1 נכניס את אותיות הקלט למחסנית ולתור בו זמנית. במחסנית נקבל את הקלט כשהוא מהסוף להתחלה ובתור הוא יהיה מההתחלה לסוף. בסיום הקריאה, נשלוף מהמחסנית ומהתור אות ונבדוק שקיבלנו אותה אות.

```
1: procedure IsPalindrome(S, Q)
2:     while the input string wasn't fully read do
3:         c = next char from the input string
4:         Push(S, c)
5:         Enqueue(Q, c)
6:     while S is not empty do
7:         if Pop(S) ≠ Dequeue(Q) then
8:             return False
9:     return True
```

2 לשם הפשטות נניח שהביטוי $-E$ ייוצג ע"י שורש עם האופרטור $-$ ובנו השמאלי הוא שורש הביטוי E. אם הבן השמאלי לא קיים, אז אנחנו בעלה ונחזיר את הערך שלנו. אם הבן הימני לא קיים (אך השמאלי כן) אז אנחנו בשורש של עץ מהצורה $-E$, ונחזיר את הנגדי של תוצאת הפונקציה על הבן השמאלי. אחרת יש לצומת הנוכחי שני בנים, נחשב את שניהם ובהתאם לפעולה בצומת הנוכחי, נחזיר את התוצאה המתאימה.

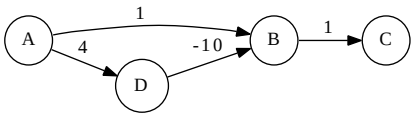
```
1: procedure Calc(Node)
2:     if Node.Left is null then
3:         return Node.Value
4:     if Node.Right is null then
5:         return − Calc(Node.Left)
6:     l = Calc(Node.Left)
7:     r = Calc(Node.Right)
8:     if Node.Value = '+' then
9:         return l + r
10:    if Node.Value = '-' then
11:        return l − r
12:    if Node.Value = '×' then
13:        return l × r
14:    if Node.Value = '÷' then
15:        return l ÷ r
```

3 א

איטרציה	צומת נבחר	שינויים
1	A	$\lambda(C) = 5, \lambda(D) = 3, \lambda(G) = 14$
2	D	$\lambda(E) = 10, \lambda(G) = 9$
3	C	$\lambda(F) = 7, \lambda(E) = 8$
4	F	$\lambda(B) = 14$
5	E	$\lambda(B) = 13$
6	B	
7	G	

3 ב האלגוריתם של דייקסטרה מניח שברגע בחירת צומת, כבר מצאנו את המסלול הקצר ביותר אליו מהצומת התחילי. במקרה שיש קשת עם משקל שלילי, הדבר לא מובטח, שכן יכול לקרות מצב שבו הערך של λ יקטן באיטרציה עתידית.

לדוגמה:



נסתכל על פעולת האלגוריתם:

איטרציה	צומת נבחר	שינויים
1	A	$\lambda(B)=1, \lambda(D)=4$
2	B	$\lambda(C)=1$
3	C	$-$
4	D	$\lambda(B)=-6$

קיבלנו שמשקלו של המסלול הקצר ביותר מ-A ל-C הוא 1 בסתירה לכך שקיים מסלול שעובר דרך D ומשקלו -5.

4 א הפתרון דומה לפתרון החמדני שמתואר במדריך למידה, למעט שינוי קטן: שם בכל שלב לוקחים כמה שאפשר מהפריט שעבורו היחס v_i/w_i מקסימלי עד שלקחנו את כל החלקים של הפריט או שנגמר המקום בשק. כאן נרשה לקחת $q_i \cdot w_i$ חלקים.

4 ב נסתכל על הוקטור $v/w = (\frac{3}{2}, \frac{21}{10}, \frac{6}{5}, 2, \frac{18}{7})$ שבו האיבר ה-i שווה ליחס v_i/w_i לכל $i \leq N$. נמיין אותו ונקבל:

$$v/w = \begin{pmatrix} 18/7 \\ 21/10 \\ 2 \\ 3/2 \\ 30/25 \end{pmatrix}, qc = \begin{pmatrix} 7 \cdot 2 = 14 \\ 20 \\ 24 \\ 30 \\ 100 \end{pmatrix}$$

כאשר qc הוא וקטור הכמויות שהאיבר ה-i שלו מכיל את כמות החלקים שאפשר לקחת מהפריט ה-i. ניקח פריטים בסדר הבא:

1. לוקחים 14 חלקים מהאיבר הראשון ב-v/w שמתאים לפריט החמישי. משקל השק 36.
2. לוקחים 20 חלקים מהאיבר השני ב-v/w שמתאים לפריט השני. משקל השק 42.
3. עוברים לפריט הבא שמתאים לפריט הרביעי ולוקחים 18 חלקים. משקל השק 70.

5 א בתת קבוצה כלשהי בת k איברים קיימות שתי אפשרויות בנוגע לאיבר ה-k: או שהוא נמצא בה, או שהוא לא. כלומר מספר תת הקבוצות בגודל k הוא הסכום של כל אותן תת קבוצות שבהן k נמצא, ותת הקבוצות שבהן נמצאים k איברים שאף אחד מהם אינו האיבר ה-k-י. אזי נוכל לרשום $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ לכל $n, k > 0$ כאשר תנאי העצירה יהיו $\binom{n}{0} = 1$ ו-$\binom{0}{k} = 0$.

5 ב האלגוריתם משתמש במערך דו ממדי A.

```
1:  procedure BINOM(n, k)
2:      A = (n + 1) × (k + 1) array
3:      for r ← 0 to n do
4:          for c ← 0 to k do
5:              if c = 0 then
6:                  A[r][c] = 1
7:              else if r = 0 then
8:                  A[r][c] = 0
9:              else
10:                 A[r][c] = A[r − 1][c − 1] + A[r − 1][c]
11:     return A[r][c]
```

5 ג נמלא את הטבלה שגודלה 8×4:

	0	1	2	3
0	1	0	0	0
1	1	1	0	0
2	1	2	1	0
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10
6	1	6	15	20
7	1	7	21	**35**

$$\uparrow$$
$$\binom{7}{3}$$

6 (ציירתי את העץ רקורסיה על דף, אבל זה ארוך ומייגע לכתוב אותו כאן...)

```
>>> def M(n):
...     if n > 100:
...         return n - 10
...     else:
...         return M(M(n + 11))
...
>>> M(87)
91
```