הרצאה 1

הקדמה, חיפוש בגרפים, BFS ו-DFS

הקדמה

> אלגוריתם הוא דרך שיטתית (כלומר כזו שצעדיה מוגדרים היטב) לביצוע של משימה מסוימת, במספר סופי של צעדים.

— ויקיפדיה

המושג אלגוריתם אינו חדש עבורנו, ראינו ומימשנו כבר אלגוריתמים בקורס מבוא למדעי המחשב, מבוא לתכנות מערכות ומבני נתונים.

חומר הקורס

בקורס נלמד מגוון אלגוריתמים שעל פי רוב נחשבים לבסיס בתחום האלגוריתמים. הרוב המוחלט של האלגוריתמים שנלמד בקורס הם אלגוריתמים על גרפים וזאת מכיוון שגרפים הוכיחו את עצמם ככלי מאוד חזק בייצוג מגוון רחב של בעיות. לחלק מהאלגוריתמים שימושים ברורים (מסלול קצר ביותר, עצי הופמן), חלק מהאלגוריתמים מהווים פתרון למגוון גדול של בעיות שניתנות לייצוג בצורה מסוימת (זרימה), וחלק מהאלגוריתמים מהווים בסיס לפיתוח אלגוריתמים מורכבים יותר (DFS ,BFS, עץ פורש). מעבר לזה נלמד טכניקות (פשוטות יחסית) כלליות לפיתוח אלגוריתמים.

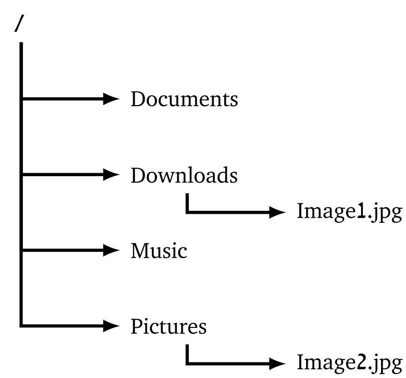
חשיבות הקורס

בתעשייה - בעיות שמצריכות פתרון אלגוריתמי צצות במגוון תחומים. על פי glassdoor, מפתח אלגוריתמים מרוויח 20 אחוז יותר מאשר מהנדס תוכנה.
במחקר האקדמי - חלק עיקרי של המחקר האקדמי הוא בפיתוח וניתוח אלגוריתמים.

אלגוריתמי חיפוש בגרפים

דוגמה 1. רוצים למצוא (ולהדפיס) את כל קבצי התמונות ששמורות על הכונן הקשיח.

למשל עבור:



כיצד עלינו לסרוק (לחפש) את מערכת הקבצים ?

1. במידה ורוצים להדפיס את (הנתיב המלא של) הקבצים בסדר לקסיקוגרפי ?
2. במידה ורוצים להדפיס קבצים לפי העומק שלהם (מספר תיקיות) ?

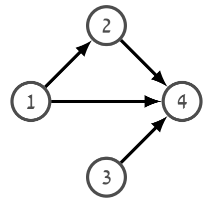
נשים לב שאפשר לייצג את מערכת הקבצים באמצעות גרף מכוון (ברוב מערכות הקבצים עץ אינו ייצוג מספק) ולכן נעביר את הדיון שלנו לחיפוש בגרפים.

ייצוג גרפים

קיימים שני ייצוגים סטנדרטים של גרפים (מכוונים או לא):

1. על ידי מטריצת שכנויות
2. על ידי רשימת שכנויות

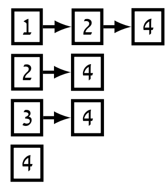
אם לא מצוין אחרת, נניח שהגרף מיוצג על ידי רשימת שכנויות.
למשל את הגרף:



ניתן לייצג על ידי המטריצה

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

וגם על ידי רשימת שכנויות:



אלגוריתם כללי

קלט: גרף G (מכוון או לא) וצומת מקור s.
מטרה: למצוא תת עץ שפורש את כל הצמתים שישיגים מ-s.
למשל:

הגדרה 1 (חתך). חתך בגרף, $G=(V,E)$, הוא תת קבוצה של צמתים. $S \subseteq V$ נאמר שקשת $uv \in E$ חוצה את החתך S אם $u \in S$ ו-$v \notin S$.

1. אתחול: $T \leftarrow \emptyset$, $U \leftarrow \{s\}$ ולכל $v \in V$ מציבים $p(v) \leftarrow \text{nil}$
2. כל עוד יש קשת uv שחוצה את U
   (א) $U \leftarrow U \cup \{v\}$, $T \leftarrow T \cup \{uv\}$, $p(v) \leftarrow u$

טענה 1. בסיום ריצת האלגוריתם U מכילה את כל הצמתים הישיגים מ-s

הוכחה. בשלילה, בוחרים מסלול מ-s לצומת v שלא נכנס ל-U ומסתכלים על הצומת הראשון במסלול שלא נכנס. □

טענה 2. בכל שלב בריצת האלגוריתם T עץ קשיר. בנוסף המסלול מצומת u ל-s הוא שרשור של הקשת $(u,p(u))$ והמסלול מ-$p(u)$ ל-s

הוכחה. באינדוקציה על צעד האלגוריתם. □

משפט 1. בסיום ריצת האלגוריתם הכללי T הוא עץ שפורש את כל הצמתים הישיגים מ-s

חיפוש לרוחב - Breadth First Search (BFS)

הגדרה 2 (מרחק). בהינתן גרף $G=(V,E)$, נגדיר את המרחק בין שני צמתים $u,v \in V$, ונסמנו $dist_G(u,v)$, כמספר הקשתות המינימלי במסלול מ-u ל-v.

הערה: כאשר ברור על איזה גרף מדובר נסתפק בסימון $dist(u,v)$.

קלט: גרף G (מכוון או לא) וצומת מקור s.
מטרה: למצוא תת עץ, T, שפורש את כל הצמתים שישיגים מ-s כך שלכל צומת $v \in V$ מתקיים $dist_T(s,v) = dist_G(s,v)$.
למשל:

1. אתחול: $i \leftarrow 0, T \leftarrow \emptyset, U \leftarrow \{s\}$, לכל $v \in V$ מציבים $d(v) \leftarrow \infty, p(v) \leftarrow nil$, $d(s) \leftarrow 0$
2. כל עוד קיים צומת u כך ש-$d(u) = i$
   (א) כל עוד ישנה קשת uv שחוצה את U ו-$d(u) = i$
      i. $U \leftarrow U \cup \{v\}, T \leftarrow T \cup \{uv\}$
      ii. $p(v) = u, d(v) = i + 1$
   (ב) $i \leftarrow i + 1$

טענה 3. בסוף ריצת האלגוריתם לא קיימת קשת uv שחוצה את U.

הוכחה. בשלילה, אם קיימת ו-$d(u) = i$ אז באיטרציה ה-i היינו מוסיפים אותה $\square$

מסקנה 1. BFS הוא מקרה פרטי של האלגוריתם הכללי

טענה 4. לכל $v \in V$ מתקיים $dist_T(s,v) = d(v)$

הוכחה. באינדוקציה על צעד האלגוריתם $\square$

טענה 5. לכל $v \in V$ מתקיים $dist_T(s,v) \leq dist_G(s,v)$

הוכחה. באינדוקציה על צעד האלגוריתם $\square$

משפט 2. לכל $v \in V$ מתקיים $d(v) \leq dist_G(s,v)$

מהו זמן הריצה של BFS ? קשה להגיד כי לא הגדרנו כיצד מתבצעת הבדיקה בשלב 2 של האלגוריתם. ניתן לממש שלב זה על ידי תור באופן הבא:

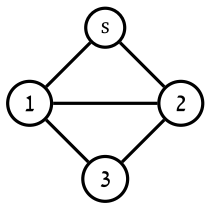
1. אתחול: $i \leftarrow 0, T \leftarrow \emptyset, U \leftarrow \{s\}$, לכל $v \in V$ מציבים $d(v) \leftarrow \infty, p(v) \leftarrow nil$, $d(s) \leftarrow 0, Q \leftarrow (s)$
2. כל עוד התור לא ריק וראש התור, u, מקיים $d(u) = i$
   (א) כל עוד ישנה קשת uv שחוצה את U
      i. $U \leftarrow U \cup \{v\}, T \leftarrow T \cup \{uv\}$
      ii. $p(v) = u, d(v) = i + 1$
      iii. $\text{Q.enqueue}(v)$
   (ב) $i \leftarrow i + 1$

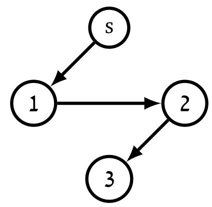
נשים לב שבמימוש הנ"ל כל צומת נכנסת ויוצאת מהתור לכל היותר פעם אחת ובנוסף כל קשת נבדקת לכל היותר פעם אחת ולכן זמן הריצה הוא $O(|V| + |E|)$

חיפוש לעומק - Depth First Search (DFS)

נגדיר $\text{dfs}(U, T, u)$

1. אם קיימת קשת uv שחוצה את U אז
   (א) $U \leftarrow U \cup \{v\}, T \leftarrow T \cup \{uv\}, p(v) \leftarrow u$
   (ב) $\text{dfs}(U, T, v)$





סיכום

דוגמה 2 (פאזל הזזה). נתון לוח משחק בגודל $n \times m$ על הלוח $nm - 1$ חלקים ממוספרים מ-1 עד $nm - 1$ ומשבצת ריקה. נתון סידור ראשוני של החלקים ואנו רוצים לסדר את החלקים לפי הסדר כך שבכל שלב מותר לנו להזיז את אחד החלקים ששכנים למשבצת הריקה אל המשבצת הריקה.

למשל עבור $n = m = 3$:

1	3	6
8	4	
5	2	7

הציעו אלגוריתם לפתרון הבעיה.