פתרונות לבחינות בקורס

מבני נתונים ומבוא לאלגוריתמים

עידן כמרה
idan@aleph.nu

במסמך זה אפשר למצוא פתרונות **תמציתיים** לבחינות בקורס "מבני נתונים ומבוא לאלגוריתמים" שנלמד באוניברסיטה הפתוחה.

לתגובות, תיקונים או בכל עניין אחר אפשר ליצור איתי קשר בכתובת שמופיעה למעלה.

עדכונים למסמך זה יופיעו ב-http://aleph.nu/university.html.
עודכן לאחרונה ב-12.02.2012.

תוכן העניינים

2005 ב'

1 Radix-Sort + Counting-Sort (בדומה לשאלה ז-18, עמ' 126 במדריך למידה).

2 א מחלקים את המערך לשני חלקים ככה שכל איבר בחלק הראשון קטן שווה לכל איבר בחלק השני. ממיינים את החלק הראשון עם מיון ערמה ואת החלק השני רקורסיבית ויוצא ששני החלקים ממויינים.

2 ב נוסחת הנסיגה היא

$$T(n)=T\left(\frac{n}{2}\right)+\Theta(n)+\Theta\left(\frac{n}{2}\lg\frac{n}{2}\right)=T\left(\frac{n}{2}\right)+\Theta(n\lg n)$$

אפשר לפתור אותה עם שיטת האב מקרה 3 כאשר $c=\frac{1}{2}$. מקבלים $T(n)=\Theta(n\lg n)$.

2 ג במקרה הגרוע מיון מהיר הוא היחיד שרץ ב-$\Theta(n^2)$. במקרה הממוצע כולם רצים ב-$\Theta(n\lg n)$.

3 א בונים עץ אדום שחור מכל הערכים **השונים** ב-A. בגלל שיש k ערכים שונים, יהיו בעץ k צמתים ולכן הגובה שלו הוא $O(\lg k)$. N הכנסות לעץ כזה לוקחות $O(N\lg k)$. לכל צומת x בעץ נשמור שדה $size$ שמכיל את מספר הפעמים שהמפתח של x מופיע בקלט - $m(key[x])$. שומרים בצד זוג מספרים: המספר שהופיע הכי הרבה עד עכשיו וכמות הפעמים שהופיע. בכל הכנסה בהתאם לשדה ה-$size$ של הצומת המוכנס מעדכנים את זוג השדות.

3 ב משתמשים בעץ מסעיף א' כדי לבנות עץ אדום-שחור נוסף שמכיל את המכפלה $key[x]\cdot size[x]$ לכל צומת x בעץ המקורי. בכל צומת שומרים רשימה מקושרת $keys$ של המפתחות.

סורקים את העץ שבנינו ולכל צומת x מחפשים את $y=z-key[x]$. אם $key[y]\neq key[x]$ אז מחזירים את האיברים הראשונים ב-$keys[x],keys[y]$. אם $key[y]=key[x]$ אז צריך לוודא שיש יותר מאיבר אחד ב-$keys[x]$, אחרת האיבר שמצאנו הוא אותו אחד.

זמן ריצה: $O(N\lg k)+O(k\lg k)+O(k)=O(N\lg k)$.

4 ב גובה עץ הרקורסיה הוא $\lg n$ כאשר עלות הרמה ה-k היא $O(\frac{n}{2^k})$:

$$\sum_{k=0}^{\lg n}\frac{n}{2^k}=O(n)$$

4 ג אחרי החלוקה המערך מחולק לשני חצאים ככה שכל איבר בחצי הראשון קטן מכל איבר בחצי השני. בחצי השני בונים ערמת מינימום ולכן האיבר הראשון שם הוא המינימום בחצי השני והוא גדול מכל האיברים בחצי ראשון, כלומר הוא ערך המיקום ה-$\lfloor\frac{n}{2^1}\rfloor+1$...

Find-Os - מחזירים את $A[\lfloor\frac{n}{2^i}\rfloor+1]$.

Del-Os - מוציאים את השורש של הערמה שמתחילה ב-$\lfloor\frac{n}{2^i}\rfloor+1$.
לכל $k=i+1,\ldots,\lfloor\lg n\rfloor$ מוציאים את שורש הערמה שמתחילה ב-$\lfloor\frac{n}{2^k}\rfloor+1$ ומכניסים אותו לערמה שמתחילה ב-$\lfloor\frac{n}{2^{k-1}}\rfloor+1$ - סוג של "בעבוע". כל הפעולות על הערמות לוקחות $O(\lg n)$ ויש $\lg n$ ערמות, סה"כ מקבלים $O(\lg^2 n)$.

5 א מתחזקים מערך ממויין (או רשימה דו מקושרת) עם זוג אינדקסים בשביל Min-Gap. אחרי שמכניסים איבר במקום המתאים בודקים אם המרחק מהמפתח המוכנס לזה שמשמאלו או ימינו קטן מה-Min-Gap הנוכחי, אם כן מעדכנים בהתאם. אם האינדקס של האיבר שמוחקים הוא אחד מאלה שיוצרים את ה-Min-Gap אז רצים על כל הזוגות הסמוכים במערך ומוצאים את ה-Min-Gap החדש (זה לינארי).

5 ב המבנה מורכב משלוש ערמות: את $\left\lceil \frac{n}{2} \right\rceil$ האיברים **הקטנים** מחזיקים פעמיים בערמות מינימום ומקסימום. כל זוג איברים זהים בכל ערמה מצביעים אחד לשני. כל פעם שמבצעים פעולה כלשהי (הזזה, מחיקה) על איבר בערמה אחת, מתבצע עדכון למצביע (או מחיקה) בערמה השניה. את שאר $\left\lfloor \frac{n}{2} \right\rfloor$ האיברים **הגדולים** מחזיקים בערמת **מינימום**. בשורשי הערמות יושבים החציונים.

Build - מוצאים את החציון עם Select ב-$O(n)$ ובונים את שתי הערמות.

Insert - משווים את המפתח שרוצים להכניס לשורש ערמת המינימום של חצי האיברים הגדולים ולשורש ערמת המקסימום של חצי האיברים הקטנים כדי לדעת לאיזו צריך להכניס. אחרי הכנסה צריך שהערמות ישמרו על גודלם היחסי, לכן במקרה הצורך מוציאים איברים מראש ערמה אחת ומכניסים לשניה. כל הפעולות לוקחות $O(\lg n)$.

Del-Med - החציונים נמצאים בשורשים של ערמת המינימום והמקסימום. מוחקים את האיברים המתאימים ומאזנים את הערמות שיהיו בגדלים המתאימים אחרי המחיקה.

Del-Min2 - ערך המיקום השני נמצא ברמה השנייה בערמת המינימום של חצי האיברים הקטנים. מוחקים אותו וגם את האיבר שאליו הוא מצביע בערמת המקסימום ומאזנים את הערמות במידת הצורך.

2006 ב'

1 עץ אדום שחור עם שדה sum בכל צומת שמכיל את סכום כל המפתחות בתת העץ המושרש בצומת זה. תיחזוק השדה הוא סטנדרטי. בנוסף שומרים שני שדות בצד $max, max2$ שיחזיקו את המפתח המקסימלי וקודמו (בשביל Max2).

Insert - בודקים אם הצומת גדול מהמקסימום, אם כן max הוא $max2$ החדש ומשימים את הערך שמוכנס ל-max. אם הערך המוכנס בין max ל-$max2$ הטיפול דומה.

Delete - אם מוחקים את המקסימום אז $max2$ הופך להיות max ו-$max2$ החדש הוא ה-Predecessor של $max2$ הישן. סה"כ כל הפעולות פה לוקחות $O(\lg n)$.

Pair-Sum - מנהלים שתי סריקות תוכיות במקביל: אחת רגילה ואחת הפוכה (הולכים ימינה לפני שהולכים שמאלה). הראשונה תתן לנו את איברי העץ בסדר עולה והשנייה בסדר יורד. בכל שלב סוכמים את שני הצמתים הנוכחיים. אם הסכום שווה ל-z סיימנו. אם הוא קטן מקדמים את הסריקה הראשונה, אחרת את השנייה. מפסיקים אם הסריקות מצביעות לאותו צומת או אם המפתח של הראשונה גדול מהמפתח של השניה. האלגוריתם הזה רץ ב-$O(n)$ ודורש שתי מחסניות בגודל $O(\lg n)$.

פתרון אחר יותר פשוט שדורש $O(n)$ מקום זה פשוט לסרוק תוכית את העץ ולשים אותו במערך ולרוץ עם שני מצביעים מההתחלה והסוף...

Sum - שומרים מונה בצד שיכיל את הסכום המבוקש. עבור כל צומת, אם המפתח קטן מ-k אז הצומת הנוכחי וכל תת העץ השמאלי שלו צריכים להיסכם, אז מוסיפים למונה את המפתח הנוכחי ואת שדה ה-sum של הבן השמאלי. ממשיכים רקורסיבית לבן הימני. אם המפתח של הצומת הנוכחי גדול מ-k, אז הצומת הנוכחי וכל התת עץ הימני לא מעניינים, ממשיכים רקורסיבית שמאלה.

2 א ממיינים את המערך ב-$\Theta(n\lg n)$. לכל איבר x במערך מחפשים בינארית את x^2-1. סה"כ n חיפושים בעלות $\Theta(\lg n)$ ⇐ $\Theta(n\lg n)$.

2 ב מכניסים לטבלת גיבוב (שומרים גם את האינדקס המקורי של האיבר) ולכל איבר בטבלה מחפשים אותו דבר כמו בסעיף הקודם. סה"כ n חיפושים בעלות $\Theta(1)$ ⇐ $\Theta(n)$.

3 א מוצאים עם Select את האיבר ה-p וקוראים ל-Partition כשהוא ה-$pivot$. אז ממיינים עם מיון מיזוג/ערימה את התת מערך שכל איבריו קטנים מ-p. סה"כ יוצא:

$$O(p\lg p) = O(\frac{n}{\lg n}(\lg n - \lg\lg n)) = O(n(1-\underset{\underset{0}{\downarrow}}{\frac{\lg\lg n}{\lg n}})) = O(n)$$

3 ב Radix-Sort + Counting-Sort (בדומה לשאלה ז-18, עמ' 126 במדריך למידה).

4 א ממזגים את כל c הקטעים בבת אחד כמו במיון מיזוג (אלא שכאן יש לנו c קטעים במקום 2). צריך לעבור על n איברים ובכל שלב לבצע $c=O(1)$ בדיקות (כדי למצוא את המינימום מבין c החלקים), סה"כ יוצא $\Theta(n)$.

4 ב + ג פותרים את הנוסחה $T(n) = c \cdot T(\frac{n}{c}) + \Theta(n)$ עם שיטת האב מקרה 2. מקבלים $T(n) = \Theta(n \lg n)$ ולכן אין הבדל בין שתי הגרסאות.

5 עץ אדום-שחור שבכל צומת בנוסף למפתח שומרים רשימה מקושרת של רשומות (שומרים גם את גודל הרשימה). בנוסף שומרים בכל צומת שדה $majority$ שמכיל את מצביע לצומת עם הרשימה הכי ארוכה בתת העץ שמושרש בצומת זה.

MODE - מחזירים את $key[majority[root[S]]]$.

INSERT - מתחילים כרגיל. כשמגיעים לצומת מוסיפים את הרשומה לרשימה המקושרת ומגדילים את הגודל ב-1. מטיילים במעלה העץ ומעדכנים את שדה ה-$majority$ בכל צומת במידת הצורך.

DELETE - מתחילים כרגיל. כשמגיעים לצומת, אם הרשימה גדולה מ-1 פשוט מוחקים את ראש הרשימה ומקטינים את שדה הגודל. אחרת מוחקים את הצומת. מטיילים במעלה העץ ומעדכנים את שדה ה-$majority$ שיכיל את הצומת שאורך הרשימה שלה מקסימלי מבין הצומת הנוכחי, השמאלי והימני.

87 מועד א' 2009

1 א מנהלים שתי סריקות תוכיות במקביל: אחת רגילה על תת העץ השמאלי של השורש ואחת הפוכה על תת העץ הימני של השורש (הולכים ימינה לפני שהולכים שמאלה). הראשונה תתן לנו רשימה של מפתחות בסדר עולה והשנייה בסדר יורד. בכל שלב סוכמים את שני הצמתים הנוכחיים. אם הסכום שווה ל-$2 \cdot key[root[T]]$ סיימנו. אם הוא קטן מקדמים את הסריקה הראשונה, אחרת את השנייה. האלגוריתם רץ ב-$O(n)$ ודורש שתי מחסניות בגודל $O(\lg n)$.

1 ב מריצים את האלגוריתם מסעיף א' לפי רמות (קודם הצמתים בעומק 0, אחרי זה 1, 2...). לשורש יש $\Theta(n)$ עבודה, לכל אחד מהבנים $\Theta(\frac{n}{2})$ וכך הלאה...

זמן הריצה הוא: $T(n) = 2 \cdot T(\frac{n}{2}) + \Theta(n) = \Theta(n \lg n)$.

2 מציבים $m = \lg n$ ומשתמשים בשיטת האב מקרה 2: $\Theta(\lg^{2/3} n \cdot \lg\lg n)$.

3 נניח שהכוונה ב-Os-Med7 היא לערך המיקום ה-$\lceil\frac{n}{2}\rceil+7$. נשים את $\lceil\frac{n}{2}\rceil$ האיברים הקטנים בערמת מקסימום. מתוך $\lfloor\frac{n}{2}\rfloor$ האיברים הגדולים, את 6 הקטנים מהם נשמור במערך ממויין ואת כל השאר בערמת מינימום. תחזוק המערך הזה לוקח $\Theta(1)$ כי גודלו קבוע.

Build - מוצאים את החציון (Select) ומכניסים את הערכים למבנים המתאימים. שומרים במשתנה בצד את המינימום.

Min - מחזירים את המשתנה ששמרנו בצד.

Del-Median - מוציאים את המקסימום מהערמה הראשונה. שומרים על איזון בין כל שלושת המבנים ע"י העברת איברים מאחד לשני במידת הצורך.

Os-Med7 - מחזירים את שורש הערמה השניה.

5 האיברים יוכנסו לעץ אדום שחור. בנוסף, כל איבר שמוכנס יוכנס גם לעץ ערכי מיקום כאשר המפתח בכל צומת יהיה מספר רץ. הצמתים יכילו מצביעים אחד לשני.

לדוגמה נניח שהכנסנו כבר 5 איברים, ועכשיו מכניסים איבר עם מפתח x: מכניסים את x לעץ הראשי. מכניסים לעץ ערכי מיקום את המפתח 6 ומעדכנים את שני הצמתים שהוכנסו שיצביעו אחד לשני.

Insert - כמו שתואר בפסקה למעלה. מקדמים את המספר הרץ.

Max - מתוחזק במשתנה בצד, (מעדכנים אותו בכל הכנסה/מחיקה מהעץ בעלות של $\Theta(\lg n)$).

Delete-Max - מוחקים את Max ואת הצומת שהוא מצביע אליו בעץ ערכי מיקום.

Delete-Old - מחפשים את ערך המיקום ה-t בעץ ערכי מיקום. מוחקים אותו ואת הצומת שאליו הוא מצביע.

2009 א' מועד 88

1 מתחילים במיון המערך. לכל $1 < j < n$ רצים עם שני אינדקסים, אחד בראש המערך והשני בסופו ובודקים אם הסכום שלהם שווה ל-$2 \cdot A[j]$. במידה וכן אז סיימנו. אם הוא קטן מקדמים את האינדקס הראשון. אם הוא גדול מזיזים את האחרון אחד אחורה. מקדמים את j כאשר אחד האינדקסים מגיע אליו ומתחילים את התהליך שוב.

המיון הראשוני לוקח $\Theta(n \lg n)$. בכל שלב זוג האינדקסים משמאל ומימין ל-j עוברים על $\Theta(n)$ איברים, ו-j גם כן, לכן זמן הריצה הכולל הוא $\Theta(n^2)$.

2 משתמשים ברמז ומקבלים לאחר הצבה $U(n) = 8 \cdot U(\sqrt{n}) + \lg^3 n$. עושים עוד הצבה $m = \lg n, S(m) = U(2^m)$ ומקבלים $S(m) = 8 \cdot S(\frac{m}{2}) + m^3$. את המשוואה האחרונה פותרים עם שיטת האב מקרה 2 ומקבלים $S(m) = \Theta(m^3 \lg m)$. אחרי חזרה לפונקציה המקורית מקבלים $T(n) = \Theta(n^3 \lg^3 n \lg \lg n)$.

3 המבנה מורכב מ-6 ערמות: מקסימום H_L ומינימום H_H ($L = low, H = high$) מכילות בהתאמה את חצי האיברים הקטנים והגדולים במבנה. מקסימום H_{E_L} ומינימום H_{E_H} מכילות את חצי האיברים הזוגיים הקטנים והגדולים ואותו דבר לאי-זוגיים ב-H_{O_L}, H_{O_H}. ארבעת הערמות האחרונות מכילות איברים שכבר הופיעו בשתי הערמות הראשונות, לכן כדי לשמור על סנכרון ביניהם, כל זוג איברים זהים יצביעו אחד לשני וכאשר אחד מהם מוזז/נמחק ההעתק שלו יעודכן בהתאם.

Build - מוציאים את החציונים המתאימים עם Select ומכניסים למבנים המתאימים את האיברים שקטנים/גדולים מהם.

Median - מחזירים את השורש של H_L.

Odd-Median - מחזירים את השורש של H_{O_L}.

Even-Median - מחזירים את השורש של H_{E_L}.

במחיקות למיניהן מוחקים את השורש של הערמה המתאימה (וגם את ההעתק שהוא מצביע אליו) ואז "מאזנים" מחדש כל זוג ערמות שישמרו על היחס ביניהם ע"י הוצאת איבר מערמה אחת והכנסה שלו לערמה השניה.

Increase - מקדמים את המפתח ומתקנים את הערמה (Max-Heapify). מסירים את האיבר מהערמה של הזוגיים או אי-זוגיים ומכניסים לערמה המתאימה בהתאם לזוגיותו ולגודל (משווים לשורש של הערמה).

4 המועמדים האפשריים לתנאי הראשון הם כל האיברים שערך המיקום שלהם גדול מ-$n-2m$. כדי שאיבר כלשהו יהיה מועמד לתנאי השני, הוא צריך להיות ערך המיקום ה-$n-m$ או ה-n. אם כך מוצאים את ערך המיקום ה-$n-m$ עם Select ועושים עוד מעבר על המערך כדי למנות את מספר המופעים שלו. אם הוא מופיע יותר מ-m פעמים אז סיימנו, אחרת עושים אותו דבר לערך המיקום ה-n. סה"כ קוראים פעמיים ל-Select ועושים עוד שני מעברים על המערך לכן זמן הריצה הוא $\Theta(n)$.

5 עץ אדום-שחור כאשר בכל צומת שומרים שדה מספרי $accum$ שיאותחל ל-0 כברירת מחדל. לכל צומת x בעץ מחברים את $accum[x]$ לכל המפתחות בתת העץ שמושרש ב-x.

למימוש חיפוש, הכנסה ומחיקה ראה שאלה 3 ב' בממ"ן 17.

```
procedure DECREASE-UPTO$(S, k, d)$
    $n \leftarrow root[S]$
    while $n \neq nil$ do
        if $key[n] \leq k$ then
            $accum[n] \leftarrow accum[n] - d$
            if $right[n] \neq nil$ then
                $accum[right[n]] \leftarrow accum[right[n]] + d$
            $n \leftarrow right[n]$
        else
            $n \leftarrow left[n]$
```

2009 ב' מועד 91

1 ממיינים את המערך עם Counting-Sort ורצים עם שני אינדקסים מההתחלה ומהסוף. אם הסכום שלהם גדול מ-65535 מזיזים את האינדקס השני אחד אחורה. אם הסכום קטן אז הראשון זז אחד קדימה. מסיימים כשהסכום שווה או שהאינדקסים נפגשים/עברו אחד את השני.

2 ממיינים את הזוגות לפי x. אם יש יותר מזוג נקודות אחד עם אותו ערך x, לוקחים את האחד עם ערך ה-y המקסימלי.

שומרים במשתנה $area$ את השטח שנצבר עד כה. האלגוריתם רץ על הזוגות הממויינים תוך שהוא שומר במשתנה $max = (x, y)$ את הנקודה עם ה-y המקסימלי. המלבן ה-i מטופל בצורה הבאה:

- אם $y_i \leq y_{i-1}$ אז $area \mathrel{+}= \ (x_i - x_{i-1}) \cdot y_i$.
- אחרת אם $y_i > max_y$ אז $area = x_i \cdot y_i$ (מעדכנים את max).
- אחרת $area = x_i \cdot max_y - (x_i - max_x) \cdot max_y$.

3 מוצאים את החציון עם Select ורצים על המערך וסופרים כמה איברים שווים לו.

4 עץ אדום שחור עם שדה $size$ בכל צומת ומחסנית (שממומשת עם רשימה דו-מקושרת) של מצביעים לצמתים בעץ. כשמכניסים צומת לעץ דוחפים למחסנית מצביע לצומת.

Delete-Old - האיבר הוותיק ביותר יהיה בסוף המחסנית, מוחקים אותו משני מבני הנתונים.

Count-Min-Old - מחפשים בעץ את המפתח של האיבר שבסוף המחסנית k. מתחילים בשורש העץ: לכל צומת x, אם $key[x] > k$ אז כל האיברים שמימין ל-x לא רלוונטים והולכים רקורסיבית שמאלה. אם $key[x] \leq k$ אז כל האיברים שמשמאל ל-x קטנים מ-k. מוסיפים את $size[left[x]] + 1$ למונה ששומרים בצד וממשיכים רקורסיבית ימינה.

5 ערמת מקסימום ומחסנית (שממומשת עם רשימה דו-מקושרת) של מצביעים לערמה. כל איבר בערמה ישמור מצביע לאיבר המתאים לו במחסנית ויעדכן אותו כשהוא מוזז.

Switch-Old-New - האיבר הראשון במחסנית הוא החדש ביותר והאחרון הוא הוותיק ביותר. מעדכנים את המפתח של אחד מהם עם המפתח השני (שומרים את הערך הישן בצד) וקוראים ל-Max-Heapify ואז מטפלים בשני (צריך לעשות את זה אחד אחרי השני כי השגרה מניחה שתת העץ שמושרש בצומת שמתקנים הוא ערמה תקינה).

מימוש שאר השגרות הוא סטנדרטי בתוספת עדכון המצביעים במקומות המתאימים ותחזוק המחסנית.