אלגוריתמים 1 - הרצאה 9

6 בינואר 2026

גיא יער־און

0.1 מבוא והגדרה

הגדרה: יהי אלגוריתם רנדומי A. אלגוריתם רנדומי הוא אלגוריתם שמשתמש במחרוזת אקראית r כקלט (פרט לקלט והפלט הרגילים). יש לכך פירושים שונים - בקורס אלגוריתמים 1: אנחנו מניחים כי כל מספר ב-r הוא מספר אקראי שמתפלג באופן אחיד מטווח המספרים השלמים $[0,n]$ או $[1,n]$ באשר n הוא גודל הקלט של האלגוריתם A.

דוגמה. אם הקלט שלנו הינו גרף $G=(V,E)$ אזי $n=|E|+|V|$ ונקבל שכל מספר ב-R הוא ממשי בטווח $[0,n]$ שמתפלג בו אחיד.

בעת שאנחנו מריצים אלגוריתמיים דטרמיניסטיים, הנכונות והסיבוכיות ברורים ואפשריים להוכחה. עם זאת, באלגוריתם רנדומי יכולים לקרות דברים שונים.

א. הריצה של אלגוריתם רנדומי A יכולה לרוץ בזמן שונה. יתכן כי זמן הריצה יהיה תלוי במחרוזת האקראית r

ב. האלגוריתם A יכול להחזיר בסוף הריצה שלו תשובה שונה. כלומר, יהיו P_1,P_2 ריצות שונות של A. יתכן כי $P_1(A)\neq P_2(A)$.

משתנים מקריים:

א. זמן הריצה של אלגוריתם רנדומי יהיה משתנה מקרי שתלוי ב-r.

ב. הנכונות של האלגוריתם (והצלחתו) היא גם כן משתנה מקרי שתלוי ב-r.

0.1.1 אלגוריתמי מונטה קרלו ואלגוריתמי לאס וגאס

הגדרה: אלגוריתמי מונטה קרלו $(Monte-Carlo)$ הם אלגוריתמים רנדומיים שזמן הריצה שלהם הוא דטרמיניסטי (ניתן לחסום אותו), אך הנכונות היא משתנה מקרי. כלומר, האלגוריתם עלול לטעות ולא להצליח. נרצה לנתח את הסיכוי לטעות: שיהיה כמה שיותר קטן. (לזכור - $Maybe\ correct$).

בהינתן גודל קלט n, נרצה כי הסיכוי לטעות $\leq \frac{1}{n^{\alpha}}$ עבור $\alpha\geq 1$ קבוע.

אם אכן הסיכוי לטעות $\leq \frac{1}{n^{\alpha}}$ אזי אנחנו נאמר שהאלגוריתם צודק בסיכוי גבוה $\leq 1-\frac{1}{n^{\alpha}}$.

הגדרה: אלגוריתמי לאס וגאס $(Las-Vegas)$ הם אלגוריתמים רנדומיים שזמן הריצה שלהם הוא משתנה מקרי, והנכונות היא דטרמיניסטית. כלומר, האלגוריתם תמיד צודק אך זמן הריצה משתנה. באלגוריתמי לאס וגאס אנחנו נרצה לנתח את התכונות ההסתברויות של זמן הריצה. למשל: לחשב את תוחלת זמן הריצה, או חסם עליון לזמן הריצה בסיכוי גבוה.

הגדרה: אלגוריתמי אטלנטיק סיטי הם אלגוריתמים שגם זמן הריצה וגם הנכונות הינם משתנים מקריים. (לא נתעסק בהם בקורס).

0.2 וידוא כפל מטריצות

קלט: 3 מטריצות בינאריות מגודל $n \times n$. נסמנן A, B, C.

פלט: לבדוק האם $C = A \times B$ באשר הכפל הוא במודולו 2. (לא בינארי, אלא כפל מעל $\mathbb{Z}_2$).

הערה. כפל במודולו 2 הכוונה היא שכפל הוא כמו AND וחיבור הוא כמו XOR. כלומר: $0 \times 1 = 0, 0 \times 0 = 0, 1 \times 0 = 0, 1 \times 1 = 1$ וכן $0 + 0 = 0, 1 + 1 = 0, 1 + 0 = 1, 0 + 1 = 1$.

אלגוריתם נאיבי: נכפיל את A בB ונבדוק האם אכן הפלט הינו C. ראינו כבר שההכפלה תעלה $O(n^{\omega})$. וזה - לא ממש טוב לנו.

0.2.1 נסיון ראשון

נראה אלגוריתם מונטה קרלו שזמן הריצה שלו הינו $O(n^2)$ שטועה בסיכוי $\frac{1}{2} \geq$.

להלן האלגוריתם:

```
Verify-Binary-MM-Basic(A, B, C)
1  Pick a random vector V⃗ = (v_1, v_2, ..., v_n) of bits
2  V⃗_B ← B · V⃗
3  V⃗_AB ← A · V⃗_B
4  if V_AB = C · V⃗
5        return "true"
6  else return "false"
```

האלגוריתם די ברור. הוא מבצע את המכפלה באמצעות וקטור ערכים אקראיים המורכב מביטים, בשני שלבים, ולאחר מכן בודק האם המכפלה הזו שקולה למכפלה של המטריצה C בוקטור. אם כן: מחזיר אמת, אחרת מחזיר שקר. נשים לב כי אם $C = AB$ אזי בהכרח לכל וקטור $\vec{v}$ יתקיים $C \times \vec{v} = AB \times \vec{v}$

נשים לב כי אם $C = AB$ האלגוריתם יהיה צודק תמיד, אם $C \neq AB$ אבל בחירה של וקטור לא טוב יוביל את האלגוריתם לטעות. מצב זה נקרא $false\ positive$ (אמרתי נכון, בזמן שלא נכון). אם קיבלנו לא מהאלגוריתם, בהכרח $C \neq AB$.

סיבוכיות זמן הריצה: נראה כי חישוב כל אחד מהשלבים 2 ו3 יעלה באופן נאיבי $O(n^2)$, כל אחד מהשלבים פולט וקטור בגודל n. שלב 4 מבצע שוב כפל שעולה $O(n^2)$. לבסוף הבדיקה האם V_{AB} שווה למכפלה זו עולה $O(n)$ שכן מעבר על n ערכי וקטור. סה"כ סיבוכיות האלגוריתם $O(3n^2 + n) = O(n^2)$.

ניתוח הסיכוי לטעות:

נגדיר $D = C - AB$. נניח $C \neq AB$. אזי המטריצה $D \neq 0$. נסמנה $D = \{d_{ij}\}$

משמעות הדבר: קיים $d_{ij} = 1$. (לפחות אחד שכזה).

הגדרה: נאמר כי וקטור $\vec{v}$ הוא רע אם מתקיים $D \times \vec{v} = (C - AB) \times \vec{v} = 0$ (וקטור שכזה יגרום לאלגוריתם שלנו לטעות). אם $\vec{v}$ אינו רע, אזי נאמר כי $\vec{v}$ הוא טוב. כלומר: אם $D \times \vec{v} \neq 0$.

למה 1: מספר הוקטורים הטובים הוא לפחות כמספר הוקטורים הרעים. (תחת הנחה ש$D \neq 0$)

מסקנה: אם בחרנו וקטור באקראי, והלמה אכן נכונה (מיד נוכיח), אזי הסיכוי לבחור וקטור טוב הוא לפחות $\frac{1}{2}$.

הוכחה: נתאר פונקציה חד חד ערכית שממפה וקטורים רעים לוקטורים טובים. מכאן שבהכרח יתקיים כי $|bad-vectors| \le |good-vectors|$ לפי בדידה, ואז סיימנו את ההוכחה.
יהי $\vec{v}$ וקטור רע. כלומר, $D \times \vec{v} = 0$. כלומר, $\forall_{1\le \ell \le n} : \sum_{k=1}^{n} d_{\ell k} \times v_k = 0$
נזכר כי קיים $d_{ij} = 1$, כיוון שהנחנו $D \neq 0$ (אם אכן $D = 0$ אזי אין סיכוי לטעות).
נגדיר וקטור

$$\vec{w_j} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ .. \\ 0 \end{pmatrix}$$

באשר הגדרת הוקטור הינה 0 פרט למיקום j (אותו j מ$d_{ij} = 1$) בו יהיה 1.
נזכיר. יצאנו מתוך נקודת הנחה ש$\vec{v}$ הינו רע (כלומר, הוא עבד עלינו). נראה כי הוקטור $\vec{v'} = \vec{v} + \vec{v_j}$ הוא וקטור טוב.
כלומר, נוכיח כי $D \times \vec{v'} \neq 0$.

נסמן
$$\vec{v'} = \begin{pmatrix} v'_1 \\ v'_2 \\ .. \\ v'_j \\ .. \\ v'_n \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ .. \\ v_j \\ .. \\ v_n \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ .. \\ w_j \\ .. \\ w_n \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ .. \\ v_j + 1 \\ .. \\ v_n \end{pmatrix}$$

נסתכל על $(D \times \vec{v})_i$. נקבל לפי הגדרה כי -

$$(D \times \vec{v}')_i = \sum_{k=1}^{n} d_{ik} v'_k = \sum_{k=1}^{n} d_{ik}(v_k + w_k) = \sum_{k=1}^{n} d_{ik} v_k + \sum_{k=1}^{n} d_{ik} w_k$$

נשים לב כי $\sum_{k=1}^{n} d_{ik} v_k = 0$ כפי שהסברנו מעלה (כי הנחנו $D \times \vec{v}$ שווה לאפס, וטענו לגבי ℓ בפרט $\ell = i$ יתקיים).
כלומר:

$$(D \times \vec{v}')_i = \sum_{k=1}^{n} d_{ik} w_k = d_{ij} w_j + 0 + 0 + ... + 0 = 1 \times 1 = 1$$

באשר כל שאר הערכים הינם אפס פרט ל$w_j = 1$, וכן $d_{ij} = 1$. סה"כ קיבלנו כי $(D \times \vec{v})_i = 1$ ולכן $D \times \vec{v}' \neq 0$.
סה"כ קיבלנו פונקציה:

$$f(\vec{v}) = \vec{v} + \vec{w_j}$$

נוכיח כי f חד חד ערכית.

נניח בשלילה כי f אינה חד חד ערכית. כלומר, קיימים $X = \begin{pmatrix} x_1 \\ x_2 \\ .. \\ .. \\ x_{n-1} \\ x_n \end{pmatrix}$ ו $Y = \begin{pmatrix} y_1 \\ y_2 \\ .. \\ .. \\ y_{n-1} \\ y_n \end{pmatrix}$
באשר $X \neq Y$. עבורם $f(X) = f(Y)$.
נסמן, $X' = f(X) = X + w_j$ וכן $Y' = f(Y) = Y + w_j$, קיבלנו כי $\forall_{1\leq\ell\leq n}$ מתקיים $x'_\ell = y'_\ell$. נשים לב כי הוספה או הזזה של w_j בוקטור בסך הכל משנה את הביט הj בוקטור. מכאן שאם $X' = Y'$ אזי לא יתכן כי $X = Y$. בסתירה. (בסה"כ מוסיפים בשתי הצדדים אותו דבר, ולכן בהכרח נקבל $X = Y$, בסתירה). מסקנה: f חד חד ערכית.

0.2.2 נסיון שני ־ אמפליפקציה $(Amplification)$

הרעיון הוא: נריץ את האלגוריתם הבסיסי k פעמים. נתבונן באלגוריתם הבא:

```
Verify-Binary-MM(A, B, C)
k ← α log n
for i = 1 to k
    if Verify-Binary-MM-Basic(A, B, C) = "false"
        return "false"
return "true"
```

אם באחת האיטרציות יוחזר לנו $false$ מהאלגוריתם הבסיסי, אז האלגוריתם יחזיר שקר. כלומר: בשביל שהאלגוריתם יטעה בכל האלגוריתם, ויחזיר אמת למרות שלא מתקיים $C = AB$ אזי צריך לבחור k פעמים וקטור רע.
נגדיר את המאורע: A= בכל איטרציה אנחנו בוחרים וקטור רע. נשים לב כי הסיכוי לטעות בכל איטרציה בלתי תלוי באחרות. לכן נגדיר B כמאורע של הסיכוי לטעות באיטרציה אחת.

$$Pr[A] = (Pr[B])^k \leq (\frac{1}{2})^k = \frac{1}{2^k}$$

נרצה כי זמן הריצה יהיה $\frac{1}{n^\alpha}$ עבור $\alpha \geq 2$ קבוע. זה יקרה באשר $\frac{1}{2^k} = \frac{1}{n^\alpha}$ כלומר $n^\alpha = 2^k$ ומכאן $k = log(n^\alpha) = \alpha log(n)$.
ואכן אם נגדיר את k להיות $\alpha log(n)$ נקבל כי

$$Pr[A] \leq \frac{1}{2^k} = \frac{1}{2^{\alpha log(n)}} = \frac{1}{n^\alpha}$$

סיבוכיות זמן הריצה: נראה כי הלולאה מתבצעת k פעמים, בכל איטרציה קוראים לאלגוריתם הבסיסי שעלותו $O(n^2)$ ונקבל $O(n^2k) = O(\alpha n^2 logn) = O(n^2 logn)$.

0.3 מיון מהיר ־ $Quick\ Sort$: מבוא

בעיית המיון:

קלט: מערך $A=[a_1,...,a_n]$ של מספרים ממשיים (שונים ־ לא מהותי, מהותי מתמטית מבחינת ההוכחות).

פלט: מיון של A בסדר עולה.

מיון מהיר:

בחירה של איבר אקראי a_r, ויצירת $pertiton$ סביב a_r. כל מה שמימינו יהיה גדול ממנו וכל מה שמשמאלו יהיה קטן ממנו. ואת הצד הימני והצד השמאלי, פותרים איך לא: ברקורסיה. לאחר הרקורסיה נקבל את L ממויין, a_r בניהם ואת R ממויין וסה"כ נשרשר את שני המערכים ונקבל את A ממויין.

באלגוריתם הקלאסי r נבחר באופן אקראי אחיד בטווח האינדקסים השלמים $[1,n]$. חשוב שנשים לב ־ תמיד האלגוריתם מצליח למיין. מה שלא תמיד קורה: הוא זמן הריצה שמשתנה. ומכאן ־ **מדובר באלגוריתם לאס וגאס**.

נסמן ב$T(n)$ את זמן הריצה על קלט בגודל n. במקרה הגרוע: $r=1$ או $r=n$ ואז אנחנו צריכים לבצע מיון על קבוצה כלשהי בגודל $n-1$ (בוודאות L או G היא קבוצה ריקה). בשני המקרים הללו נקבל כי $T(n)=T(n-1)+O(n)=O(n^2)$ כאשר הקלט קטן באחד, וכן נדרש עוד $O(n)$ לביצוע ה$partion$ (האיחוד לבסוף).

באופן כללי ־

$$T(n)=T(|L|)+T(|G)+O(n)$$

ומכאן כי במקרה הגרוע עלות האלגוריתם יכולה להגיע ל$O(n^2)$.

מקרה טוב עבורנו ־ a_r הינו החציון של A. במקרה זה נקבל:

$$T(n)=T(\frac{n}{2})+T(\frac{n}{2})+O(n)=2T(\frac{n}{2})+O(n)\in O(nlogn)$$

כלומר: אם החלוקה תהיה תמיד בדיוק בחצי, אזי עלות זמן הריצה של האלגוריתם תהיה $O(nlogn)$.

נשים לב כי ־ לא צריך חלוקה עד כדי כך טובה. גם אם החלוקה היא כזו שגודל כל צד הוא לפחות רבע מn, זמן הריצה יצא עדיין $O(nlogn)$. ויותר מזה ־ אם כל צד הוא לפחות $\frac{n}{a}$ עבור a קבוע כלשהו, אזי עדיין זמן הריצה יהיה $O(nlogn)$. הבחנה זו, תוביל אותנו לגרסה מעט שונה של $Quick\ Sort$.

0.4 מיון מהיר פרנואידי

האלגוריתם מבצע שינוי מאוד פשוט באלגוריתם המקורי: לאחר שמבצעים חלוקה, אם $|L|<\frac{n}{4}$ או $|G|<\frac{n}{4}$ אזי האלגוריתם מבטל את בחירת ה$Pivot$ הנוכחי ומחפש $Pivot$ חדש. (באותו האופן).

זמן ריצה: אנחנו נרצה לנתח את תוחלת זמן הריצה, שכן זמן הריצה אינו קבוע. חשוב להדגיש שכל חישובי $T(n)$ קודם לכן נבעו מאינטואיציה, ואינם היו מדוייקים. נרצה לחשב כעת את תוחלת זמן הריצה שהיא הגורם המכריע באלגוריתמי לאס וגאס.

נזכר בכלל התוחלת $E[X+Y]=E[X]+E[Y]$.

$$E[T(n)]=E[T(|L|)+T(|G|)+T(pivot)]=E[T(|L|)]+E[T(|G|)]+E[T(pivot)]$$

באשר $T(pivot)$ זהו הזמן שלוקח למצוא $pivot$ טוב. נשים לב כי $|G| + |L| = n - 1$. מכאן:

$$E[T(n)] = E[T(n-1-|G|] + E[T(|G|)] + E[T(pivot)]$$

נתמקד כעת ב$E[T(pivot)]$. נשים לב כי $T(pivot)$ הינו מס' הפעמים שמחפשים $pivot$ כפול $O(n)$, שכן בכל חיפוש $Pivot$ סורקים את המערך פעם אחת (בשביל למצוא מי מעליו ומתחתיו ולגלות את גדלי הקבוצות). נזכר כי $E[aX] = aE[X]$ ומכאן שנסמן T: מס' הפעמים שמחפשים $Pivot$. נקבל:

$$E[T(n)] = E[T(n-1-|G|] + E[T(|G|)] + O(n) \times E[T]$$

בחירת ה$Pivot$ היא ניסוי שמבצעים שוב ושוב, עד שמצליחים למצוא $Pivot$ טוב. מספר הנסיונות עד ההצלחה מתפלג גאומטרית. ולכן אם ההצלחה בכל ניסוי הינה p, אזי תוחלת מס' הנסיונות עד ההצלחה הינה $\frac{1}{p}$. נחשב כעת את p. על מנת ש$Pivot$ יהיה רע, הוא צריך להיות או ברבע האיברים הכי קטנים כי אז $|L| < \frac{n}{4}$. או ברבע האיברים הכי גדולים, ואז $|G| < \frac{n}{4}$. כלומר: בשביל ש$pivot$ יהיה טוב הוא צריך להיות בטווח $[\frac{n}{4}+1, \frac{3n}{4}-1]$. נשים לב כי אלו בדיוק חצי מהאפשרויות ל$pivot$. מכאן המסקנה כי

$$p = Pr[Good - Pivot] = \frac{1}{2}$$

מכאן, $E[T] = \frac{1}{p} = \frac{1}{\frac{1}{2}} = 2$. סה"כ קיבלנו עד כה:

$$E[T(n)] = E[T(n-1-|G|] + E[T(|G|)] + 2 \times O(n)$$

נגדיר פונקציה $T'(k) = E[T(k)]$ ונסמן $|G| = x$. נקבל

$$E[T(n)] = T'(n-1-x) + T'(x) + 2 \times O(n)$$

כיוון שאנו יודעים כי $\frac{n}{4} \leq x \leq \frac{3n}{4}$ נוכל לראות כי $E[T(n)] = T'(n) = O(nlogn)$. כי: $\frac{n}{4} \leq x \leq \frac{3n}{4} \Longleftarrow -\frac{n}{4} \geq -x \geq -\frac{3n}{4} \Longleftarrow n-1-\frac{n}{4} \geq n-1-x \geq n-1-\frac{3n}{4}$ $\Longleftarrow \frac{3n}{4}-1 \geq n-1-x \geq \frac{n}{4}-1$ ונקבל כי

$$E[T(n)] \leq T(\frac{3n}{4}-1) + T(\frac{3n}{4}) + 2O(n) \leq 2T(\frac{3n}{4}) + O(n)$$

והביטוי מימין חסום ב$O(nlogn)$ עם עץ רקורסיה או סתם אינדוקציה שנחסוך כעת. מש"ל.

0.5 תוחלת זמן הריצה של מיון מהיר "קלאסי"

בהתייחס לאלגוריתם הכללי של מיון מהיר, נרצה לחסום את תוחלת מס' ההשוואות.

מתי האלגוריתם מבצע השוואות? האם האלגוריתם משווה בין כל זוג איברים? האלגוריתם לא משווה בין כל זוג איברים. ננסה להבין למה: האלגוריתם מקבל כקלט את המערך A ובוחר r. נניח כי בצד L ישנו a_i ובצד R ישנו a_j. אזי נשים לב: a_i וa_j לא ישוו זה מול זה לעולם.

אזי, מתי האלגוריתם כן ישווה בין a_i לa_j? נהפוך את צורת ההסתכלות. אמרנו כי $A = (a_1, ..., a_n)$. נסמן את הפלט $(y_1, ..., y_n)$. (בהכרח מתקיים $y_1 < y_2 < ... < y_n$).

כעת נשאל ־ מתי האלגוריתם משווה בין y_i לy_j? זה קורה מתי שy_i או y_j נבחרו ל$pivot$ ועד לאותו רגע, עבור כל $i \leq k \leq j$ אף אחד מהאיברים y_k לא נבחר להיות $pivot$. (אם נבחר ־ הם לא ישוו). כל עוד לא נבחר y_k שכזה להיות ה$pivot$ אזי y_i וy_j יהיו יחד בקריאה הרקורסיבית.

אם בפעם הראשונה שהאלגוריתם בוחר $pivot$ מתוך האיברים $y_i, y_{i+1}, .., y_j$ הוא בחירה של y_i או y_j אזי באותה בחירה האלגוריתם משווה בין y_i לבין y_j.

נגדיר משתנה מקרי ונסמן מאורע A: האלגוריתם משווה בין y_i לy_j.

$$X_{ij} = \left\{ \begin{array}{cc} 1 & A : exsist \\ 0 & o.w \end{array} \right\}$$

$$E[T(n)] = (*)E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

$(*)$־נשים לב כי זהו בדיוק מס' ההשוואות.

$$E[X_{ij}] = 1 \times Pr[x_{ij} = 1] + 0 \times Pr[x_{ij} = 0] = Pr[x_{ij} = 1]$$

כפי שאמרנו, ההסתברות ש$Pr[x_{ij} = 1]$ היא ההסתברות שתהיה השוואה בין y_i לבין y_j. אמרנו קודם שהם ישוו רק אם כל האיברים בניהם לא נבחרו להיות $Pivot$ ואחד מהם נבחר. בטווח ישנם $j - i + 1$ איברים. מתוכם, שני איברים אם נבחרים ראשונים (y_i, y_j) יגררו ש$X_{ij} = 1$. הבחירה כאן אקראית ־ יוניפורמית. הסיכוי שx_i או x_j יבחרו ראשונים בטווח הינו $\frac{2}{j-i+1}$. מכאן נקבל $E[x_{ij}] = Pr[x_{ij} = 1] = \frac{2}{j-i+1}$.

$$E[T(n)] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = 2\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j-i+1} =$$

$$2\sum_{i=1}^{n-1}(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + .. + \frac{1}{n-i+1}) \leq 2\sum_{i=1}^{n-1} \sum_{\ell=1}^{n-j+1} \frac{1}{\ell} = 2\sum_{i=1}^{n-1} ln(n-i+1)$$

$$\leq 2\sum_{i=1}^{n-1} log(n) = O(nlogn)$$

כנדרש.

מסקנה: מיון מהיר מבצע בתוחלת $O(nlogn)$ השוואות.

0.6 מיון דלי $(Bucket\ Sort)$

האלגוריתם הינו דטרמיניסטי. אם כן, נניח שהקלט מיוצר בצורה אקראית. נפתור את בעיית המיון כאשר כל מספר בקלט נבחר בהתפלגות אחידה בקטע $[0,1]$. הרעיון יהיה שהאלגוריתם אינו מבוסס השוואות, ולכן בתוחלת נצליח לרדת מ$O(nlogn)$. עם זאת במקרה הכי גרוע נגיע לסיבוכיות כזו גם.
האלגוריתם יעבוד כך: נקח את טווח המספרים $[0,1]$ ונחלק אותו לn דליים. כלומר $[0, \frac{1}{n}, \frac{2}{n}, \frac{3}{n}, ..., \frac{n-1}{n}, 1]$.
נתחיל "לזרוק" את האיברים אחד אחד. לכל דלי יכנסו מס' של איברים. בתוחלת: בכל דלי, יש איבר אחד. שכן יש n מספרים והסיכוי של כל איבר להכנס לתא הוא $\frac{1}{n}$ ליפול בתא מסויים. מכאן שתוחלת מס' האיברים בדלי תהיה $n \times \frac{1}{n} = 1$. אם תאורטית - בכל דלי נפל איבר אחד, אזי קיבלנו מיון של האיברים שכן הדליים בסדר עולה. אם לא, ויתכן מצב תאורטי שבו נפלו הרבה בתא אחד, אז אנחנו בבעיה. פסודו לאלגוריתם יראה כך -

מיון דלי $(A = (a_0, .., a_n))$
- נכניס כל איבר לדלי המתאים (האיבר נבחר מראש אקראית!)
- נמיין כל דלי באמצעות מיון הכנסה
- נשרשר את הדליים לפי סדרם

זמן הריצה:
נסמן $B_i = [\frac{i-1}{n}, \frac{i}{n}]$ הדלי הi. נסמן בn_i את מס' האיברים בB_i. נראה כי n_i הוא משתנה מקרי. כמו כן $\sum_i n_i = n$. נראה כי

$$\forall_{1 \le i \le n} : E[n_i] = \frac{1}{n} \times n = 1$$

להכניס כל איבר לדלי המתאים עולה $O(n)$ זמן. שרשור הדליים גם כן יעלה $O(n)$ זמן. נראה כי השלב השני, מיון כל דלי באמצעות מיון הכנסה, יעלה לכל n_i $O(n_i^2)$ כי משתמשים במיון הכנסה. מכאן, זמן הריצה הינו $O(n) + \sum_{i=1}^{n} n_i^2$
נסתכל על תוחלת זמן הריצה.

$$E[n + \sum_{i=1}^{n} n_i^2] = E[n] + \sum_{i=1}^{n} E[n_i^2] = n + \sum_{i=1}^{n} E[n_i]^2$$

נתמקד בבאקט B_i ובn_i. נשים לב שהאלגוריתם לוקח n איברים וכל אחד מהם מצליח להכנס אל הבאקט B_i בסיכוי $\frac{1}{n}$. n_i סופר את מספר ההצלחות. כלומר: יש לנו ניסוי שחוזר n פעמים עם סיכוי הצלחה $p = \frac{1}{n}$. ספירת מס' ההצלחות היא התפלגות בינומית $n_i \sim Bin(n, \frac{1}{n})$. מכאן ש

$$E[n_i] = \frac{1}{n} \times n = 1$$

$$Var[n_i] = np(1-p) = 1 - \frac{1}{n}$$

נזכר כי $Var[n_i] = E[n_i^2] - E[n_i]^2$ ומכאן נוכל לקבל כי

$$E[n_i^2] = Var[n_i] + E[n_i]^2 = 1 - \frac{1}{n} + 1^2 = 2 - \frac{1}{n}$$

סה"כ נחזור מעלה ונקבל

$$E[n + \sum_{i=1}^{n} n_i^2] = n + \sum_{i=1}^{n} E[n_i]^2 = n + n(2 - \frac{1}{n}) = n + 2n - 1 = 3n - 1 = O(n)$$

כלומר, תוחלת זמן הריצה הינה $O(n)$.

נותרת השאלה: למה בחרנו במיון הכנסה? באמצעות מניפולציות מתמטיות קל לטפל בn_i^2 הרבה יותר מ$n_i log(n_i)$, כמו כן: בחרנו תאים מאוד קטנים ואנחנו מצפים שלא יהיה שם הרבה ערכים. מיון הכנסה הוא המיון הכי טוב עבור קלטים מאוד מאוד קטנים - הקבועים די קטנים. לעומת זאת, האלגוריתמים של $O(nlogn)$ מחזיקים קבועים גדולים ונהיים יעילים רק עבור n גדול.