Itai Katz ©1995

Trie

מבנה חיפוש re*trie*val

קלט: מחרוזות ארוכות
הפעולות: הכנסה, הוצאה, חיפוש, מינימום (לקסיקוגרפי).

פתרון סטנדרטי: עץ חיפוש (מאוזן).
מבנה צומת

```
typedef struct bin_tree_node{
   char *key;
   bin_tree_node *left, *right;
} BIN_TREE_NODE;
```

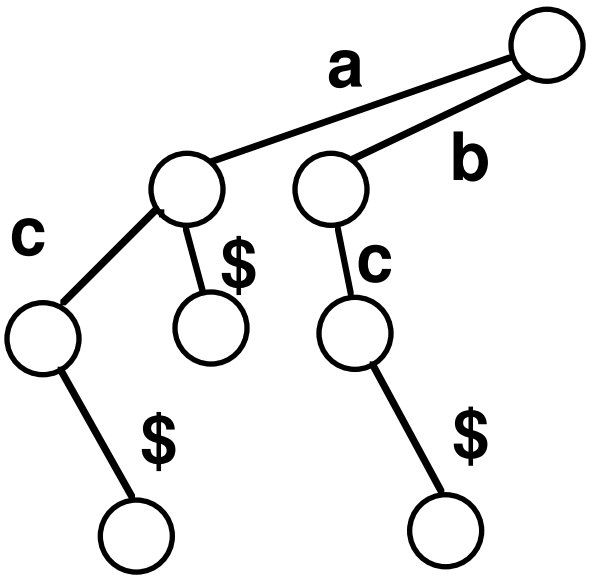
עבור קלט:
"aa...ab", "aa...ac",...,"aa...az"
(n מחרוזות באורך m)
אם נחפש את "aa...ab"
כל השוואה תיקח $\theta(m)$ זמן.
סה"כ החיפוש $\theta(m\log n)$ זמן.

trie עבור א"ב Σ

עץ מדרגה $\sigma =|\Sigma|$, (לכל צומת $\sigma \geq$ בנים).
כל קשת תסומן באות $a \in \Sigma$
(מכל צומת יש לכל היותר קשת אחת המסומנת ב-a).
נוסיף לכל מחרוזת אות $\$ \notin \Sigma$

<u>דוגמא</u>: trie עבור

"a$", "ac$", "bc$"



מבנה צומת:

```
typedef struct trie_node{

   struct trie_node *a[σ+1];
} TRIE_NODE;
```

כל מחרוזת במבנה מתאימה למסלול מהשורש לעלה של ה-trie.

אלגוריתם החיפוש למחרוזת s:

1. עקוב אחרי המסלול בעץ שמסומן ב-s
2. אם יש מסלול כזה:
   (כיון שהאות האחרונה היא $)
   הגענו לעלה והחיפוש הצליח;
3. אחרת: (אין מסלול כזה) -- החיפוש נכשל.

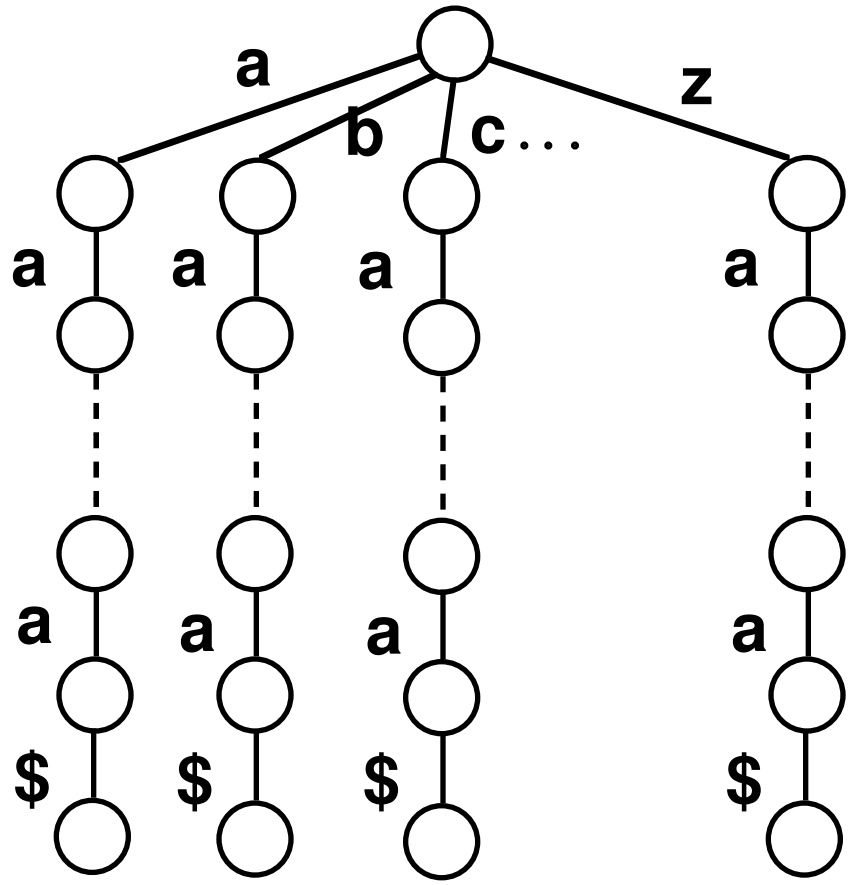
```
TRIE_NODE *trie_search(char *s,
                       TRIE_NODE *v){
   for(; *s; s++){
      if(v->a[*s] == NULL)
         return NULL;
      v = v->a[*s];
   }
   return v;
}
```

סיבוכיות זמן: length(s)

שימו לב: האלגוריתם לא משווה בין מחרוזות.

סיבוכיות מקום

trie עם n מחרוזות, כל-אחת באורך m,
"aa...a$", "ba...a$", "ca...a$", ...



סה"כ nm צמתים, שבכל אחד מקום ל- σ בנים.
$\theta(\sigma mn)$
שימו לב שאורך הקלט הוא nm.

לפעמים גודל ה-trie קטן מגודל הקלט:
דוגמא
"a...a$", "a...ab$","a...ac$",...,
"a...az$"

היורוסטיקות לחסכון במקום

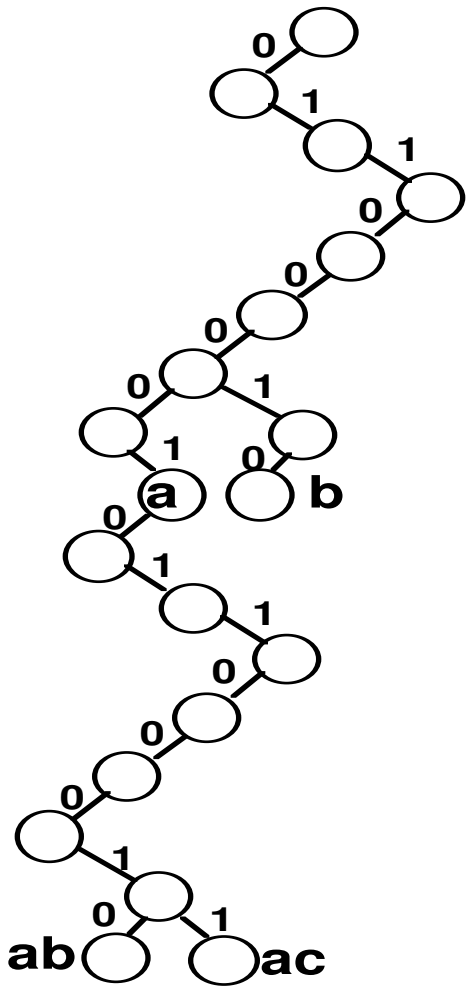
א. הקטנת הא"ב

במקום א"ב Σ נשתמש בא"ב בינארי:
כלומר, במקום לבנות trie על Σ נבנה trie בינארי
על ייצוג ascii של המחרוזת

דוגמא:

"ab","ac","b"

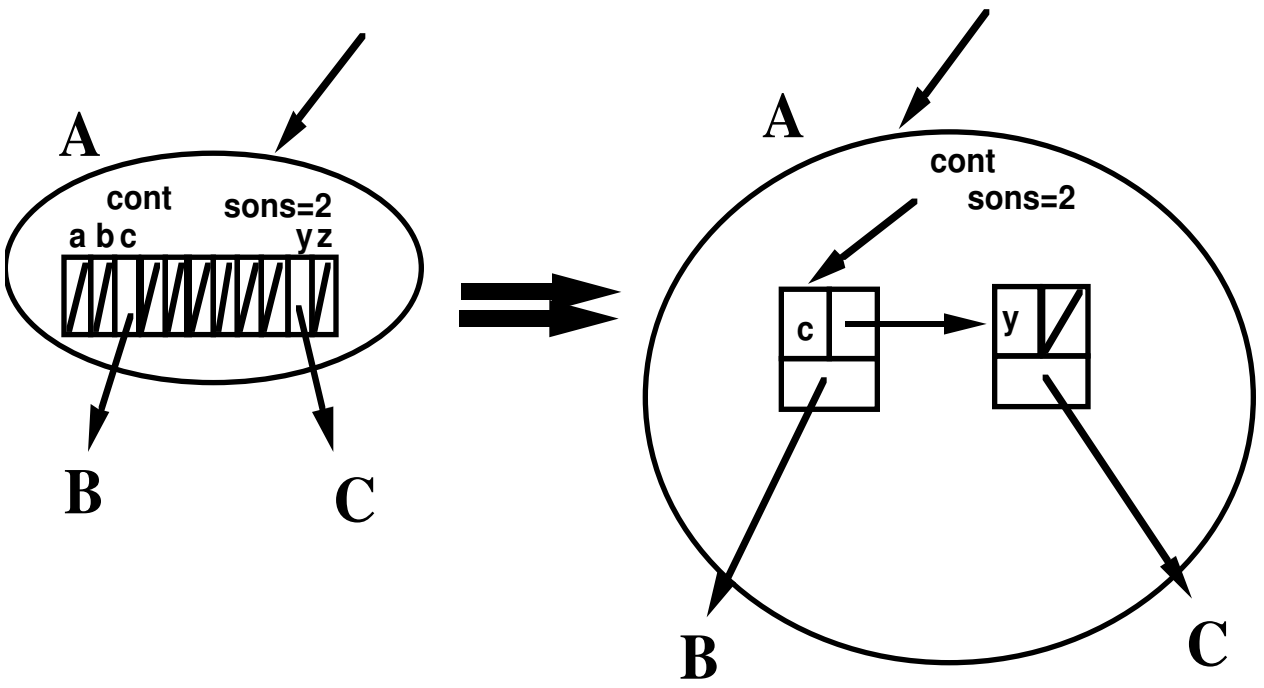
ascii של "a" הוא 0110 0001 = 97



כל קשת הופכת למסלול באורך $\log_2 \sigma$
סיבוכיות המקום היא $O(mn \log_2 \sigma)$
מה עם סיבוכיות הזמן?

ב. מימוש צמת מעוט בנים ע"י רשימה

נקבע סף t, כך שצומת בעל פחות מ-t בנים יוחלף במבנה פשוט יותר -- למשל, רשימה מקושרת.

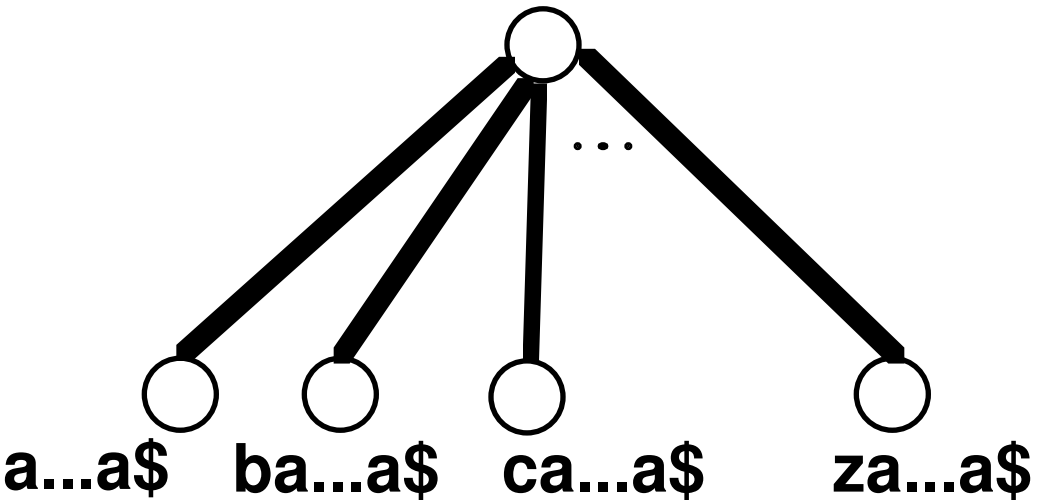


מבנה צומת של ה-trie:

```
typedef struct trie_node{
   int sons;
   union {
      struct trie_node *a[σ];
      NODE *first_son;
   } children;
} TRIE_NODE;
```

ג. trie_דחוס

נסלק מהעץ צמתים בעלי בן אחד,
נחליף "שרוך" בקשת שתסומן במחרוזת.



a[i] יצביע על מבנה

```
TYPEDEF struct {
      char *label;
      TRIE_NODE *head;
   } PATH;
```

סיבוכיות מקום:

מספר העלים $n \geq$,
לכל צומת פנימי, דרגה ≥ 2,
לכן יש לכל היותר $n-1$ צמתים פנימיים.
סה"כ מספר הצמתים $\geq \quad 2n-1$.

לפעמים אפשר לדחוס עוד,
כך שגודל ה-trie קטן מגודל הקלט.

מיון מחרוזות

<u>קלט</u>: מחרוזות $s_1,\ldots,s_n$

שאורכן הכולל $I=\sum_{i=1}^{n}|s_i|$

<u>פלט</u>: אותן המחרוזות בסדר לקסיקוגרפי.

<u>האלגוריתם</u>:

(נניח ש-'$' מיוצג ע"י 0)

1 הכנס את $s_1,\ldots,s_n$ ל-trie

2 עבור על ה-trie לפי סדר preorder וכתוב לפלט את המסלול לכל עלה

<u>סיבוכיות</u>

זמן: צעד 1: הכנסת s_i דורשת זמן $|s_i|$,

סה"כ $I=\sum_{i=1}^{n}|s_i|$ (גודל הקלט).

צעד 2: גם-כן זמן $O(I)$.

סה"כ : זמן $O(I)$.

שיטת מיון קונבנציונלית:

$|s_i|\cong\frac{I}{n}$, השוואות, כל-אחת לוקחת זמן $n\log n$

לכן סה"כ זמן $\frac{I}{n}n\log n=I\log n$.