סיבוכיות

במדעי המחשב, סיבוכיות (complexity) היא כלי מדד מתמטי של משאבי המערכת הנחוצים לפתרון בעיה נתונה באמצעות מחשב. המשאב העיקרי הנבחן הוא **זמן הריצה** 🏃, כלומר נבחן משך הזמן הנחוץ לשם ביצוע האלגוריתם. משאב נוסף הוא ה**זיכרון** Ⓜ️ הנחוץ לשם ביצוע האלגוריתם.

לאינטליגנציה הכללית

ניתן להביא בחשבון משאבים נוספים, כגון כמה מעבדים נחוצים לשם פתרון הבעיה בעיבוד מקבילי. התורה החוקרת סיבוכיות קרויה תורת הסיבוכיות. ענף הסיבוכיות נבדל מענף החישוביות, שבו נבחנת השאלה האם ניתן בכלל לפתור בעיה נתונה, בלא קשר לכמות המשאבים הנחוצה.

(ויקי פ. ה׳תשפ)

מאפייני הסיבוכיות

בעיה, בהקשר זה, היא **קבוצה של שאלות** בעלות קשר הדוק. דוגמה: בעיית הפירוק לגורמים היא: כאשר נתון מספר שלם כלשהו, הצג את כל הגורמים הראשוניים שלו. **שאלה ספציפית** קרויה **מופע** של הבעיה. דוגמה: השאלה "הצג את הגורמים של המספר 15" היא מופע של בעיית הפירוק לגורמים. עד כאן הקדמה – כדי להבין את ההמשך. בעיה = קבוצת שאלות. מופע – שאלה ספציפית.

סיבוכיות הזמן של בעיה נתונה היא **מספר הצעדים הנחוצים** לפתרון מופע שלה כפונקציה של גודל הקלט של מופע זה, תוך שימוש באלגוריתם היעיל ביותר למטרה זו. אם לפתרונו של מופע שאורך הקלט שלו הוא n סיביות נחוצים n^2 צעדים, סיבוכיות הזמן שלו היא n^2. מספר הצעדים המדויק תלוי במחשב המסוים שישמש לפתרון הבעיה.

*** לאינטליגנציה הכללית***

כדי להימנע מהתייחסות למחשב מסוים נהוג להשתמש במודל מתמטי עבור מחשב: מכונת טיורינג.

(ויקי פ. ה׳תשפ)

כדי להימנע מחישובים טכניים מדי ומכיוון שעיקר העניין הוא הקצב שבו כמות המשאבים הנדרשים גדלה ככל שהקלט לבעיה גדל, נהוג לסמן את סיבוכיות הזמן באמצעות **סימון אסימפטוטי** שמייצג **סדר גודל** של זמן הריצה, ולמדוד את זמן הריצה על פי מספר הביצועים של פעולות בסיסיות (כמו ביצוע פעולה אריתמטית, קריאת ערך של תא בזיכרון וכדומה). בצורה זו סיבוכיות הזמן מייצגת את הזמן הנחוץ לפתרונה בכל מחשב מסוים (עד כדי הכפלה או הוספה של קבוע שתלוי ברמת הביצועים של המחשב המסוים הזה), כך שהסיבוכיות המוצגת אינה תלויה במחשב שישמש לפתרון הבעיה ואינה מציגה את הזמן הדרוש לפתרון הבעיה בשניות, אלא מייצגת את **סדר הגודל של הזמן הנחוץ לפתרון הבעיה כפונקציה של אורך הקלט**.

© MORIYAT

דוגמה: לכיסוח דשא 🌲 יש סיבוכיות לינארית, משום שהכפלת שטח הדשא מכפילה את הזמן הנחוץ להשלמת המשימה. לחיפוש במילון, לעומת זאת, יש סיבוכיות לוגריתמית בבסיס 2: בהנחה שהמחפש מחפש במילון בחיפוש בינארי, הכפלת גודל המילון תוסיף רק צעד אחד - פתיחת המילון באמצעו - למספר הצעדים הנחוץ לביצוע החיפוש, משום שצעד זה מקטין לחצי את גודל הבעיה.

דוגמה זו ממחישה ש**לבעיות שונות עשויה להיות רמת סיבוכיות שונה**.

טבלת סיבוכיות

הטבלה הבאה מציגה רמות אחדות של סיבוכיות בסדר יעילות יורד (n הוא גודל הקלט, c הוא קבוע גדול מ-1):

שם	סימון
קבוע	$O(\mathbf{1})$
לוגריתמי	$O(log(n))$
פולילוגריתמי	$O(logc(n))$
לינארי	$O(n)$
פולינומי	$O(n^c)$
מעריכי	$O(c^n)$
עצרתי	$O(n!)$

כאשר רמת הסיבוכיות של בעיה היא פולינומית (או פחות מזה) הבעיה נחשבת כבעלת פתרון "יעיל", משום שהגדלת אורך הקלט אינה מביאה לשינוי דרמטי בזמן הנחוץ לפתרון הבעיה. לעומת זאת, כאשר לבעיה סיבוכיות מעריכית (אקספוננציאלית), שינוי קטן באורך הקלט מביא לשינוי מהותי בזמן הנחוץ לפתרון הבעיה. לדוגמה, כאשר הזמן הנחוץ לפתרונה של בעיה הוא $O(2^n)$ (סיבוכיות מעריכית) הגדלת אורך הקלט ב-1 מביאה להכפלת הזמן הנחוץ לפתרון הבעיה.

© MORIYAT

סימון אסימפטוטי

סימון אסימפטוטי (ידוע גם כסימון לנדאו) משמש במתמטיקה כסימון מקוצר שמתאר את התנהגותן של פונקציות עבור ערכים הולכים וגדלים (או הולכים וקטנים), וזאת באמצעות השוואתן לפונקציות אחרות.
היתרון שבשימוש בסימונים אסימפטוטיים הוא שהוא מאפשר לקבל הערכה טובה על אופן הגידול של ערכי הפונקציה מבלי שיהיה צורך לדעת אותו במדויק. לסימונים אסימפטוטיים שני תחומים מרכזיים שבהם הם יעילים: במדעי המחשב (זה אנחנו😉) הם משמשים כדי להעריך את הסיבוכיות של אלגוריתמים,
ובמתמטיקה הם משמשים על מנת להעריך את גודל השגיאה בקירובים שונים.

הסימונים והחסמים

הסימון O big / חסם עליון

הסימון הנפוץ ביותר הוא הסימון "O" (קרי: אוֹאוּ גדול). משום שבד"כ, נרצה לחסום את זמן הריצה "מלמעלה".
נניח תוכנית A רצה בזמן $f(n)$ עבור קלט מאורך n .

אם $f(n) = O(g(n))$ אנו בעצם אומרים שהפונקציה היא "קטנה שווה" לפונקציה $g(n)$.

הקבוצה $O(g(n))$ מוגדרת בתור אוסף כל הפונקציות $f(n)$ בעלות אותו תחום וטווח כמו g , עבורן קיימים קבועים חיוביים c n_0 . כך שלכל $n > n_0$ מתקיים $f(n) \leq c \cdot g(n)$ כלומר: עבור ערכי n הולכים וגדלים שמקבלת f, היא קטנה יותר מ-g עד כדי כפל בקבוע (c).

מטעמי פשטות, נהוג לכתוב $f(n) = O(g(n))$ כאשר הכוונה היא שמתקיים $f(n) \in O(g(n))$ (שייך)
כאשר אנחנו אומרים "זמן הריצה הוא $O(n^2)$" הכוונה שזמן הריצה הגרוע ביותר, לקלט הגרוע ביותר חסום ע"י $c \cdot n^2$. c זה איזשהו קבוע. אנחנו בעצם מסתכלים על המקרה הגרוע ביותר.

ביטוי זה עשוי לגרום לבלבול אם מתייחסים אליו כמציין שוויון ולא שייכות; בפרט, כשהוא אינו סימטרי. למשל, מתקיים $n = O(n^2)$ אבל לא $O(n^2) = O(n)$

הסימון o / חסם עליון לא הדוק

בעוד הסימון O מציין כי הפונקציה $f(n)$ חסומה בקצב גידולה על ידי קצב הגידול של $g(n)$ הרי שהסימון "o" (קרי: אוֹאוּ קטן) בא לציין כי קצב הגידול של f קטן ממש יחסית לקצב הגידול של g
בצורה פורמלית, אומרים ש- $f(n)$ היא $o(g(n))$ אם לכל קבוע c קיים n_0 כך שלכל $n > n_0$ מתקיים $f(n) < c \cdot g(n)$

הסימון Θ (תטא / Theta) / חסם הדוק

פירוש הסימון $f = \Theta(g)$ הוא שקצבי הגידול של f ו-g הם שווים אסימפטוטית (כלומר, שתי הפונקציות הן מאותו סדר גודל).

הסימון Ω / חסם תחתון

לעיתים נרצה לדבר על חסם תחתון לבעיה מסויימת. לדוגמא, ברור שמיון של n מספרים דורש לפחות n פעולות נרצה לחסום את זמן הריצה מלמטה.
נאמר ש- $f(n) = \Omega(g(n))$ אם קיים קבוע $c > 0$, וקיים n_0 כך שלכל $n > n_0$
מתקיים: $0 \leq c \cdot g(n) \leq f(n)$

© MORIYAT

מיונים

מיון אקראי (לא יעיל ולא שימושי, אפשר לדלג על הקטע הזה😂)

מיון אקראי (בנוסף מיון טיפש😏, מיון איטי🦕, מיון קוף🙈, ובאנגלית bogosort) הוא אלגוריתם מיון מאד לא יעיל המבוסס על ניסוי וטעייה. האלגוריתם לא שימושי מאד למיון, אך שימושי למטרות לימוד, לדוגמה השוואה עם אלגוריתמים יותר שימושיים.

אם ננסה למיין חפיסת קלפים על ידי מיון זה היינו בודקים כל פעם אם החפיסה ממוינת, אם לא - נזרוק את החפיסה לאוויר, נרים את הקלפים בסדר אקראי. נחזור על התהליך עד שהחפיסה ממוינת.

תיאור האלגוריתם:

הדרך שהאלגוריתם ממומש מיוצגת בפסאודו קוד (קוד דמה המיועד לבני אדם ולא למחשבים) כך:

כל עוד החפיסה לא מסודרת > סדר את החבילה באקראיות

זמן ריצה וסיום:

בהנחה שכל האיברים שונים, מספר ההשוואות הממוצע ישאף ל $(e-1)n!$
ומספר ההחלפות הממוצע יהיה $(n-1)n!$

הסיבה להפרש בין מספר ההשוואות המצופה למספר ההחלפות המצופה הוא שמגלים שלא כל האיברים ממוינים רק אחרי מספר השוואות, ללא תלות במספר האיברים הכולל, בעוד שמספר ההחלפות תלוי במספר האיברים.

בכל זאת קראת?! הבנת למה המיון לא טוב? גם אם לא – לא נורא😊

מיון בחירה

מיון בחירה (באנגלית: selection sort) הוא אלגוריתם מיון השוואתי פשוט אך לא יעיל.

תיאור האלגוריתם:

1. מצא את האיבר בעל הערך הנמוך ביותר במערך
2. החלף אותו עם האיבר הראשון במערך
3. המשך באותה שיטה על שאר המערך (ללא האיבר הראשון)

זמן ריצה

זמן הריצה הממוצע של האלגוריתם הוא $\Theta(n^2)$ פעולות (כמו, למשל, מיון בועות).
מבחינת צריכת זיכרון האלגוריתם חסכוני, והוא דורש $\Theta(1)$

חישוב סיבוכיות / ניתוח זמן הריצה

אפשר לראות שהלולאות יעברו על $n-1$ איברים בלי שום קשר לערכים במערך. לכן בהינתן מערך בגודל n, זמן הריצה של האלגוריתם הוא קבוע, כלומר אין מקרה טוב ומקרה רע. בכל מקרה לפי צעד 3, סורקים את שארית המערך n פעמים. זה מביא אותנו לסיבוכיות של $\Theta(n^2)$.

זו כנראה הדרך האינטואיטיבית ביותר למיין, אך כאמור היא איננה יעילה במיוחד במונחים של זמן ריצה.

© MORIYAT

מיון בועות

מיון בועות 💭 (באנגלית: Bubble Sort), הידוע גם בכינוי מיון החלפה הוא מיון השוואתי פשוט. המיון קיבל את שמו מהדרך בה מבעבעים אלמנטים במערך: האלמנטים הכבדים בכיוון אחד, והקלים בכיוון ההפוך, וכך גם בינם לבין עצמם.

תיאור האלגוריתם

למערך בגודל n לכל $1 \leq i \leq n-1$, בצע -

אם $a[i] > a[i+1]$ (כלומר, אם האיבר במקום ה-i וזה שאחריו לא מצויים בסדר הנכון) החלף ביניהם.

חזור על התהליך עד שלא ימצאו שני מספרים במיקומים עוקבים שאינם לפי הסדר.

זמן ריצה

המיון פועל בסיבוכיות של $\Theta(n^2)$

חישוב סיבוכיות / ניתוח זמן הריצה

סיבוכיות הזמן של האלגוריתם היא $\Theta(n^2)$ (כיוון שעבור קבוצה של n מספרים דרושים עד n מעברים על הקבוצה, ויהיה צורך לבצע n מעברים במקרה ש- a_n הוא המספר הקטן ביותר בסדרה) דבר שהופך אותו ללא יעיל עבור נתונים רבים. לעומת זאת, עבור נתונים מעטים מאד זהו המיון היעיל ביותר בשל פשטותו. עבור קלט ממוין חלקית או כמעט ממוין ייתכן שזמן הריצה יהיה נמוך יותר כיוון שהאלגוריתם יסיים את סידור המערך בפחות מ- n מעברים על המערך.

צבים וארנבים

🐢🐇🐢🐇🐢🐇

חישוב סיבוכיות / ניתוח זמן ריצה

<u>המקרה הטוב ביותר</u> 🤩 הוא כאשר הקלט הוא מספר ממיון של אברים, כך שבכל איטרציה מעבר לפעולת השמירה של האיבר הנוכחי במשתנה העזר, נעשית רק פעולת השוואה אחת לאיבר הצמוד אליו מהאוסף שכבר מוין, זמן הריצה כולל מעבר על כל איברי האוסף כשלכל מעבר עוד שני פעולות (השוואה ושמירה) כך שסך הכל יש כאן $3n$ פעולות שזה בחישוב סיבוכיות $O(n)$

<u>המקרה הגרוע (ביותר!)</u> 😕הוא כאשר כל האברים ממוינים בסדר הפוך כך שהמיון כולל מעבר על כל אברי האוסף, ובכל איטרציה מספר של i בדיקות לאורך כל המערך הממויין (כי כל איבר מגיע למקום הראשון – אז יש i בדיקות למשל – $A_{0\,1\,2\,3\,4}^{5\,4\,3\,2\,1}$: 4 עובר מקום 1 -45321 - , 3 עובר 2 מקומות – 34521 - , 2 עובר 4 מקומות – 23451 - , 1 עובר 5 מקומות 12345 – ואז המערך ממוין. ו5 לא עובר ולכן זה n-1 שימי לב שכל אחד עבר מקומות כפי המיקום i שלו)

ראינו אם כן שבמקרה זה סך כל הבדיקות שווה ל $1+2+3+\cdots+n-1$ השווה ל $\frac{n(n-1)}{2}$ פעולות (ככה זה על פי הנוסחה. אפשר לקחת דוגמה ולבדוק😊) ולכן סיבוכיות זמן הריצה במקרה זה הוא $O(n^2)$

<u>המקרה הממוצע</u> 🙂 הוא כאשר המערך אינו ממוין וכולל השוואות נוספות מעבר לn איטרציות, כמובן שגם מקרה זה חסום ב$O(n^2)$. אולם למרות סיבוכיות זמן הריצה, לאוספים ממוינים ברובם (כגון אוסף ממוין שהוכנס לו איבר בודד שאינו ממוין) או לאוספים קטנים מאד מיון הכנסה יעיל יותר אפילו ממיונים מהירים כמו מיון מהיר, לעיתים ניתן לממש את המיון מהיר כך שאוספים קטנים עד סף מסוים (לרוב עד עשר) המיון יהיה לפי מיון הכנסה.

מיון מהיר 👊

מיון מהיר (באנגלית: Quicksort) הוא אלגוריתם מיון השוואתי אקראי מהיר במיוחד. סיבוכיות הזמן הממוצעת של האלגוריתם היא $\Theta(n\log n)$ פעולות (כמו, למשל, מיון מיזוג), אך במקרה הגרוע עלול האלגוריתם לדרוש $O(n^2)$ פעולות (כמו, למשל, מיון בועות).
בפועל, אלגוריתם זה נחשב לאלגוריתם המיון ההשוואתי היעיל ביותר הידוע זאת מאחר שהסיכוי למקרה הגרוע הוא מאד נמוך.

תיאור האלגוריתם

אלגוריתם מיון מהיר הוא אלגוריתם רקורסיבי הפועל בשיטת הפרד ומשול. צעדיו הם כדלקמן:

1. בהינתן סדרת איברים, בחר איבר מהסדרה באקראי (נקרא: pivot, או "איבר ציר").
2. סדר את כל האיברים כך שהאיברים הגדולים מאיבר הציר יופיעו אחרי האיברים הקטנים מאיבר הציר.
3. באופן רקורסיבי, הפעל את האלגוריתם על סדרת האיברים הגדולים יותר ועל סדרת האיברים הקטנים יותר.
4. תנאי העצירה של האלגוריתם הוא כאשר ישנו איבר אחד, ואז האלגוריתם מודיע כי הסדרה ממוינת.

שיפורים ווריאציות של האלגוריתם

יעילות האלגוריתם תלויה בבחירת איבר הציר. אם - כתוצאה מ"מזל טוב"😊 - איבר הציר הוא תמיד האיבר האמצעי בגודלו בסדרה, האלגוריתם לוקח $\Theta(n\log n)$
אם, מאידך - כתוצאה מ"מזל רע"☹ - איבר הציר הוא האיבר הקטן ביותר או האיבר הגדול ביותר, אזי האלגוריתם לוקח $O(n^2)$. לכן, ישנה חשיבות רבה למציאת איבר הציר:

© MORIYAT

לעיתים משתמשים אלגוריתמים בפועל בשיטת "חציון משלושה" על מנת לבחור איבר שעשוי להיות "מתאים יותר" מאשר איבר שרירותי.

באופן תאורטי, ניתן למצוא את החציון בסדרה בזמן לינארי, מה שמבטיח זמן ריצה של $\Theta(n \log n)$ אך על אף העובדה שהאלגוריתם למציאת החציון רץ בזמן לינארי, הגורמים הקבועים הכרוכים בו גבוהים למדי, מה שמוביל לזמן ריצה גבוה, ולחוסר שימוש בשיטה זו בפועל.

אחת התכונות של האלגוריתם היא שזמן הריצה שלו עבור קלטים קטנים במיוחד גדול ביחס לאלגוריתמים פשוטים, כגון מיון בחירה (selection sort). לכן, ביישומים רבים תנאי העצירה של האלגוריתם הוא עבור מספר קבוע כלשהו של איברים (7, למשל), ומשלב זה ואילך מתבצע המיון באמצעות אלגוריתם "מיון בחירה" או אלגוריתם דומה.

ל"אינטליגנציה הכללית"

קיימת גרסה איטרטיבית (לא רקורסיבית) של המיון המהיר, אולם היא מורכבת לכתיבה ולא אינטואיטיבית.

(ויקי פ. / 2019)

מיון מנייה (מערך מונים)

במדעי המחשב, מיון מנייה או מיון ספירה (counting sort) הוא אלגוריתם מיון עבור מספרים שלמים המתבסס על העובדה שהמספרים נמצאים בטווח חסום כדי לבצע את המיון בזמן מהיר יותר מזה שמסוגלים לו אלגוריתמי המיון הכלליים. בצורה אינטואיטיבית, די למיון לעבור על קבוצת האיברים שרוצים למיין ולמנות את מספר המופעים של כל אחד מהאיברים, ומכאן שמו של האלגוריתם.

© MORIYAT