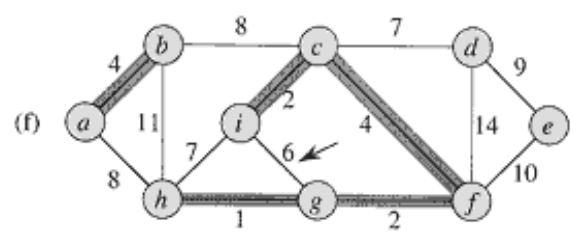
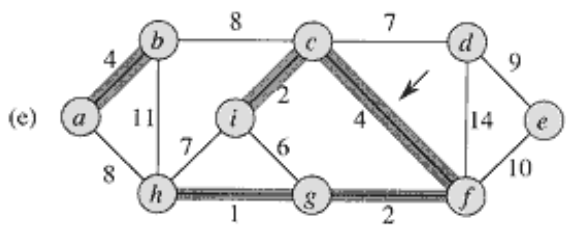
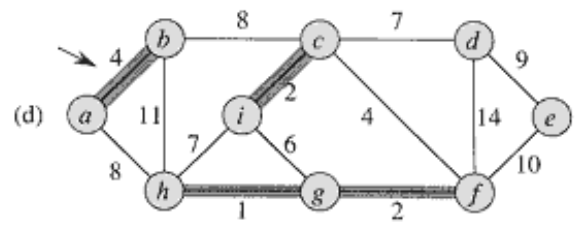
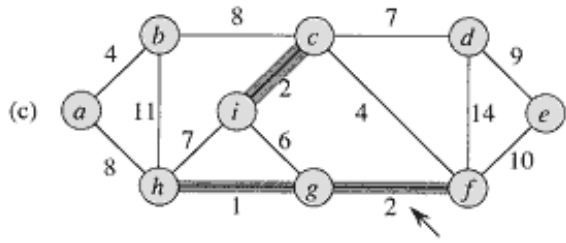
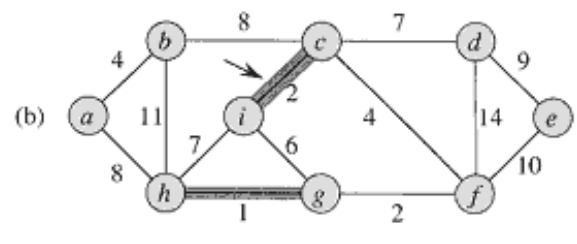
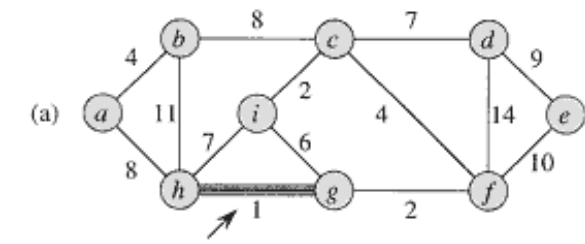
אם הגרף G ממושקל, אז נגדיר משקל של עץ בתור סכום המשקלים שעל הקשתות המרכיבות אותו. מכיוון שבגרף נתון יכולים להיות עצים פורשים רבים, ייתכנו גם משקלים רבים עבור עצים אלו. אנחנו נתעניין בשאלה: האם קיים אלגוריתם, אשר בהינתן גרף ממושקל, מוצא את העץ הפורש אשר משקלו הוא הקטן ביותר מבין כל משקלי העצים הפורשים של הגרף. כלומר, עץ פורש מינימלי (Minimal Spanning Tree - MST).

קיימים מספר אלגוריתמים כאלה. הראשון מביניהם, נקרא האלגוריתם של קרוסקל (פורסם בידי המתמטיקאי ומדען המחשב Joseph Kruskal, בשנת 1956). הרעיון מאחורי האלגוריתם הוא לבנות אוסף של עצים (יער – forest), כך שבתחילת הריצה כל עץ מורכב מקודקוד יחיד. בכל איטרציה, האלגוריתם מוצא קשת (u, v) בעלת משקל מינימלי מבין כל הקשתות המחברות שני עצים כלשהם ביער, ומאחד את שני העצים לעץ אחד. בסיום התהליך מתקבל עץ יחיד שהוא פורש ומינימלי.

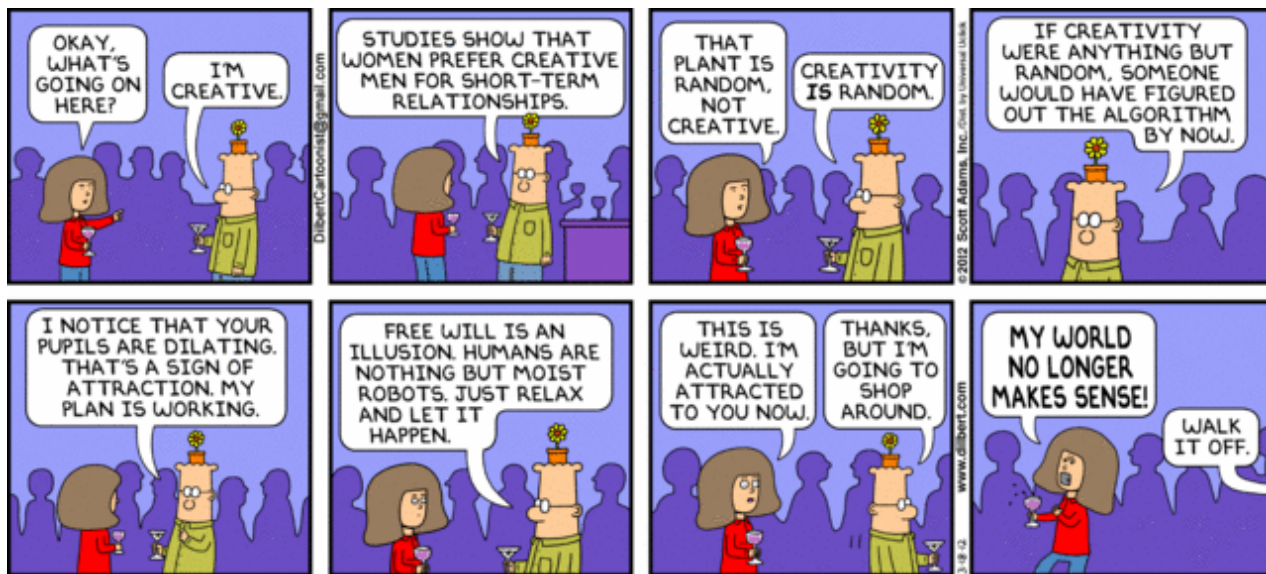
במהלך האלגוריתם משתמשים בפעולה MAKE-SET המקבלת קודקוד בודד ויוצרת קבוצה המכילה אותו, בפעולה FIND-SET המקבלת קודקוד בודד ומחזירה את הקבוצה אליה הוא שייך, ובפעולה UNION המקבלת שני קודקודים ומאחדת את שתי הקבוצות אליהם הקודקודים שייכים.

```
KRUSKAL(G)
1   A ← ∅
2   for each vertex v ∈ V
3     do MAKE-SET(v)
4   sort the edges of E by increasing weight
5   for each edge (u,v) ∈ E in order by increasing weight
6     do if FIND-SET(u) ≠ FIND-SET(v)
7       then A ← A ∪ {(u,v)}
8           UNION(u,v)
9   return A
```

שורות 1-3 באלגוריתם מאתחלות את הקבוצה A לקבוצה הריקה ויוצרות $|V|$ עצים אשר כל אחד מהם מכיל קודקוד אחד. שורה 4 ממיינת את הקשתות ב- E לפי סדר עולה של משקלות. לולאת ה- for בשורות 5-8 בודקת, עבור כל קשת (u, v) , האם הקודקודים בקצוותיה, u ו- v , שייכים לאותו עץ. אם אכן כך הדבר, אזי לא ניתן להוסיף את הקשת (u, v) בלי ליצור מעגל, ולפיכך אין משתמשים בה. אחרת, שני הקודקודים שייכים לעצים שונים, ואז מוסיפים את הקשת (u, v) ל- A בשורה 7, ובשורה 8 מתבצע מיזוג של שני העצים הללו לעץ אחד.



זמן הריצה של האלגוריתם של קרוסקל על גרף $G = (V, E)$ תלוי במימוש מבנה הנתונים עבור הקבוצות. מקובל להשתמש במימוש שבו האיתחול יתבצע בזמן $\Theta(V)$, הזמן הדרוש למיון הקשתות בשורה 4 יהיה $\Theta(E \log E)$, והזמן הדרוש לביצוע הפעולות על הקשתות בשורות 5-8 יהיה $\Theta(E \log E)$. לכן, בסך-הכל, זמן ריצת האלגוריתם יהיה $\Theta(E \log E)$. ניתן להראות כי ביטוי זה שקול גם ל- $\Theta(E \log V)$.



4. פתרון משוואות נסיגה לינאריות הומוגניות

משוואות נסיגה (משוואות רקורסיביות) מהצורה

$$f(n) = c_1 \cdot f(n-1) + c_2 \cdot f(n-2) + c_3 \cdot f(n-3) + \dots + c_k \cdot f(n-k)$$

נקראת משוואה נסיגה לינארית הומוגנית. פותרים אותה באופן הבא:

א. מציבים α^i במקום $f(i)$, ומקבלים:

$$\alpha^n = c_1 \cdot \alpha^{n-1} + c_2 \cdot \alpha^{n-2} + c_3 \cdot \alpha^{n-3} + \dots + c_k \cdot \alpha^{n-k}$$

ב. מצמצמים ב- α^{n-k} , ומקבלים: $\alpha^k = c_1 \cdot \alpha^{k-1} + c_2 \cdot \alpha^{k-2} + c_3 \cdot \alpha^{k-3} + \dots + c_k$

ג. מעבירים אגפים, ומקבלים: $\alpha^k - c_1 \cdot \alpha^{k-1} - c_2 \cdot \alpha^{k-2} - c_3 \cdot \alpha^{k-3} - \dots - c_k = 0$. המשוואה הזו

נקראת המשוואה האופיינית (characteristic equation) של נוסחת הנסיגה.

ד. פותרים את המשוואה האופיינית, ואם יש k פתרונות שונים $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \dots, \alpha_k$, אז מתקיים:

$$f(n) = A_1 \cdot \alpha_1^n + A_2 \cdot \alpha_2^n + A_3 \cdot \alpha_3^n + \dots + A_k \cdot \alpha_k^n$$

ה. את הקבועים $A_1, A_2, A_3, A_4, \dots, A_k$ מחשבים לפי תנאי ההתחלה של נוסחת הנסיגה.

נדגים את השיטה על-ידי פתרון מספר משוואות נסיגה לינאריות הומוגניות.

שאלה

נתונה נוסחת הנסיגה $f(n) = f(n-1) + 2f(n-2)$ עם תנאי ההתחלה $f(1) = 1, f(2) = 3$. מצאו ביטוי מפורש עבור $f(n)$.

תשובה

נציב α^i במקום $f(i)$, ונקבל: $\alpha^n = \alpha^{n-1} + 2\alpha^{n-2}$.

נצמצם ב- α^{n-2} , ונקבל: $\alpha^2 = \alpha + 2$.

נעביר אגפים, ונקבל את המשוואה האופיינית: $\alpha^2 - \alpha - 2 = 0$.

נפתור את המשוואה הזו לפי נוסחת השורשים, ונקבל: $\alpha_1 = 2$ ו- $\alpha_2 = -1$.

לכן, צורת הפתרון היא: $f(n) = A_1 \cdot 2^n + A_2 \cdot (-1)^n$.

כדי לחשב את ערכם של הקבועים A_1, A_2 נשתמש בתנאי ההתחלה.

נציב $n = 1$ ונקבל: $1 = 2 \cdot A_1 - A_2$

נציב $n = 2$ ונקבל: $3 = 4 \cdot A_1 + A_2$

קיבלנו מערכת של שתי משוואות בשני נעלמים, שפתרונה: $A_1 = \frac{2}{3}, A_2 = \frac{1}{3}$.

ואז הביטוי המפורש של נוסחת הנסיגה הוא: $f(n) = \frac{2}{3} \cdot 2^n + \frac{1}{3} \cdot (-1)^n$.

ניתן להוציא גורם משותף, ולקבל: $f(n) = \frac{1}{3} \cdot (2^{n+1} + (-1)^n)$.

מבחינה אסימפטוטית: $f(n) = \Theta(2^n)$.

שאלה

נתונה נוסחת פיבונצ'י $f(n) = f(n-1) + f(n-2)$ עם תנאי ההתחלה $f(0) = f(1) = 1$. מצאו ביטוי מפורש עבור $f(n)$.

תשובה

נציב α^i במקום $f(i)$, ונקבל: $\alpha^n = \alpha^{n-1} + \alpha^{n-2}$.

נצמצם ב- α^{n-2} , ונקבל: $\alpha^2 = \alpha + 1$.

נעביר אגפים, ונקבל את המשוואה האופיינית: $\alpha^2 - \alpha - 1 = 0$.

נפתור את המשוואה הזו לפי נוסחת השורשים, ונקבל: $\alpha_1 = \frac{1+\sqrt{5}}{2}$ ו- $\alpha_2 = \frac{1-\sqrt{5}}{2}$.

לכן, צורת הפתרון היא: $f(n) = A_1 \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n + A_2 \cdot \left(\frac{1-\sqrt{5}}{2}\right)^n$.

כדי לחשב את ערכם של הקבועים A_1, A_2 נשתמש בתנאי ההתחלה.

נציב $n = 0$ ונקבל: $1 = A_1 + A_2$.

נציב $n = 1$ ונקבל: $1 = A_1 \cdot \left(\frac{1+\sqrt{5}}{2}\right) + A_2 \cdot \left(\frac{1-\sqrt{5}}{2}\right)$.

קיבלנו מערכת של שתי משוואות בשני נעלמים, שפתרונה: $A_1 = \frac{1+\sqrt{5}}{2\sqrt{5}}$, $A_2 = \frac{\sqrt{5}-1}{2\sqrt{5}}$.

ואז הביטוי המפורש של נוסחת הנסיגה הוא: $f(n) = \frac{1}{\sqrt{5}} \cdot \left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1} \right)$.

מבחינה אסימפטוטית: $f(n) = \Theta(1.6^n)$.