מדריך למתקשים בתכנות דינאמי

כל שאלות התכנות הדינאמי בקורס בנויות על אותם שניים-שלושה רעיונות. ברגע שמבינים טוב את הרעיונות הללו, כמעט כל השאלות בנושא נפתרות בקלות. המדריך הקצר הזה נועד להסביר את הרעיונות הללו, אך הוא אינו מספיק. על מנת להבין את הנושא בצורה טובה, חשוב לשבור את הראש לבד על מספר תרגילים לבד (זה כמובן חשוב בכל נושאי הקורס, אך במיוחד בנושא זה).

הפתרון לשאלות תכנות דינאמי מורכב ממספר שלבים קבועים: הגדרת תתי בעיות, הקשר בין תתי הבעיות שהגדרנו לפתרון השאלה, חישוב תתי הבעיות הקטנות ביותר, חישוב תתי בעיות באמצעות תתי בעיות קטנות מהן, וניתוח זמן הריצה. להלן דוגמא להמחשת כל השלבים הללו:

שאלה: נתון מערך A של n מספרים (חלקם שליליים וחלקם חיוביים). מצאו זוג אינדקסים $i \leq j$ שימקסם את הסכום $A[i] + A[i+1] + \cdots + A[j]$.

לדוגמא, אם המערך מכיל את רצף המספרים $5-, 3, 20, 18-, 20, 3, 1-$, זוג האינדקסים שימקסם את הסכום הינו $i = 3, j = 6$ ($20 - 18 + 20 + 3 = 25$). מעבר סטנדרטי על כל האפשרויות ידרוש $O(n^3)$ זמן, ועם שיפור קל תוכלו להגיע ל$O(n^2)$. כעת נראה כיצד לפתור את הבעיה בזמן לינארי עם תכנות דינאמי.

פתרון: נחלק את הפתרון לפי השלבים שתיארנו מעל:

- השלב הראשון הוא **להגדיר תתי בעיות**. הכוונה היא למופעים קטנים יותר של הבעיה הגדולה. במקרה הנוכחי נוכל להגדיר את $c[i]$ להיות ערך הסדרה המקסימאלית שמסתיימת בתא ה-i של המערך. למשל, עבור המערך שבדוגמא מעל, נקבל $c[1] = 1, c[2] = 0, c[3] = 20, c[4] = 2,\ c[5] = 22, c[6] = 25, c[7] = 20$.
- כעת עלינו להגדיר את **הקשר בין תתי הבעיות לפתרון של השאלה**. במקרה הזה, הפתרון של השאלה יהיה $\max_i c[i]$. כלומר, אם נחשב את ערכי כלל תתי הבעיות, נוכל למצוא את הפתרון לשאלה בזמן לינארי.
- נחשב את **תת הבעיה הקטנה ביותר** (לפעמים יש יותר מאחת כזו ואז צריך לחשב את כולן). במקרה הזה, הבעיה הכי קטנה היא $c[1]$, ולא קשה לראות ש-$c[1] = \max(0, A[1])$.
- השלב הבא הוא לחשב **את שאר תתי הבעיות**. הרעיון הוא לחשב את הבעיות לפי הגודל. בשלב הקודם חישבנו את תתי הבעיות הקטנות ביותר. בעזרתן נחשב תתי בעיות קצת יותר גדולות, ואז בעזרתם תתי בעיות יותר גדולות, וכך הלאה. במקרה שלנו, נוכל לחשב $c[i] = \max(0, c[i-1] + A[i])$. שימו לב שכאשר אנו באים לחשב תת בעיה, אנחנו מסתמכים על ערך של תת בעיה קטנה יותר שכבר חישבנו.

- כעת נותר **לחשב את זמן הריצה**. לרוב זה תהליך מאוד פשוט – ניקח את מספר תתי הבעיות ונכפיל אותו בזמן לחישוב כל תת בעיה. במקרה שלנו, ישנן n תתי בעיות, וכל אחת ניתן לפתור בזמן קבוע. לכן, נקבל זמן ריצה של $O(n)$.

מה לגבי הוכחת הנכונות? לרוב אין ממש מה להסביר, ומספיק לרשום שורה או שתיים על איך בדיוק הנוסחה הרקורסיבית עובדת. במקרים נדירים הנוסחה הרקורסיבית לא טריוויאלית ואז צריך להסביר למה בדיוק היא עובדת.

כפי שאולי שמתם לב, החלק הקשה בתהליך הוא למצוא את ההגדרה הנכונה של תתי הבעיות. לרוב שאר השלבים לא דורשים הרבה מחשבה נוספת. כיצד מוצאים את ההגדרה הנכונה לתתי הבעיות? **פשוט מנחשים!** מנסים הגדרה אחת לתת בעיה, ובודקים האם אפשר למצוא נוסחה רקורסיבית עבורה. אם לא מצליחים לגרום לרקורסיה לעבוד, מנסים לשנות את הגדרת תת הבעיה על מנת שתפתור את הקושי שנתקלנו בו. מומלץ לא לשבור את הראש יותר מדי על מהי ההגדרה הנכונה, ופשוט לנסות את ההגדרה שעולה לכם בראש. להלן דוגמא שתמחיש את התהליך הזה.

שאלה: נתון סדרה $s_1, s_2, \dots, s_n$ של מספרים. תארו אלגי יעיל אשר מוצא את אורך תת הסדרה הארוכה ביותר שהינה מונוטונית עולה. האיברים בתת הסדרה אינם צריכים להיות עוקבים בסדרה המקורית, אך כן לשמר את הסדר ביניהם.

לדוגמא, בסדרה 1,2,5,10,5,7,1,3 תת הסדרה המונוטונית-עולה הארוכה ביותר הינה 1,2,5,5,7 (והסדרות 1,1,3 ו-1,5,7, הן גם סדרות חוקיות, אך לא הארוכות ביותר).

פתרון:

קודם כל, עלינו למצוא דרך להגדיר תתי בעיות. הדבר שאולי נראה הכי אינטואיטיבי הוא להגדיר את $c[i]$ כאורך המקסימלי של תת סדרה המוכלת בתת הסדרה $s_1, s_2, \dots, s_i$. במקרה זה, הערך שהאלגוריתם צריך להחזיר הוא $c[n]$. גם את מקרה הבסיס קל לחשב: $c[1] = 1$ (תת סדרה באורך 1).

כעת נרצה לחשב את $c[i]$ באמצעות ערכי ה-$c[\cdot]$ הקודמים (כלומר, לחשב את יחס הרקורסיה). אולי נרצה לקחת את הסדרה שמצאנו באורך $c[i-1]$ ולהוסיף בסופה את s_i. אבל אנחנו לא יודעים האם מותר לנו לעשות זאת! ספציפית, אנחנו לא יודעים האם האיבר s_i קטן מהאיבר האחרון בסדרה שהתייחסנו אליה ב-$c[i-1]$. באופן כללי, לא ברור איך לחשב את $c[i]$ באמצעות האיברים הקודמים.

כיוון שנתקענו, ננסה לנסח את תת הבעיה בצורה כזו שתפתור את הבעיה. ניתן לעשות זאת בכמה דרכים שונות. להלן שתי דוגמאות:

1. נגדיר את $c[i]$ להיות אורך הסדרה המקסימאלית **שמסתיימת באיבר s_i.** כלומר, הסדרה חייבת להכיל את האיבר s_i. הפעם נוכל להגדיר את יחס הרקורסיה: $c[i] = \max_{j<i, s_j \le s_i}(c[j]+1)$. כלומר, אנחנו מחפשים את הסדרה הארוכה ביותר

שמצאנו עד כה שהסתיימה באיבר שאינו גדול מ-s_i, ומוסיפים לה 1 (כלומר, מוסיפים את s_i בסופה). חישבנו n איברים, כל אחד ב-$O(n)$ זמן, ולכן זמן הריצה הוא $O(n^2)$.

2. אפשרות אחרת היא להוסיף פרמטר נוסף להגדרת תת הבעיה. למשל, נוכל להגדיר את $c[i,j]$ להיות האורך המקסימאלי של תת סדרה של $s_1, s_2, \dots, s_i$ שאין בה אף איבר גדול מ-s_j. אם $s_i > s_j$ אז $c[i,j] = c[i-1,j]$, כיוון שאסור להוסיף את s_i לסדרה. אם $s_i \leq s_j$, אז מותר להשתמש ב-s_i ויחס הרקורסיה הוא $c[i,j] = \max\{c[i-1,i]+1, c[i-1,j]\}$. האיבר הראשון במקסימום הינו אורך הסדרה המקסימאלית שמכילה את s_i, והאיבר השני הוא אורך הסדרה המקסימאלית שאינה מכילה את s_i. במקרה זה נחשב $O(n^2)$ איברים, כל אחד בזמן קבוע, ולכן זמן הריצה הוא שוב $O(n^2)$.

לסיכום, ניסינו את הגדרת תתי הבעיות הראשונה שחשבנו אליה, נתקלנו בקושי כאשר ניסינו למצוא את יחס הרקורסיה, ועל מנת להתגבר על הקושי שינינו את הגדרת תת הבעיה. קושי נפוץ בשאלות תכנות דינאמי הוא פחד לנסות הגדרת תת בעיה שאולי אינה נכונה. **אל תפחדו!** פשוט נסו, ואם אתם חושבים שנתקעתם, נסו לעבור לתת בעיה אחרת כדי לעקוף את הקושי.

מעבר למה שדיברנו עליו עד כה, יש רק עוד טריק אחד שחוזר על עצמו בשאלות תכנות דינאמי – טריק שמקל על מציאת יחס הרקורסיה. בגדול, הרעיון הוא לשים לב שיש פיסת מידע שאם היינו יודעים אותה היינו יכולים לחשב את יחס הרקורסיה בקלות. לדוגמא, ניקח את בעיית סידור הפסקה שראינו בתרגול (ונמצאת במצגת באתר). שם תת הבעיה $c[j]$ מוגדרת כעלות הסידור המינימלית עבור פסקה שמכילה רק את j המילים הראשונות. קשה למצוא כאן את יחס הרקורסיה, אבל אם היינו יודעים שהשורה האחרונה בפסקה צריכה להתחיל במילה ה-i, זה היה הרבה יותר פשוט. במקרה הזה, נוכל לחשב $c[j] = c[i-1] + lc[i,j]$ (כאשר $lc[i,j]$ היא עלות השורה שמתחילה במילה ה-i ומסתיימת במילה ה-j). כיוון שאנחנו לא באמת יודעים מהו הערך הנכון של j, נוכל לעבור על כל הערכים האפשריים, ולקחת את הטוב ביותר. כלומר $c[j] = \min_i\{c[i-1] + lc[i,j]\}$.

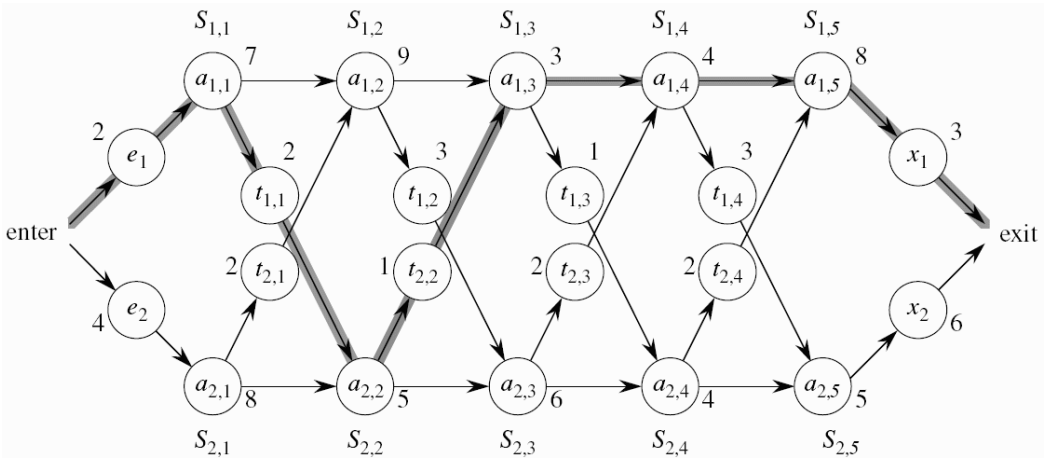
הטריק הזה חוזר על עצמו בכל מני שאלות, כמו בשאלת הסוכן הנוסע שראינו בתרגול, ובשאלה עם המטבעות שבתרגיל הבית. החלק הקשה בטריק הוא להבין מהי פיסת המידע שרלוונטית לנו. כאשר מנסים למצוא את יחס הרקורסיה, מנסים לחשוב "מה קורה לקראת סוף הפתרון של תת הבעיה?" (או איזה מידע מסוף הפתרון היה יכול לעזור לנו לחשב את הפתרון). למשל, בשאלה עם סידור הפסקה, מה שקורה "לקראת הסוף" הוא המילה שמתחילה את השורה האחרונה. באופן דומה, בבעיית הסוכן הנוסע מדובר בעיר לפני האחרונה במסלול.

בגדול עברנו על כל הטריקים הנפוצים בשאלות תכנות דינאמי. אבל למקרה שלא ברור לכם מה ההבדל בין תכנות דינאמי לבין סתם אלגוריתם רקורסיבי, להלן דוגמא קצרה אשר ממחישה את זה (הדוגמא והציור לקוחים מהספר).

שאלה. במפעל מכוניות, כל מכונית צריכה לעבור ב-n תחנות לפני שהיא מוכנה. המפעל מחזיק בשני פסי יצור מקבילים, אשר כל אחד מורכב מ-n התחנות הללו. נסמן את התחנות

בפס הראשון כ-$a_{1,1}, a_{1,2}, \dots, a_{1,n}$ ואת התחנות בפס השני כ-$a_{2,1}, a_{2,2}, \dots, a_{2,n}$. עבור כל פס יצור, נתון לנו הזמן שלוקח להעביר מכונית מכל תחנה לתחנה הבאה במסלול (המחירים אינם זהים בשני המסלולים).

מדי פעם מתקבלת הזמנה דחופה של מכונית, ואז משתמשים בשני פסי היצור בו זמנית על מנת ליצר מכונית אחת. הסיבה לכך היא שייתכן שיותר מהיר להעביר את המכונית מתחנה $a_{1,i}$ אל $a_{2,i+1}$ (כלומר, אל התחנה הבאה בפס האחר) מאשר אל $a_{1,i+1}$. לצורך כך, נתונים לנו גם מחירי ההעברה מכל תחנה לתחנה הבאה בפס האחר. תארו אלגוריתם למציאת מסלול היצור המהיר ביותר.



פתרון:

בכל שלב במסלול יש לנו שתי אפשרויות – להישאר באותו פס יצור, או לעבור לפס האחר. בפתרון רקורסיבי, בכל שלב נריץ רקורסיבית את האלגוריתם עבור שתי תתי הבעיות הללו. זמן הריצה של אלגוריתם כזה הוא $O(2^n)$. לעומת זאת, נסו לפתור את הבעיה עם תכנות דינאמי, ותראו שבלי להתאמץ מקבלים זמן ריצה לינארי.

נכתב ע"י אדם שפר.

גרסא מ-08.01.2012.