התאמת

מחרוזות

תווים

- מציאת כל המופיעים של תבנית בטקסט היא בעיה החוזרת וצצה תדיר בתכניות עיבוד התמלילים.
- בד"כ, הטקסט הוא מסמך שיש לערוך, והתבנית היא מילה מסוימת שהמשתמש מחפש.

- אלגוריתמים יעילים לפתרון בעיה זו יכולים לשפר במידה ניכרת את מהירות התגובה של מעבד התמלילים.
- אלגוריתמים להתאמת מחרוזות משמשים גם למשל, לחיפוש אחר תבניות מסוימות ברצפים של DNA.

- להלן הניסוח הפורמלי של בעיית התאמת המחרוזות:
- מניחים כי הטקסט הוא מערך [T [1.. n באורך n
- והתבנית היא מערך [P [1.. m באורך m כאשר $n \leq m$.
- עוד מניחים שאיברי P ו- T הם תווים הלקוחים מאלפבית סופי Σ.

לדוגמה , האלפבית עשוי להיות

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

מערכי התווים P ו- T נקראים

לעתים קרובות מחרוזות (strings)

של תווים.

AFEKA המכללה האקדמית להנדסה בתל־אביב TEL-AVIV ACADEMIC COLLEGE OF ENGINEERING אפקה

נאמר שהתבנית P מופיעה עם היסט s (occurs with shift s) בטקסט T, או, באופן שקול, שהתבנית P מופיעה החל מהמיקום s+1 (p occurs beginning at position s+1) בטקסט T אם

$0 \le s \le n-m$ ו-

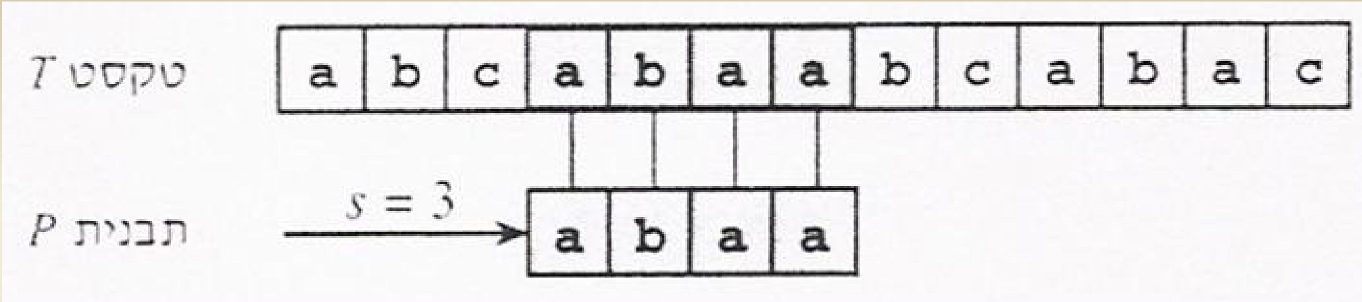
T[s+1..s+m] = P[1..m]

AFEKA אפקה המכללה האקדמית להנדסה בתל-אביב TEL-AVIV ACADEMIC COLLEGE OF ENGINEERING

(כלומר, אם $P[J] = T[s+j]$ עבור $1 \leq j \leq m$).

אם P מופיעה ב-T עם היסט s, נאמר ש-s הוא היסט תקף (invalid shift).

בעיית התאמת המחרוזות היא הבעיה של **מציאת כל ההיסטים התקפים שבהם מופיעה תבנית נתונה P בטקסט נתון T.**
איור 34.1 מדגים הגדרות אלה.



איור 34.1 בעיית התאמת המחרוזות. המטרה היא למצוא את המופיעים של התבנית P=abaa בטקסט T=abcabaabcabac.

- התבנית מופיעה בטקסט פעם אחת בלבד, עם היסט של s=3.
- אנו אומרים שההיסט s = 3 הוא היסט תקף.
- באיור, כל תו בתבנית מחובר בקו אנכי לתו המתאים לו בטקסט, וכל התווים התואמים מוצללים.

פרק זה מאורגן באופן הבא:

- בהתחלה נסקור אלגוריתם הנאיבי להתאמת מחרוזות, שרץ במקרה הגרוע בזמן $O((n-m+1)\ m)$.

לאחר מכן, נתאר אלגוריתם להתאמת מחרוזות המתחיל בכך שהוא בונה אוטומט סופי המיועד במיוחד לחיפוש המופעים של תבנית נתונה P בטקסט.

• אלגוריתם דומה, אך חכם בהרבה, הוא אלגוריתם קנות' מוריס פראט (KNUTH-MORRIS-PRATT או KMP), שיוצג .

אלגוריתם KMP רץ בזמן $O(n+m)$.

סימון ומינוח

נסמן ב- Σ^* (קרי: "סיגמה כוכב") את קבוצת כל המחרוזות בעלות אורך סופי שאפשר ליצור מתווים הלקוחים מן האלפבית Σ.

בפרק זה נעסוק רק במחרוזות בעלות אורך סופי.

- המחרוזת הריקה (empty string) שאורכה 0, המסומנת ב- ε ,

  שייכת גם היא ל- Σ^* .
- אורכה של מחרוזת x מסומן על ידי |x|.

שרשור (concatenation) של שתי מחרוזות x ו- y, המסומן על ידי xy , הוא מחרוזת באורך |x|+|y| המורכבת מתווי x ואחריהם תווי y.

אנו אומרים שמחרוזת w היא רישא (prefix) של מחרוזת x, ומסמנים זאת על ידי $w \prec x$, אם קיימת מחרוזת $y \in \Sigma^*$ שעבורה x=wy. ברור כי: $|x| \geq |w|$

באופן דומה, אנו אומרים שמחרוזת w היא סיפא (suffix) של מחרוזת x, ומסמנים זאת על ידי $\mathrm{w} \prec \mathrm{x}$, אם קיימת מחרוזת $y \in \Sigma^*$ שעבורה x= yw.

מ- $x \succ w$ נובע $|\mathrm{x}| \geq |\mathrm{w}|$.

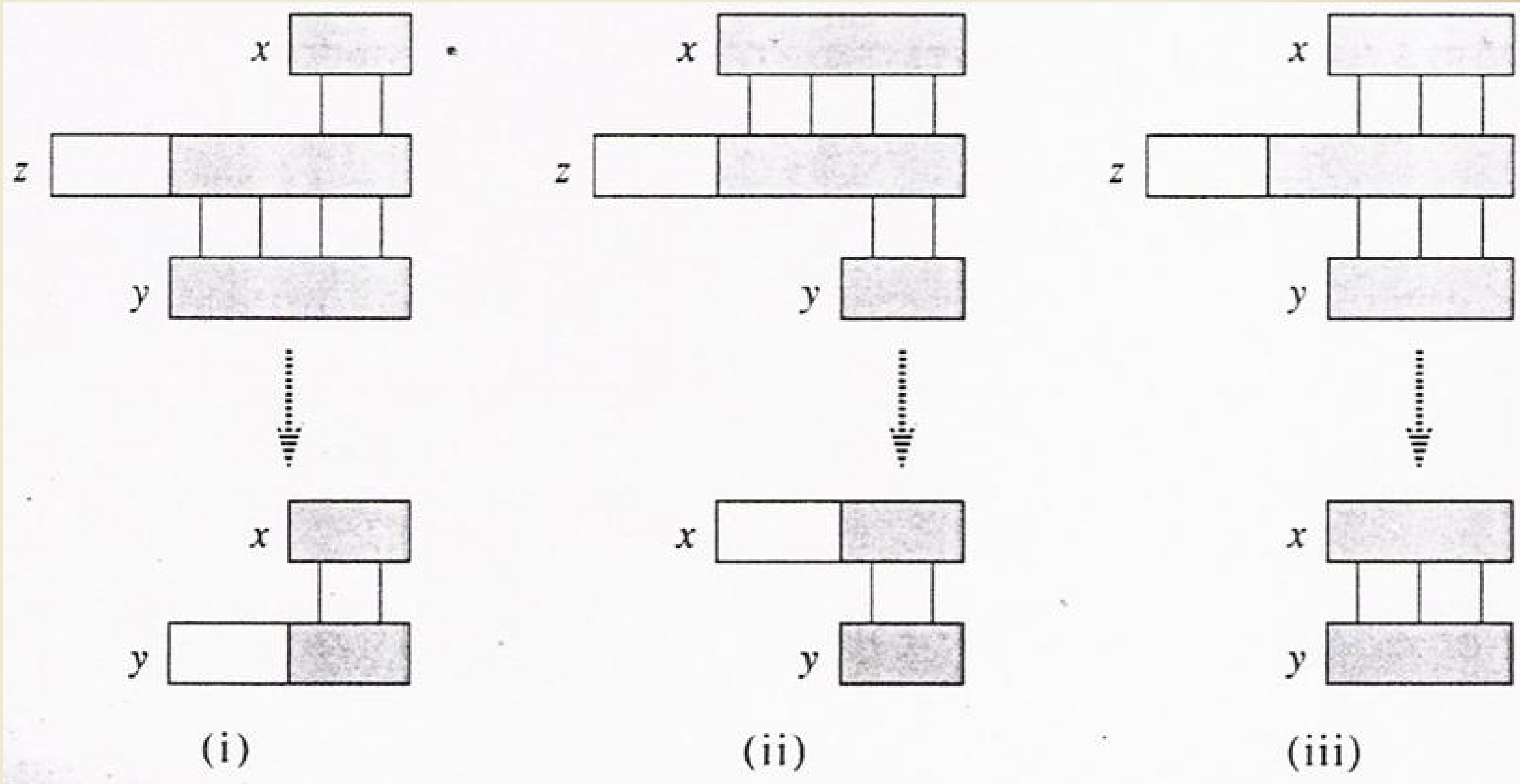
המחרוזת הריקה ε היא גם רישא וגם סיפא של כל מחרוזת.

- לדוגמה, $abcca \prec ab$ ו- $Abcca \succ cca$.
- כדאי לשים לב שעבור כל שתי המחרוזות x ו -y ותו כלשהו a, מתקיים $x \succ y$ אם ורק אם $xa \succ ya$.
- עוד נשים לב כי $\prec$ ו- $\succ$ הם יחסים טרנזיטיבים. הלמה שלהלן תועיל לנו בהמשך.

למה 34.1
(למת הסייפות החופפות)

נניח ש-x, y ו- z הן מחרוזות המקיימות $z \succ x$ ו- $z \succ y$. אם $|x| \le |y|$, אזי $y \succ x$. אם $|x| \ge |y|$, אזי $x \succ y$. אם $|x|=|y|$, אזי x=y.

הוכחה ראה גראפית באיור 34.2



איור 34.2 הוכחה גראפית ללמה 34.2 אנו מניחים כי $z \succ x$ ו- $z \succ y$. שלושת חלקי האיור מדגימים את שלושת המקרים המופיעים בלמה. קווים אנכים מחברים אזורים תואמים (המופיעים כשהם מוצללים) של המחרוזות.

- אם $[y] \le [x]$, אזי $y \succ x$. (ii) אם $[y] \ge [x]$, אזי $y \succ x$.
- (ii) אם $|y| \ge |x|$, אזי $x \succ y$.
- (iii) אם $|y|=|x|$, אזי $x=y$.

כדי לקצר את הסימונים, נסמן ב- P_k את הרישא בת k התווים $P(1..k)$ של התבנית $P[1..m]$.

- בסימון זה, $P_0 = \varepsilon$ ו- P[1..m] = P_m.
- באופן דומה, נסמן ב- T_k את הרישא בת k התווים של הטקסט T.
- באמצעות סימון זה, נוכל לנסח את הבעיה של התאמת מחרוזות **כבעיית מציאתם של כל ההיסטים s** בתחום $0 \le s \le n-m$ שעבורם $P \succ T_{s+m}$.

- בפסאודו קוד שלנו, נתייחס אל פעולת ההשוואה של שתי מחרוזות שוות אורך כאל פעולת יסוד.
- אם השוואת המחרוזות נעשית משמאל לימין ונפסקת ברגע שמתגלה חוסר התאמה, נניח כי הזמן לביצוע בדיקה כזאת היא פונקציה ליניארית של מספר התווים התואמים שנמצאו.

ביתר דיוק, אנו מניחים שהבדיקה "x=y" מתבצעת בזמן $O(t+1)$, כאשר t הוא אורכה של המחרוזת הארוכה ביותר z המקיימת $x \prec z$ ו- $y \prec z$.

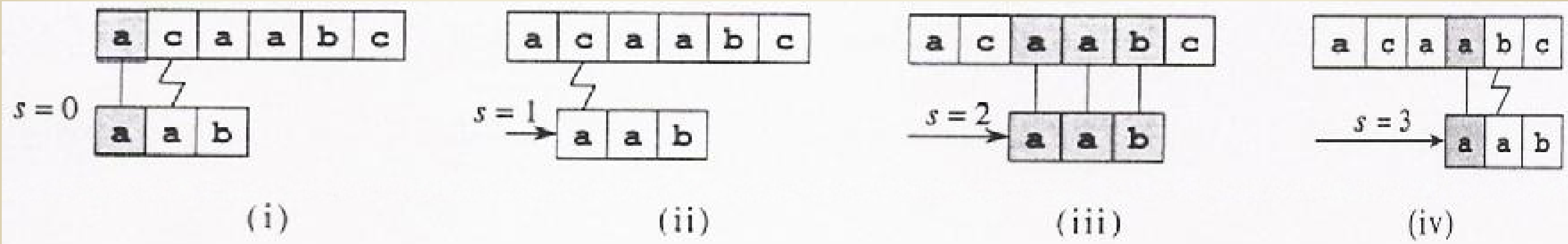
34.1 האלגוריתם הנאיבי להתאמת מחרוזות

האלגוריתם הנאיבי מוצא את כל ההיסטים התקפים באמצעות לולאה הבודקת אם התנאי $P[1..m]=T[s+1..s+m]$ מתקיים עבור כל אחד מ- $(n-m+1)$ הערכים האפשריים

```
Naive-String-Matcher (T, P)
n ← Length [T]
m ← Length [P]
for s ← 0 to n-m
    do if P [1..m] = T [ s+1..s+m]
        then print "pattern occurs with shift" s
```

- הבדיקה בשורה 4 קובעת אם ההיסט הנוכחי תקף או לא.
- בדיקה זו כרוכה למעשה בלולאה מובלעת המשווה בין תווים הנמצאים במיקומים מתאימים:
- עד שמתגלה חוסר התאמה **או** עד שהיא מוצאת כי בכל המיקומים נמצאים תווים שווים.

שורה 5 מדפיסה כל היסט תקף s.



איור 34.3 פעולתו של האלגוריתם הנאיבי להתאמת מחרוזות על התבנית P=aab והטקסט T=acaabc ניתן לדמות את פעולתו של האלגוריתם להחלקת סרגל המכיל את התבנית P לאורך הטקסט.

- בחלקים (i) – (iv) באיור מופיעים ארבעת ההיסטים העוקבים שבודק האלגוריתם הנאיבי.
- בכל חלק, קווים אנכיים מחברים אזורים מתאימים שנמצאו בהם תווים שווים (מוצללים באיור),
- וקו זיגזג מחבר את התווים הראשונים שנמצאו שונים זה מזה, אם יש כאלה.

במקרה זה נמצא בטקסט מופע אחד של התבנית, בהיסט s=2, והוא מוצג בחלק (iii).

- השגרה NAIVE-STRING-MATCHER רצה במקרה הגרוע בזמן $\theta((n-m+1)m)$.
- לדוגמה, נתבונן במחרוזת הטקסט a^n (מחרוזת של n תווי a) ובתבנית a^m.
- עבור כל אחד מ- (n-m+1) הערכים האפשריים של ההיסט s, הלולאה המובלעת בשורה 4 המשווה תווים במיקומים מתאימים חייבת להתבצע m פעמים עד שהיא מוודאת שההיסט אכן תקף.

- זמן הריצה במקרה הגרוע הוא **אפוא** $\theta((\text{n-m+1})\text{m})$, **השווה ל-** $\theta(n^2)$ **אם** .m=[n/2]
- כפי שנראה בהמשך, NAIVE-STRING-MATCHER אינה שגרה אופטימאלית עבור בעיה זו. ואכן, בפרק זה נראה אלגוריתם שזמן הריצה שלו במקרה הגרוע הוא O(n+m).

האלגוריתם הנאיבי להתאמת מחרוזות אינו יעיל משום שהוא מתעלם לחלוטין מן המידע על הטקסט שנרכש עבור ערך אחד של s, בעת שהוא בוחן ערכי s אחרים.

- אבל מידע כזה יכול להיות בעל ערך רב.
- לדוגמה, אם P=aaab ואנו מוציאים כי s=o הוא היסט תקף, אזי אף אחד מן ההיסטים 1,2 או 3 אינו יכול להיות תקף, שכן b = T[4]. בסעיפים הבאים נבחן כמה דרכים לשימוש יעיל בסוג זה של מידע.

תרגיל למחשבה

נניח שכל התווים בתבנית P שונים זה מזה . הראה כיצד ניתן להחיש את פעולתה של NAIVE-STRING-MATCHER כך שתרוץ בזמן O(n) על טקסט T בן n תווים.

34.3 התאמת מחרוזות באמצעות אוטומטים סופיים

אלגוריתמים רבים להתאמת מחרוזות בונים אוטומט סופי הסורקת את מחרוזת הטקסט T כדי לגלות בה את כל המופעים של התבנית P.
בסעיף זה נציג שיטה לבניית אוטומט כזה.

אוטומטים כאלה להתאמת מחרוזות הם יעילים מאוד: הם בוחנים כל תו בטקסט בדיוק פעם, בזמן קבוע לכל תו. הזמן הדרוש- לאחר בניית האוטומט- הוא אפוא $\theta(n)$.

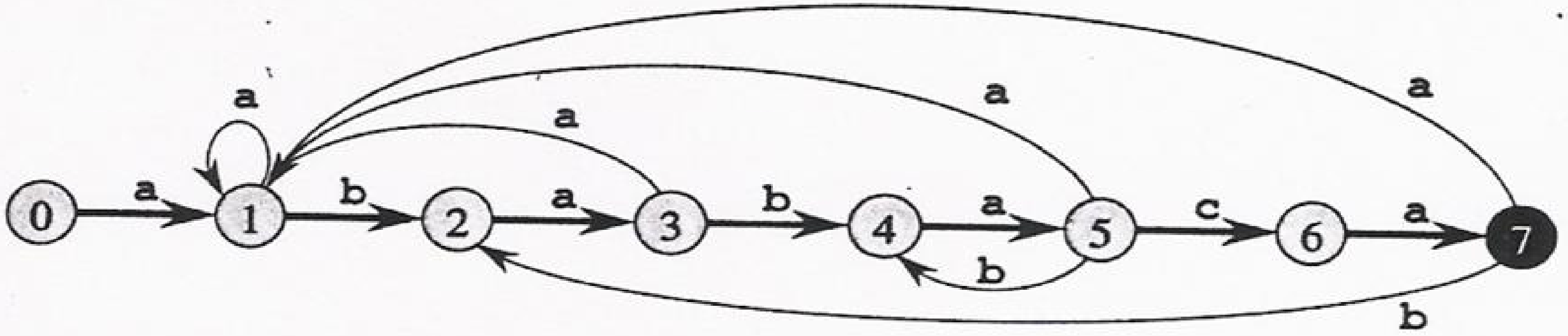
אולם , הזמן לצורך בניית האוטומט עשוי להיות ארוך אם Σ גדול.

אוטומטים להתאמת מחרוזות

- ניתן לבנות אוטומט להתאמת מחרוזות עבור כל תבנית P;
- את האוטומט יש לבנות עפ"י התבנית בעיבוד מקדים לפני שניתן להשתמש בו לחיפוש התבנית בטקסט.

באיור מודגמת בנייתו של אוטומט סופי עבור התבנית P=ababaca.

מכאן ואילך נניח ש- P היא מחרוזת נתונה בתבנית קבועה; לשם קיצור, לא נציין את התלות ב- P בסימונים שלנו.



(i)

מצב	קלט a	b	c	P
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

(ii)

i	—	1	2	3	4	5	6	7	8	9	10	11
$T[i]$	—	a	b	a	b	a	b	a	c	a	b	a
מצב $\phi(T_i)$	0	1	2	3	4	5	4	5	6	7	2	3

(iii)

- (iii) פעולתו של האוטומט על הטקסט T=abababacaba.
- מתחת לכל תו T[i] של הטקסט מופיע המצב $\phi(T_I)$ שבו נמצא האוטומט לאחר עיבוד הרישא T_I.
- האוטומט מוצא מופע אחד של התבנית, המסתיים במיקום 9.

- נגדיר תחילה פונקצית עזר σ, הנקראת **פונקצית הסיפא** (suffix function) המתאימה ל-P.
- הפונקציה σ היא מיפוי מ- $\sum^*$ ל- { 0 ,1,..., m } כך ש- $\sigma(x)$ הוא אורכה של הרישא הארוכה ביותר של P שהיא סיפא של x:

$$\sigma(x) = \max\{k : p_k \succ x\}$$

- פונקצית הסיפא σ מוגדרת היטב, שכן המחרוזת הריקה $p_0 = \varepsilon$ היא סיפא של כל מחרוזת.
- לדוגמא, עבור התבנית $p = ab$, נקבל $\sigma(\varepsilon) = 0$, $\sigma(ccaca) = 1$ ו- $\sigma(ccab) = 2$.
- עבור תבנית P באורך m מתקיים $\sigma(x) = m$ אם ורק אם $p \succ x$.

- מהגדרתה של פונקצית הסיפא נובע שאם $x \succ y$, אזי $\sigma(x) \leq \sigma(y)$.

את האוטומט להתאמת מחרוזות המתאים לתבנית נתונה $p[1..m]$ אנו מגדירים באופן הבא:

- קבוצת המצבים Q היא הקבוצה [0,1,…,m]. המצב ההתחלתי q_0 הוא המצב 0, והמצב m הוא המצב המקבל היחיד.

- פונקצית המעברים δ מוגדרת ע"י המשוואה שלהלן. עבור כל מצב q וכל תו a:

$$\delta(q,a) = \sigma(p_q a) \quad (34.3).$$

להגדרה $\delta(q,a)=\sigma(p_q a)$ ניתן להציע את ההסבר האינטואיטיבי שלהלן.

המכונה משמרת בעת פעולתה את האינוואריאנטה:

$$\phi(T_i)=\sigma(T_i) \quad (34.4)$$

בניסוח מילולי,

פירוש הדבר שאחרי סריקת i התווים הראשונים של מחרוזת הטקסט T, המכונה נמצאת במצב $\phi(T_i) = q$, כאשר $q = \sigma(T_i)$ הוא אורכה של הסיפא הארוכה ביותר של T_i שהיא גם הרישא של התבנית P.

- אם התו הבא שנסרק הוא כי $T[i+1]=a$, אזי המכונה צריכה לעבור למצב $\sigma(T_{i+1})=\sigma(T_i a)$. הוכחת המשפט מראה כי $\sigma(T_i a)=\sigma(p_q a)$: דהינו, כדי לחשב את אורכה של הסיפא הארוכה ביותר של $T_i a$ שהיא גם רישא של P, נוכל לחשב את הסיפא הארוכה ביותר של $p_q a$ שהיא גם רישא של P.

בכל מצב, המכונה צריכה לדעת רק את אורכה של הרישא הארוכה ביותר של P שהיא סיפא של המחרוזת שנקראה עד כה. לכן, ההצבה $\delta(q,a) = \sigma(p_q a)$ משמרת את האינוואריאנטה הרצויה (34.3). טיעון בלתי פורמלי זה ינוסח במדויק בהמשך.

לדוגמה, באוטומט להתאמת מחרוזות שבאיור 34.6 -***בשקופית 40*** -

מקבלים $\delta(5,b)=4$.

דבר זה נובע מן העובדה שאם האוטומט קורא תו b במצב q=5, אזי $p_q b = ababab$ והרישא הארוכה ביותר של P שהיא גם סיפא של ababab היא

$p_4 = abab$.

כדי להבהיר את אופן פעולתו של אוטומט להתאמת מחרוזות, נציג עתה תכנית פשוטה ויעילה המדמה את התנהגותו של אוטומט כזה (המיוצג על ידי פונקציית המעברים שלו δ במציאת המופעים של תבנית P באורך m בטקסט קלט $T[1..n]$. כמו בכל אוטומט להתאמת מחרוזות עבור תבנית באורך m, קבוצת המצבים Q היא $\{0,1,\ldots,m\}$, המצב ההתחלתי הוא 0, והמצב המקבל היחיד הוא המצב m.

FINITE-AUTOMATON-MATCHER (T, δ, m)

$n \leftarrow length[T]$

$q \leftarrow 0$

for $i \leftarrow 1$ to n

  do $q \leftarrow \delta(q, T[i])$

    if $q = m$

      then $s \leftarrow i - m$

        print "Pattern occurs with shift" s

- מבנה הלולאה הפשוט של FINITE-AUTOMATON-MATCHER נובע שזמן הריצה שלה על מחרוזת טקסט באורך n הוא $O(n)$.
- אולם, זמן ריצה זה אינו כולל את הזמן הנדרש לחישוב פונקצית המעברים δ.

נתבונן בפעולתו של האוטומט על טקסט קלט T[1..n].

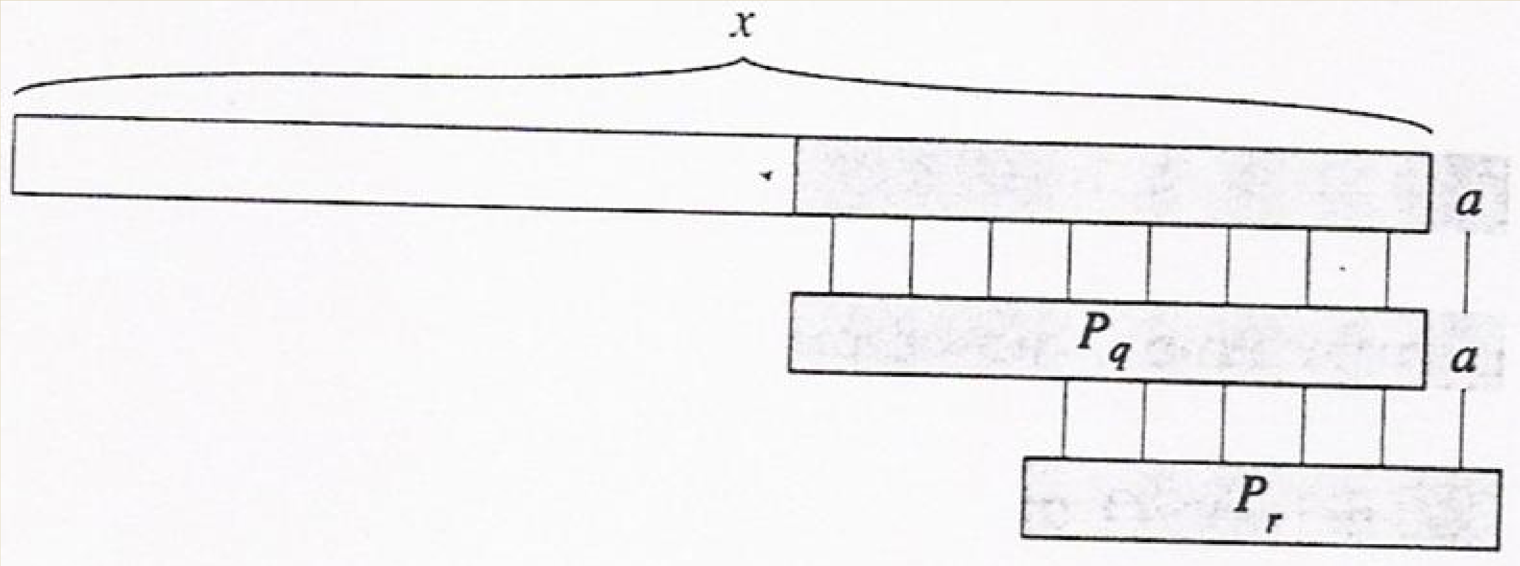
נוכיח שלאחר סריקת התו T[i] האוטומט נמצא במצב $\sigma(T_i)$.

מאחר ש- $\sigma(T_i)$ אם ורק אם $P \succ T_i$, הרי שהמכונה תימצא במצב m אם ורק אם התבנית P נסרקה זה עתה.

למה 34.3 (למת הרקורסיה של פונקצית הסיפא)

לכל מחרוזת x ותו a, אם $q = \sigma(x)$, אזי

$\sigma(xa) = \sigma(P_q a)$.



איור 34.8 הדגמת ההוכחה של למה 34.3. האיור מראה כי $r=\sigma(P_q a)$, כאשר $q=\sigma(x)$ ו- $r=\sigma(xa)$.

- כאן לשתול את הקוד של חישוב פונקציית הכישלון ואת בניית האוטומט שבקובץ word

שגרה זו מחשבת את $\delta(q,a)$ באופן ישיר, על-פי הגדרתה של δ. הלולאות המקוננות המתחילות בשורות 2 ו- 3 בוחנות את כל המצבים q והתווים a, ושורות 4-7 מציבות ב- $\delta(q,a)$ את ה- k הגדול ביותר שעבורו $P_k \succ P_q a$.

AFEKA הסכללה האקדמית להנדסה בתל-אביב TEL-AVIV ACADEMIC COLLEGE OF ENGINEERING אפקה

זמן הריצה של COMPUTE-TRANSITION-FUNCTION הוא

$O(m^3|\Sigma|)$, משום שהלולאות החיצוניות תורמות לזמן הריצה גורם של $m|\Sigma|$, לולאת ה- **repeat** הפנימית יכולה להתבצע $m+1$ פעמים לכל היותר, והבדיקה $P_k \succ P_q a$ בשורה 6 כורכה בהשוואה של m תווים לכל היותר.

- קיימות שגרות מהירות בהרבה;
- את הזמן הדרוש לחישוב δ בהינתן P ניתן לשפר ל- $O(m|\Sigma|)$ על-ידי חישוב מתוחכם של מידע אודות התבנית P וניצולו .
- כאשר משתמשים באורך m בטקסט באורך n מעל אלפבית Σ , הוא $O(n+m|\Sigma|)$.

תרגילים
34.3-1
בנה את האוטומט להתאמת מחרוזות עבור התבנית P=aabab והדגם את פעולתו על מחרוזת הטקסט T=aaababaabaababaab.

34.3-2
שרטט תרשים מצבים של אוטומט להתאמת מחרוזות עבור התבנית ababbabbababbababbabb מעל האלפבית $\Sigma = \{a,b\}$.

34.4 אלגוריתם קנות'-מוריס-פראט

עתה נציג אלגוריתם התאמת-מחרוזות של קנות', מוריס ופראט (Knuth-Morris-Pratt) שרץ בזמן ליניארי. האלגוריתם משיג זמן ריצה של $\Theta(n+m)$ על-ידי כך שהוא נמנע לחלוטין מחישוב פונקצית המעברים δ,

- ומשתמש לשם התאמת התבנית רק בפונקציית עזר $\pi[1..m]$ המחושבת מראש מן התבנית בזמן $O(m)$.
- המערך π מאפשר חישוב יעיל של פונקציית המעברים δ "תוך כדי ריצה" בעת הצורך.

- באופן כללי, עבור כל מצב q=0,1,..,m וכל תו $a \in \Sigma$, הערך $\pi[q]$ מכיל את המידע הדרוש לחישוב $\delta(q,a)$ שאינו תלוי ב-a.

(הערה זו תובהר מיד) מאחר שהמערך π מכיל רק m תאים, ואילו ל- δ יש $O(m|\Sigma|)$ ערכים, הרי שאנו חוסכים גורם של $|\Sigma|$ בעיבוד המקדים על-ידי כך שאנו מחשבים את π ולא את δ.

פונקצית הרישא של תבנית

- פונקצית הרישא של תבנית אוצרת ידע אודות התאמת התבנית להיסטים של עצמה.
- במידע זה ניתן להשתמש כדי להימנע מ-

- בדיקה מיותרת של היסטים חסרי תועלת כפי שעושה האלגוריתם הנאיבי להתאמת מחרוזות,
- או כדי להימנע מחישוב מראש של עבור אוטומט להתאמת מחרוזות.

נתבונן בפעולתו של האלגוריתם הנאיבי להתאמת מחרוזות. באיור 34.9(i) ***שבשקופית78*** ניתן לראות היסט מסוים S של התבנית P=abababaca ביחס לטקסט T.

בדוגמא זו, q=5 מהתווים הותאמו בהצלחה, אבל התו השישי של התבנית אינו מתאים לתו המקביל בטקסט.

- המידע ש- q תווים הותאמו בהצלחה מלמד מהם q תווי טקסט האלה, וידיעתם מאפשרת להסיק מיד שהיסטים מסוימים אינם תקפים.

- בדוגמא שבאיור, ההיסט s+1 בהכרח אינו תקף, שכן במקרה זה, התו הראשון בתבנית, a, יושווה לתו שעליו ידוע כי הוא מתאים לתו השני בתבנית, שהוא b.
- לעומת זאת, עבור ההיסט s+2 המוצג בחלק (ii) של האיור, מסתדרים שלושת תווי התבנית הראשונים מול תווי טקסט שבהכרח זהים להם.

ככלל, מועיל לדעת את התשובה לשאלה הבאה:
כאשר ידוע שקיימת התאמה בין תווי התבנית $P[1..q]$ לתווי הטקסט $T[s+1..s+q]$, מהו ההיסט s' הקטן ביותר כך ש: $s<s'$ שעבורו:

$$P[1..k] = T[s'+1..s'+k] \quad (34.5)$$

כאשר $s'+k = s+q$?

היסט כזה 's הוא ההיסט הראשון הגדול מ- s שאינו בהכרח לא תקף על פי ידיעתנו את $T[s+1..s+q]$.

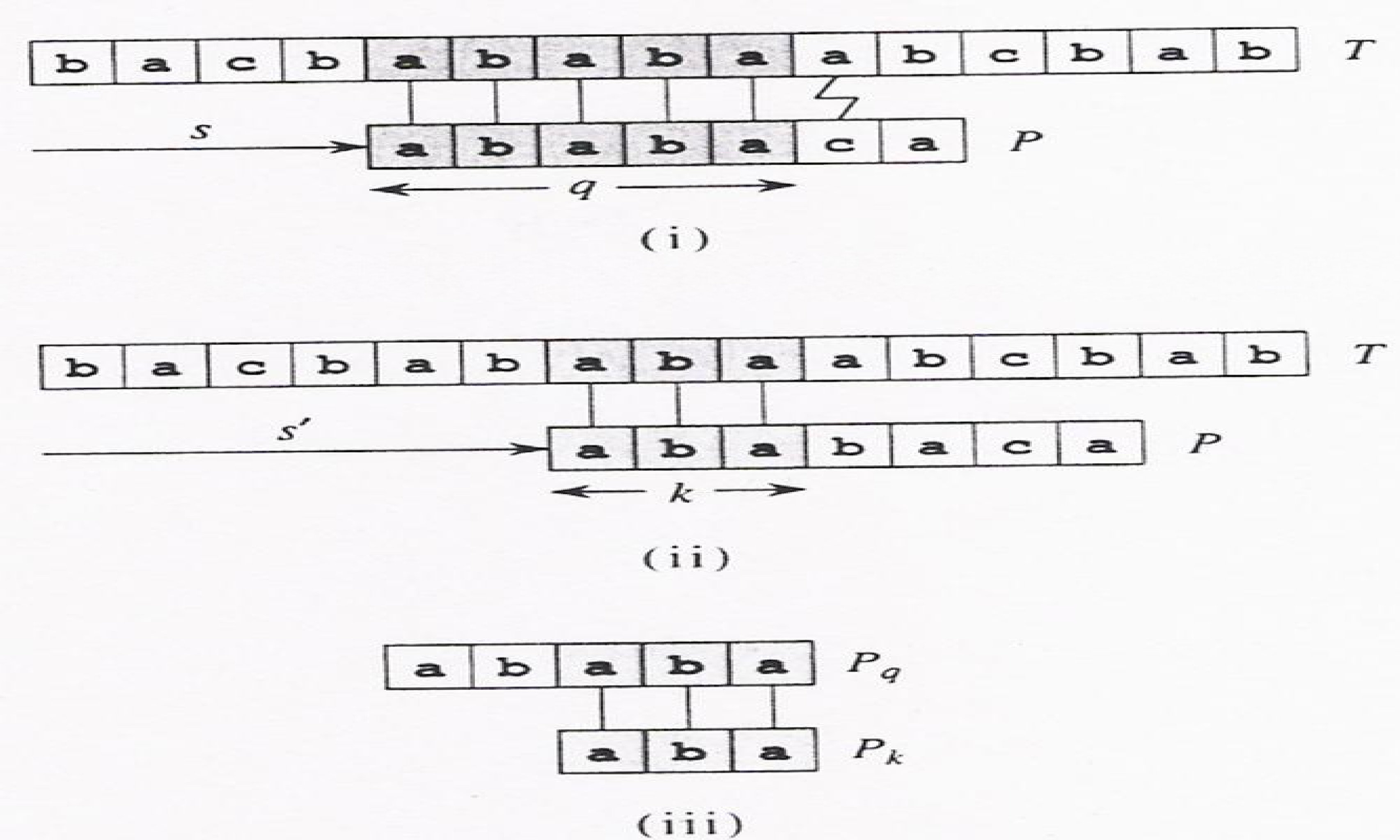
במקרה הטוב ביותר, אנו מקבלים כי $s'=s+q$, וניתן לפסול מיד את ההיסטים $s+1, s+2,\ldots,s+q-1$.

- בכל מקרה, עבור ההיסט החדש 's אין צורך להשוות את k התווים הראשונים של Pעם התווים המקבילים להם ב-T, שכן על פי משוואה (34.5) מובטח לנו שהם שווים.

- את המידע הנחוץ ניתן לחשב מראש על-ידי השוואת התבנית לעצמה, כמודגם באיור 34.9(iii).
- מאחר ש- T[s'+1..s'+k] הוא חלק מהקטע הידוע של הטקסט, הוא מהווה סיפא של המחרוזת P_q.

- ניתן אפוא לפרש את משוואה (34.5) כשאלה מהו ה- $q<k$ הגדול ביותר שעבורו $P_k \succ P_q$. אזי, $s'=s+(q-k)$ הוא ההיסט הבא שעשוי להיות תקף.

- מתברר שנוח לאחסן את המספר k של תווים תואמים עבור ההיסט החדש 's במקום לאחסן, למשל, את s'-s.
- ניתן להשתמש במידע זה כדי להחיש את פעולתו של האלגוריתם הנאיבי להתאמת מחרוזות וגם את פעולתו של האוטומט הסופי להתאמת מחרוזות.



b a c b a b a b a a b c b a b
T
s
a b a b a c a
P
q
(i)
b a c b a b a b a a b c b a b
T
s′
a b a b a c a
P
k
(ii)
a b a b a
P_q
a b a
P_k
(iii)

איור 34.9 פונקצית הרישא π. (i) התבנית P=ababaca מסתדרת מול הטקסט T כך ש- q=5 התווים הראשונים תואמים. באיור, התווים התואמים מוצללים ומחוברים בקווים אנכיים. (ii) ידיעתנו את חמשת התווים התואמים די בה כדי להסיק שההיסט s+1 אינו תקף.

אבל ההיסט של s'=s+2
קונסיסטנטי עם כל הידוע לנו על
הטקסט ולכן עשוי להיות היסט
תקף. (iii) את המידע המשמש
להסקת מסקנות כאלה ניתן לחשב
מראש על ידי השוואת התבנית
לעצמה.

באיור, אנו רואים שהרישא הארוכה ביותר של P שהיא גם סיפא של P_5 היא P_3. מידע זה מחושב מראש ומאוחסן במערך π, כך ש- $\pi[5]=3$.
אם ידוע שבהיסט s נמצאה התאמת בין q תווים, אזי ההיסט הבא שעשוי להיות תקף הוא $s'=s+(q-\pi[q])$.

AFEKA אפקה המכללה האקדמית להנדסה בתל־אביב TEL-AVIV ACADEMIC COLLEGE OF ENGINEERING

את החישובים המקדימים הדרושים ננסח באופן פורמאלי כדלקמן.
בהינתן תבנית P[1..m], **פונקצית הרישא** (prefix function) עבור התבנית P היא הפונקציה $\pi:\{1,2,\ldots,m\}\rightarrow\{0,1,\ldots,m-1\}$ המקיימת:

$\pi[q]=\max\{k:k<q$ **וגם** $P_k \succ P_q\}$

AFEKA הַמִּכְלָלָה האקדמית להנדסה בתל־אביב TEL-AVIV ACADEMIC COLLEGE OF ENGINEERING אפקה

כלומר, $\pi[q]$ הוא אורכה של הרישא הארוכה ביותר של P שהיא סיפא ממש של P_q. כדוגמה נוספת, באיור 34.10(i) מופיעים כל הערכים של פונקצית הרישא π עבור התבנית ababababca.

אלגוריתם התאמת-מחרוזות של קנות'-מוריס-פראט נתון בפסאודו-קוד שלהלן כשגרה KMP-MATCHER. כפי שנראה בהמשך, המבנה של שגרה זו מבוסס במידה רבה על המבנה של השגרה FINITE-AUTOMATON-MATCHER.

השגרה KMP-MATCHER קוראת לשגרת העזר -COMPUTE PREFIX לשם חישוב π.

```
KMP-MATCHER(T, P)
    n ← length[T]
    m ← length[P]
    π ← COMPUTE-PREFIX-FUNCTION(P)
    q ← 0
    for i ← 1 to n
        do while q > 0 and P[q + 1] ≠ T[i]
                do q ← π[q]
           if P[q + 1] = T[i]
              then q ← q + 1
           if q = m
              then print "Pattern occurs with shift" i − m
                   q ← π[q]
```

```
COMPUTE-PREFIX-FUNCTION(P)
    m ← length[P]
    π[1] ← 0
    k ← 0
    for q ← 2 to m
        do while k > 0 and P[k + 1] ≠ P[q]
                do k ← π[k]
           if P[k + 1] = P[q]
              then k ← k + 1
           π[q] ← k
    return π
```

נתחיל בניתוח זמני הריצה של שתי שגרות אלה.

ניתוח זמן הריצה

באמצעות ניתוח נקבל כי זמן הריצה של COMPUTE-PREFIX-FUNCTION הוא O(m).

אלגוריתם קנות'-מוריס-פראט רץ בזמן של O(m+n) הקריאה ל-COMPUTE-PREFIX-FUNCTION, צורכת זמן של O(m) כפי שראינו זה עתה, וניתוח לשיעורין דומה, תוך שימוש בערכו של q כפונקצית הפוטנציאל, מראה שיתרת השגרה KMP-MATCHER מתבצעת בזמן O(n).

בהשוואה ל- -FINITE
AUTOMATON-MATCHER על
ידי שימוש ב- π במקום ב- δ קיצרנו
את זמן העיבוד המקדים של
התבנית מ- $O(m|\Sigma|)$ ל- O(m), תוך
שמירה על חסם של O(m+n) על
הזמן שעורכת ההתאמה עצמה.

34.4-5

כתוב אלגוריתם שזמן ריצתו ליניארי, אשר קובע אם טקסט T מהווה רוטציה מעגלית של מחרוזת אחרת 'T. לדוגמא, המחרוזות "arc" ו- "car" הן רוטציות מעגליות זו של זו.

סיום

התאמת

מחרוזות

תווים