

סיכום משפטים ואלגוריתמים

14 בפברואר 2017

1. משוואת master

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

- (א) $f(n) < O(n^{\log_b a}) \Rightarrow T(n) = O(n^{\log_b a})$
(ב) $k \geq 0 \wedge f(n) = O(n^{\log_b a} \log^k n) \Rightarrow T(n) = O(n^{\log_b a} \log^{k+1} n)$
(ג) $f(n) > O(n^{\log_b a}) \wedge f\left(\frac{n}{b}\right) \leq kf(n) \wedge k < 1 \Rightarrow T(n) = O(f(n))$

2. מבני נתונים: כולם זמינים בקישור הנפלא הבא <https://www.cs.usfca.edu/~galles/visualization/Algorithms.html>

- (א) תור: האיבר האחרון שנכנס הוא האיבר האחרון שיוצא
(ב) מחסנית: האיבר האחרון שנכנס הוא האיבר הראשון שיוצא
(ג) רשימה מקושרת: לכל תא יש מצביע לתא שאחריו, בצורה זו מימוש מחסנית ותור דורש סיבוכיות $O(1)$ להכניס ולהוציא אברים (במערך צריך להזיז את כל האיברים).
(ד) עץ: אין מסלולים סגורים. יש קודקוד שורש, צמתים ועלים.
(ה) עץ בינארי: לכל קודקוד עד שני ילדים
(ו) ערימת מינימום: עץ בו האב קטן מילדיו. הכנסה והוצאה היא לוגריתמית (בהנחה שגובה הערימה הוא לוגריתמי) מציאת מינימום היא בסיבוכיות קבועה
(ז) עץ חיפוש בינארי: הילד הימני קטן מהאבא שקטן מהילד השמאלי סיבוכיות מציאת איבר הוצאה והכנסה היא לוגריתמית (אם העץ בגובה לוגריתמי)
(ח) עץ חיפוש 2-3 (ישנם כמה הגדרות): לכל קודקוד יש ערך אחד ועד שני ילדים או 2 ערכים ושלוש ילדים. העץ תמיד מאוזן (גובה תמיד לוגריתמי) סיבוכיות לוגריתמית לחיפוש הכנסה והוצאה
(ט) ערימת פיבונצ'י: קבוצת ערימות מינימום. הוצאת מינימום בסיבוכיות לוגריתמית, הכנסה בזמן קבוע.
(י) עץ avl עץ חיפוש מאוזן: חיפוש הכנסה והוצאה כולם לוגריתמים

3. מיון: bubble cocktail quicksort heap insertion library merge buckt radix

4. עץ פורש מינימלי: עץ המכיל את כל הקודקודים של הגרף הנתון עם כמות מינימלית של קשתות

(א) אלגוריתמים למציאתו: prim kruskel reverse-delete

5. סדר מעבר על עצים: (כלומר סידור של הקודקודים)

(א) סדר bfs ביקור כל הקודקודים במרחק 1 מהשורש ואז ביקור בכל הקודקודים במרחק 2 וכך הלאה.

(ב) סדר dfs ביקור ענף עד עלה, אחר כך ביקור בענף אחר עד העלה שלו וכך הלאה

(ג) סדר preorder אב, ילד שמאל, ילד ימין

(ד) סדר inorder ילד שמאל, אב, ילד ימין (הערה: בעץ חיפוש בינארי נקבל את הקודקודים לפי הערכים שלהם)

(ה) סדר postorder ילד שמאל ילד ימין אב

6. מציאת המרחק (אורך מסלול הקצר ביותר בין שני קודקודים)

(א) אלגוריתם bfs עובר על הקודקודים לפי המרחק מהשורש. בהנחה של משקל זהה לכל הקשתות והוא מחשב מרחק מהשורש לכל שאר הקודקודים

(ב) אלגוריתם dijkstra המשקלים לא שליליים,

(ג) אלגוריתם bellman ford : המשקלים יכולים להיות שליליים

(ד) אלגוריתם floyd-warshall מוצא מרחק בין כל הקודקודים לכל הקודקודים

7. דחיסה

(א) אלגו LZW (שני גרסאות lz77 lz78) : lz78 מקבל מחרוזת ובונה ממנה עץ (שמאוחסן בצורה יעילה) וקריאת ענפי העץ לפי סדר העלים נותנת את המחרוזת המקורית

(ב) הופמן : מחשבים את התדירויות של כל אות במחרוזת, לפי זה בונים עץ קידוד שמאפשר להמיר כל אות למספר המתאים לה, ואז להציג את המחרוזת עם המספרים האלה. (החסם תחתון לדחיסה הוא האנטרופיה של השכיחויות)

8. התאמת מחרוזות

(א) אלגו kmp מחשב את ה prefix function המתאימה למילה אותה מחפשים. לפי זה הוא מחפש את המילה במחרוזת

(ב) אלגו kr מחשב ערך גיבוב של המילה אותה מחפשים. וכאשר עוברים על המחרוזת לכל מיקום במחרוזת מחשבים את ערך הגיבוב שלו ומשווים לערך הגיבוב של המילה

(ג) אלגו boyer-moore עם כללי דילוגים במחרוזת הסיפא הטובה, האות הרעה.

9. עצי סיפא: כלומר לכל קשת מתאימים תת-מילה. כל ענף מתאים לסיפא.

(א) הבניה של אקונון: מוסיפים בהדרגה אותיות של המילה, אחרי כל הוספה נקבל עץ סיפא חלקי המתאים לכל האותיות עד אותה אות.

10. תכנון דינמי: בדרך כלל מזהים נוסחה רקורסיבית שמתאימה לבעיה אותה מנסים לפתור. ומזהים שבנוסחה הרקורסיבית יש חישובים שחוזרים על עצמם. מקרה קיצון הוא הפרד-ומשול שבו הרקורסיבה מפרקת את הביה לשתי תתי-בעיות בלי שום חפיפה ביניהם.

11. סימפלקס:

(א) בהינתן בעיה מהצורה הבאה: (כל האותיות הקטנות הם וקטורי עמודה אותיות גדולות מטריצות, תג מסמל שיחלוף)

$$\begin{aligned} \max c'x \\ Ax \leq b \\ x \geq 0 \end{aligned}$$

(ב) ממירים את האילוצים לשיוויונות על ידי הוספת משתני slack :

$$\begin{aligned} Ax + s = b \\ s \geq 0 \end{aligned}$$

(ג) רושמים בטבלה את כל האילוצים ומבצעים דירוג של גאוס לפי סדר מסוים של בחירת איבר ציר

(ד) סימפלקס דואלי של הבעיה הפרימלית בסעיף a:

$$\begin{aligned} \min b'y \\ A'y \geq c \\ y \geq 0 \end{aligned}$$

(ה) קשרים בין הבעיות:

i. כל פתרון של הדואלי (כלומר מצאנו y שמקיים את האילוצים והצבנו ב $b'y = D$) תמיד גדול שווה מהפתרון של הפרימלי (מצאנו x שמקיים את

$$D \geq P : (c'x = P$$

ii. אם מתקיים שוויון $D = P$ אז בהכרח ה- x וה- y שמצאנו הם פתרון אמיתי.

iii. נשים לב שמספר המשתנים (גודל הוקטור x) בפרימלי שווה למספר משתני ה slack בדואלי. ואכן יש ביניהם התאמה: אם המשתנה ה- i בפרימלי הוא חיובי אז בהכרח משתנה ה slack המתאים לו בדואלי הוא אפס (ולחפך עבור הדואלי).

12. אלגוריתם fft : בהינתן וקטור באורך n , טרנפורם פוריה של הוקטור הוא:

$$\begin{aligned} F_n(v) &= A_n v \\ (A_n)_{j,k} &= \frac{1}{\sqrt{n}} e^{-2\pi i(j-1)(k-1)/n} \end{aligned}$$

(א) ניתן לחשב את הטרנספורם פוריה השל הוקטור בזמן $n \log n$ עם אלגוריתם fft
 (ב) מהסיבה הזאת ניתן לבצע מכפלה של פולינומים באותה סיבוכיות: נניח שנתונים שני וקטורים p_1, p_2 עם וקטורי מקדמים (נניח שדרגת הפולינומים היא n) v_1, v_2 שגודלם $2n$ (החצי הראשון שלהם אפסים). אזי וקטור המקדמים של המכפלה v מקיים:

$$v = F_{2n}^{-1}(F_{2n}(v_1) \cdot F_{2n}(v_2))$$

- i. כאשר F^{-1} הוא הטרנספורם ההפוך והוא כמעט זהה לפוריה הישר-בפרט יש לו אותה סיבוכיות
- ii. כאשר הנקודה \cdot מסמלת מכפלה אבר-אבר של שני וקטורים.
- iii. ישנן בעיות (כמו שראינו בתרגול) שניתן לנסח אותם מחדש כמקדמי פולינומים מסויימים. ומכאן התועלת שבשימוש ב-fft