מבני נתונים למחרוזות

חומר קריאה לשיעור זה

Algorithms on Strings, Trees, and Sequences, Dan Gustfield

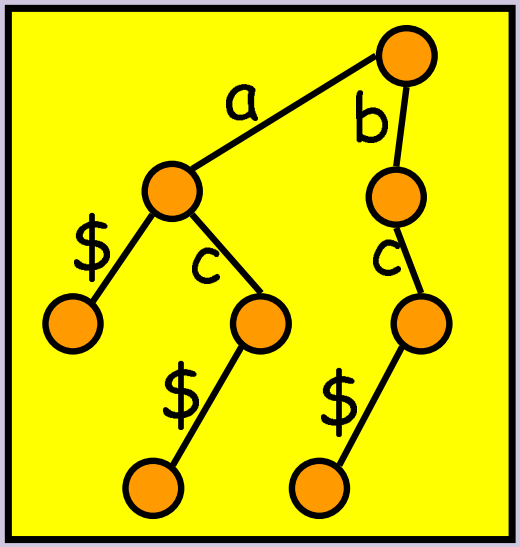
Chapter 5, 7.3, 7.4, 7.17

©Geiger & Itai, 2001

מבנה נתונים Trie

מבנה נתונים Trie מאפשר חיפוש,הכנסה, הוצאה, ומציאת מינימום (לקסיקוגרפי) של מחרוזות.

המימוש באמצעות עץ. לכל צומת פנימי יש לכל היותר מספר ילדים כגודל האלף-בית + אחד. כל קשת מסומנת בתו. התו $ (שאינו שייך ל- Σ) מסמן סיום מחרוזת. אנו נתייחס לגודל של Σ **כקבוע**.



דוגמא: trie עבור המחרוזות ac, a, bc, כאשר תו סיום-המחרוזת $ הוא הקטן ביותר לקסיקוגרפית.

הערה: בכל צומת מוחזק מערך באורך $|\Sigma|+1$ של מצביעים.

כל מחרוזת במבנה מגדירה מסלול מהשורש אל עלה. כל הפעולות מתבצעות ע"י מעקב לאורך המסלול המתאים. הזמן הנדרש לביצוען נקבע ע"י אורך המחרוזת $O(|s|)$ ולא ע"י מספרן של המחרוזות n.

©Geiger & Itai, 2001

מיון מחרוזות באיבי

קלט: מחרוזות $S_1,\ldots,S_n$ שאורכן הכולל $m = \sum_{i=1}^{n} |S_i|$.

פלט: הדפסת המחרוזות בסדר לקסיקוגרפי.

נניח לרגע (לשם פשטות) שאורך כל המחרוזות אחיד ושווה ל- m/n. השוואת שתי מחרוזות לוקחת זמן $S_i = O(m/n)$.
לפתרון באמצעות השוואות נדרשות $\Theta(n \log n)$ השוואות
ולכן סה"כ נדרש זמן $O((m/n)\, n \log n) = O(m \log n)$.

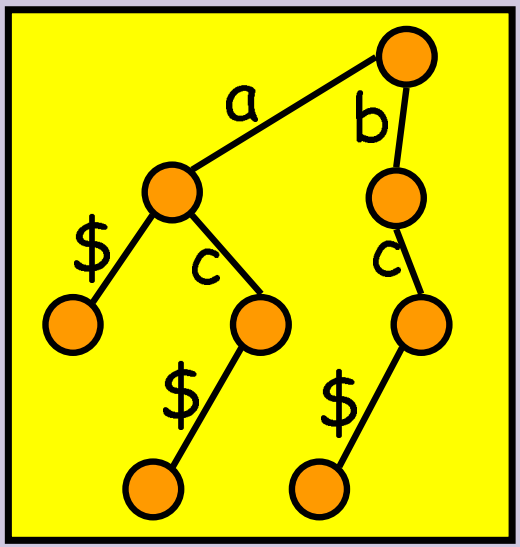
נראה כעת פתרון בזמן $O(m)$.

מיון מחרוזות באמצעות Trie

ניתן להשתמש בעובדה שלמחרוזות יש מבנה - שרשרת תווים מעל אלף-בית סופי Σ- כדי למיינן מהר יותר מאשר ע"י השוואות של מחרוזות. נשתמש במבנה נתונים Trie.

<u>האלגוריתם</u>

1. הכנס את $S_1,\ldots,S_n$ ל- trie
2. עבור על ה- trie לפי סדר preorder וכתוב לפלט את המסלול לכל עלה. (המסלול נמצא במחסנית הרקורסיה).



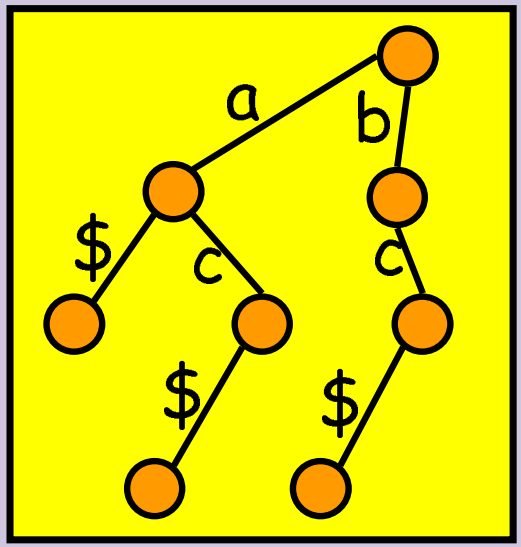
<u>דוגמא</u>: נתונות המחרוזות ac, a, bc, כאשר תו סיום-מחרוזת הוא $ (הקטן ביותר לקסיקוגרפית). המחרוזות הממוינות הן a$, ac$, bc$.

©cs,Technion

ניתוח זמנים

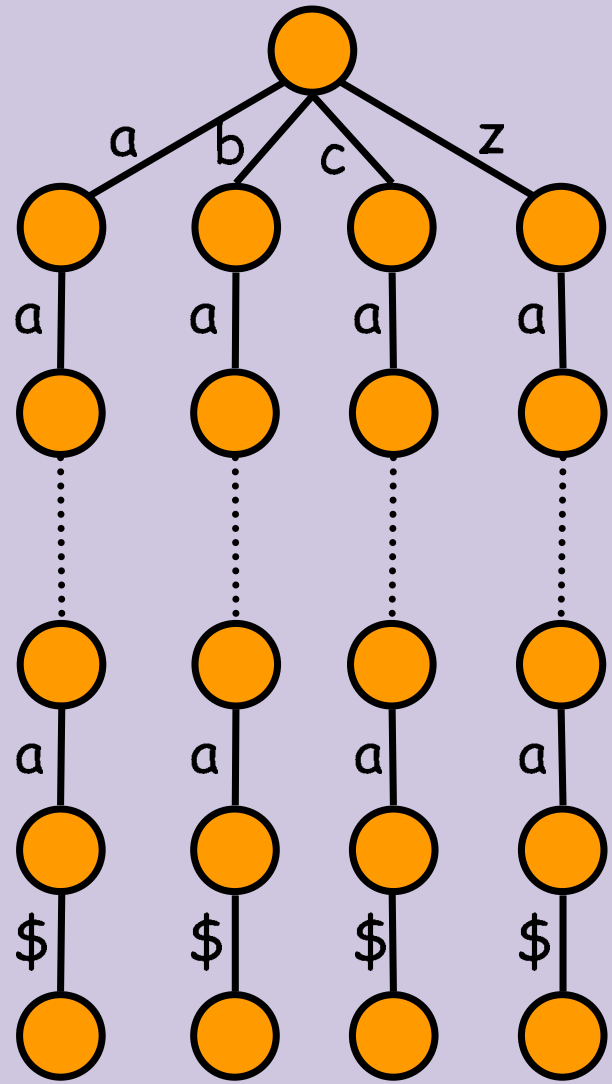
<u>זמן הריצה</u>: הכנסת s_i דורשת זמן $O(|s_i|)$. לכן זמן הכנסת כל המחרוזות הוא $O(m)=O(\Sigma_i|s_i|)$. סיור ה-preorder ב-trie דורש זמן כמספר הצמתים ומספר זה חסום ע"י $O(m)$.

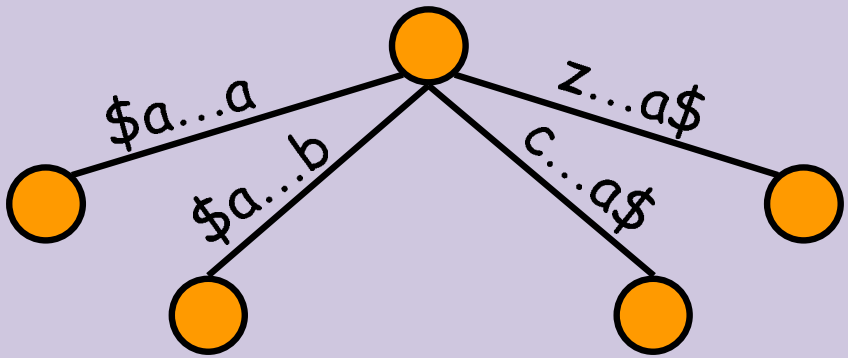
<u>דוגמא</u>:



דחיסת - Trie

נסלק מהעץ צמתים בעלי בן אחד ע"י החלפת שרשרת קשתות בקשת בודדת שתסומן בתווית המקודדת את המחרוזת המתאימה.





<u>סיבוכיות מקום:</u>

מספר העלים הוא n.

לכל צומת פנימי לפחות שני בנים.

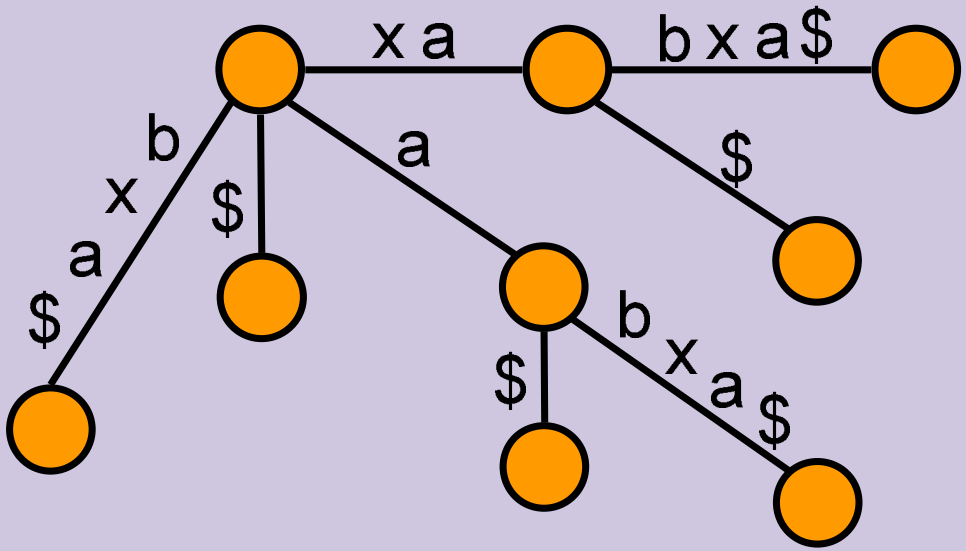
לפיכך יש לכל היותר n-1 צמתים פנימיים.

לכן סה"כ הצמתים קטן שווה 2n-1.

החסכון יהיה משמעותי יותר בהמשך כאשר תוויות הקשתות יהיו מקודדות בצורה קומפקטית.

עץ סיומות (Suffix tree)

עץ סיומות של מחרוזת S הוא Trie שבו הוכנסו כל הסיומות של המחרוזת S עם תו סיום $.



דוגמא: עץ סיומות עבור S=xabxa$

לעץ סיומות עשרות שימושים במסגרת אלגוריתמים הפועלים על מחרוזות. אנו נבחן שלושה שימושים (שימושים רבים נוספים מתוארים בספר של Gusfield):

- מציאת תת מחרוזת בתוך מחרוזת נתונה (או בתוך רשימת מחרוזות נתונה).
- מציאת תת מחרוזת ארוכה ביותר המשותפת לשתי רשימות נתונות.
- מימוש אלגוריתם לדחיסת אינפורמציה (Ziv-Lempel compression).

אלגוריתם לבניית עץ סיומות

נניח לאורך ההרצאה שאורך המחרוזת S הוא m.

אלגוריתם נאיבי לבניית עץ סיומות עבור S:

- הכנס את המחרוזות S[1...m], S[2...m], ..., S[m...m] ל-Trie
- דחוס את ה- Trie שנוצר.

ניתוח זמנים: $Time(m) = cm+c(m-1)+\ldots+1 = O(m^2)$

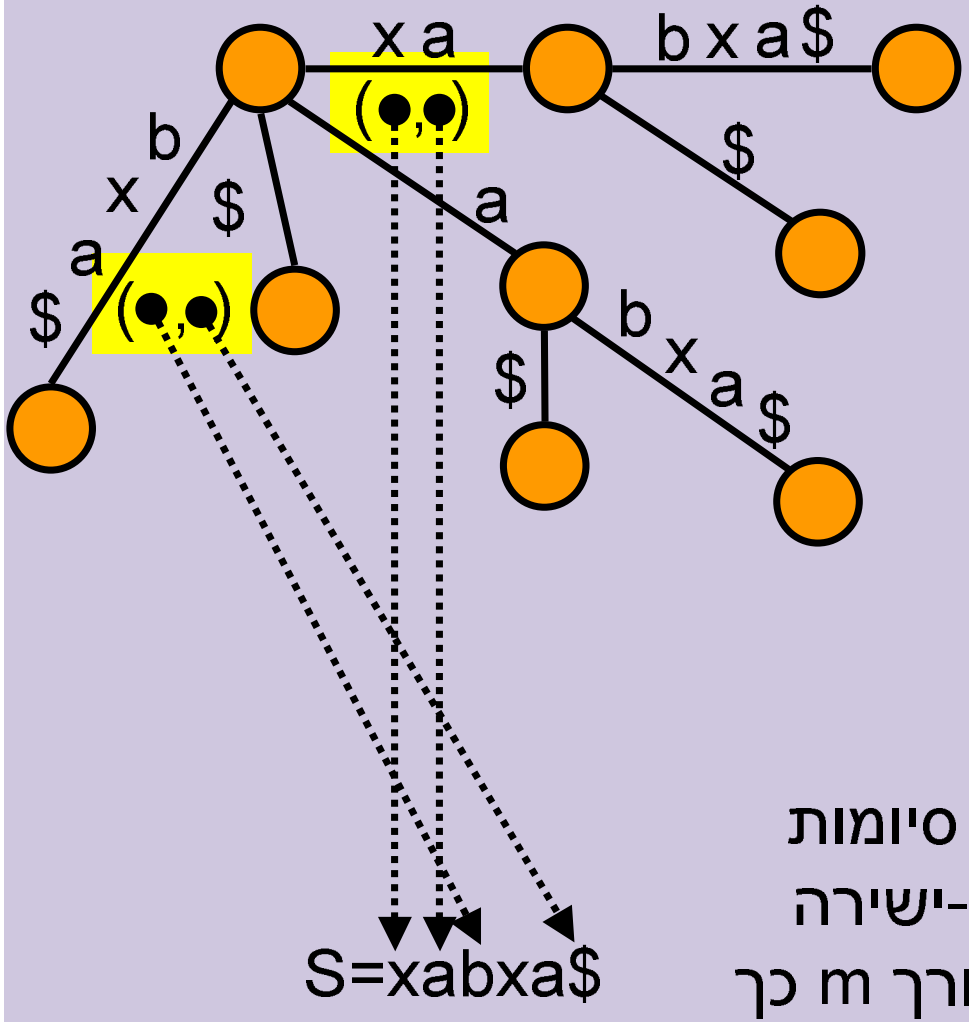
קיימים מספר אלגוריתמים מסובכים בהרבה המאפשרים לבנות עץ סיומות בזמן O(m) (כאשר גודל האלף-בית קבוע).האלגוריתם היעיל ביותר מתואר במאמר:

Esco Ukkonen. On-line construction of suffix trees. Algorithmica, 14:249-60, 1995

ובספר של Gusfield. אנו נשתמש באלגוריתם זה כ"קופסא שחורה".

©cs,Technion

חיסכון הכרחי במקום



ניזכר בעץ הסיומות עבור S=xabxa$

<u>חיסכון במקום</u>: נשים לב שהתווית של כל קשת יכולה להיות בגודל m=|S| ושחלקים ממנה מופיעים שוב ושוב.

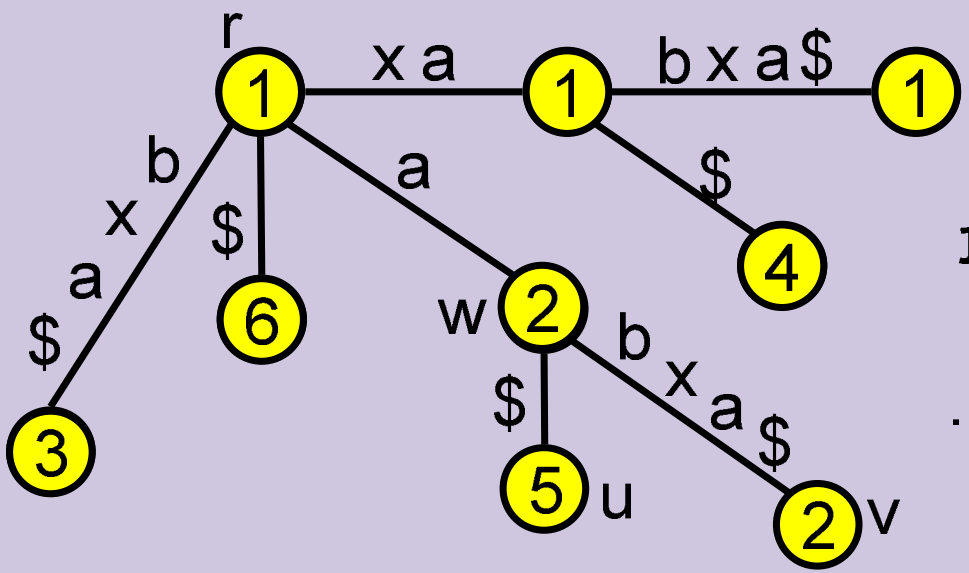
לכן נשמור העתק נפרד של המחרוזת S וכל תווית תהיה זוג מצביעים המציינים את מיקום התווית במחרוזת S.

סך המקום הנדרש הוא O(m).

למעשה <u>כל אלגוריתם ליניארי</u> לבניית עצי סיומות חייב לייצג את תוויות הקשתות בצורה לא-ישירה כיון שקיימות סדרות של מחרוזות S_m באורך m כך שסכום האורכים של תוויות הקשתות של S_m גדול מ- $\Theta(m)$ (תרגיל בית. רמז: הסתכלו והכלילו את המחרוזת (111 110 101 100 011 010 001 000)).

אינפורמציה נוספת בעץ סיומות

ניבחן שוב את עץ הסיומות עבור S=xabxa$



ע"י סיור preorder בעץ נוכל בזמן ליניארי O(m) לחשב לכל צומת z את המיקום הראשון של תת המחרוזת המיוצגת ע"י המסלול מהשורש ועד z. נסמן מיקום זה ע"י c_z.

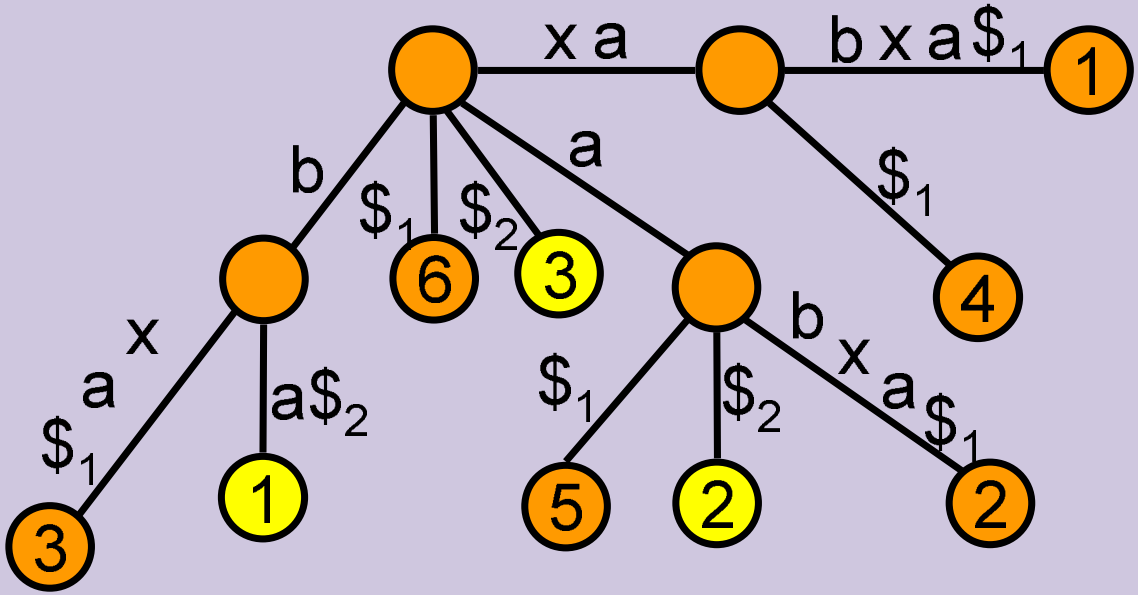
למשל המחרוזת abxa$ המיוצגת ע"י המסלול (r,v) מתחילה במקום השני ב-S והמחרוזת המיוצגת ע"י המסלול (r,u) מתחילה במקום החמישי ב- S. מספרים אלה מתקבלים ע"י חיסור מספר התווים המופיעים על המסלול לצומת z מהאורך הכללי m ועוד אחד (m=6 בדוגמא זו).

בצמתים פנימיים יירשם המינימום של ערכי הילדים. למשל 2 בצומת w, כלומר $c_w = 2$. אמנם המיקום השמאלי ביותר של המחרוזת a ב- s הוא 2.

©cs,Technion

עץ סיומות מוכלל

עץ סיומות מוכלל הוא Trie שבו הוכנסו כל הסיומות של קבוצת מחרוזות $\{S_1,..,S_n\}$ עם תו סיום שונה $\$_i$ לכל מחרוזת S_i.



דוגמא: עץ סיומות מוכלל עבור $\{S_1=xabxa\$_1,\ S_2=ba\$_2\}$.

אלגוריתם נאיבי לבניה: נכניס את כל הסיומות אחת אחת ל-Trie בודד.

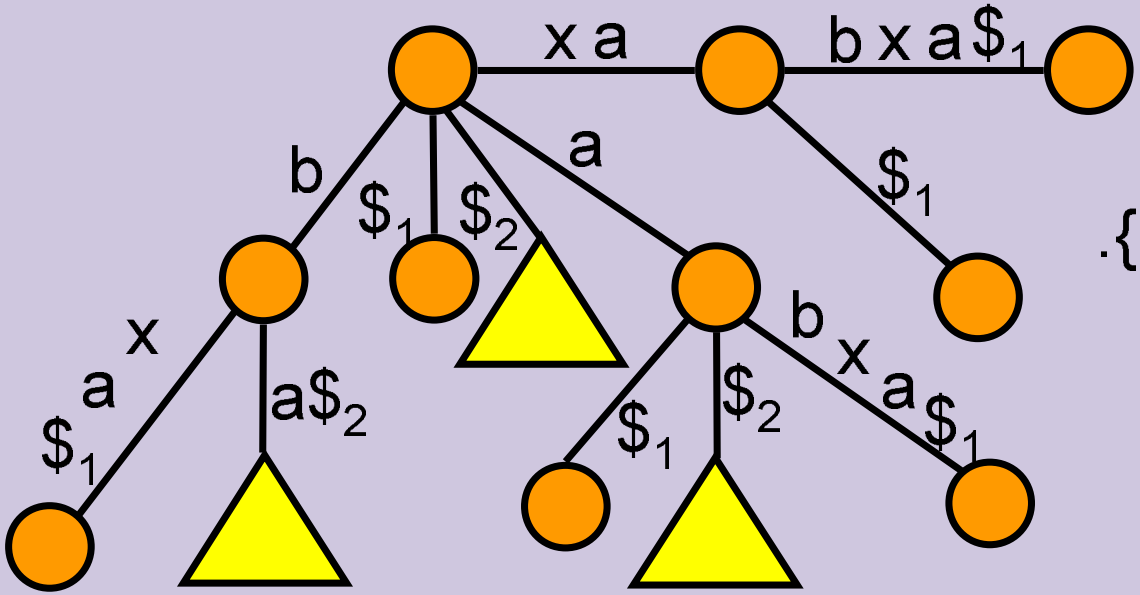
זמן הריצה כאורך סכום כל הסיומות של כל המחרוזות (במקרה הגרוע גדול הרבה מסכום אורכי המחרוזות).

בנית עץ סיומות מוכלל

אלגוריתם ליניארי: נבנה עץ סיומות עבור המחרוזת $S_1\$_1 S_2\$_2 \ldots S_n\$_n$

נקצץ את כל תתי העצים מתחת לקשתות שהתווית שלהן היא $\$_i$.

סיבוכיות זמן $O(\sum_i |s_i|)$.



דוגמא: $\{S_1=xabxa\$_1,\ S_2=ba\$_2\}$.
עץ סיומות עבור $ba\$_2xabxa\$_1$
נראה כך:

העצים המצויירים כמשולש מייצגים סיומות המכילות $\$_i$. סיומות אלה אינן שייכות לאף אחת מהמחרוזות המקוריות. לכן ניתן לגזום את המשולשים הנ"ל מהעץ.

מציאת מחרוזות קצרות בטקסט ארוך

הבעיה: נתונה מחרוזת T מאורך m הנקראת טקסט. לאחר זמן עיבוד ליניארי O(m) של הטקסט, יש להיות מוכנים לקבל מחרוזת s לא ידועה באורך n ולמצוא מופע של s בטקסט T (המופע הראשון) או לקבוע שהמחרוזת s אינה נמצאת בטקסט.

שימו לב שכל פתרון העובר על הטקסט T בזמן קבלת המחרוזת s ללא עיבוד מוקדם של T יאלץ לבצע לפחות $\Theta(m)$ פעולות.

דוגמאות לשימושים: הטקסט הוא האנציקלופדיה בריטניקה והמחרוזת s היא מילה. הטקסט הוא הגנום של אורגניזם כלשהו, כלומר מחרוזת ארוכה של האותיות {A,C,T,G}, והמחרוזת s היא סדרה של אותיות כאלה המקודדת גן אותו יש למצוא בגנום הנתון.

הערה: כמובן שקיימות וריאציות לבעיה זו כגון מציאת כל המופעים של s בטקסט הנתון, התאמה חלקית של s לטקסט, או חיפוש s בתוך אוסף טקסטים.

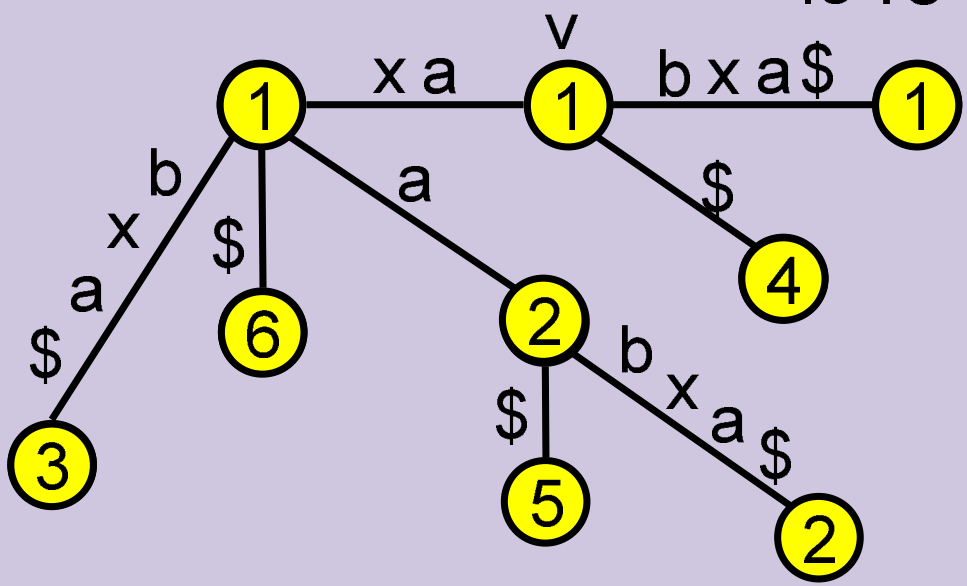
הנחה: $m \gg n$ כלומר המחרוזת המבוקשת קצרה יחסית לאורך הטקסט.

©cs,Technion

אלגוריתם למציאת מחרוזות בטקסט

בנה בזמן O(m) עץ סיומות עבור הטקסט T.
הוסף לכל צומת v בעץ הסיומות את המספר c_v.

בהינתן מחרוזת s, עקוב על המסלול מהשורש של עץ הסיומות לפי התווים שבמחרוזת s. אם נמצאה המחרוזת s, אזי מקום המחרוזת הוא c_v כאשר v הוא הצומת האחרון במסלול החיפוש של s.



<u>דוגמא</u>: ניבחן שוב את עץ הסיומות עבור T=xabxa$. המחרוזת xa נמצאת במקום הראשון (והרביעי) והמחרוזת a נמצאת במקום השני (והחמישי).

<u>הערה</u>: למציאת כל המופעים של s בטקסט, האלגוריתם מוצא את c_u לכל עלה u בתת העץ ששורשו v. זמן החיפוש הוא O(k) כאשר k הוא מספר המופעים של s בטקסט. החסם נובע מכך שמספר הצמתים בתת העץ קטן מ-2k. בדוגמא, עבור הצומת v מתקבל c_v= 1,4.

©cs,Technion

דחיסת אינפורמציה

הבעיה: טקסט מילולי (ואחר) המקודד בצורה מפורשת הוא לעיתים ארוך מהנחוץ שכן מילים וחלקי מילים חוזרים על עצמם לאורך הטקסט.

המטרה: בהינתן מחרוזת s, לייצר מחרוזת "s התופסת פחות מקום מ- s והמכילה את אותה האינפורמציה.

השימוש: העברה יעילה של קבצי אינפורמציה במדיום אלקטרוני כגון בדיסקטים, ברשתות תקשורת, וכדומה. למשל פקודות הדחיסה המקובלות winzip ו-compress במערכת Windows ובמערכת Unix .

הרעיון: נעבור על המחרוזת הנתונה s משמאל לימין, בכל פעם שעוברים על תת מחרוזת z שכבר ראינו נחליף את z עם האינדקס והאורך של המופע השמאלי ביותר של z במחרוזת s.

האלגוריתם המתבסס על רעיון זה נקרא Ziv-Lempel compression והוא מתואר במאמרים:

Ziv, Lempel, IEEE Trans on Information Theory, 23:337-43, 1977.

Ziv, Lempel, IEEE Trans on Information Theory, 24:530-368, 1978.

כמו כן מוכח במאמרים אלה שאסימפטוטית, זהו אלגוריתם דחיסה אופטימלי.

©cs,Technion

דוגמאות לדחיסה

דוגמא I:

המחרוזת הנתונה: S = x a b a c aba x ab y

המחרוזת הדחוסה: S” = x a b (2,1) c (2,3) x (2,2) y

במקרה זה אין דחיסה כלל. זו תופעה נפוצה עבור מחרוזות קצרות.

דוגמא II:

המחרוזת הנתונה:

S = a b ab abab abababab ababababababababab

המחרוזת הדחוסה: S” = a b (1,2) (1,4) (1,8) (1,16)

עבור k חזרות של ab נקבל מחרוזת דחוסה באורך $\Theta(\log k)$.

©cs,Technion

הגדרות וסימונים

הגדרה: לכל אינדקס i במחרוזת S[1..m], נגדיר את תת המחרוזת $Prior_i$ להיות הרישא (Prefix) הארוכה ביותר של S[i..m] ואשר מופיעה כתת מחרוזת בתוך S[1.. i-1].

דוגמא:

1 2 3 4 5 6 7 8 9

S = a b a x c a b a x a b y

$Prior_7$=bax

נסמן ב- L_i את אורך המחרוזת $Prior_i$.

כאשר $Prior_i$ היא מחרוזת ריקה אז L_i=0.

נסמן ב- s_i את מיקום המחרוזת $Prior_i$ כאשר L_i>0.

בדוגמא: $s_7 = 2$ L_7=|bax|=3

האלגוריתם וניתוחו

נתונה המחרוזת S[1..m].

```
for (i=1; i <=  m;;  )
  {  Compute(s_i ,L_i) ;
     if L_i > 0 {output(s_i, L_i);  i = i + L_i };
     else       {output(S[i]) ;  i = i +1 }  }
```

נקודת המפתח במימוש האלגוריתם הוא חישוב (s_i,L_i) בכל איטרציה. נניח שניתן לממש פעולה זו בזמן $O(L_i)$ (כפי שנראה), מה יהיה זמן הריצה של האלגוריתם ?

האלגוריתם קורא את המחרוזת משמאל לימין. האלגוריתם לא מחשב את (s_i,L_i) עבור אינדקס i שכבר נמצא באזור הדחוס. בכל איטרציה האלגוריתם דוחס L_i תווים וקופץ קדימה L_i תווים במחרוזת S. לפיכך סכום האורכים L_i שחושבו ע"י האלגוריתם קטן מ-m (אין חפיפות), וזמן הריצה O(m).

S = a b ab abab abababab abababababababab

L_i = 0 0 2 4 8 16 האורכים שחושבו:

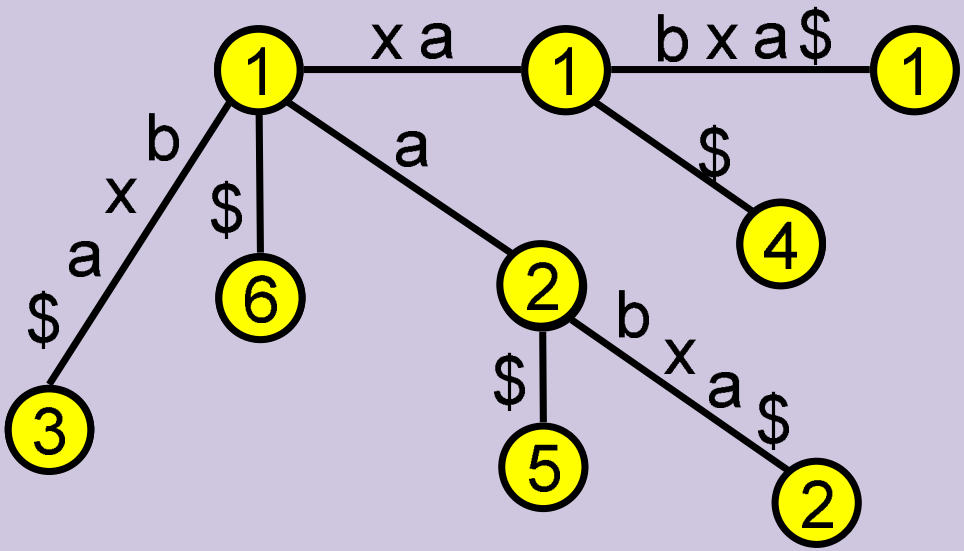
מימוש יעיל של Ziv-Lempel

בנה עץ סיומות T עבור המחרוזת S בזמן O(m).
חשב את c_v לכל צמתי T בזמן O(m).

בכל איטרציה, כאשר האלגוריתם נדרש לחשב את (s_i,L_i), צעד על המסלול מהשורש של T לפי התווים במחרוזת S[i..m] כל עוד $c_v<i$.
כאשר החיפוש נעצר, לאחר L_i תווים, המקום s_i שווה ל- c_u כאשר u הוא הצומת האחרון על מסלול הצעידה.

<u>דוגמא</u>: דחיסת S=xabxa$

S"= xab(1,2)$



<u>פענוח מחרוזת דחוסה</u>: עבור על המחרוזת "S משמאל לימין. במקרה של תו,העתק אותו ל- S, ובמקרה של זוג (s_i,L_i), שכפל את L_i התווים המתחילים במקום s_i.

©cs,Technion

מציאת תת מחרוזת ארוכה משותפת

הבעיה: מצא תת מחרוזת ארוכה ביותר המשותפת לשתי מחרוזות נתונות S_1,S_2 בזמן $O(L_1+L_2)$ כאשר L_i הוא אורך המחרוזת S_i.

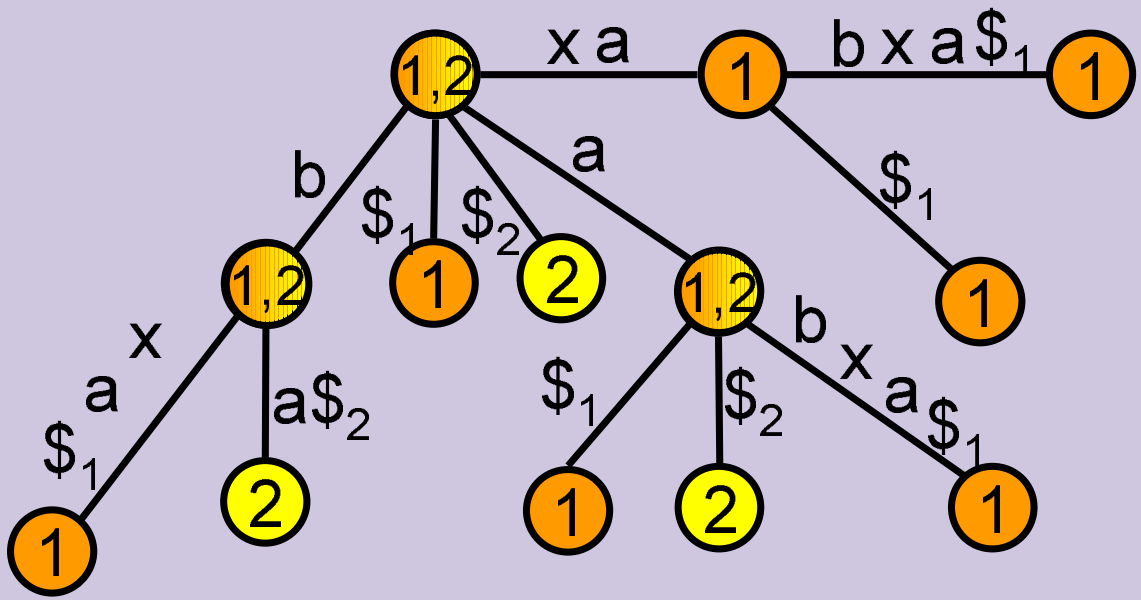
דוגמא: S_1= superiocalifornialives S_2=sealiver

פתרון לדוגמא: תת המחרוזת הארוכה ביותר המשותפת לשתי המחרוזות הנתונות היא alive.

תרגיל: מצאו אלגוריתם יעיל ככל שתוכלו ללא שימוש בעצי סיומות.

מציאת תת מחרוזת ארוכה משותפת

הרעיון: נבנה עץ סיומות מוכלל המכיל את הסיומות של שתי המחרוזות. נסמן עלה בתווית 1 אם העלה מתאים לסיומת של S_1 ובתווית 2 אם העלה מתאים לסיומת של S_2. נסמן צומת פנימי ב-1 אם כל ילדיו מסומנים ב-1, נסמנו ב-2 אם כל ילדיו מסומנים ב-2, ונסמנו ב- {1,2} אם תוויות ילדיו מכילים גם 1 וגם 2. סימון זה לוקח זמן ליניארי בגודל עץ הסיומות.



דוגמא: עץ סיומות מוכלל עבור $\{S_1=xabxa\$_1,\ S_2=ba\$_2\}$.

שורת המחץ: תת המחרוזת הארוכה ביותר המשותפת היא זאת המיוצגת ע"י המסלול הארוך ביותר מהשורש אשר סימון כל הצמתים שלו הוא {1,2}. בדוגמא,המחרוזת המשותפת היא a או b.

מציאת תת מחרוזת ארוכה משותפת ל- k מחרוזות

הבעיה: מצא תת מחרוזת ארוכה ביותר המשותפת ל-k מחרוזות נתונות $S_1,S_2,\ldots,S_k$ בזמן $O(L_1+L_2+\ldots+L_k)$ כאשר L_i הוא אורך המחרוזת S_i.

הרעיון כמו קודם: נבנה עץ סיומות מוכלל המכיל את הסיומות של k המחרוזות. נסמן את הצמתים ונמצא את המסלול הארוך ביותר מהשורש אשר מסומן בכול אורכו ע"י $\{1,\ldots,k\}$.

דוגמא לשימוש:המחרוזות הנתונות הם הגנום של מספר אורגניזמים, והמחרוזת הארוכה ביותר המשותפת להם עוזרת למציאת התאמות בן הגנומים השונים.

כמובן שבשימוש זה נצטרך להכניס מגבלות נוספות, כגון מיקום תת המחרוזת בכל גנום, התאמה חלקית לחלק מהגנומים,מחרוזות משותפות נוספות, וכו.

©cs,Technion

strings