מבני נתונים - תרגול 8
עצים

גלעד אשרוב

29 באפריל 2014

לפני שנתחיל עם עצים, נראה תרגיל אחרון (לפחות לשלב הזה) בגרפים...

תרגיל 1. נתון גרף מכוון $G=(V,E)$ על ידי מטריצת שכנויות. הציעו אלגוריתם יעיל המוצא קודקוד $v\in V$ שדרגת הכניסה שלו היא $|V|-1$ ודרגת היציאה שלו היא 0. על האלגוריתם לרוץ בזמן $O(|V|)$.

פתרון: נקרא לקודקוד כזה "בור" (רק נכנסים אליו קשתות, ולא יוצאים ממנו). נשים לב שיכול להיות לכל היותר רק בור אחד בגרף (למה?).

נשים לב שאם קיימת קשת $(u,v)\in V$, אזי u אינו קשת (שכן יוצאת ממנו בור), ומצד שני, ייתכן ו-v בור. כלומר, ע"י בדיקת קשת אחת, אנחנו יכולים לשכוח מהקודקוד u. למעשה, אנחנו מחפשים אינדקס i, כך שהשורה i היא כולה אפסים, והעמודה ה-i היא כולה אחדות (מלבד האינדקס (i,i)).

- נאתחל אינדקסים $i=1,k=1$. נגדיר משתנה `כיוון` המקבל ערכים $\{$ימינה, למטה$\}$. ערך התחלתי יהיה ימינה.
- כל עוד i או k קטנים מ-n:
  - נבדוק את המיקום (i,k).
    * אם 0 - ייתכן והשורה i היא בור. נקדם את k באחד.
    * אחרת, אם 1 - ייתכן והעמודה ה-k היא בור. נקדם את i באחד.
- כאשר סיימנו את הלולאה, אחד המשתנים הגיע ל-n. נתבונן במשתנה השני ונקרא לו j. j הוא המועמד היחיד להיות בור. נבדוק האם באמת השורה ה-j היא שורת אפסים, והעמודה ה-j היא עמודת אחדות (מלבד המיקום (j,j)).

זמן ריצה: כל אחד מהאינדקסים i,k אינו מתקדם מעבר ל-n. בנוסף, הבדיקה בשלב האחרון עולה $O(n)$. לכן, סה"כ נשלם $O(n)$.

1 עצים

עד כה ראינו בעיקר מבנה נתונים לינאריים. ראינו שהמבנים הללו – גם רשימה ומקושרת, וגם מערך – שניהם לא גמישים מספיק. מערך – עלות גבוהה להיכנס להכניס איבר באמצע המערך, וברשימה - קשה לגשת לאיברים באמצע המערך. ראינו גם "רשימת דילוגים", שנתנה לנו גמישות מסויימת, אבל העלות בה היא מבוססת על רנדומיות[1], ולכן לפעמים נוכל להגיע לחוסר איזון (בהסתברות נמוכה).

בשיעורים הבאים נלמד - עצים. אלו מבנה נתונים שבהם יהיה קל (יחסית) להכניס איברים, וגם יהיה קל למצוא אותתם. נעבור להגדרה:

[1] ישנה גם גירסא דטרמיניסטית, אבל היא מסובכת יותר ולא ראינו אותה..

הגדרה 2. עץ בינארי מוגדר בצורה רקורסיבית:

- עץ בינארי הינו עץ ריק (ללא צמתי), או:
- עץ המורכב משורש ושני תתי עצים בינאריים, ימני ושמאלי.

בהינתן שורש, אפשר לדבר על עומק של קודקוד (המרחק מהשורש). שתי הגדרות נוספות:

- **עץ בינארי מלא:** עץ בינארי שכל קודקוד בו הוא או עלה, או שיש לו בדיוק שני בנים.
- **עץ בינארי שלם:** עץ בינארי מלא שבו כל העלים נמצאים בעומק d. בצורה זו, בעץ יש $2^0+2^1+\dots 2^d = 2^{d+1}-1$ קודקודים בסה"כ.

לעץ בינארי מלא, ניתן לתת גם את ההגדרה (הרקורסיבית) הבאה: עץ בינארי מלא הוא:

- עלה, או
- עץ המורכב משורש ושני תתי עצים בינאריים מלאים.

שימו לב שכאן מכיוון ש"תנאי" העצירה הוא קודקוד בודד (עלה), בהכרח לכל קודקוד יש או שום בן (עלה), או שני בנים (ולא אחד. בהגדרה המקורית של עץ בינארי - יכל להיות לקודקוד מסוים בעץ בן שהוא עץ ריק - ולכן רק בן יחיד).

תרגיל 3. הוכח כי בעץ בינארי מלא עם n עלים יש $n-1$ קודקודים פנימיים.

לפני שנוכיח, נשים לב להוכחה **לא נכונה**.

הוכחה (שגויה!): בסיס - אם בעץ יש קודקוד בודד, אזי יש בו עלה בודד ו-0 קודקודים פנימיים.
נניח שהטענה מתקיימת עבור עץ בינארי מלא עם k עלים, כלומר, עבור עץ כזה יש $k-1$ קודקודים פנימיים. נראה שהטענה מתקיימת עבור עץ עם $k+1$ עלים. לשם כך, ניקח עץ מלא עם k עלים (שכאמור, עפ"י ההנחה - יש בו $k+1$ קודקודים פנימיים) נקח איזשהו עלה ונהפוך אותו לקודקוד פנימי ע"י כך שנוסיף לו שני בנים. קיבלנו כמובן - עץ מלא. מספר העלים גדל באחד (לקחנו עץ עם k עלים, הורדנו עלה אחד (שהפכנו אותו לקודקוד פנימי), והוספנו לו שני בנים - שני עלים). מספר הקודקודים הפנימיים גדל באחד. קיבלנו עץ עם $k+1$ עלים, ו-k קודקודים פנימיים. ■

ההוכחה הזו - לא נכונה! בנינו בצעד האינדוקציה עץ מסויים בעל $k+1$ עלים. בכדי שההוכחה תהיה נכונה נצטרך להראות שתמיד, הדרך היחידה לקבל עץ בעל $k+1$ עלים היא ע"י אותה הבנייה כמו שעשינו בהוכחה. הדרך הנכונה להוכיח זאת היא ע"י לקיחת עץ כלשהו מגודל $k+1$, הפיכתו לעץ בגודל k, הפעלת הנחת האינדוקציה על עץ בגודל k והסקת המסקנה על עץ בגודל $k+1$. נראה הוכחה אחרת:

הוכחה: (אינדוקציה רגילה) בסיס - כמו למעלה. הנחה - כמו למעלה.
כעת, ניקח עץ מלא כלשהו בעל $k+1$ עלים. נראה שיש לו k קודקודים פנימיים. לשם כך, נתבונן בקודקוד (פנימי) ששני ילדיו הם עלים (בהכרח קיים אחד כזה). נמחוק את שני בניו (את שני העלים). ע"י כך, נהפוך את הקודקוד הפנימימי הנ"ל לעלה.
כעת, בעץ המקורי היו $k+1$ עלים. הורדנו שני עלים ו"הוספנו" עלה אחד, כך שבסה"כ קיבלנו עץ בעל k עלים. על עץ כזה, לפי הנחת האינדוקציה, יש בדיוק $k-1$ קודקודים פנימיים. מכיוון שהפכנו קודקוד פנימי לעלה, נקבל שבעץ המקורי היו k קודקודים פנימיים, כפי שהיינו צריכים להוכיח. ■

לבסוף, נראה הוכחה יותר פשוטה בעזרת אינדוקציה על מבנה העץ:

1.1 אינדוקציה על מבנה העץ

בצורה זו יותר קל להוכיח טענות מהסוג שרק ראינו. הנחת האינדוקציה תעשה כמו אינדוקציה רגילה, כלומר, אינדוקציה על מספר הקודקודים, או, על מספר העלים, או על גובה העץ. ההבדל יהיה בצעד האינדוקציה - שנוכיח אותו ישירות מההגדרה של מבנה העץ, או מההגדרה הרקורסיבית של מהו עץ בינארי. לפני שנתחיל, ניזכר בהגדרה הרקורסיבית של עץ מלא: עץ מלא הוא או עלה, או שורש עם שני בנים שהם עצים מלאים. כעת, נוכיח שוב את הטענה:

הוכחה: (אינדוקציה על מבנה העץ) בסיס: לעץ בעל עלה יחיד (1) אין צמתים פנימיים (0).

צעד: נניח שעבור כל העצים המלאים בעלי $k < n$ עלים יש $k-1$ קודקודים פנימיים.

נוכיח עבור העצים המלאים בעלי n עלים:

יהי עץ מלא בעל n עלים. מכיוון שהוא עץ מלא, לכל קודקוד שני בנים, ובפרט לשורש. נתבונן בבנים של השורש. העץ הימני הוא עץ מלא בעל n_1 עלים, ו $n_1 - 1$ קודקודים פנימיים. העץ השמאלי הוא עץ מלא בעל $n - n_1$ עלים, ובעל $n - n_1 - 1$ קודקודים פנימיים. לכן, בסה"כ הקודקודים הפנימיים בעץ (כולל השורש) הוא:

$$n_1 - 1 + n - n_1 - 1 + 1 = n - 1$$

והוכחנו את הטענה. ■

1.2 תרגיל נוסף

תרגיל 4. בהינתן גרף $G = (V, E)$ ובהינתן זוג קודקודים $u, v \in V$, נגדיר את $d(u, v)$ להיות אורך המסלול הקצר ביותר בין u ל- v (נאמר שהמרחק הוא ∞ אם אין כזה). נגדיר את הקוטר ($diameter$) של הגרף להיות:

$$D(G) = \max_{u,v} d(u, v)$$

בשאלה זו נחקור את הקוטר של עץ בינארי T.

1. הוכח או הפרך את הטענה הבאה: אם קוטר של עץ בינארי T (בעל לפחות 2 קודקודים) הוא D, אזי בהכרח קיים מסלול באורך D העובר דרך השורש.
2. כתוב אלגוריתם המקבל כקלט עץ, ומחזיר את הקוטר של הגרף.

פתרון: הטענה אינה נכונה. קל לראות שאם לשורש יש בן שמאלי יחיד, ובן ימני שהוא עץ שלם, קוטר הגרף יהיה פעמיים עומק העץ השלם, והוא גדול (כמעט) פי שתיים מכל המסלולים שעוברים דרך השורש.

הטענה אבל "כמעט" נכונה: קל לראות שקוטר הגרף הוא בעצם המקסימום של שלושת האפשרויות הבאו: (1) עומק תת העץ השמאלי + עומק תת העץ הימני + 2, (2) הקוטר של תת העץ הימני, (3) הקוטר של תת העץ השמאלי.

נכתוב אלגוריתם לחישוב קוטר, המשתמש בפונקציית העזר "חשב עומק וקוטר" שמחזיר זוג ערכים. האלגוריתם לחישוב קוטר ייקרא לפונקציית העזר הנ"ל על השורש, יקבל חזרה את הקוטר ואת העומק, ופשוט יחזיר את הקוטר.

הפונקציה לחישוב עומק וקוטר מקבלת עץ בינארי ועובדת בצורה הבאה:

- אם העץ הוא עלה, החזר עומק - 0, וקוטר - 0.
- אחרת, קרא רקורסיבית לתת העץ הימני ולתת העץ השמאלי. קבל שני זוגות של ערכים - (קטור ימני, עומק ימני), (קוטר שמאלי, עומק שמאלי).
- החזר כעומק את המקסימום מבין שני העומקים $+1$. החזר כקטור את המקיסמום מבין הערכים הבאים: (1) עומק תת העץ השמאלי + עומק תת העץ הימני + 2, (2) הקוטר של תת העץ הימני, (3) הקוטר של תת העץ השמאלי.

1.3 סריקות בעצים בינאריים

ראינו סריקות בגרפים כלליים BFS, DFS. מכיוון שלכל קודקוד יש לכל היותר שני בנים, סריקות בגרפים אלו הרבה יותר פשוטים מאשר בגרפים כלליים. ישנם שלושה סוגים לסריקות בעץ בינאריים:

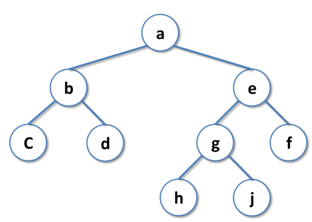
- ***preorder:*** סרוק את השורש. הפעל $preorder$ על תת העץ השמאלי (בצורה רקורסיבית). הפעל $preorder$ על תת העץ הימני.
- ***inorder:*** הפעל $inorder$ על תת העץ השמאלי. סרוק את השורש. הפעל $inorder$ על תת העץ הימני.
- ***postorder:*** הפעל $postorder$ על תת העץ השמאלי. הפעל $postorder$ על תת העץ הימני. סרוק את השורש.

תרגיל 5. נתונות סריקות $inorder$ ו-$preorder$ של עץ בינארי:

- $preorder$: $a, b, c, d, e, g, h, j, f$.
- $inorder$: $c, b, d, a, h, g, j, e, f$.

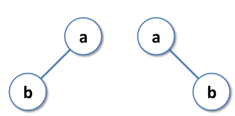
שחזרו את העץ.

פתרון: האיבר שנראה ראשון ב-$preorder$ הוא השורש. לכן, שורש העץ הוא a. כעת, נתבונן בפלט ה$inorder$, ונדע בדיוק אלו קודקודים הם בתת העץ הימני, ואלו בתת העץ השמאלי. נפעיל רקורסיבית בצורה דומה. נקבל את העץ:



תרגיל 6. הוכיחו או הפריכו: תמיד ניתן לשחזר עץ בינארי על סמך סריקות ה-$preorder$ וה-$postorder$ שלו.

פתרון: לא נכון. נראה דוגמא נגדית: תוצאת סריקת ה-$perorder$, היא a, b בשני העצים, ותוצאות ה-

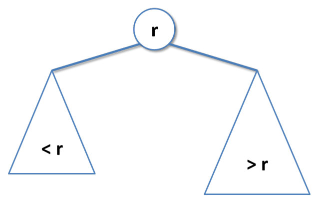


$postorder$ היא b, a בשני העצים.

אנחנו לומדים שבכדי לשחזר את העץ, אנחנו זקוקים לסריקת $inorder$ בנוסף לסריקה אחרת כלשהי ($preorder$ או $postorder$).

2 עץ חיפוש בינארי

נשמור את המפתחות בעץ בצורה הבאה: בכל תת עץ, כל המפתחות בתת העץ הימני קטנים מהשורש, וכל המפתחות בתת העץ השמאלי - גדולים מהשורש. קיבלנו למעשה מבנה נתונים סטאטי, שבו ניתן לבצע חיפוש בצורה יעילה: בכדי לחפש מפתח k נתחיל בשורש. אם האיבר שלנו קטן מהמפתח בשורש - נחפש בצורה רקורסיבית בתת העץ השמאלי. אם המפתח גדול מהמפתח בשורש - נחפש בתת העץ הימני.



תרגיל 7. הראה שניתן לשחזר עץ חיפוש בינארי בהינתן תוצאת ריצת ה-$postorder$ שלו בלבד.

הוכחה: בהינתן תוצאת ה-$postorder$, שורש העץ יופיע אחרון. בהינתן השורש, אנחנו יודעים שכל המפתחות שקטנים ממנו יימצאו בתת העץ השמאלי, וכל המפתחות שגדולים ממנו נמצאים בתת העץ הימני. נעבור על תוצאת ה$postorder$ ונחפש את כל המפתחות שקטנים ממנו. מכיוון שבסריקה הדפסנו קודם את תת העץ השמאלי, המפתחות הנ"ל יימצאו בתחילת פלט $postorder$, אך הפעם נדע בדיוק מתי מסתיים תת העץ השמאלי (בניגוד למקרה שהעץ הוא לא עץ חיפוש, ששם לא ידענו אילו קודקודים שייכים לתת העץ השמאלי). בצורה דומה, נדע בדיוק מהו תת העץ הימני. נפעיל רקורסיבית על כל צד, נקבל שני תתי עצים, ונסיים. ■

2.1 עצי חיפוש מאוזנים

מהי עלות החיפוש בעץ חיפוש בינארי? התשובה תלויה כמובן באיזון של העץ. עד כמה העץ הוא מאוזן. אם גובה העץ הוא $O(\log n)$, עלות חיפוש תהיה $O(\log n)$, שכן אנחנו מבצעים פעולה בודדת בכל רמה במהלך החיפוש. בזאת סיימנו את הגדרת מבנה הנתונים הסטאטי: בהינתן הנתונים, נבנה עץ חיפוש עליהם, כך שסה"כ הרמות יהיו $\log n$. אנחנו יודעים לייצר עץ חיפוש כזה בצורה מושלמת בהינתן הנתונים.

בהמשך הקורס, נראה כיצד מבצעים הכנסה והוצאה לעץ חיפוש, תוך שמירת התנאי הבא: גובה העץ תמיד $O(\log n)$. ע"י מימושים שונים של הכנסות והוצאות, נקבל עצים שונים, לדוגמא - עץ אדום שחור, עץ AVL ועוד.

תרגיל 8. הוכיחו את המשפט הבא: בעץ חיפוש בינארי, סריקת $inorder$ מדפיסה את כל הערכים בצורה ממויינת (עולה).

הוכחה: נוכיח באינדוקציה על מבנה העץ (על מספר הקודקודים בעץ).

עבור עץ ריק ועבור עץ עם קודקוד בודד - הטענה ודאי מתקיימת.

כעת, נוכיח שהטענה מתקיימת עבור עץ עם $k < n$ קודקודים.

נוכיח עבור עץ עם n קודקודים: נתבונן בשורש. מכיוון שהעץ הוא עץ חיפוש, כל המפתחות עם ערכים קטנים מהמפתח בשורש נמצאים בתת העץ השמאלי, וכל המפתחות עם ערכים גדולים ממפתח השורש נמצאים בתת העץ השמאלי. נריץ $inorder$ על העץ. לפי אלגוריתם $inorder$, אנחנו מפעילים $inorder$ על תת העץ השמאלי. בעץ זה פחות מ-n קודקודים, ולכן ניתן להפעיל עליו את הנחת האינדוקציה, ונקבל בצורה ממויינת את כל הערכים שקטנים מהשורש. כעת, עפ"י $inorder$ אנחנו מבקרים בשורש, ושוב - נפעיל רקורסיבית על תת העץ השמאלי, שעל פי הנחת האינודקציה יחזיר לנו סדרה ממויינת של המפתחות שגדולים מהשורש. קיבלנו אם כן שהרצת $inorder$ מחזירה לנו בצורה ממויינת את כל המפתחות שקטנים מהשורש, את המפתח בשורש, ולאחר מכן את כל המפתחות שגדולים מהשורש בצורה ממויינת, ולכן בסה"כ הפלט ממויין. ■