

# מבני נתונים - תרגול 5

## רשימת דילוגים - Skip-List

גלעד אשרוב

28 במרץ 2014

### תקציר

בתרגול זה נלמד מבנה נתונים חדש המשלב מעין חיפוש בינארי עם רשימה מקושרת. כך, אנו מקבלים גם את הגמישות של רשימה מקושרת (בכל הקשור להכנסת איבר לאמצע הרשימה), בנוסף לחיפוש וגישות מהירות אל איברים בתוך הרשימה, כמו במערך.

### 1 השלמה - נוסחאות נסיגה - החלפת משתנים

נתבונן בדוגמא הבאה:

$$\begin{aligned}T(n) &= 2T(\sqrt{n}) + \log n \\T(2) &= 1\end{aligned}$$

**פתרון בעזרת החלפת משתנים.** נגדיר משתנה חדש:  $m = \log n$ . נשים לב שמתקיים בעצם:  $n = 2^m$ , ולכן:

$$\sqrt{n} = n^{1/2} = (2^m)^{1/2} = 2^{m/2}$$

לכן, ניתן לכתוב את הנוסחא הרקורסיבית שלנו בצורה הבאה:

$$\begin{aligned}T(2^m) &= 2T(2^{m/2}) + m \\T(2) &= 1\end{aligned}$$

כעת, נגדיר  $S(m) \stackrel{\text{def}}{=} T(2^m)$ . לכן, מתקיים גם:  $S(m/2) = T(2^{m/2})$ . נקבל:

$$S(m) = 2S(m/2) + m$$

בנוסף, עבור תנאי העצירה, מתקיים  $S(1) = T(2^1) = T(2) = 1$ . כלומר, למעשה קיבלנו את הנוסחא הבאה:

$$\begin{aligned}S(m) &= 2S(m/2) + m \\S(1) &= 1\end{aligned}$$

נשים לב שכבר פתרנו את התרגיל הנ"ל, בתחילת חלק 3, והפתרון היה  $O(m \log m)$ . כלומר, קיבלנו:

$$S(m) = T(2^m) \in O(m \log m)$$

נציב חזרה  $m = \log n$  (או  $2^m = n$ ), נקבל:

$$\begin{aligned}T(2^m) &= O(m \log m) \\T(n) &= O(\log n \log \log n)\end{aligned}$$

## 2 רשימת דילוגים - מוטיבציה

בניגוד למבני נתונים שנלמד בהמשך הקורס, שבהם נתקל בפרטים קטנים שלפעמים קשים לזכירה - רשימת דילוגים הוא מבני נתונים פשוט מאוד, וקל לזכור כיצד הוא פועל (אם כי, ניתוחי זמן הריצה עלולים להיות מעט מסובכים).

מבני הנתונים הזה הוא כנראה גם החדש ביותר שנלמד בקורס, ופורסם ע"י Pugh ב-1990. ראה [?]. נתבונן בפעולה  $search(x)$  - מצא את האיבר  $x$  במבנה הנתונים. אם נשמור את הנתונים בתוך מערך - ונשמור תמיד שהמערך יהיה ממוין, חיפוש יעלה לנו  $O(\log n)$  (ניתן לבצע חיפוש בינארי). גם אם  $x$  לא נמצא בקבוצה, נוכל להחזיר את הקודם או העוקב במבנה הנתונים.

הבעיה במערך היא הכנסה והוצאה. כל עוד שומרים שהמערך ממוין, בכדי להכניס איבר בין שני איברים - יש צורך "להזיז" את שאר האיברים במערך, מה שיכול להעלות  $O(n)$ . כנ"ל לגבי הוצאה.

אם עוברים לרשימה מקושרת - בעיית ההכנסה וההוצאה נפתרת. אך כעת, גם כאשר הרשימה ממויינת - לא ניתן לבצע חיפוש איבר בזמן  $O(\log n)$ . זאת מכיוון שלא ניתן לבצע גישה ישירה לאיבר במקום ה- $i$  (רשימה מקושרת לא תומכת ב-*random access*). אם כן, נרצה לשפר את החיפוש ברשימה מקושרת; נרצה לבצע מעין חיפוש בינארי - ברשימה מקושרת. זהו הרעיון שעומד מאחורי רשימת דילוגים. מימוש מעין *random access* ברשימה מקושרת.

נדבר כרגע רק על מבנה נתונים סטטי. כלומר - כל האיברים כבר נתונים וידועים מראש, והפקודה היחידה שעליה אנו מדברים היא  $search(x)$  - פעולת שאילתא. נתבונן בפעולה זו ברשימה מקושרת ממויינת, ונרצה להוסיף "מידע נוסף" שיעזור לנו לבצע את החיפוש. נעיר שהסיבה היחידה שבה אנו מדברים על רשימת דילוגים ולא על מערך היא מכיוון שאנו בכל זאת רוצים לדבר על *insert* ו-*delete*, ולכן נרצה בהחלט לתמוך בפעולות אלו. בכל אופן, בשלב ראשון, בזמן שאנו חושבים רק על הרעיון שמאחורי מבני הנתונים, נתמקד רק ב-*search*.

## 3 רעיון ראשון - שתי רשימות

כאשר אנחנו מבצעים חיפוש בינארי במערך, אנחנו נכנסים לאיברים באמצע המערך. איך ניתן לעשות זאת ברשימה מקושרת? הרעיון הראשון הוא להחזיק - שתי רשימות מקושרות. נתבונן ברשימה מקושרת, שבה כל איבר מצביע לאיבר שאחריו, ולאיבר שלפניו (רשימה מקושרת דו כיוונית). נוסיף רשימה מקושרת נוספת מעליו. רשימה זו תהיה רשימה של "נציגים" של הרשימה המקורית. לא נשמור את כל האיברים ברשימה זו, אלא רק נציגים מהרשימה שמתחתיה. כל נציג יצביע לאיבר שמתחתיו - לאיבר האמיתי. כך, כאשר נרצה לבצע חיפוש, נבצע חיפוש רגיל ברשימה מקושרת ברשימת הנציגים. כאשר נמצא את הקודם והעוקב של האיבר אותו אנו מחפשים, נרד לרשימה מתחתיה (שבה כל האיברים) - ושם נחפש את האיבר עצמו, בין שני הנציגים.

כמה נציגים כדאי לבחור? מה המרחק בין כל שני נציגים?

לשאלה השנייה נענה כרגע - מרחק אחיד (נעיר שזוהי תשובה אופטימלית בסדר גודל, אם כי, ניתן לשפרה בקבועים - תרגיל?). באשר לשאלה הראשונה, נתבונן בעצם כמה עולה לנו חיפוש. תהי  $L_1$  הרשימה המלאה, ותהי  $L_2$  רשימת הנציגים. חיפוש איבר  $x$  במבנה הנתונים שיצרנו עולה:

$$T(search_2) = |L_2| + \frac{|L_1|}{|L_2|}$$

כלומר, במקרה הגרוע ביותר אנו עוברים על כל רשימת הנציגים. לאחר מכן, אנו עוברים על חלק מהרשימה המלאה. מכיוון שחילקנו את הנציגים באופן אחיד, המרחק בין כל שני נציגים הוא  $|L_1|/|L_2|$  (אם נניח ברשימה המלאה יש 20 איברים, וברשימת הנציגים יש 4 איברים, בין כל שני נציגים יש בערך 5 איברים; בכל מעבר ברשימת הנציגים, אנו "מדלגים" על 5 איברים אמיתיים).

מתי ביטוי זה מקבל מינימום? בכדי לגלות - נגזור את הביטוי. נסמן את  $|L_1|$  ב- $n$  (מספר האיברים ברשימה הגדולה), ואת  $|L_2|$  ב- $k$ .

$$T(search_2) = k + \frac{n}{k}$$

כאמור, אנחנו רוצים לדעת מהו מספר הנציגים האופטימלי, ולכן נגזור את הביטוי לפי  $k$ . נקבל:

$$T'(search_2) = 1 - \frac{n}{k^2}$$

נשווה ל-0, ונקבל כי:  $k^2 = n$ , כלומר,  $k = \sqrt{n}$ . במילים אחרות, כאשר רשימת הנציגים היא  $\sqrt{n}$ , עלות החיפוש היא:  $\sqrt{n} + n/\sqrt{n} = 2\sqrt{n}$ , וכאשר עוברים בין מנציג אחד לשני מדלגים על  $\sqrt{n} = n/\sqrt{n}$  איברים אמיתיים. למעשה, הצלחנו להוריד את זמן החיפוש מ- $n = |L_1|$  ל- $2\sqrt{n}$ . סיבוכיות המקום עלתה מ- $n$  ל- $n + \sqrt{n} = O(n)$ , כלומר - אין הבדל בסדר גודל בכל הקשור לסיבוכיות מקום. כבר יש לנו שיפור משמעותי.

### Ideal Skip List 3.1

נמשיך לבנות רשימת נציגים לנציגים, נקבל שלוש רמות. נניח שכל רשימה של נציגים מחולקת באופן אחיד. נקבל אם כן, שימך החיפוש בשלוש רשימות הינו:

$$T(search_3) = |L_3| + \frac{|L_2|}{|L_3|} + \frac{|L_1|}{|L_2|}$$

ביטוי זה מקבל מינימום כאשר  $|L_3| = \sqrt[3]{|L_1|}$ ,  $|L_2| = \sqrt[3]{|L_1|^2}$  ואז נקבל:  $T(search_3) = 3\sqrt[3]{n}$ . אם נמשיך ונבנה עוד ועוד רמות, נקבל בצורה כללית:

$$\begin{aligned} T(search_1) &= n \\ T(search_2) &= 2 \cdot \sqrt{n} \\ T(search_3) &= 3 \cdot \sqrt[3]{n} \\ T(search_4) &= 4 \cdot \sqrt[4]{n} \\ &\vdots \\ T(search_k) &= k \cdot \sqrt[k]{n} \\ &\vdots \\ T(search_n) &= n \cdot \sqrt[n]{n} = n \end{aligned}$$

נשים לב שהביטוי מורכב מ- $k \cdot \sqrt[k]{n}$  - כאשר החלק הראשון במכפלה ( $k$ ) הוא מספר הרמות, והחלק השני - הוא מספר האיברים שמדלגים עליהם בכל רמה.

אם כן, מהו מספר הרמות האופטימלי?

שוב, בכדי להביא למינימום, נגזור את הביטוי (לפי  $k$ ) ונשווה ל-0. נקבל:

$$\begin{aligned} T(search_k) &= k \sqrt[k]{n} = k \cdot n^{1/k} \\ T'(search_k) &= n^{1/k} + k \cdot \ln n \cdot n^{1/k} \cdot \left(-\frac{1}{k^2}\right) \\ T'(search_k) &= n^{1/k} + k \cdot \ln n \cdot n^{1/k} \cdot \left(-\frac{1}{k^2}\right) \\ T'(search_k) &= n^{1/k} - \frac{\ln n \cdot n^{1/k}}{k} \end{aligned}$$

נשווה ל-0 ונקבל:

$$\begin{aligned} n^{1/k} &= \frac{\ln n \cdot n^{1/k}}{k} \\ k &= \ln n \end{aligned}$$

כלומר, הביטוי  $k \cdot \sqrt[k]{n}$  מקבל מינימום כאשר  $k = \log n$ . במקרה זה נקבל כי  $T(\text{search}_k) = \log n \cdot \sqrt[\log n]{n}$ . כולומר, יש לנו במבנה הנתונים  $T(\text{search}_{\log n}) = \log n \cdot 2$ . בצורה כזו, נקבל כי: (רשימות), כל אחת מכילה נציגים של הרמה מתחתיה, והרווח בין כל שני נציגים הוא 2.

- ברמה התחתונה יהיה כל האיברים.
- ברמה שמעליה יהיו  $1/2$  מהאיברים.
- ברמה שמעליה יהיו  $1/4$  מהאיברים (חצי מהאיברים שברמה האחת לפני האחרונה).
- ברמה שמעליה יהיו  $1/8$  מהאיברים,
- וכן הלאה.

נקבל אם כן, שסיבוכיות הזיכרון היא:

$$n \cdot \left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots\right) = 2n = O(n)$$

חיפוש איבר עולה לנו  $O(\log n)$  - בדיוק כמו בחיפוש בינארי. נעיר כי מה שהצגנו פה הוא "Ideal Skip List", כלומר - רשימת דילוגים אידיאלית. המצב "יתקלקל" כאשר נגיע ל-*insert* ו-*delete*. לסיכום, חיפוש איבר ברשימה ייתבצע באופן הבא:

#### חיפוש (x).

1. נחפש את  $x$  ברשימה העליונה, ונעצור כאשר עברנו את  $x$ . נלך צעד אחד שמאלה ונקבל את ה"קודם" של  $x$  ברשימה העליונה.
  2. נלך "בן" של הקודם של  $x$  (כלומר, לאיבר שמתחתיו). הגענו לרשימה השנייה.
  3. נחפש את  $x$  ברשימה השנייה כפי שחיפשנו ברשימה הראשונה, רק שנתחיל את החיפוש באיבר שאיתו הגענו לרשימה זו (בנו של הקודם של  $x$  ברשימה הראשונה).
  4. נמשיך את התהליך עד שנגיע רמה התחתונה. ברמה זו, או שנמצא את  $x$ , או שנראה שאינו קיים ונוכל להחזיר את הקודם שלו.
- קל לראות שברשימת דילוגים "אידיאלית", זמן הריצה של חיפוש הוא  $\Theta(\log n)$ .

## 4 תרגיל

**שאלה.** נתון מערך  $A$  של מספרים חיוביים (ושונים מ-0) בגודל  $n$ . השאילתא שעליה נרצה לענות בצורה מהירה היא:

$$mult(i): \text{ החזר את מכפלת כל המספרים במערך פליכד } A[i]. \text{ כלומר, החזר } \prod_{j \neq i} A[j].$$

1. תארו אלגוריתם המבצע עיבוד מקדים על המערך<sup>1</sup>, כך שניתן לענות על שאילתא בזמן  $O(1)$ . אלו נתונים? מהי עלות העיבוד המקדים?

<sup>1</sup> כלומר, אנו מניחים שני שלבים - בשלב הראשון אנחנו מקבלים את המערך כקלט, מבצעים עליו איזשהו עיבוד, ושומרים את התוצאות. בשלב השני, אנחנו מקבלים סידרה של שאילתות שעליהן נרצה לענות בצורה יעילה ומהירה בזכות המידע ששמרנו בשלב העיבוד המקדים.

**פתרון:** העיבוד המקדים פשוט יעבור על המערך ויחשב את מכפלת כל המספרים. נשמור את המכפלה בצד, במשתנה  $mult$ . בזמן שאילתא על אינדקס  $i$ , פשוט נחשב:  $mult/A[i]$ .  
 עלות עיבוד מקדים:  $O(n)$ . עלות שאילתא -  $O(1)$ .

2. לסעיפים הבאים אנו מניחים שהמחשב לא יודע לבצע פעולות חילוק. בכל סעיף יש לציין מהי עלות העיבוד המקדים. תארו את תהליך העיבוד המקדים, כך שעלות שאילתא תהיה  $O(\sqrt{n})$ . יש לציין במפורש כיצד ייתבצע העיבוד המקדים, וכיצד תתבצע שאילתא. (רמז: חישבו על "נציגים")

**פתרון:** נחלק ל- $\sqrt{n}$  קבוצות בגודל  $\sqrt{n}$  כל אחת, ונשמור מערך בגודל  $\sqrt{n}$  שישמור את כל המכפלות של המספרים בכל קבוצה. כלומר, נשמור מערך  $B$  בגודל  $\sqrt{n}$ , כך ש:

$$B[i] = \prod_{k=i\sqrt{n}}^{i\sqrt{n}+\sqrt{n}-1} A[k]$$

בנוסף, נשמור אינדקס התחלה וסוף של כל קבוצה.

בהינתן שאילתא, נחשב את מכפלת כל המערך  $B$  מלבד הקבוצה בה נמצא האיבר  $B[i]$  (ניתן לעשות זאת בלולאה, ונדע מתי מתחילה הקבוצה שבה- $i$  נמצא בזכות האינדקסי התחלה וסוף ששמרנו לכל תא). בנוסף, נחשב את מכפלת כל איברי הקבוצה בה נמצא  $A[i]$ , מלבד האיבר  $A[i]$ .

עלות העיבוד המקדים היא  $O(n)$ . בשאילתא אנחנו מבצעים  $2\sqrt{n}$  מכפלות, ולכן עלות  $O(\sqrt{n})$ .

3. תארו תהליך של עיבוד מקדים ואלגוריתם לשאילתא, כך שעלות שאילתא תהיה  $O(\sqrt[3]{n})$ .

**פתרון:** בדומה למה שראינו בכיתה. מחלקים את המערך לקבוצות בגודל  $\sqrt[3]{n}$ . מחשבים מכפלה של כל קבוצה כזו, ושומרים הכל במערך  $B$ . גודל המערך  $B$  הוא  $n/\sqrt[3]{n} = \sqrt[3]{n^2}$ . כעת, נחלק גם את המערך  $B$  לקבוצות בגודל  $\sqrt[3]{n}$ , נקבל  $\sqrt[3]{n}$  קבוצות. נשמור גם אינדקסים בגל קבוצה. עלות בנייה -  $O(n)$ . שאילתא מתבצעת בצורה דומה לסעיף הקודם. עלות  $\sqrt[3]{n}$  מכפלות, ולכן עלות כוללת  $O(\sqrt[3]{n})$ .

4. הכלילו את התהליך, כך שעלות שאילתא תהיה  $O(k\sqrt[k]{n})$ . מהי כמות הזיכרון הנצרכת? מהו  $k$  האופטימלי? מהי עלות שאילתא במקרה זה?

**פתרון:** בצורה דומה. נחלק את המערך לקבוצות בנות  $\sqrt[k]{n}$  כל קבוצה. נקבל  $n/\sqrt[k]{n} = n^{1-1/k}$  קבוצות. גם אותם נחלק לקבוצות בנות  $\sqrt[k]{n}$ , נקבל  $n^{1-2/k}$  קבוצות. נמשיך כך  $k$  רמות, עד שנקבל ברמה העליונה  $n^{1/k} = \sqrt[k]{n}$  איברים. עלות שאילתא תהיה -  $O(k\sqrt[k]{n})$ . אם נגזור כדי למצוא את  $k$  האופטימלי, נקבל  $k = \log n$  ואז:  $O(k\sqrt[k]{n}) = O(\log n \cdot \log n \sqrt[n]{n}) = O(2 \log n)$ , וזוהי עלות השאילתא. עלות העיבוד המקדים -  $O(n)$ .

5. חשבו על רעיון אחר לחלוטין, כך שעלות שאילתא תהיה  $O(1)$ , ועלות העיבוד המקדים -  $O(n)$ .

נשמור שני מערכים  $B, C$ , שניהם מגודל  $n$ . במערך  $B$  נשמור:

$$B[i] = \prod_{k=1}^i A[k] \quad \text{and} \quad C[i] = \prod_{k=i}^n A[k]$$

קל לראות שניתן לבנות את המערכים ב- $O(n)$ : בכדי לחשב את  $B[i]$ , פשוט נחשב  $B[i-1] \cdot A[i]$ . לכן, ניתן לרוץ בלולאה מ-1 ועד  $n$  בכדי לחשב את  $B$ . בצורה דומה, ע"י לולאה הפוכה, ניתן לחשב את המערך  $C$ .

כעת, בהינתן שאילתא  $i$ , נחזיר  $B[i-1] \cdot C[i+1]$  ונקבל:

$$B[i-1] \cdot C[i+1] = \prod_{k=1}^{i-1} A[k] \cdot \prod_{k=i+1}^n A[k] = \prod_{k \neq i} A[k]$$

עלות שאילתא היא  $O(1)$  (ניתן לאחר חישוב המערכים הנ"ל לשמור מערך של תוצאות ועלות שאילתא תהיה גם כן  $O(1)$ ).