מבני נתונים ואלגוריתמים

תרגיל בית 3

טל פאר וענת לורמן

שאלה 1

נשתמש באלגוריתם BFS ונבצע עליו מודיפיקציות אשר יאפשרו לו למצוא את מספר המסלולים הקצרים ביותר מצומת נתונה s.

Vertex attribute:

v.neighbors
v.color
v.parent
v.d
v.spn – initialization: v.spn = 0

Q1 : **Modified_BFS**(G,s):

1. BFS Initialization (G,s, Q)
2. s. spn = 1
3. while Q $\neq \emptyset$ do
4.   u = Dequeue(Q)
5.   for all v $\in$ u. Neighbors do
6.   if v.color == white then
7. **   v. spn = u. spn**
8.   v.color = gray
9.   v.d = u.d + 1
10.   v.π = u
11.   Enqueue(Q, v)
12. **   Else**
13. **   If v.d = u.d +1**
14. **   v. spn = v. spn + u. spn**
15.   u.color = black

הסבר האלגוריתם המוצע:

- נשים לב כי מבחינת סיבוכיות זמן, אין כל הבדל בין האלגוריתם המתואר לאלגוריתם BFS רגיל משום שהפעולות שנוספו מתבצעות ב- O(1) בלבד. לפיכך סיבוכיות האלגוריתם הינה: $O(V+E)$.
- נשים לב כי הקשת $(u,v) \in E$, מתאפיינת בכך שבמידה ולצומת v נכנסת הקשת הנ"ל והיא בלבד, מספר המסלולים הקצרים ביותר של הצמתים u, v שווה.
  באותו אופן, גם במידה וצומת v בעלת קשתות נוספות הנכנסות אליה מלבד u, מספר המסלולים הקצרים מצומת v הוא סכום מספר המסלולים הקצרים של כלל הקשתות הנכנסות אליה.
  בדיוק לשם כך הוספנו את התכונה לכל צומת, הסופרת את מספר המסלולים הקצרים.
  כל צומת "בן" יורשת מצומת האב שלה את מספר המסלולים, באופן זה הצומת הראשון באלגוריתם BFS יורשת את המספר 1 מהצומת הראשית s, וכך הלאה.

שאלה 2

א. יהי $G = (V, E)$ גרף מכוון פשוט.

אלגוריתם מוצע:

נשתמש באלגוריתם $BFS(G, s)$ עם השינוי הבא-

נחליף את השדה v.d בכל צומת v in V, בשני השדות הבאים-

1. v.s_d - המרחק של צומת v מצומת s.
2. v.t_d - המרחק של צומת v מצומת t המגדירה את t -מסלול.

בהחלפה זו נריץ את האלגוריתם $BFS(G, s)$ לעדכון השדה v.s_d ואת האלגוריתם $BFS(G, t)$ לעדכון השדה v.t_d .

עבור כל צומת $u \in V$ הערך המוחזר הוא $t.s\_d + v.t\_d$.

נכונות

מאופן הגדרת האלגוריתם לעיל ומנכונות האלגוריתם BFS, לפי (1) ו-(2) מתקיים כי לכל צומת v הערך החוזר הוא -

$\delta(s,t) + \delta(t,v)$ במידה ואכן קיים מסלול העובר דרך t.

אחרת, השדות המתוארים נותרים בערכם הדפולטיבי ∞ , ולכן הפלט הוא ∞ **.**

עלינו להראות כי $length(p_t(s,v))$ = $\delta_t(s,v)$ = $\delta(s,t) + \delta(t,v)$

מאופן הגדרת , $\delta_t(u,v)$ בתור המסלול הקצר ביותר בין u ל- v העובר דרך t, יתקיים:

$$length(P(s,t)) + length(P(t,v)) \geq \delta(s,t) + \delta(t,v)$$

נסמן את המסלול הקצר ביותר מ - s ל - t בתור $P^*(s,t)$ ומסלול קצר ביותר מ-t ל- v בתור $P^*(t,v)$.

$$\min_{P_t(u,v)}\{length(P_t(u,v))\} \leq length(P^*(s,t)) + length(P^*(t,v)) = \delta(s,v) + \delta(t,v)$$

משילוב שני אי-השוויונות שהתקבלו, מתקיים כי -

$$\min_{P_t(u,v)}\{length(P_t(u,v))\} = \delta_t(s,v) = \delta(s,t) + \delta(t,v)$$

ולכן האלגוריתם המוצע אכן מחזיר את הפלט המבוקש.

ניתוח סיבוכיות זמן:

ראינו כי סיבוכיות זמן-ריצת אלגוריתם BFS היא$O(V + E)$

באלגוריתם המוצע סה"כ מבצעים 2 ריצות על אלגוריתם BFS, כאשר אחת היא עם צומת s כקלט והשניה עם צומת t כקלט. סה"כ:

$2 * O(n+m) = O(n+m)$ וסיבוכיות זמן האלגוריתם היא כנדרש.

ב. הראינו בסעיף 1 כי פלט האלגוריתם אמור לספק עבור כל צומת $u \in V$ את המרחק לצומת t ומצומת t אל הצומת s.

נגדיר G^+ בתור הגרף ההופכי של G, כלומר G לאחר הפיכת הכיוונים של הקשתות המגדירות אותה.

כעת עבור כל צומת $v \in V$, המסלול ממנה לצומת t הוא מסלול מצומת t אליה בגרף G^+.

נשתמש באלגוריתם $BFS(G, t)$ עם השינוי הבא-

נשמור את ערך $s.d$ במשתנה חדש בשם $distance\_t\_s$.

לאחר מכן נריץ את האלגוריתם $BFS(G^+, t)$, כך שעבור כל צומת $u \in V$ הערך המוחזר הוא- $v.d + distance\_t\_s$.

באופן זה כפי שהראינו בסעיף א, מתקבל הפלט הנדרש -

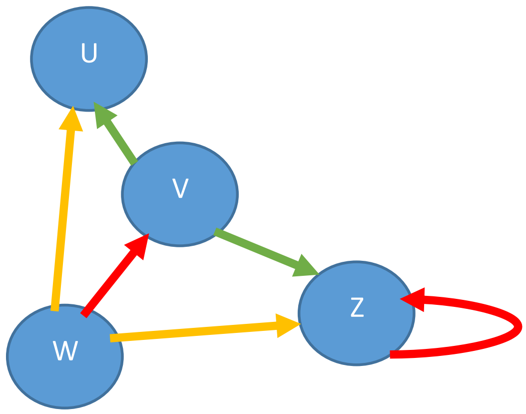
$$\min_{P_t(u,v)}\{length(P_t(u,v))\} = \delta_t(s,v) = \delta(s,t) + \delta(t,v)$$

ניתוח סיבוכיות זמן:

ראינו כי סיבוכיות זמן-ריצת אלגוריתם BFS היא$O(V + E)$

באלגוריתם המוצע סה"כ מבצעים ריצה על אלגוריתם BFS ואת פעולת הפיכת הקשתות, שמתבצעת ב- $O(n+m)$ – מתבצע מעבר יחיד על כל הצמתים והקשתות, בהתאם לאופן הגדרתם.

$3 * O(n+m) = O(n+m)$ וסיבוכיות זמן האלגוריתם היא כנדרש.



שאלה 3

הטענה אינה נכונה.

הפרכה

יהי הגרף $G = (V,E)$ מכוון ומעגלי:
בגרף מכוון לולאה עצמית היא קשת אחורית,

ובנוסף יתכנו המצבים:
(u.color=gray&v.color=white) ו/או *(z.color=gray & v.color=white)*

סה"כ קבוצת הקשתות האחוריות $F = \{(z,z),(v,z),(v,u)\}$
אך על מנת להפוך את G לDAG מספיק להסיר את הקשתות $2 = |\{(z,z),(w,v)\}| \leq |F| = 3$.

משמע יצרנו גרף מכוון חסר מעגלים על ידי הסרה של פחות מגודל קבוצת הקשתות האחוריות.
זה הוא מקרה נגדי לטענה ולכן הטענה אינה נכונה.

שאלה 4

הטענה נכונה.

$\leftarrow$:

יהי $G = (V,E)$ גרף מכוון, פשוט וחסר מעגלים DAG.
נרצה להראות כי קיימת ריצה של $DFS(G)$ בה לא מתקבלות קשתות עץ.
G גרף חסר מעגלים, לפיכך לא קיים מסלול: $p = < v_0, \ldots, v_k >$ כך ש $v_0 = v_k$ לכל $v \in V$.
בנוסף G גרף מכוון פשוט ללא לולאות עצמיות.
נניח כי קיימת קשת $e = (u,v)$ כך ש- $u.\pi = v$
כלומר e קשת אחורית, אשר אינה מובילה לצמתים אחרים בגרף (לדוגמה הגרף G מורכב מהקשת e והצמתים הנתונים בלבד)
לפיכך אם גרף G הוא DAG - גרף מכוון פשוט חסר מעגלים – אזי קיימת ריצה של $DFS(G)$ שבה לא התקבלה קשת עץ.

$\rightarrow$:

יהי $G = (V,E)$ גרף מכוון פשוט.
קיימת ריצה $DFS(G)$ כך שלא מתקבלות קשתות עץ, נרצה להראות כי G הוא DAG.
נניח בשלילה כי G אינוDAG , כלומר קיים מסלול: $p = < v_0, \ldots, v_k >$ כך ש $v_0 = v_k$, והוא מעגל בגרף.
נתון כי G חסרת קשתות עץ, כלומר לכל קשת $e = (u,v)$ מתקיים $v.color\ != white$.
לפיכך כפי שראינו בהרצאה מתקיים $v_0 = v_1.\pi$ וגם v_k צאצא של v_1, אך זאת בסתירה לכך שקיים מעגל המוגדר על ידי $v_0 = v_k$.
לכן אם קיימת ריצה $DFS(G)$ כך שלא מתקבלות קשתות עץ, G הוא DAG.

<u>ומשילוב שני הכיוונים:G הוא DAG אם ורק אם קיימת ריצה של $DFS(G)$ שבה לא מתקבלות קשתות עץ , כנדרש להוכיח ■</u>

שאלה 5

א. אופן פעולת האלגוריתם -

1. נעבור על כל צומת v in V, ונחפש צומת ללא קשתות נכנסות.
2. על הצומת מסעיף 1, נבצע הרצת DFS_VISIT
3. נעבור על כל צומת v in V ונחפש צומת w כך ש w.color = white
4. אם נמצאה כזו בסעיף 3, נדפיס הודעת "אין סיור". אחרת נחזיר את עץ הDFS שנוצר.

זמן ריצה:

1. מעבר על n צמתים, כך שעל כל צומת נבצע בדיקה ב- O(1) סה"כ O(n).
2. כפי שנלמד בהרצאה, הרצת DFS_VISIT היא O(n+m).
3. מעבר על n צמתים, כך שעל כל צומת נבצע בדיקה ב- O(1) סה"כ O(n).
4. פעולת סיום O(1).

סך הכל – סיבוכיות זמן ריצת האלגוריתם היא O(n+m) כפי שנדרש.

נכונות:

1. G הוא DAG ולכן לא יתכן שלכל הצמתים יש קשתות נכנסות. לפיכך קיימת צומת v המקיימת את התנאי.
3. נניח ונמצא באלגוריתם צומת u כך ש- u.color = white:

לפי אופן סריקת האלגוריתם ניתן להסיק כי צומת u אינו נגיש מ- v ,אחרת היינו מגיעים אליו במעבר על רשימת השכנויות של V, ו-U היה צאצא של V ביער Gπ.

מכיוון שלצומת v אין קשתות נכנסות, אם אכן היה קיים סיור בגרף הרי שצומת v היא נקודת ההתחלה שלו.
לפיכך,שילוב העובדות כי "u אינו נגיש מצומת v " ו- "סיור חייב להתחיל מ-v"
מוביל לכך כי אכן לא קיים סיור ונדפיס את התוצאה הנכונה בשלב 4.

נניח ונמצא באלגוריתם צומת u כך ש- u.color = white :

מכיוון ש-G הוא DAG אז בהכרח קיימת ריצה בה לא מתקבלות קשתות עץ - הוכח בשאלה 4.
לפיכך סיור המתחיל מצומת v בהכרח בעל מוצא בצומת u, ולכן קיים סיור ונחזיר אותו.

ב. אופן פעולת האלגוריתם -

1. נריץ את אלגוריתם SCC(G), כפי שנלמד בהרצאה.
2. נריץ את האלגוריתם בסעיף א' על G(S) שהוא DAG.
3. אם מוחזרת הודעת "אין סיור", נחזיר את ההודעה. אחרת נחזיר שקיים סיור.

זמן ריצה:

1. הראינו בהרצאה כי מתבצע בסיבוכיות זמן O(n+m)
2. מתבצע ככל היותר בסיבוכיות זמן O(n+m), והחסם אכן מתקבל במקרה בו כל צומת v ב קבוצת הצמתים V מהווה רכיב קשיר היטב בגרף G(S).

סך הכל – סיבוכיות זמן ריצת האלגוריתם היא O(n+m) כפי שנדרש.

נכונות:

האלגוריתם יחזיר גרף DAG והוא G(S), בו כל צומת מהווה רכיב קשיר היטב בגרף G המקורי, וכל קשת בין שני צמתים קשירים היטב כמתואר מסמלת כי קיימת קשת בגרף G המקורי מאחד הצמתים ברכיב הראשון אל אחד הצמתים ברכיב השני.
לכן, קיומו של סיור בגרף DAG המתקבל מעיד על קיום מסלול כלשהוא בגרף G המקורי, העובר בכל |V| הצמתים המהווים רכיבי קשירות בגרף, לפי סדר הסיור ב-G(S).
במידה ואין סיור בגרף G, לא יתקבל סיור בגרף המתואר ולכן האלגוריתם יחזיר כי אין סיור.