<u>בעיית התאמת מחרוזות</u>

בעיה: נתון טקסט של תווים ומטרתנו למצוא בטקסט הנתון את כל המופעים של "מילה" מסוימת.

קלט: מערך T באורך n המכיל טקסט (מחרוזת).
מערך P באורך m (כאשר $m<n$) המכיל <u>תבנית</u> שהיא מילה (מחרוזת) מסוימת.

פלט: מספר המופעים של התבנית (P) במחרוזת הנתונה (T).

טרם נדון בבעיית התאמת מחרוזות נדון בהגדרות, סימונים מקובלים ופעולות על מחרוזות של תווים.

כידוע מידע אינו בהכרח מספרים בלבד, אלא גם צירופי אותיות ומילים. מידע לא מספרי מיוצג לרוב בצורת מחרוזות של תווים.

בהמשך נשתמש בשני מונחים: מילה או מחרוזת, אשר שקולים זה לזה.

<u>מילים וקבוצות מילים</u>

<u>הגדרה</u>: <u>אלפבית (א"ב)</u>- זוהי קבוצה סופית לא ריקה של סימנים.

<u>סימון</u>: $\underline{\Sigma}$ - מסמן את האלפבית (א"ב).

דוגמאות: הקבוצה הבאה: $\Sigma=\{1,0\}$ היא א"ב של 2 סימנים אפס ואחד .
הקבוצה הבאה: {א ,ב, ג,...ת}=Σ היא א"ב בן 22 סימנים (אותיות).
הקבוצה הבאה: $\Sigma=\{0, 1, 2, ..., 9\}$ היא א"ב בן 10 סימנים (ספרות).

<u>הגדרה</u>: <u>מילה (מחרוזת)</u> – זוהי סדרה סופית של אותיות מתוך הא"ב (Σ).

<u>הערה חשובה! בהמשך נשתמש הרבה פעמים במונח "מעל" אשר מציין "מתוך האותיות של".</u>

דוגמא: נתון: {א,ב,ג,...,ת}= $\sum$ (א"ב בן 22 אותיות).
"אמא" היא מילה מעל (מתוך האותיות של) $\sum$.
"אבא" היא מילה מעל (מתוך האותיות של) $\sum$.
"אאמ" היא מילה מעל (מתוך האותיות של) $\sum$.
"באא" היא מילה מעל (מתוך האותיות של) $\sum$.

נשים לב לכך ש- 2 מילים "אמא" ו- "אאמ" שונות זו מזו ולכן מילה היא סידרה של אותיות ולא קבוצה של אותיות.

<u>הגדרה</u>: <u>מילה ריקה –</u> זוהי מילה שאינה מכילה שום אות או זוהי סידרה סופית של אפס אלמנטים (כלומר זוהי סידרה שאינה מכילה שום איבר).

<u>סימון</u>: ⌣ או $\in$ מסמן את המילה הריקה. (כלומר ישנם שני סימנים מקובלים לציון מילה ריקה).

<u>הגדרה</u>: <u>אורך של מילה</u> – הינו מספר האותיות שבמילה.

<u>סימון</u>: $|w|$ מסמן את אורך המילה w.

דוגמאות: נתון: $\sum=\{0,1\}$, אז

$|0|=1$, כי במילה 0 ישנו רק סימן אחד והוא אפס.

$|010|=3$, כי במילה 010 ישנן 3 אותיות בסידרה (במילה).

$|1101|=4$ כי במילה 1101 ישנן 4 אותיות בסידרה (במילה).

$|\in|=0$, כי במילה ריקה ישנם 0 תווים.

<u>הגדרה</u>: <u>שרשור של שתי מילים</u> – היא המילה הנוצרת מ- "הדבקת" המילה השניה מימין למילה הראשונה.

<u>סימון</u>: $w_1\bullet w_2$ – מסמן את שרשורן של שתי מילים w_1 ו- w_2.
נקודה "." - מציינת את אופרטור השרשור.

דוגמאות:

1. אם w_1 היא המילה aba כלומר $w_1=aba$ ו- $w_2=ima$
אזי $w_1\bullet w_2=abaima$ ו- $w_2\bullet w_1=imaaba$.

מסקנה: $w_1\bullet w_2$ לא בהכרח זהה ל- $w_2\bullet w_1$, כלומר סדר השרשור חשוב.

2. אם w_1 היא המילה aba, כלומר $w_1=aba$ ו- w_2 היא המילה הריקה, כלומר $w_2=\in$
אזי $w_1\bullet w_2=aba\bullet\in=aba$
וגם $w_2\bullet w_1=\in\bullet aba=aba$

מסקנה: לכל מילה w: $w=w\bullet\in=\in\bullet w$

3. אם $w_1=ab$ ו- $w_2=ab$
אזי $w_1\bullet w_2=ab\bullet ab=abab$
וגם $w_2\bullet w_1=ab\bullet ab=abab$

<u>הערה</u>: בהמשך אנו נוותר על הנקודה " • " . המציינת את אופרטור השרשור ובמקום $w_1\bullet w_2$ נרשום w_1w_2 (בלי ציון הנקודה).

דוגמא: אם $w_1=ab$ ו $w_2=ba$
אזי $w_1w_2=abba$ ו $w_2w_1=baab$

שים לב לעובדות הבאות:

1. כל סימן (אות שבאלפבית) הינו מילה באורך 1.
 דוגמא: אם $\sum=\{0,1\}$
 אז $|0|=1$ (0 – היא גם מילה באורך 1),וגם
 $|1|=1$ (1 היא גם מילה באורך 1).
2. לכל שתי מילים מעל (מתוך האותיות של) $\sum$ w_1 ו- w_2 מתקיים:
 א. $|w_1w_2|=|w_1|+|w_2|$
 ב. לכל סימן (אות) a, כלומר $a\in\sum$, $|w_1a|=|w_1|+1$ כיוון שהאורך של כל סימן (אות) שבא"ב הוא 1.

דוגמא: אם $\sum=\{0,1\}$ וניקח 2 מילים: $w_2=10$ $w_1=010$
נקבל: $|w_1w_2|=|01010|=5$
$|w_1|=|010|=3$
$|w_2|=|10|=2$
ואכן: $3+2=5$ כנדרש.
כמו כן: $|w_11|=|0101|=4$ ו $|w_1|=|010|=3$ וגם $|1|=1$
ולכן $3+1=4$.

<u>הגדרה</u>: <u>חזקה של מילה</u>
נתונה מילה w מעל $\sum$. החזקה של w מוגדרת ומסומנת כדלהלן: $w^0=\in$
לכל $i>0$ $w^i=w^{i-1}\bullet w$

השרשור של מילה w לעצמה מסומן ב- w^2. באופן דומה נסמן ב- w^3 את שרשור המילה 3 פעמים.
כלומר $w^3=w\bullet w\bullet w$ ו- w^3 מורכבת משלושה מופעים של w.
דוגמא: אם $\sum=\{a,b\}$ אזי $(abb)^3=abb\bullet abb\bullet abb=abbabbabb$
באופן כללי $w^n=\underbrace{w\bullet w\bullet w\bullet\ldots\bullet w}_{}$

n מופעים של w

שים לב ל- w^0 אשר מציינת 0 מופעים של w. זוהי בעצם סדרה של 0 אלמנטים שהיא מילה ריקה.

<u>הגדרה</u>: <u>היפוך של מילה</u> – הוא המילה הנוצרת מהיפוך סדר האותיות שבמילה.
<u>סימון</u>: w^r – מסמן את האופרטור היפוך של המילה w.

דוגמאות: אם $\Sigma=\{0,1\}$ ו- $w=1101$ אזי $w^r=1011$
<u>הערה</u>: אם $w=\in$ אזי גם $w^r=\in$

<u>הגדרה</u>: <u>רישא (prefix) של מילה</u> – אם x,w,y מילים כך ש- $x=wy$ אזי המחרוזת w נקראת רישא של המחרוזת x.

<u>סימון</u>: $w=prefix(x)$ – מציין שהמחרוזת w הינה רישא של המחרוזת x.

דוגמא: נתון $\Sigma=\{0,1\}$ והמילה $x=010$ מעל Σ.

$010=\underline{\in}\bullet 010$

$010=\underline{0}\bullet 10$

$010=\underline{01}\bullet 0$

$010=\underline{010}\bullet \in$

ולכן $\in$, 010,01,0 הן הרישות של x.

הערה: כאשר $w=prefix(x)$ ברור כי $|w|\le|x|$

<u>הגדרה</u>: <u>סיפא (suffix) של מילה</u> – אם x,w,y מילים כך ש- $x=wy$ אזי המחרוזת y נקראת הסיפא של המחרוזת x.

<u>סימון</u>: $w=suffix(x)$ – מציין שהמחרוזת w סיפא של המחרוזת x.

דוגמא: נתון $\Sigma=\{0,1\}$ והמילה $x=010$ מעל Σ כידוע:

$010=010\bullet \underline{\in}$

$010=01\bullet \underline{0}$

$010=0\bullet \underline{10}$

$010=\in\bullet \underline{010}$

ולכן $\in$, 010,10,0 הן סיפות של x.

<u>הערה</u>: כאשר $w=suffix(x)$ ברור כי $|w|\le|x|$

<u>מסקנה</u>: המילה הריקה ($\in$) היא גם רישא וגם סיפא של כל מחרוזת.

<u>הגדרה</u>: <u>תת-מילה של מילה (substring)</u> – אם x,w,y,z מילים כך ש- $x=wyz$, אז y נקראת תת-מילה של x.

דוגמא: נתון $\Sigma=\{0,1\}$ והמילה $x=010$ מעל Σ. כידוע:

$010=\in\bullet 010\bullet\in$	$010=\in\bullet 010\bullet\underline{\in}$	$010=0\bullet 10\bullet\in$
$010=\underline{0}\bullet 010\bullet\in$	$010=\in\bullet 01\bullet\underline{0}$	$010=0\bullet 1\bullet\underline{0}$
$010=\underline{01}\bullet 0\bullet\in$	$010=\in\bullet 0\bullet\underline{10}$	$010=0\bullet\in\bullet\underline{10}$
$010=\underline{010}\bullet\in\bullet\in$	$010=\in\bullet\in\bullet\underline{010}$	$010=\in\bullet\in\bullet\underline{010}$

וכך הלאה.

קל לראות כי כל רישא או כל סיפא היא תת מילה אך לא כל תת-מילה היא רישא או סיפא.

- שים לב! עבור שתי מחרוזות כלשהן x ו- y ותו כלשהו a מתקיימים:

א. x היא סיפא של y ‏[x=suffix(y)] אם ורק אם xa היא סיפא של ya ‏[xa=suffix(ya)].

ב. x היא רישא של y ‏[x=prefix(y)] אם ורק אם ax היא רישא של ay ‏[ax=prefix(ay)].

<u>הגדרה</u>: <u>שרשור של קבוצות מילים</u>

שרשור של שתי קבוצות מילים הוא הקבוצה שמכילה את כל אפשרויות השרשור של מילה מהקבוצה הראשונה עם מילה מהקבוצה השניה.

תהיינה L_1 ו- L_2 קבוצות מילים מעל Σ. כאמור פעולת השרשור מוגדרת ומסומנת כך: $L_1 \bullet L_2=\{xy|\ x\in L_1, y\in L_2\}$

דוגמא: אם $L_1=\{0,1,\in\}$ ו- $L_2=\{1,01\}$

$0\bullet1=01$

$\underline{0\bullet01=001}$

$1\bullet1=11$

$\underline{1\bullet01=101}$

$\in\bullet1=1$

$\in\bullet01=01$

ולכן $L_1\bullet L_2=\{01,001,11,101,1\}$

המילה 01 נוצרה פעמיים באופנים הבאים: $0\bullet1=01$ ו- $\in\bullet01=01$ אך היא תרשם רק פעם אחת ב- $L_1\bullet L_2$ כיוון שהתוצאה של פעולת השרשור $L_1\bullet L_2$ הינה קבוצה וכל איבר בקבוצה מופיע רק פעם אחת(לפי הגדרת הקבוצה).

שים לב! יש חשיבות לסדר השרשור (בדומה לשרשור המילים).

<u>הגדרה</u>: קבוצת כל המילים מעל א"ב Σ – זוהי קבוצת כל המחרוזות שאפשר ליצור מן האלפבית Σ.

<u>סימון</u>: Σ^* (כוכב קליני) – מסמן את קבוצת כל המילים מעל א"ב Σ.

$$\Sigma^*=\bigcup_{i\geq0}\Sigma^i$$

דוגמא: נניח כי $\Sigma=\{0,1\}$ נקבל:

$\Sigma^0=\{\in\}$

$\Sigma^1=\Sigma=\{0,1\}$

$\Sigma^2=\Sigma^1\bullet\Sigma^1=\Sigma\bullet\Sigma=\{0,1\}\bullet\{0,1\}=\{00,01,10,11\}$

Σ^2 מציין אוסף כל המחרוזות, שאורכן <u>בדיוק</u> 2, ושאפשר ליצור אותן מן האלפבית Σ.

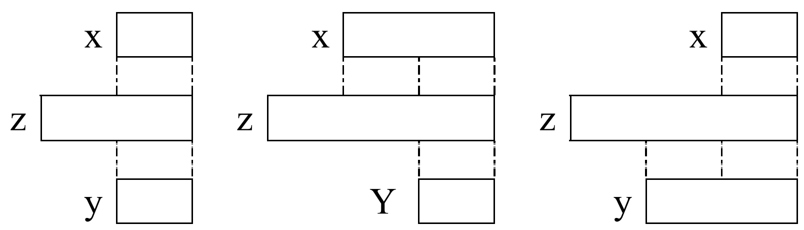
$\Sigma^3=\Sigma^2\bullet\Sigma=\{00,01,10,11\}\bullet\{0,1\}=\{000,001,010,011,100,101,110,111\}$

Σ^3 מציין אוסף כל המחרוזות, שאורכן בדיוק 3, ושאפשר ליצור אותן מן האלפבית Σ.

לכן נסיק כי: Σ^i - תציין אוסף כל המחרוזות, שאורכן בדיוק i, ושאפשר ליצור אותן מן האלפבית Σ.
לכן $\Sigma^*=\bigcup_{i\geq 0}\Sigma^i$ - זוהי קבוצת כל המחרוזות בכל האורכים האפשריים שאפשר ליצור אותן מן האלפבית Σ.

למה: תהיינה x,y ו- z מחרוזות המקיימות:
x=suffix(z) וגם y=suffix(z).
אם |x|≤|y| אזי x=suffix(y)
ואם |x|≥|y| אזי y=suffix(x)
ואם |x|=|y| אזי x=y

להלן התיאור הגרפי של הלמה:



x=y y=suffix(x) x=suffix(y)

<u>טענה</u>: תהי מילה x כאשר $x \in \{a,b\}^*$ אם המילה x מקיימת את התכונה הבאה:
$xa=ax$, אזי $x=a^n$ עבור n כלשהו כאשר $n \in N$.

<u>הוכחה</u>:
נוכיח את הטענה בדרך השלילה. נניח ש $xa=ax$ והמילה x מכילה תו b. לכן נוכל לכתוב כי $x=a^n bu$ ואז מתקיים: $a^n bu \bullet a = a \bullet a^n bu$
לכן מתקיים: $bua = abu$
אך לא יתכן ששתי המילים מקיימות את השוויון, כלומר $abu=bua$ כיוון ש $a \neq b$, כי אם שתי מחרוזות שוות אזי גם התו הראשון שלהן צריכים להיות זהים. לכן הגענו לסתירה. הסתירה נובעת מההנחה שהמילה x מכילה תו b ולכן המסקנה היא שהמילה x לא מכילה תו b כלומר $x= a^n$ עבור n כלשהו. מש"ל

<u>טענה</u>: לא קיימת מילה x כאשר $x \in \{a,b\}^*$ כך שיתקיים $ax=xb$.

<u>הוכחה</u>: אם המילה $x \in \{a,b\}^*$ כך ש $|x| = n$ (האורך של המילה x) אזי עלינו להוכיח כי לכל $n \in N$ $ax \neq xb$.

נוכיח את הטענה באינדוקציה מתמטית.
<u>בסיס</u>: כאשר $n=0$ כלומר $|x|=0$ אזי המילה x היא מילה ריקה ולכן $ax \neq xb$ כיוון ש $x=\smile$ ו $a \neq b$.

<u>הנחת האינדוקציה</u>: נניח את נכונות הטענה לכל m כאשר $m<k$ עבור k כלשהו נתון, כלומר אם $|x|=m$ ו $m<k$ אזי $ax \neq xb$.

עתה נוכיח את נכונות הטענה עבור המילה x, כאשר $|x|=k$. נראה זאת בדרך השלילה, כלומר נניח ש $|x|=k$ ו $ax=xb$ מאחר ו $a \in \Sigma$, $b \in \Sigma$ ו $x \in \Sigma^*$ כאשר $\Sigma = \{a,b\}$ ו $ax=xb$ כלומר:

a	x

שווה ל-

x	b

אזי ברור כי: המילה x מתחילה באות a ומסתיימת ב b.
אי לכך נוכל לכתוב כי $x=awb$ כאשר $w \in \{a,b\}^*$ ו $|w|<|x|$. אז נקבל:

$$ax=xb$$
$$a \bullet awb = awb \bullet b$$
כלומר $aw=wb$

אך כאמור $|w|<|x|$ לכן לפי הנחת האינדוקציה היינו צריכים לקבל כי: $aw \neq wb$
קיבלנו סתירה עקב הנחתינו ש $ax=xb$ ו נסיק כי $ax \neq xb$ לכל מילה x.
מש"ל.

<u>תרגילים</u>:

1. נתון כי $A1 = \{a,b\}$ ו $A2=\{1,2,3\}$
   א. מצא את: $A=A1{\cdot}A2$.
   ב. בהמשך לסעיף א' מצא את A^r כאשר $A^r=\{w^r|w\in A\}$
   ג. מצא את כל הרישות (prefixes) ואת כל הסיפות (suffixes) של כל מילה שבקבוצה A.

2. נתון כי:
   א. $\Sigma=\{0,1\}$
   ב. $A=\{0^n1^n|n\geq 1\}$ כלומר A היא אוסף המילים שהן שרשרת של מספר כלשהו של aיים ולאחריה שרשרת של אותו מספר bיים.
   ג. $B=\{w\in\{0,1\}^*|\#_0(w)=\#_1(w)\}$ כלומר B הוא אוסף של המילים מעל $\{0,1\}$ שבהן מספר האפסים שווה למספר האחדים.
   הסימון $\#(w)$ פירושו מספר המופעים של האות a במילה w.
   כך למשל $\#_a(bb)=0$ ו-$\#_a(baa)=2$.

   1. עבור כל המילים הבאות קבע אם היא שייכת לקבוצה A או לאו.
      $\in$, 0011, 0101, 1100, 00111, 00011. נמק תשובתך.

   2. אותה שאלה כמו בסעיף א' כאשר נבדקת השייכות לקבוצה B .

   3. האם $A\subset B$ או $B\subset A$ או $A=B$? נמק תשובתך.

   4. האם $A\subset \Sigma^*$? נמק

3. נניח ש- $v,w,x,y \in\Sigma^*$ ונניח ש: $xy=vw$ אזי קיימת $z\in\Sigma^*$ כך ש:
   1. $x=vz$ ו- $w=zy$
   או
   2. $v=xz$ ו- $y=zw$

4. נניח ש- $x,y \in\Sigma^*$ הראה שאם $xy=yx$ אזי קיימת $z \in\Sigma^*$ כך ש- $x=z^m$ ו- $y=z^n$ עבור $m,n\geq 0$.

5. נניח $x,y,z \in\Sigma^*$. הראה שאם $xz=zy$ אזי $x^kz=zy^k$ לכל $k\geq 0$.

6. שתי מילים $x,y\in\Sigma^*$ נקראות צמודות אם קיים $z\in\Sigma^*$ כך ש- $xz=zy$. הראה ש- x ו- y מילים צמודות אם ורק אם קיימות $u,v \in\Sigma^*$ כך ש: $x=uv$ ו- $y=vu$.

7. נניח ש- $x,y,z \in\Sigma^*$ הראה שאם $x^2y^2=z^2$ אזי $xy=yx$.

8. הראה כי לא קיימות שתי מחרוזות $x,y\in \{a,b\}^*$ כך ש: $xay=ybx$.

9. לפניכם טענה: אם $u,v \in\{a,b\}^*$ ו- $u\neq v$, אזי לכל שתי מלים $x,y \in \{a,b\}^*$ מתקיים $xuy\neq yvx$.

א. הראו בעזרת דוגמה נגדית שהטענה <u>איננה</u> נכונה.

ב. נתונה ה"הוכחה" הבאה לטענה:

נעיין בפרדיקט הבא: אם $x,y \in\{a,b\}^*$ ו- $|xy| =n$, אז לכל שתי מלים u ו- v מעל $\{a,b\}$, $u\neq v$, מתקיים $xuy\neq yvx$.

נוכיח את נכונותו של הפרדיקט לכל $n\in N$:

עבור $n=0$ הוא נכון.

נניח את נכונותו לכל $m<k$, עבור k נתון כלשהו, ונוכיח את נכונותו ל- k:

נניח בשלילה שקיימות שתי מלים x,y כך ש- $|xy| =k$, ושתי מלים שונות u ו- v כך ש- $xuy=yvx$.

מן השוויון הזה נובע שהאות הראשונה של x זהה לאות הראשונה של y, והאות האחרונה של x זהה לאות האחרונה של y.

נסמן: $x=\alpha x_1\beta$, $y=\alpha y_1 \beta$ ($\alpha,\beta\in\{a,b\}$).

נקבל: $\alpha x_1\beta u\alpha y_1\beta=\alpha y_1\beta v\alpha x_1 \beta$

ומכאן: $x_1\beta u\alpha y_1=y_1\beta v\alpha x_1$

מכיוון ש- $u\neq v$, גם $\beta u\alpha\neq\beta v\alpha$

נסמן: $u_1=\beta u\alpha$ ו $v_1=\beta v\alpha$

נקבל: $x_1u_1y_1=y_1v_1x_1$.

אבל $|x_1y_1| <k$ ולכן על-פי הנחת האינדוקציה, $x_1u_1y_1\neq y_1v_1x_1$. סתירה!

מהסתירה ניתן להסיק שהפרדיקט נכון גם ל- k, ועל כן הוא נכון לכל $n\in N$.

מה כאן לא בסדר?

עתה נחזור לבעיה המקורית – בעיית התאמת מחרוזות
(string –matching problem). בבעיה זו נתון טקסט (מחרוזת של תוים) באורך n שנמצא במערך [T[0…(n-1) ונתונה תבנית (pattern) (מחרוזת של תוים) שאורכה m, כאשר m<n, והיא מוחזקת במערך [P[0…(m-1). המחרוזות T ו- P מעל אותו אלפבית -Σ.

המטרה העיקרית בבעיה זו למצוא את כל המופעים של התבנית P בטקסט הנתון T, כלומר עלינו למצוא את כל האינדקסים k במערך T כך ש: $T[k..k+m-1]=P[0…m-1]$
כאשר $0 \le k \le n-m$

דוגמא: נתון:

T טקסט	b	a	b	c	a	b	a	b	b	a	b	a	b
P תבנית	0	1	2	3	4	5	6	7	8	9	10	11	12

aba

התבנית P מופיע בטקסט T פעמיים. החל מאינדקס 4, כלומר $T[\underset{456}{aba}]=P[\underset{012}{aba}]$,

והחל מאינדקס 9, כלומר $T[\underset{91011}{a\ b\ \ a}]=P[\underset{012}{aba}]$

כלומר עבור כל j מ- 0 עד 2 מתקיים: $T[k+j]=P[j]$ כאשר $k=4$ או $k=9$.

אלגוריתם א' – אלגוריתם הנאיבי

עתה נתאר פונקציה (קטע קוד) אשר מקבלת שתי מחרוזות T ו- P. הפונקציה תדפיס את כל השלמים (k) המציינים את מיקום המופע של התבנית P בטקסט T.
עבור כל אינדקס k שמקבל ערכים מאפס עד n-m נבדוק האם $T[k…k+m-1]=P[0..m-1]$.

האיור הבא מתאר את הרעיון של האלגוריתם הנאיבי להתאמת מחרוזות. לפעמים תמונה אחת שווה אלף מילים וגם במקרה זה האיור שמטה מתאר את ה"אלף מילים" ללא מילים.

T[0]	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]	T[7]	T[8]	T[9]	T[10]
A	B	C	E	F	G	A	B	C	D	E
P[0]	P[1]	P[2]	P[3]							
A	B	C	D							
	A	B	C	D						
		A	B	C	D					
					…					
						A	B	C	D	

```
n=length(T);
m=length(P);
for(k=0;k<=n-m;k++)
{
        /*החל מהנקודה הזו מתחילה לולאה המבצעת בדיקה האם P[0..m-1]=T[k…k+m-1]*/
    count=0;
    for(j=0;j<=m-1;j++)
    if(P[j]==T[k+j]) count ++;

        /*עתה נבדוק האם היתה התאמה מלאה*/
    if(count==m)printf("%d\n",k);
}
```

הלולאה החיצונית רצה בזמן $0(n-m+1)$ ו- הלולאה הפנימית רצה בזמן $0(m)$. כיוון שהלולאות מקוננות זה בזה, אזי לפי עיקרון הכפל סיבוכיות זמן הריצה של האלגוריתם הינה $0((n-m+1)m)$.

<u>שיפור לאלגוריתם א'</u>

נתבונן על המקרה הבא כאשר k=6.

T[	a	b	c	a	b	c	a	b	a	a	c	d…]
	0	1	2	3	4	5	6	7	8	9	10	11 ….

P[	a	b	a	a	b	e]
	0	1	2	3	4	5

באלגוריתם א' אנו מבצעים את ההשוואה הבאה: T[6..11]=P[0..5]

ניתן לראות כי: P[3]=T[9],P[2]=T[8],P[1]=T[7],P[0]=T[6]

אך P[4]=b שונה מ- T[10]=c.

לכן מיותר להמשיך את ההשוואות הבאות ואין טעם לעבור על ה-j - ים הגדולים מ- j_1 כאשר $P[j_1]\neq T[k+j_1]$.

לאור זאת נציע אלגוריתם משופר יותר אשר משתמש במשתנה בוליאני הבא:

$$finis=\begin{cases} true & \text{כאשר נמצא אינדקס כלשהו } j_1\text{, } 0\leq j_1\leq m \text{ כך ש- } P[j_1]\neq T[k+j_1] \\ false & \text{אחרת} \end{cases}$$

ברור כי בתחילת האלגוריתם לפני שמתחילים את הבדיקה האם P[0..m-1]=T[k..k+m-1] המשתנה finis יקבל את הערך false כיוון שלפני הסריקה עדין לא מצאנו אינדקס j_1 כך ש- $P[j_1]\neq T[k+j_1]$.

להלן האלגוריתם המשופר

```
#define true 1
#define false 0
 n=length(T);                                  /* n ← T אורך המחרוזת*/

 m=length(P);                                  /*m ← P אורך המחרוזת*/
/* החל מנקודה זו מתחילה הלולאה המבצעת בדיקה האם P[0..m-1]=T[k..k+m-1] */
                    for (k=0;k<=n-m;k++)
{ finis=false; j=0;
   while ((j<=m-1)&&(!finis))
      if(T[k+j]==P[j]) j++;
      else finis=false;
   if(j>m-1)
                                               /*האם היתה התאמה מלאה*/
   printf("%d\n",k);
}
```

נתבונן על קלט המייצג את המקרה הגרוע ביותר.

נניח : $\Sigma=\{0,1\}$

$$T=\underbrace{11111\ldots1}_{n\ \text{פעמים}}=1^n$$

$$P=\underbrace{11\ldots.1}_{m\ \text{פעמים}}=1^m$$

וברור $m\leq n$

בהינתן קלט כזה, עבור כל ערך של k, לולאת while תתבצע בכל מקרה m פעמים.
לכן באלגוריתם המשופר אנו לא מקטינים את מספר הצעדים במקרה הגרוע ביותר
ולכן זמן הריצה של האלגוריתם המשופר נשאר $0((n-m+1)m)$.

<u>הערה</u> : כאשר $m=0(n)$ אז סיבוכיות זמן הריצה של אלגוריתם א' (וגם של הגירסה המשופרת) היא $0(n^2)$.
נראה בהמשך אלגוריתמים אחרים הפותרים את הבעיה הנתונה בזמן שהוא יותר קטן מ- $0((n-m+1)m)$ ולכן אלגוריתם א' אינו אופטימלי עבור בעית התאמת מחרוזות.

<u>הערות</u>:

א. ניתן לממש את האלגוריתם המשופר, ביחס לאלגוריתם א', גם ללא שימוש במשתנה עזר – משתנה בוליאני (finis) כדלקמן:

```
n=length(T);
m=length(P);
for (k=0;k<=n-m;k++)
{i=0;
while((i<m)&&(P[i]==T[k+j])) i++;
if(i==length(P)) printf("%d\n",i);
}
```

ב. אם כל התוים של התבנית P שונים זה מזה, אז ניתן לפתור את בעיית התאמת מחרוזות בזמן לינארי – 0(n). (מדוע? חשוב היטב!)