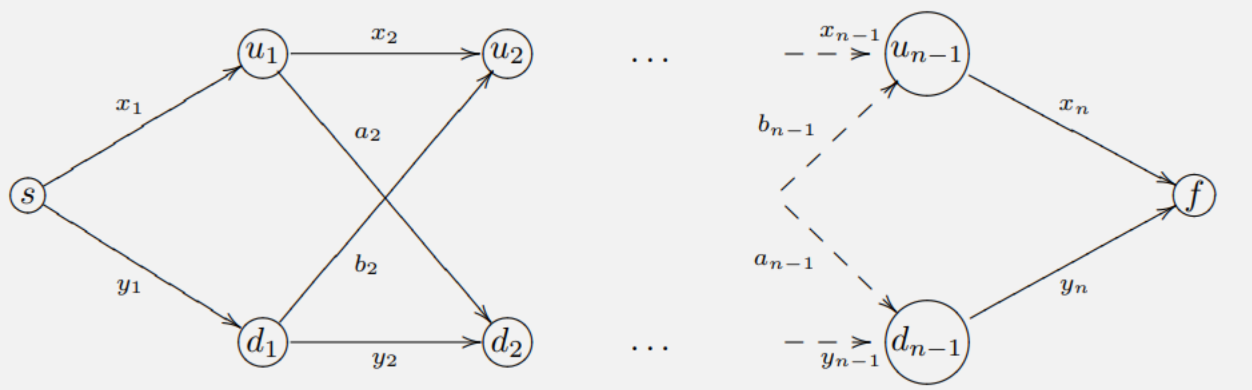
1 אלגוריתמים דינאמיים

1.1 בעיית ניתוב משימות

1.1.1 הצגת הבעיה

בעיית ניתוב משימות

סיפור מסגרת: פריט עובר תהליך ייצור. ישנם 2 פסי ייצור, עם $n-1$ תחנות עבודה על כל פס. הפריט יכול לעבור בין הפסים. לכל מעבר יש מחיר, רוצים לבצע תהליך ייצור במחיר מינימלי.

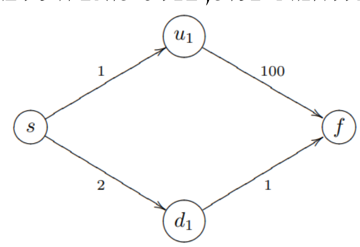


הקלט: גרף מכוון $G=(V,E)$ כנ"ל ורשימות מחירי המעברים: $x_1,...,x_n$ הפס העליון, $y_1,...,y_n$ הפס התחתון, $a_2,...,a_{n-1}$ המעברים מלמעלה למטה, $b_2,...,b_{n-1}$ המעברים מלמטה למעלה.

פלט: המסלול מ-s ל-f בעל המשקל המינימלי.

1.1.2 פתרונות אפשריים:

- הבעיה היא מקרה פרטי של בעיית מציאת מסלול מינימלי בגרף, שניתן לפתור בעזרת אלגוריתם דייקסטרה, בזמן $\Theta(nlog(m))$ נציע אלגוריתם יעיל יותר.
- כלל חמדני פשוט, בכל סיטואציה של בחירה נבחר את האופציה שעולה פחות. אבל זה לא עובד. דוגמא נגדית:



1.1.3 רעיונות בסיסיים כלליים לאלגוריתם דינאמי:

1. נחפש תת-בעיות קטנות יותר (מאותו סוג) שאם נפתור אותן, נוכל לפתור את גם את הבעיה הגדולה. **קריטי** - פתרון אופטימלי של הבעיה המקורית צריך לפתור בצורה אופטימלית את תת-הבעיות.
2. נמפה את אוסף כל תת הבעיות שנצטרך לפתור כדי לפתור את הבעיה המקורית, ונפרמל את הקשר בין תת-בעיה לתת-בעיות קטנות יותר שפתרונן יפתור את תת הבעיה באמצעות נוסחאת רקורסיה. **קריטי** - האוסף הכולל של תת-הבעיות שנצטרך לפתור צריך להיות קטן - כלומר פולינומי בגודל הקלט (כדי לאפשר פתרון יעיל).
3. בניית טבלה שכל תא בה מתאים לאחת מתת-הבעיות שזיהינו.
4. מילוי הטבלה, כאשר מתחילים מתת הבעיות הקטנות ביותר וממלאים תא המתאים לתת בעיה, רק אחרי מילוי התאים המתאימים לתת הבעיות שלה.
5. חילוץ הפתרון של הבעיה המקורית מהטבלה המלאה.

1.1.4 הפתרון שלנו:

אלגוריתם 1 פתרון לבעיית ניתוב המשימות

---

1. נתאים לבעיה המקורית של ניתוב משימות את שתי תת-הבעיות המתקבלות אחרי הצעד הראשון על המסלול. נסמן ב-p^* את ערך הפתרון האופטימלי של הבעיה המקורית. נסמן ב-$p_u(1)$ מחיר המסלול האופטימלי מ-u_1 ל-f. נסמן ב- $p_d(1)$ מחיר המסלול האופטימלי מ-d_1 ל-f.
אז נוסחאת הרקורסיה בשלב הזה היא:
$$P^* = min(u_1 + p_u(1),\ d_1 + p_d(1)\}$$
2. נסמן ב-$p_u(k)$ מחיר המסלול האופטימלי מ-u_k ל-f. נסמן ב- $p_d(k)$ מחיר המסלול האופטימלי מ-d_k ל-f. לכל $1 \le k \le n-1$. אז נוסחאת הרקורסיה הכללית היא:
$$P_u(k) = \begin{cases} x_n & k = n-1 \\ min\{x_{k+1} + p_u(k+1), a_{k+1} + p_d(k+1)\} & k < n-1 \end{cases}$$
$$P_d(k) = \begin{cases} y_n & k = n-1 \\ min\{y_{k+1} + p_d(k+1), b_{k+1} + p_u(k+1)\} & k < n-1 \end{cases}$$
3. הטבלה:

	1	....	k	$k+1$	...	$n-1$
~~p^*~~						x_n
						y_n

4. מילוי הטבלה מתבצע ב-n איטרציות, כך:
   (א) באיטרציה הראשונה נמלא את $T_u(n-1) = x_n$, $T_d(n-1) = y_n$.
   (ב) באיטרציות $2 \le t \le n-1$ נמלא את העמודה ה-$k = n-t$ (הולכים מימין לשמאל) באמצעות נוסחאת הרקורסיה ו-$lookup$ בעמודה ה-$k+1$ של t.
   (ג) נמלא את התא המתאים ל-P^* באמצעות נוסחאת הרקורסיה ובדיקה של העמודה הראשונה של T.
   (ד) נחזיר את P^*.

<u>זמן ריצה</u>

בטבלה $O(n)$ תאים. ממלאים כל תא ב-$O(1)$, סך הכול $O(n)$.

<u>חילוץ המסלול האופטימלי</u>

בזמן מילוי הטבלה, נזכור בכל תא את הבחירה הנכונה שעשינו כדי להגיע למינימום בנוסחאת הרקורסיה.

1.2 בעיית כפל מטריצות

1.2.1 כמה עולה לכפול 2 מטריצות (מלבניות)?

תהיינה A מטריצה בגודל $m \times t$ ו-B מטריצה בגודל $t \times n$. אזי ${}_m C_n = {}_m A_t \cdot_t B_n$ היא בגודל $m \times n$.

לכל $1 \le i \le m$, $1 \le j \le n$, $C_{i,j} = \sum_{k=1}^{t} a_{i,k} \cdot b_{k,j}$. לכן החישוב של $C_{i,j}$ מצריך t פעולות כפל, t פעולות חיבור. לכן לכפל של מטריצות

צריך בסך הכול $m \cdot n \cdot t$ פעולות כפל.

(הערה: ניתן לכפול שתי מטריצות $n \times n$ בזמן של $O(n^{2.37...})$) (השערה: ניתן לכפול בזמן של n^2).

1.2.2 כפל של 3 מטריצות:

דוגמא

$$_{10}D_{100} =_{10} A_{50}B_{20}C_{100}$$

דרך א':

$$D =_{10} (AB)_{20}C_{100}$$

דרך ב':

$$D = A(BC)$$

המחיר של דרך א' הוא: 10^4 עבור (AB), $2 \cdot 10^4$ עבור $(AB)C$. סה"כ: $3 \cdot 10^4$

המחיר של דרך ב' הוא: 10^5 עבור (BC).... (**זה יותר** wtf)

1.2.3 הצגת הבעיה

בעיית כפל מטריצות

הקלט: $n+1$ מספרים טבעיים $p_0, p_1, ..., p_n$ המסמלים מימדים של n מטריצות $A_1, A_2, ..., A_n$ כאשר המטריצה A_i היא מטריצה בגודל $p_{i-1} \times p_i$.

פלט: חלוקת סוגריים המתארת את הדרך היעילה ביותר לחישוב מטריצת המכפלה $B = A_1 \cdot A_2 \cdot ... \cdot A_n$.

(מרחב הפתרונות החוקיים - חלוקת $n-2$ סוגריים וימניים ושמאליים באופן חוקי. (מספרי קטלן - גודל המרחב בערך 4^n).

1.2.4 פתרונות שגויים:

- חמדני: בכל שלב נבחר את אפשרות של שתי מטריצות סמוכות שחישוב המכפלה שלהן הוא הזול ביותר. (**לא עובד**).
- דינמי: לכל $1 \le i \le n-1$ נגדיר $B_{i,i+1}$, נפתור את כל תת-הבעיות מהסוג:

$$A_1 \cdot ... \cdot A_{i-1} \cdot B_{i,i+1} \cdot A_{i+2} \cdot A_n$$

(לא טוב כי יש כמות אקספוננציאלית של תתי בעיות).

1.2.5 הפתרון שלנו (דינמי):

1. תת הבעיות הראשוניות: נפתור את כל $n-1$ תת-הבעיות הבאות. עבור $1 \leq i \leq n-1$ נחשב את:

$$B_{1,i} = A_1 \cdot ... \cdot A_i$$

$$B_{i+1,n} = A_{i+1} \cdot ... \cdot A_n$$

אז $B = B_{1,i} \cdot B_{i+1,n}$.

לכל $1 \leq i \leq n-1$ נסמן ב-$p[1,i]$ אז המחיר האופטימלי לחישוב $B_{1,i}$. נשים לב כי $p[1,n]$ זה המחיר האופטימלי לחישוב $B = A_1 \cdot ... \cdot A_n$.

לכל $2 \leq j \leq n-1$ נסמן $p[j,n]$ את המחיר האופטימלי לחישוב $B_{j,n}$.

אז נוסחאת הרקורסיה הראשונית היא:

$$p[1,n] = \underset{1 \leq i \leq n-1}{min} \{p[1,i] + p[i+1,n] + p_0 \cdot p_i \cdot p_n\}$$

2. תת הבעיות הכלליות: אם נמשיך לחלק את תת הבעיות, נגיע לצורך לפתור את כל הבעיות מהצורה הבאה: $B_{i,j} = A_i \cdot ... \cdot A_j$ לכל $i \leq j$. סה"כ יש $n + \binom{n}{2}$ בעיות (ה-n+ עבור המקרים בהם $i = j$).

לכל $i \leq j$, נסמן ב-$p[i,j]$ את המחיר האופטימלי לחישוב המטריצה $B_{i,j}$.

עבור $i > j$, נגדיר $p[i,j] = 0$.

אז נוסחאת הרקורסיה הכללית היא:

$$p[i,j] = \begin{cases} 0 & i \geq j \\ \underset{i \leq k < j}{min} \{p[i,k] + p[k+1,j] + p_{i-1} \cdot p_k \cdot p_j\} & i < j \end{cases}$$

3. בניית הטבלה ומילויה: נגדיר טבלה T עם n שורות ו-n עמודות. נרצה למלא את הטבלה כך שלכל $1 \leq i,j \leq n$ יתקיים $T[i,j] = p[i,j]$.

דוגמא עבור הקלט $10, 50, 20, 100$:

3	0	0	0
2	0	0	10^5
1	0	10^4	x
	1	2	3

כך ש:

$$x = min\{[p[1,1] + p[2,3] + p_0p_1p_3,$$
$$p[1,2] + p[3,3] + p_0p_2p_3\} = 3 \cdot 10^4$$

ממלאים את הטבלה מפינה שמאלית עליונה לפינה ימנית תחתונה, אלכסון אלכסון.

אלגוריתם 2 פתרון לבעיית כפל מטריצות

נמלא את הטבלה T ב-n איטרציות.

1. באיטרציה הראשונה $(d = 1)$ נגדיר $T[i,j] = 0$ לכל $i \geq j$.
2. עבור $d = 2, .., n$, נמלא באיטרציה ה-d את התאים $T[i,j]$ שעבורם $j - i + 1 = d$ (כלומר כל תת הבעיות המתאימות למכפלה של d מטריצות בדיוק) (מפינה שמאלית עליונה לפינה ימנית תחתונה, אלכסון אלכסון) לפי נוסחאת הרקורסיה הבאה:

$$p[i,j] = \min_{i \leq k < j} \{T[i,k] + T[k+1,j] + p_{i-1} \cdot p_k \cdot p_j\}$$

3. בסיום מילוי הטבלה נחזיר את $T[1,n]$.

<u>זמן ריצה</u>

נראה כי לכל $1 \leq i,j \leq n$ התא $T[i,j]$ מתמלא בזמן $O(n)$. לשם כך מספיק לוודא כי בעת מילוי התא $T[i,j]$, כל התאים $T[i,k]$, ו-$T[k+1,j]$ לכל $i \leq k < j$ כבר מולאו באיטרציות קודמות. לשם כך נשים לב כי התא $T[i,k]$ מולא באיטרציה $k - i + 1 < j - i + 1$ כי $k < j$. באופן דומה, התא $T[k+1,j]$ מולא באיטרציה $j - k + 1 < j - i + 1$ כי $k \geq i$. סה"כ מילוי הטבלה עולה $O(n^3)$.

<u>נכונות</u>

נראה כי לכל $1 \leq i \leq j \leq n$ מתקיים $T[i,j] = p[i,j]$. נעשה זאת באינדוקציה על $1 \leq d \leq n$ כאשר d מסמן את מספר האיטרציה של האלגוריתם למילוי הטבלה.

$d = 1$: לכל $1 \leq i \leq n$, $j = i$ מתקיים:

$$T[i,i] = 0 = p[i,i]$$

נניח נכונות עבור $1, ..., d-1$ ונוכיח עבור d. יהיו i, j כך ש-$j - i + 1 = d$. אזי:

$$T[i,j] = \min_{i \leq k < j} \{T[i,k] + T[k+1,j] + p_{i-1} \cdot p_k \cdot p_j\}$$
$$= \min_{i \leq k < j} \{p[i,k] + p[k+1,j] + p_{i-1} \cdot p_k \cdot p_j\}$$
$$= p[i,j]$$

כאשר $*$ נובע מנוסחאת הרקורסיה למילוי הטבלה, $**$ נובע מהנחת האינדוקציה ו-$***$ נובע מנוסחאת הרקורסיה הכללית. דברים כאלה לא

יופיעו במבחן/בוחן. (לא נידרש להוכיח נכונות של אלגוריתם דינמי כך).

חילוץ הפתרון

כדי לחלץ את הפתרון (חלוקת סוגריים אופטמלית) נזכור בעת מילוי כל תא $T[i, j]$ את ערך ה-k המשיג מינימום בנוסחאת האינדוקציה.

1.3 בעיית התרמיל השלם

1.3.1 הצגת הבעיה:

בעיית התרמיל השלם

הקלט: W - המשקל המירבי של התרמיל. n זוגות של מספרים $(v_1, w_1), ..., (v_n, w_n)$. כאשר v_i הוא הערך של הפריט ה-i, ו-w_i הוא המשקל של הפריט ה-i. הפריטים **לא** ניתנים לחלוקה.

פלט: תת-קבוצה $S \subseteq [n]$ כך ש-$\sum_{i \in S} w_i \leq W$ וכך ש- $\sum_{i \in S} v_i$ מקסימלי.

1.3.2 פתרונות שגויים:

- מעבר נאיבי על תת-הקבוצות S של $[n]$ דורש $\Omega(2^n)$ - יקר מדי.
- זה לא מטרואיד אז האלגוריתם החמדן הגנרי לא עובד (וגם שיטות חמדניות אחרות).
- רמז לעתיד: אפשר למצוא אלגוריתם קירוב!

1.3.3 הפתרון שלנו (דינמי):

1. נוסחאת רקורסיה ראשונית: נסמן $k[\{1, .., n\}, W]$ הערך האופטימלי של פתרון הבעיה. נסמן $k[\{2, ..., n\}, W]$ את ערך הפתרון האופטימלי של בעיית התרמיל השלם עם הפריטים $\{2, .., n\}$ ומשקל מירבי W. ובאופן דומה $k[\{2, ..., n\}, W - w_1]$. אז נוסחת הרקורסיה הראשונית היא:

$$k[\{1, ..., n\}, W] = max\left\{v_1 + k[\{2, ..., n\}, W - w_1], k[\{2, ..., n\}, W]\right\}$$

2. תת-הבעיות: מהצורה $k[\{i, ..., n\}, u]$ כך ש-$1 \leq i \leq n$, $u = W - \sum_{i \in R} w_i$ כך ש-$R \in [n]$ זו קבוצת האינדקסים של הצלעות שהכנסנו. כמה ערכים שונים הסכום $\sum_{i \in R} w_i$ יכול לקבל? 2^n לכל היותר. זו בעיה, אז נניח הנחה מקלה (והגיונית) שכל המשקלים $W, w_1, ..., w_n$ הם מספרים טבעיים. בהנחה זו כל המשקלים $W - \sum_{i \in R} w_i$ הם מספרים טבעיים בין 0 ל-W (ויש בסה"כ $W + 1$ כאלה).

3. נוסחאת רקורסיה כללית: סימון עבור $1 \leq i \leq n$ ועבור $0 \leq u \leq W$ נסמן ב-$k[i,u]$ את הערך האופטימלי של בעיית התרמיל השלם עם קבוצת הפריטים $\{i,...,n\}$ והמשקל המירבי u. אז נוסחאת הרקורסיה הכללית היא:

$$k[i,u] = \begin{cases} 0 & i = n, w_n > u \\ v_n & i = n, w_n \leq u \\ k[i+1,u] & i < n, w_i > u \\ max\{k[i+1,u], k[i+1,u-w_i] + v_i\} & i < n, w_i \leq u \end{cases}$$

4. בניית הטבלה ומילויה: נגדיר טבלה T מלבנית עם n שורות ו-$W+1$ עמודות ונמלא אותה כך ש $T[i,u] = k[i,u]$.

$T[n,0]$	...	$T[n,W]$
$\vdots$		$\vdots$
$T[1,0]$	...	$T[1,W]$

נמלא את הטבלה ב-n איטרציות כאשר באיטרציה ה-$1 \leq k \leq n$ את השורה ה-$n-k+1$ (כלומר מלמעלה למטה). ב-$k = 1$ נגדיר:

$$T[n,u] = \begin{cases} 0 & w_n > u \\ w_n & w_n \leq u \end{cases}$$

עבור $k > 1$ נמלא את השורה ה- $i = n-k+1$ לפי נוסחאת הרקורסיה.

$$T[i,u] = \begin{cases} T[i+1,u] & i < n, w_i > u \\ max\{T[i+1,u], T[i+1,u-w_i] + v_i\} & i < n, w_i \leq u \end{cases}$$

זמן ריצה

כל תא בטבלה מתמלא באמצעות $lookup$ בשני תאים שכבר מולאו ב-$O(1)$. בסה"כ $O(n \cdot W)$.

חילוץ הפתרון

נזכור בעת מילוי כל תא את ההחלטה (לקחת את הפריט ה-i או לא) המשיגה את המקסימום.