מבני נתונים - תרגול 12
פונקציית ערבול מושלמת (Perfect Hash)

גלעד אשרוב

11 ביוני 2013

תקציר

בתרגול זה נלמד על $perfect\ hash$ (פונקציית ערבוב מושלמת). המטרה היא בהנתן קבוצה של מפתחות S מתוך עולם U למצוא פונקצית ערבוב **מושלמת**, כלומר, פונקציה ללא התנגשויות כלל בקבוצה הנתונה S. בצורה זו ברור כי החיפוש אורך $O(1)$ זמן. נראה שלם כך כמות הזיכרון נדרש היא בגודל $O(n^2)$, כאשר n הוא מספר האיברים ב - S. בנוסף, נראה שניתן לקבל מבנה לינארי בגודל של S, המשתמש בלכל היותר שתי פונקציות ערבוב לשם מציאת איבר (כלומר, חיפוש עדיין לוקח $O(1)$). נעיר שמבנה הנתונים שלנו הוא סטאטי - אנחנו לא דנים בהוצאה ובהכנסה במסגרת דיוננו זה.

1 הקדמה

נתבונן בבעיה הבאה:
נתונה קבוצה של מספרים $U = \{0, \ldots, u-1\}$, ותת קבוצה $S = \{x_1, \ldots, x_n\} \subseteq U$. נרצה לענות על השאילתות הבאות:

- $query(x)$ - האם $x \in S$. אם יש מידע - החזר (כלומר, x הוא רק מפתח).
- $insert(x)$ - הכנס את x ל- S.
- $delete(x)$ - מחק את x מ- S.

פתרון. פתרון אפשרי לבעיה הוא:

- מערך בגודל U. מעין $bit-vector$. הבעיה - אם $|S| << |U|$, יש לנו המון בזבוז של מקום...
- פונקציית $hash$: נשמור את כל הנתונים במערך בגודל m, ונגדיר פונקצייה מ-$U \mapsto \mathbb{Z}_m$. כאשר נרצה לגשת לאיבר x, נחשב את $h(x)$, נקבל אינדקס בין $0, \ldots, m-1$, ונדע לאן לגשת למערך בשביל למצוא את האיבר.

2 מה למדנו בהרצאה?

בהרצאה הנחנו שהפונקציה היא קבועה ונבחרה "איכשהו", ובכל פעם בוחרים איבר כלשהו באקראי ומכניסים אותו לטבלה שלנו. הפונקציה תהיה טובה כל עוד אנחנו מניחים שהאיברים שאנחנו מכניסים לטבלה ייבחרו באקראי.
כמובן, לפי עיקרון שובך היונים, ומכיוון ש - $m << |U|$, יהיו איברים שיתנגשו. ראינו בהרצאה כיצד מתמודדים עם התנגשויות. להלן סיכום קצר:

- ראינו למשל - $hash$ פתוח ($chaining$). במקרה זה, המערך שלנו בגודל m יהיה מערך של רשימות מקושרות. בהינתן הקבוצה S של האיברים שבאמת נמצאים לנו במבנה הנתונים, בכניסה ה-i של המערך תהיה רשימה מקושרת המכילה את כל האיברים: $A[i] = \{x \in S \mid h(x) = i\}$.
- ראינו גם - $hash$ סגור. במקרה זה, לא יהיו יותר משני איברים באותו התא (בפרט, המערך שלנו הוא מערך של איברים ממש, ולא מערך של "רשימות מקושרות"). לפיכך, הטיפול בהתנגשויות מעט מסובך יותר. ראינו שלוש שיטות:
  - $Linear\ probing$: נחשבים את $h(x)$. אם המקום תפוס, מתקדמים למקום $h(x)+1$. אם גם הוא תפוס - מנסים את $h(x)+2$, וכן הלאה.... בחיפוש - מחשבים את $h(x)$ ומחפשים אותו בצורה איטרטיבית עד שמגיעים לתא ריק. בהוצאה - נצטרך אולי לצמצם רווחים, או שנסמן שאיבר הוצא.
  - $Rehashing$: יהיו לנו מספר פונקציות ערבול, $h_1, \ldots, h_t$. אם $h_1(x)$ תפוס, ניגש ל- $h_2(x)$ וכן הלאה. המקרה הקודם הוא למעשה מקרה פרטי שבו הגדנו את $h_i(x) = h_1(x) + i$.
  - $quadratic\ hashing$: אותו רעיון. $h_i(x) = h_1(x) + c_1 \cdot i + c_2 \cdot i^2$, עבור קבועים כלשהם c_1, c_2. נשים לב ש- $linear\ probing$ הוא למעשה $c_1 = 1, c_2 = 0$.
  - $double\ hashing$: באיטרציה ה-k בחיפוש נבצע: $h_k(x) = (h(x) + k \cdot h'(x)) \bmod m$ עבור שתי פונקציות $hash$ התחלתיות כלשהן h, h'.

בנוסף, ראיתם בהרצאה:

הגדרה 1. מקדם העומס α של טבלה בגודל m השומרת n ערכים הוא $\alpha = n/m$.

בנוסף, ראיתם את הטענות הבאות:

	average search time ($x \notin S$)	average search time ($x \in S$)
chaining	$\Theta(1+\alpha)$	$\Theta(1+\alpha/2) = \Theta(1+\alpha)$
Linear Probing	$\leq \frac{1}{2}\left(1 + 1/(1-\alpha)^2\right)$	$\leq \frac{1}{2}\left(1 + 1/(1-\alpha)\right)$
Open Addressing (assuming $h_0(x), \ldots, h_m(x)$ is a random permutation of $[m]$. As in quadratic and double probing.)	$O(1 + 1/(1-\alpha))$	$O(1 + \frac{1}{\alpha}\ln\frac{1}{1-\alpha})$

תרגיל 2. נתון מערך A של מספרים. תארו אלגוריתם המקבל מספר k ובודק האם קיימים ב-A זוג מספרים a, b כך ש- $a + b = k$. על האלגוריתם לרוץ בזמן $O(n)$ בממוצע.

פתרון. האלגוריתם ייבחר פונקציית $hash$, ויכניס את כל האיברים לטבלה (מכיוון שיודעים בדיוק כמה איברים יש, ניתן לבחור את m כך ש- α יהי קטן מספיק בכדי שנקבל זמן חיפוש קבוע בממוצע). כעת, בהינתן k, נעבור שוב על המערך A, כאשר באיטרציה ה-i נבדוק האם האיבר $k - A[i]$ נמצא בטבלה שלנו (עלות - $O(1)$). ברגע שמצאנו אחד כזה, מצאנו למעשה זוג איברים $(a, b) = (A[i], k - A[i])$, ומתקיים $a + b = k$.

תרגיל 3. נתונה סידרה של - n מספרים $a_1, \ldots, a_n$ קטנים מערך מסויים X. מצאו האם קיימים אינדקסים i ו- j שונים כך ש- $a_i = a_j$.

פתרון. מכיוון שאנחנו יודעים את הגבול העליון X, ניתן לקחת פונקציה $h : [1, \ldots, X] \mapsto m$, ולבנות טבלה בגודל m כלשהו, עבור α יוצא קטן מספיק. כעת, נתחיל להכניס את האיברים לתוך הטבלה שלנו בעזרת פונקצייה $hash$, כאשר ליד כל איבר נכתוב גם את האינדס ממנו הגיע במערך המקורי. כעת, אם קיימים i, j עבורם $a_i = a_j$, יתקיים $h(a_i) = h(a_j)$ ולשניהם יהיה אותו אינדקס בטבלה. נמצא את הזוג הנ"ל ב-$O(n)$ (בממוצע).

3 פונקציית ערבול מושלמת

בתרגול זה, אנחנו נעסוק בשאלה קצת שונה - במקום לשאול איך מתמודדים עם התנגשויות, אנו נניח שהקבוצה S היא ידועה ונתנונה. מה שנרצה זה לבחור פונקציה h טובה עבור הקבוצה הידועה והנתונה הזו. למעשה, אנחנו מתעלמים מהפעולות - $insert, delete$, ונדבר רק על הפעולה $query$. נדבר על מבנה נתונים **סטטי**.

כיצד נבחר פונקציית $hash$ טובה? נניח $|S| = n$. אנו רוצים לבחור פונקציה שלוקחת אלמנטים מ - U ומעבירה אותם ל - $\mathbb{Z}_m$ (למערך בגודל m (נקבע את גודלו מאוחר יותר. נרצה שגודלו יהיה לינארי ב- $|S|$). פונקציה טובה כזו היא פונקציה אקראית לחלוטין. כלומר, לכל איבר ב - U, נבחר איבר באקראי מהקבוצה $\{0, \ldots, m-1\}$. עבור שני איברים x, y, קבועים, ההסתברות להתנגשות היא:

$$\Pr_h[h(x) = h(y)] = \frac{1}{m}$$

חישוב זה הוא למעשה אותה השאלה שהייתה לנו בתחילת הסמסטר - בהינתן שני כדורים שנזרקים לתאים - מהי ההסתברות ששני כדורים נתונים ייפלו לאותו התאה? לשם תזכורת, נסביר מדוע חישוב ההסתברות הוא $1/m$. מתבוננים במספר התא ש - x הגיע אליו, ושואלים מהי ההסתברות ש - y יגיע לאותו התא. מכיוון שהתא אליו y מגיע נבחר באקראי מבין כל התאים, נקבל הסתברות של $1/m$.

בכלל, עבור k איברים מתוך S, נניח $s_1, \ldots, s_k$, ההסתברות שכולם ייתנגשו לאותו התא היא:

$$\Pr_h[h(x_1) = \ldots = h(x_k)] = \left(\frac{1}{m}\right)^{k-1}$$

כאשר ההסתברות נלקחת על הבחירה של h. הסבר לחישוב ההסברות: x_1 קובע תא שאליו כולם צריכים ללכת. ההסתברות שכל אחד מ - $x_2, \ldots, x_k$ נפלו בתא ש - x_1 נפל אליו היא $1/m$. מכיוון שכל המאורעות בלתי תלויים, נקבל $(1/m)^{k-1}$.

הבעיה. כזכור, בחרנו פונקצייה אקראית בשביל פונקציית הערבול ($hash$). כדי לייצג את הפונקציה, לכל ערך ב- U נצטרך לשמור לאיזה תא הוא ממופה, או בעצם - ערך ב - $\mathbb{Z}_m$. בכדי לייצג ערך ב - $\mathbb{Z}_m$ דרושים לנו $\log m$ ביטים (למה?), ולכן, כדי לייצג את כל הפונקצייה אנו נדרשים ל - $|U| \log m$ ביטים, שזה המון!

3.1 דרישות

למעשה, נרצה מספר תכונות מפונקציית ה - $hash$:

- **אתחול:** בהינתן קבוצה S נרצה לבנות בצורה מהירה.
- **זמן:** נרצה לענות על שאילתות בצורה מהירה.
- **מקום:** פונקציית ה - $hash$ צריכה להיות בעלת תיאור קומפקטי. כלומר, ייצוג הפונקצייה צריך להיעשות בעזרת מספר מועט של ביטים.

4 פונקציית ערבוב מושלמת - Perfect Hash

המטרה בסופו של דבר היא להגיע לפונקצייה ערבוב מושלמת. כלומר:

הגדרה 4. בהינתן קבוצה $S \subseteq U$, נאמר שהפונקציה - $h : U \to \mathbb{Z}_m$ היא פונקציית ערבוב (גיבוב) מושלמת אם לכל $x \neq y \in S$, מתקיים $h(x) \neq h(y)$.

במילים אחרות, הפונקציה h היא פונקציית ערבוב מושלמת עבור הקבוצה S אם אין בכלל התנגשויות בתוך הקבוצה.

5 פונקציית hash אוניברסלית

נדבר על משפחה של פונקציות, ונבחר פונקציה אחת מתוך המשפחה באקראי. כאשר דיברנו על בחירת פונקציה אקראית, למעשה דיברנו על כלל הפונקציות $U \to \mathbb{Z}_m$, ואמרנו ש־ h נבחרת באקראי מתוכם. עבור משפחה זו, ראינו שקיימות שתי תכונות:

- לכל $x, y \in U$, $x \neq y$, מתקיים:
$$\Pr_h[h(x) = h(y)] = \frac{1}{m}$$

- ובאופן כללי, לכל $x_1, \ldots, x_k \in U$, $x_1 \neq \ldots \neq x_k$, מתקיים:
$$\Pr_h[h(x_1) = \ldots = h(x_k)] = \left(\frac{1}{m}\right)^{k-1}$$

אמרנו שמשפחה זו בעייתית מכיוון שאין לפונקציית הגיבוב ייצוג קומפקטי. נתבונן כעת במשפחה חדשה של פונקציות, המתקיימת רק את התכונה הראשונה. לאחר מכן, נראה שתכונה זו מספיקה לנו, וכמו־כן, שקיימת משפחת פונקציות כאלה בעלי ייצוג קומפקטי. פורמלית, נגדיר:

הגדרה 5. משפחת פונקציות $hash$ $\mathcal{H} = \{h_i \mid h_i : U \to \mathbb{Z}_m\}$ תקרא משפחה אוניברסלית אם לכל $x, y \in U$, $x \neq y$ מתקיים:
$$\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{m}$$
כאשר ההסתברות נלקחת על פני בחירת h מתוך $\mathcal{H}$.

למעשה, המשפחה הקודמת שהגדרנו (כלל הפונקציות מ־ U ל־ $\mathbb{Z}_m$) הייתה עמידה בפני התנגשויות לכל תת קבוצה של איברים. אצלינו - אנחנו עמידים בפני התנגשות רק לזוג.

השאלה הראשונה היא האם קיימת משפחה של פונקציה כזאת. השאלה השנייה־ האם ייצוגה קומפקטי, והשאלה השלישית היא.... למה הסתברות נמוכה יחסית לקבלת התנגשות רק לזוג - באמת מספיק לנו??

ראשית, קיימת משפחה כזאת של פונקציות. נתבונן במשפחה הבאה:

$$\mathcal{H}_{p,m} = \{h_{a,b} \mid 1 \leq a \leq p-1, 0 \leq b \leq b-1\}$$

וכאשר:

$$h_{a,b} = ((ax + b) \bmod p) \bmod m$$

כמה פונקציות במשפחה? עבור p מסויים, יש כאן p^2 פונקציות. הטענה אומרת שהמשפחה הנ"ל היא משפחה אוניברסלית. כלומר, אם a ו־ b נבחרים באקראי כפי שצויין, התכונה שביקשנו־ מתקיימת. ההוכחה היא בספר, מבוא לאלגוריתמים, קורמן.

נשים לב שייצוג הפונקציה הוא קומפקטי־ בכדי לייצג את הפונקצייה דרושים לנו רק a ו־ b, שלשניהם אנו צריכים $\log p$ ביטים.

6 בניית פונקציית Perfect Hash עבור $S \subseteq U$

כעת, בהינתן קבוצה S נרצה לחפש עבור הקבוצה פונקציה מושלמת. לשם כך, נתבונן באלגוריתם הבא. האלגוריתם בוחר באקראי פונקציה מתוך המשפחה, ופשוט בודק האם היא באמת מושלמת. אם הפונקציה מושלמת - האלגוריתם מפסיק, ומחזיר את הפונקציה. אם היא לא - הוא חוזר להתחלה ומחפש פונצקיה מחדש (שוב, באקראי). כפי שנראה בניתוח, ההסתברות שהפונקציה תהיה טובה יהיה לנו מספיק גדול. באופן מפורט יותר:

- **קלט:** הקבוצה $S \subseteq U$ (נניח, נתונה לנו רשימה מקושרת של כל האיברים).
- **פלט רצוי:** מערך מגודל m, ופונקציה $h : U \to \mathbb{Z}_m$, כל שלכל זוג $x \neq y \in S$, $h(x) \neq h(y)$.
- **האלגוריתם:**
  1. נקבע משפחה אוניברסילית בצורה שרירותית (התלויה ב- m).
  2. נבחר $h \in \mathcal{H}$ בצורה אקראית.
  3. ניצור "קבוצות" (רשימות) של $S_i = \{x \in S \mid h(x) = i\}$. (כלומר, לכל אינדקס, בודקים כמה איברים מופו לאותו האינדקס).
  4. אם קיים i כך ש- $|S_i| > 1$, חזור ל (2). (אם קיימת איזושהי התנגשות - בחר פונקציה מחדש).

ניתוח מספר ההתנגשויות. אנו מעוניינים לחשב מהי ההסתברות שכאשר בחרנו פונקציית $hash$ מתוך $\mathcal{H}$, נקבל איזושהי התנגשות. נקבל: (הסבר על החישוב מובא לאחר החישוב)

$$\Pr\left[\exists\ collision\right] = \Pr\left[\exists x, y \in S, x \neq y, h(x) = h(y)\right] \leq \sum_{x,y \in S, x \neq y} \Pr\left[h(x) = h(y)\right] \leq \binom{n}{2} \cdot \frac{1}{m} \leq \frac{n^2}{2m}$$

מספר הערות על חישוב זה:

- החישוב מאוד דומה לניתוח מספר התנגשויות לכפי k כדורים שנזרקים ל-n תאים כפי שראינו בתרגול 3.
- בחישוב זה נעזרנו ב- *union bound*. הכלל אומר כי: $\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$.
- באשר לאי השיוויון השלישי - אנו בודקים את כל הזוגות x, y האפשריים שיש, ללא חזרות. ישנן $\binom{n}{2}$ זוגות כאלו. עליהן, אנו מחשבים מה ההסתברות שיש התנגשות בין כל זוג וזוג. ההסתברות להתנגשות של כל זוג היא קטנה מ- $1/m$, וזה נובע מתכונת האוניברסליות של $\mathcal{H}$.

אם ניקח $m = n^2$, נקבל כי:

$$\Pr\left[\exists\ collision\right] \leq \frac{n^2}{2m} = \frac{n^2}{2n^2} = \frac{1}{2}$$

כלומר בהסתברות קטנה מחצי - תהיה בעיה.

ניתוח זמן ריצה. בכל סיבוב מבצעים עבודה התלויה באורך של $|S| = n$. השאלה היא כמה סיבובים ישנם. מכיוון שאנחנו עובדים באלגוריתם הסתברותי, זמן הריצה של האלגוריתם יכול להמשיך עד אינסוף (אף פעם לא נמצא פונקציה שאין בה התנגשות...), אבל באופן כללי, אנו **מצפים** שמספר הסיבובים יהיה קטן. למעשה, אינטואיטיבית, מכיוון שההסתברות שקיימת התנגשות לפונקציה אקראית קטנה מ- $1/2$, אנו מצפים שנצטרך לבדוק 2 פונקציות עד שנקבל פונקציה טובה.

כאמור, כאשר נבחר $m = n^2$, ההסתברות שנצטרך לחזור לשלב (2) היא קטנה מ־ $1/2$. לכן, נקבל:

$$E(number\ of\ rounds) = \sum_{i=1}^{\infty} i \cdot \Pr[number\ of\ rounds = i] \leq \sum_{i=1}^{\infty} i \cdot \frac{1}{2^i} = 2$$

כלומר, אנו **מצפים** שלאחר שני סיבובים, האלגוריתם יעצור.

לסיכום, קיבלנו פונקציית $hash$ מושלמת. הבנייה (צפוייה) ב - $O(n)$, הפונקציה יעילה, קלה לחישוב, וקומפקטית. הבעיה היחידה - אנו צריכים $O(n^2)$ מקום!

גודל טבלה לינארי במספר האיברים. ניתן לייצר גודל טבלה לינארי במספר האיברים, ע"י שימוש באלגוריתם מעט מתוחכם יותר, שנקרא FKS (על שמם של - $Fridman\ Komels\ Szomeredi$). הרעיון של האלגוריתם הוא להשתמש ב-$hash$ אחד גדול, ולהרשות שאיברים ייתנגשו, אבל "לא יותר מדי". אם "יותר מדי איברים מתנגשים" - נחפש פונקציה אחרת. כעת, עבור כל תא שבו האיברים מתנגשים - נייצר פונקציית ערבול מושלמת עבור אותו התא (עם טבלה חדשה שגודלה הוא הריבוע של מספר האיברים באותו התא. אין פה יותר מדי בזבוז - מכיוון שמספר האיברים בתא הוא קטן). בניתוח ניתן להראות שאכן המבנה נתונים הוא לינארי.