DFS וכיצד להשתמש בו עבור מיון טופולוגי

אלגוריתם DFS

חיפוש לעומק הידוע גם כ-Depth First Search או DFS הוא אלגוריתם חיפוש בגרף (מכוון) ולו מגוון שימושים רחב.

במימוש להלן שומרים לכל צומת v שלושה נתונים:

- $Color(v)$ - הצבע של v. יהיו 3 צבעים אפשריים[1] (בהתחלה כל הצמתים לבנים):
  - לבן – עוד לא ביקרנו ב-v
  - אפור – v נמצא במחסנית
  - שחור – v יצא מהמחסנית
- $Father(v)$ - הצומת שממנו הגענו ל-v. אם אין כזה אז הוא NULL. $Father(v)$ תמיד מאותחל ל-NULL.
- $Order(v)$ – זמן היציאה מהמחסנית של v. (לצומת שיצא ראשון מהמחסנית יתקיים Order=1, לצומת שיצא שני מהמחסנית יתקיים Order=2 וכן הלאה.)

מימוש טיפוסי:

1. $i = 1$
2. עבור כל $v \in G$:
   a. אם $Color(v) \neq white$:
      i. חזור ל-2.
   b. תהי S מחסנית ריקה
   c. $Color(v) = grey$
   d. $S.push(v)$
   e. כל עוד S לא ריקה:
      i. $v = S.top()$
      ii. אם יש צומת לבן u כך שיש קשת מ-v ל-u:[2]
         1. $Color(u) = grey$
         2. $Father(u) = v$
         3. $S.push(u)$
      iii. אחרת:
         1. $S.pop()$
         2. $Color(v) = black$
         3. $Order(v) = i$
         4. $i = i + 1$
         5. לא חובה: לעשות כאן משהו מגניב.

הערה: החלק שמתחיל מהוראה b (בצבע כחול) הוא DFS המתחיל מקודקוד v ולא בהכרח עובר בכל הצמתים בגרף. אם משלבים את DFS באלגוריתמים אחרים, לרוב מבצעים מספר פעולות בפקודה 5.

---

[1] בפועל מספיקים שני צבעים, אבל זה יקל על הניסוח.

[2] איך נמצא קודקוד כזה? לכל צומת v נחזיק מונה $couter(v)$ שבהתחלה יהיה 1 עבור כל הקודקודים. כאשר נרצה קודקוד כנ"ל, נגדיל את $counter(v)$ עד שהקשת ה-$counter(v)$ שיוצאת מ-v תגיע לקודקוד לבן. אם $counter(v)$ גדול ממספר הקשתות היוצאות מ-v, סימן שאין צומת לבן עם קשת מ-v אליו.

תכונות כלליות:

1. כל צומת ב-G נכנס ויוצא מהמחסנית בדיוק פעם אחת.
2. בסוף הריצה של האלגוריתם כל הצמתים שחורים.
3. זמן הריצה של האלגוריתם הוא $O(|V|+|E|)$ (במימוש חכם) וסיבוכיות הזיכרון היא $O(|V|)$.

תכונות הקשורות למסלולים בין הצמתים:

4. אם u בראש המחסנית ו-v מתחתיו אז $(u,v)\in E$.
5. אם u נמצא במחסנית ו-v היכנשהו מתחתיו אז יש מסלול מ-v ל-u.
6. יהיו u,v צמתים אזי התנאים הבאים שקולים:
   a. v נכנס אחרי u למחסנית ויצא לפניו.
   b. v היה לבן כש-u נכנס למחסנית ויש מסלול מ-u ל-v.

התכונה הבאה של הסדר שמשרה DFS הופכת אותו לשימושי במיוחד:

טענה: אם $Order(u)<Order(v)$ אז <u>יש מסלול מ-v ל-u</u> **או** <u>אין מסלול מ-u ל-v</u>.

הוכחה: יש שתי אפשרויות: <u>אם v הוכנס למחסנית לפני u</u> אז u נכנס אחרי v ויצא לפניו (כי $Order(u)<Order(v)$). לכן, לפי תכונה 6 יש מסלול מ-v ל-u. <u>אם v הוכנס למחסנית אחרי u</u> אז הוא היה לבן כאשר u הוכנס למחסנית (אחרת הוא לא היה מוכנס). היות ו-v גם <u>יצא</u> אחרי u אז בהכרח v נכנס למחסנית אחרי ש-u יצא ממנה. לכן, לפי תכונה 6, בהכרח אין מסלול מ-u ל-v. **משל.**

מסקנה: אם G חסר מעגלים ו- $Order(u)<Order(v)$ אז <u>אין מסלול מ-u ל-v</u>.

הוכחה: לפי הטענה הקודמת, יש מסלול מ-v ל-u או אין מסלול מ-u ל-v. במקרה השני סיימנו. במקרה הראשון נקבל שלא יתכן שיש מסלול מ-u ל-v כי G חסר מעגלים. **משל.**

שימוש ב-DFS למיון טופולוגי

המסקנה האחרונה מאפשרת להשתמש ב-DFS כדי לבצע מיון טופולוגי.

הגדרה: יהי $G=(V,E)$ גרף (מכוון). מיון טופולוגי של G הוא יחס סדר מלא $\le$ על V כך שאם $E \ni (u,v)$ מתקיים $u\le v$. סדר מלא אפשר לייצג ע"י רשימה של קודקודים $v_1,v_2,\dots,v_n$ המכילה כל קודקוד בדיוק פעם ולכל $(u,v)\in E$ הקודקוד u מופיע לפני v.

הערה: בדרך כלל יש יותר ממיון טופולוגי אחד. אם יש מעגל בגרף, לא קיים מיון טופולוגי.

טענה: יהי G גרף ללא מעגלים. נבצע DFS על G ונסדר את הקודקודים לפי $Order$ <u>מהגדול</u> <u>לקטן</u>: $v_1,v_2,\dots,v_n$ (כלומר $Order(v_1)>Order(v_2)>\cdots$). אזי $v_1,v_2,\dots,v_n$ הוא מיון טופולוגי של G.

הוכחה: תהי $(u,v)\in E$. אזי יש מסלול מ-u ל-v. לכן לפי המסקנה האחרונה לא ייתכן $Order(u)<Order(v)$ וזה אומר ש-u מופיע לפני v ברשימה $v_1,v_2,\dots,v_n$. **משל.**

פסאודו קוד לחישוב מיון טופולוגי בעזרת DFS: נחזיק את המיון הטופולוגי בתור Q.

0. יהי Q תור ריק
1. $i=1$
2. עבור כל $v\in G$:
   a. אם $Color(v)\neq white$:

i. חזור ל-2.

b. תהי S מחסנית ריקה

c. $Color(v) = grey$

d. $S.push(v)$

e. כל עוד S לא ריקה:

i. $v = S.top()$

ii. אם יש צומת לבן u כך שיש קשת מ-v ל-u:

1. $Color(u) = grey$
2. $Father(u) = v$
3. $S.push(u)$

iii. אחרת:

1. $S.pop()$
2. $Color(v) = black$
3. $Order(v) = i$
4. $i = i + 1$
5. $Q.push(v)$

3. החזר את Q

הערה: כל הפקודות המחשבות את $Father$ ואת $Order$ מיותרות, אבל ברגע שהן שם יותר קל להוכיח את נכונות האלגוריתם.