Lecture 11 of Geiger & Itai's slide brochure
www.cs.technion.ac.il/~dang/courseDS

מבני נתונים למחרוזות

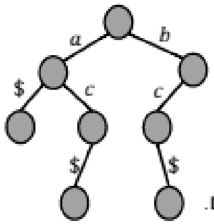
חומר קריאה לשיעור זה

Algorithms on Strings, Trees, and Sequences, Dan Gustfield
Chapter 5, 7.3, 7.4, 7.17

מילון למחרוזות - Trie

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 למקרים בהם הגודל של Σ **קבוע**.



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

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

כל מחרוזת במבנה מגדירה מסלול מהשורש אל עלה.
כל הפעולות מתבצעות ע"י מעקב לאורך המסלול המתאים.

סיבוכיות המימוש:
- הזמן הנדרש לביצוען נקבע ע"י אורך המחרוזת $O(|s|)$ **(ולא ע"י מספרן של המחרוזות n).**
- המקום: לינארי בסכום אורכי המחרוזות במבנה.

מבנה צומת בTrie

הגדרת צומת:

a	b	c	...	z	$\$$
			...		

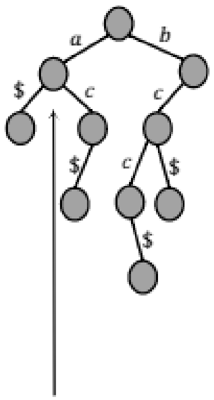
```
#define SIGMA 26;

typedef struct node {
        struct node[ SIGMA + 1] *sons ;
        } NODE ;

NODE *root ;
```


מימוש חיפוש, הכנסה והוצאה

חיפוש(s): נתחיל בשורש. $i \leftarrow 0$.
כל עוד $i \le |s|$: אם התו ה-i במחרוזת הוא התו ה-j בא"ב, נעקוב אחרי המצביע ה-j במערך. אם המצביע הזה NULL, נחזיר "לא נמצא". אחרת, $i \leftarrow i + 1$.
אם $i > |s|$, החזר "נמצא".

הכנסה(s): בדומה לחיפוש, אבל אם ניתקל במצביע NULL במהלך החיפוש, נקצה צומת חדש ונדאג שהמצביע המתאים בצומת הנוכחי יצביע אליו, ונמשיך בחיפוש בצומת החדש.

הוצאה(s): נחפש את המחרוזת s ונשמור את הצומת האחרון v בעל יותר מבחן אחד לאורך מסלול החיפוש ואת האות הבאה של s. נמחק את תת העץ המתאים של s.



הצומת v עבור
הסרת המחרוזת $ac\$$

מיון מחרוזות נאיבי

קלט: מחרוזות $S_1, \dots, S_n$ שאורכן הכולל הוא $m = \sum_{i=1}^{n} |S_i|$
פלט: הדפסת המחרוזות בסדר לקסיקוגרפי.

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

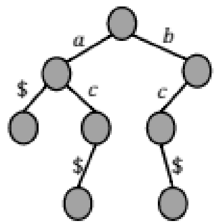
נראה כעת פתרון בזמן $O(m)$.

מיון מחרוזות באמצעות Trie

ניתן להשתמש בעובדה שלמחרוזות יש מבנה - שרשרת תווים מעל אלף-בית מאורך קבוע $\sum$ - כדי למיינן מהר יותר מאשר ע"י השוואות של מחרוזות. נשתמש ב- Trie באופן הבא:

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

דוגמא: נתונות המחרוזות ac, a, bc, כאשר תו סיום-מחרוזות הוא \$ (הקטן ביותר לקסיקוגרפית). המחרוזות הממוינות הן $a\$, ac\$, bc\$$.

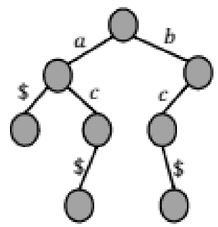


ניתוח זמנים

זמן הריצה:
- 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 הוא $O(m)$.
- ההדפסות המתבצעות במהלך הסיור דורשות זמן כסכום אורכי המחרוזות $O(m) = O(\Sigma_i |s_i|)$.

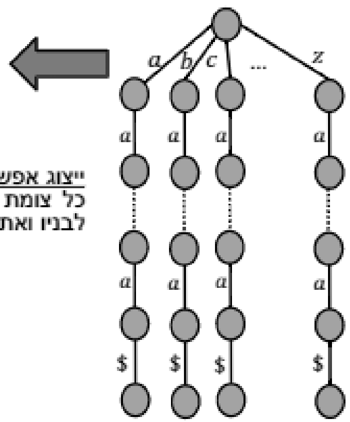
סה"כ זמן ריצת האלגוריתם: $O(m)$.

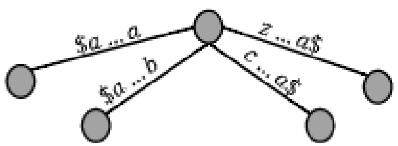
דוגמא:



דחיסת Trie

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





ייצוג אפשרי:
כל צומת יחזיק מערך בגודל $|\Sigma| + 1$ של מצביעים לבניו ואת המחרוזות המתאימות לקשתות.

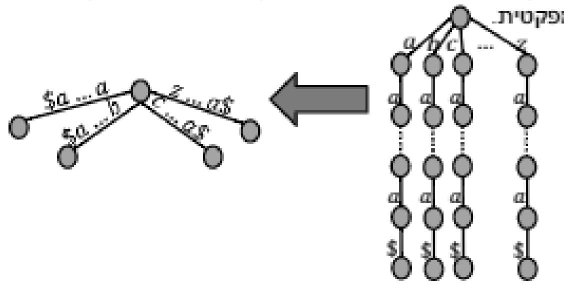
דחיסת Trie – השפעה על מספר הצמתים

יהא M מספר הצמתים בעץ ו- n מספר העלים.
עקב הדחיסה, לכל אחד מ $(M-n)$ הצמתים הפנימיים יש לפחות שני בנים.

<u>טענה</u>: לעץ בו לכל צומת פנימי לפחות 2 בנים, מספר הצמתים מקיים $2n-1 \ge M$

<u>הוכחה</u>: שיקולי ספירת קשתות. כיוון שמדובר בעץ, ולפי התכונות הנ"ל, מתקיים:
$$M-1=\Sigma_v d_{out}(v) \ge 2(M-n)$$
ומכאן מתקיים $2n-1 \ge M$.

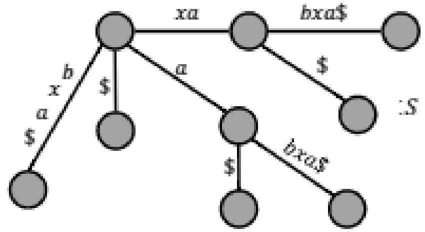
לכן לאחר דחיסת Trie מספר הצמתים לינארי במספר העלים (מספר המחרוזות בTrie). החסכון יהיה משמעותי יותר בהמשך כאשר תוויות הקשתות יהיו מקודדות בצורה קומפקטית.



עץ סיומות (Suffix Tree)

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

<u>דוגמא</u>: 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.

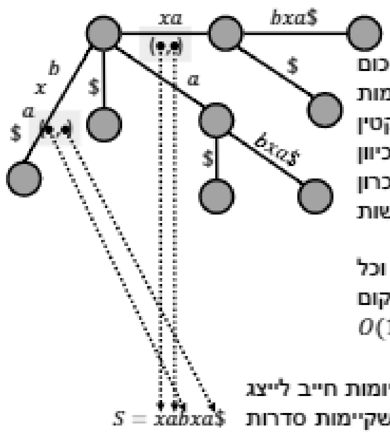
<u>אלגוריתם נאיבי לבניית עץ סיומות עבור S</u>:
- הכנס את המחרוזות $S[1 \ldots m], S[2 \ldots m], \ldots, S[m \ldots m]$ ל-Trie
- דחוס את ה-Trie שנוצר.

<u>ניתוח זמנים</u>: $Time(m) = cm + c(m-1) + \cdots + c \;=\; O(m^2)$

קיימים מספר אלגוריתמים מסובכים בהרבה המאפשרים לבנות עץ סיומות בזמן $O(m)$ (כאשר גודל האלף-בית קבוע). אלגוריתם עם זמן ריצה זה מתואר למשל במאמר:
Esco Ukkonen. "On-line construction of suffix trees." Algorithmica, 14:249-60, 1995
ובספר של Gusfield. אנו נשתמש באלגוריתם זה כ"קופסא שחורה".

חיסכון הכרחי במקום

ניזכר בעץ הסיומות עבור $S = xabxa\$$:



<u>חיסכון במקום</u>: Trie דחוס דורש מקום כסכום אורכי המחרוזות על הקשתות. עבור עץ סיומות מדובר ב- $\Theta(m^2)$ מקום. הראינו כיצד להקטין את מספר הצמתים ל- $O(m)$, אך כיוון ששומרים מחרוזות על הקשתות סך הזיכרון נשאר $\Omega(m^2)$. כיצד ניתן לשפר את דרישות המקום?

<u>פתרון</u>: נשמור העתק נפרד של המחרוזת S וכל תווית תהיה זוג מצביעים המציינים את מיקום התווית במחרוזת S. כל קשת תדרוש $O(1)$ מקום. סך המקום הנדרש הוא $O(m)$.

למעשה <u>כל אלגוריתם לינארי</u> לבניית עצי סיומות חייב לייצג את תוויות הקשתות בצורה לא-ישירה כיון שקיימות סדרות של מחרוזות באורך m כך שסכום האורכים של תוויות הקשתות של S_m גדול מ- $\Theta(m)$ ולכן זמן כתיבת עץ הסיומות ידרוש יותר מ-$\Theta(m)$ זמן (תרגיל בית).

<u>רמז</u>: הסתכלו והכלילו את המחרוזת (000 001 010 011 100 101 110 111).