



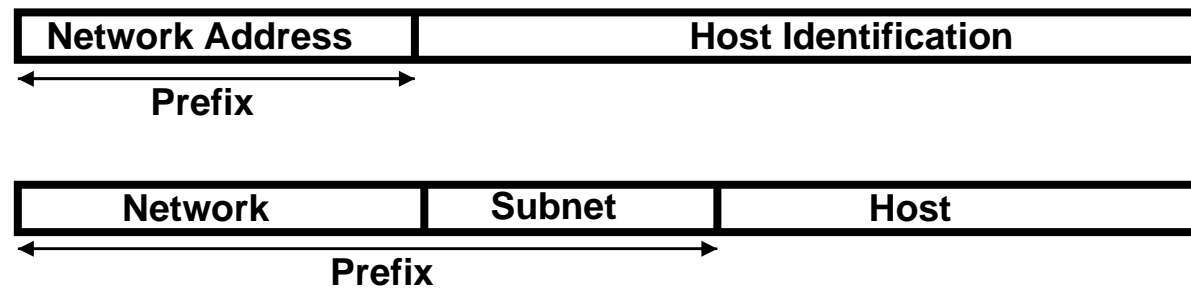
Hálózatok II

2006

5: Az IP Prefix Lookup Probléma III.

Internet Protocol IP

- Az adatok a küldőtől a cél állomásig IP-csomagokban kerülnek átvitelre
- A csomagok fejléce tartalmazza a cél IP-címét



- Minden csomópont (router) a cél címe alapján dönt, hogy melyik szomszédos csomópontnak kell továbbítania a csomagot
 - Az ehhez szükséges információt minden csomópontban egy routing-tábla tárolja

IP Prefix Lookup Probléma

- A címek W hosszúságú bináris sztringek.

Egy router routing-táblája R

- bejegyzéseket tartalmaz (x,y) formában
 - x : legfeljebb W hosszú bináris sztring, melynek a neve prefix. x lehet az üres sztring is, amit ε -nal jelölünk,
 - y : egy a router-ből kivezető link.
- Minden x bináris sztringhez R legfeljebb egy (x,y) bejegyzést tartalmaz.
- A bejegyzések száma a routing-táblában N .

Feladat:

- A routerhez érkező csomaghoz, melynek cél címe k ,
- találjuk meg az (x,y) bejegyzést R -ben a leghosszabb egyező prefixszel (BMP).

Prefix expanzió [Srinivasan, Varghese 99]

- Az IP Prefix Lookup bináris kereséssel a prefixhossz szerint $O(\log r)$ időt igényel, ahol r a nemüres hosszosztályok száma.
- A routing-táblákban IPv4 esetén
 - Az osztályok L_1 és L_7 között legtöbbször üresek, így legtöbbször $r \leq 25$;
 - Gyakran a L_{16} , L_{24} osztályok tartalmazzák messze a legtöbb bejegyzést.
- Cél: A routing-táblát úgy módosítani, hogy a hosszosztályok száma csökkenjen.
- Alapötlet: egy $i < W$ hosszú x prefixet kicserélhetünk két $(i+1)$ hosszú prefixre $x0$ -ra és $x1$ -re anélkül, hogy a routing-tábla viselkedését megváltoztatnánk: Minden célcím, amelyben x prefix, vagy $x0$ -al vagy $x1$ -gyel kezdődik.

Prefix expanszió

- Azt mondjuk, hogy két routing-tábla R és R' **ekvivalens**, ha minden k célcímre teljesül, hogy mindazon csomagok, melyekben a cél címe k , R és R' által ugyanazon a linkeken továbbítódnak.
- Az x prefix helyettesítését x_0 és x_1 által az x prefix **expansziójának** nevezzük.
- Az x prefix expansziója a routing-bejegyzésekre nézve azt jelenti, hogy (x,y) -t helyettesítjük (x_0,y) -nal és (x_1,y) -nal. Pontosabban:
 - Töröljük (x,y) -t R -ből;
 - Megvizsgáljuk, hogy egy (x_0,y') bejegyzés már benne van-e R -ben. Ha még nincs, befűzzük (x_0,y) -t R -be.
 - Megvizsgáljuk, hogy egy (x_1,y') bejegyzés már benne van-e R -ben. Ha még nincs, befűzzük (x_1,y) -t R -be.
- A prefix expanszió után a routing-tábla ekvivalens marad az eredetivel.

Prefix expanzió

- Egy i hosszú x prefix j -szeri expanziója után legfeljebb 2^j új prefix áll elő R -ben, melyek hossza $i+j$.
- Ha R -t prefix expanziók útján úgy módosítjuk, hogy csak s különböző prefixhossz maradjon és az új tábla az eredetivel ekvivalens marad, akkor
 - először rögzíthetjük az s hosszt $l_1 < l_2 < \dots < l_s$, amely osztályok majd a nem üres osztályok lesznek. Az l_s hossz legalább olyan nagy, mint a leghosszabb prefix hossza az eredeti routing-táblában.
 - R bejegyzéseit hossz szerint növekvő sorrendben dolgozzuk fel. Minden nemüres L_j hosszosztálynál a következő eseteket különböztetjük meg:
 - Ha $i \in \{l_1, l_2, \dots, l_s\}$, akkor folytatjuk a következő hosszosztállyal.
 - Különben L_j minden x prefixéhez expanziót hajtunk végre. Ha $x0$ ill. $x1$ még nem volt benne L_{i+1} -ben, akkor befűzzük L_{i+1} -be.

Prefix expanszió

Példa:

- R a prefix expanszió előtt: 7 nem üres hosszosztály összesen 8 bejegyzés:

- $L_1 = \{(0, y_1), (1, y_2)\}$
- $L_2 = \{(10, y_3)\}$
- $L_3 = \{(111, y_4)\}$
- $L_4 = \{(1000, y_5)\}$
- $L_5 = \{(11001, y_6)\}$
- $L_6 = \{(100000, y_7)\}$
- $L_7 = \{(1000000, y_8)\}$

- R a prefix expanszió után, amelyben csak $l_1=2$, $l_2=5$, $l_3=7$ hossz fordul elő, összesen 13 bejegyzés:

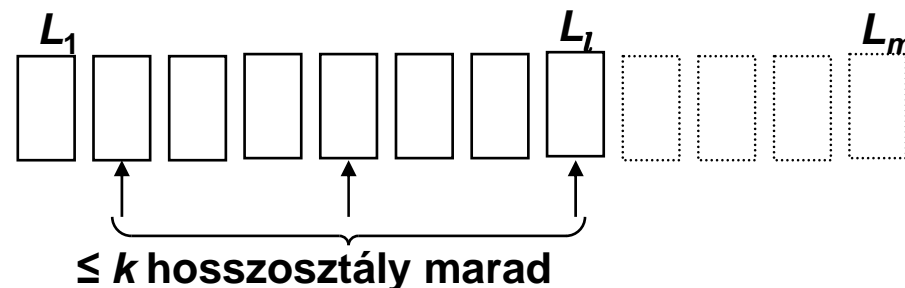
- $L_2 = \{(00, y_1), (01, y_1), (10, y_3), (11, y_2)\}$
- $L_5 = \{(11100, y_4), (11101, y_4), (11110, y_4), (11111, y_4), (10000, y_5), (10001, y_5), (11001, y_6)\}$
- $L_7 = \{(1000000, y_8), (1000001, y_7)\}$

A hosszosztályok választása – Dinamikus programozás

- Tehát prefix expanzióval konstruálható egy ekvivalens routing-tábla, ami
 - kevesebb hosszosztályt tartalmaz. Ebből rövidebb keresési idő adódik (pl. a bináris kereséssel a prefixhossz szerint);
 - rendszerint több bejegyzést tartalmaz.
- Hogyan lehet a l_1, l_2, \dots, l_s hosszosztályokat kedvezően megválasztani, ha s előre adott? Hogyan lehet a bejegyzések számát minimalizálni?
- Egy optimális megoldást **dinamikus programozás** segítségével számíthatunk ki. (az optimális megoldást a részfeladatok előzetesen kiszámított optimális megoldásaiból állítjuk össze.)

A hosszosztályok választása – Dinamikus programozás

- Legyen m a maximális prefixhossz az adott R routing-táblában.
- Jelölje $M[l,k]$, $l \in \{0,1,\dots,m\}$ -re és $k \in \{1,2,\dots,s\}$ -re, a prefixek minimális számát, amit úgy kaphatunk, hogy az eredeti hosszosztályokat L_1, \dots, L_l ($l = 0$ esetén ez az üres halmaz) prefix expanzióval úgy változtatjuk meg,
 - hogy legfeljebb k nem üres hosszosztály marad és
 - L_l a hosszosztály a maximális hosszal.



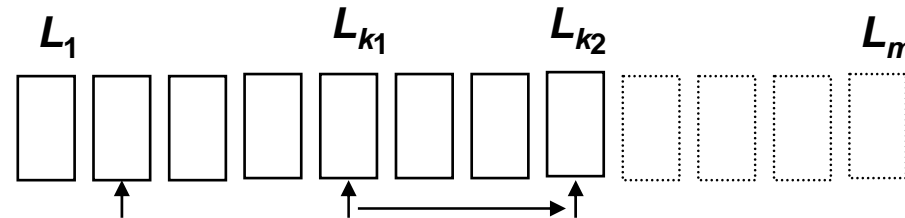
- $M[m,s]$ értéke ekkor megadja a minimális számú prefixet, amit akkor kapunk, ha egy routing-táblát s nem üres hosszosztállyal optimális prefix expanzióval konstruálunk.

A hosszosztályok választása – Dinamikus programozás

- Legyen $E[k_1, k_2]$, $1 \leq k_1 < k_2 \leq m$ az a segédváltozó, ami megadja, hány prefix áll elő a L_{k_2} hosszosztályban, ha minden prefixet a $L_{k_1}, L_{k_1+1}, \dots, L_{k_2-1}$ hosszosztályokból prefix expanzióval k_2 hosszúságúra bővítünk.

A példánkban pl. $E[3,5] = 7$.

$E[k_1, k_2]$ minden (k_1, k_2) párra hatékonyan kiszámítható.



- Ekkor a következő érvényes

$$\bullet M[l, 1] = E[1, l] \quad \forall l \geq 1 \quad (1)$$

$$\bullet M[0, k] = 0 \quad \forall k \geq 1 \quad (2)$$

$$\bullet M[l, k] = \min_{0 \leq j < l} \{ M[j, k-1] + E[j+1, l] \} \quad \forall l > 0, k > 1 \quad (3)$$

A hosszosztályok választása – Dinamikus programozás

- (1) ($M[l,1] = E[1,l]$, $\forall l \geq 1$) érvényes, mert $M[l,1]$ és $E[1,l]$ definíciója azonos.
- (2) ($M[0,k] = 0$, $\forall k \geq 1$) érvényes, mert hosszosztályok egy üres halmazából prefix expanszió útján sem keletkezik új prefix.
- (3)-ban ($M[l,k] = \min_{0 \leq j < l} \{ M[j,k-1] + E[j+1, l] \}$, $\forall l > 0, k > 1$) j a második legnagyobb hossz l után, amelyre egy nem üres hosszosztályt választunk.
 - Minden rögzített j -re egy optimális prefix expansziót a $1,2,\dots,l$ hosszosztályokból k nem üres hosszosztállyal úgy kapunk, hogy vesszünk egy optimális prefix-expansziót a $1,2,\dots,j$ hosszosztályokból $k-1$ nem üres osztállyal és a $j+1,\dots,l$ osztályok egy prefix-expanszióját L_l hosszosztályra.
 - A minimum minden j fölött, $0 \leq j < l$, megadja az optimális prefix-expansziót a $1,2,\dots,l$ hosszosztályokhoz k nem üres osztállyal.

A hosszosztályok választása – Dinamikus programozás

- Legyen $P[l,k]$ az a j érték, melyre $M[l,k] = \min_{0 \leq j < l} \{ M[j,k-1] + E[j+1,l] \}$ a minimumot adja. A számítás végén
 $m, P[m,s], P[P[m,s],s-1], \dots$
megadják a prefixek hosszait az s hosszosztályban.

Algoritmus Optimális_Prefix_Expanzió

Input: $E[k_1,k_2]$, $\forall 1 \leq k_1 < k_2 \leq m$; és a hosszosztályok száma s

Output: $M[l,k]$ és $P[l,k]$, $\forall 0 \leq l \leq m, 1 \leq k \leq s$

1. for $l = 0$ to m do
2. for $k = 1$ to s do
3. if $l = 0$ then $M[l,k] := 0$; $P[l,k] := 0$;
4. else if $k = 1$ then $M[l,k] := E[1,l]$; $P[l,k] := 0$;
5. else
6. $M[l,k] := \min_{0 \leq j < l} \{ M[j,k-1] + E[j+1,l] \}$;
7. $P[l,k] :=$ az a j , amelyre az előző sor a minimumot adta;
8. fi;
9. od;
10. od;

A hosszosztályok választása – Dinamikus programozás

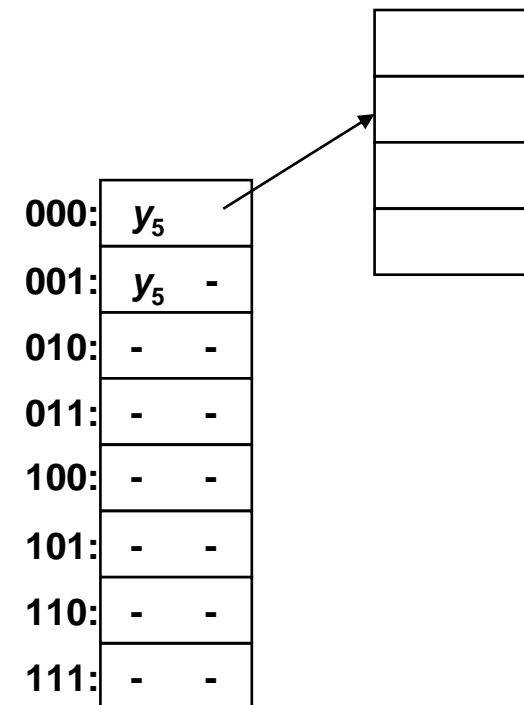
Tétel 1: Az Optimális_Prefix_Expanzió algoritmus kiszámítja $M[m,s]$ -ben egy adott s -re a prefixek minimális számát, amit akkor kapunk, ha egy routing-táblát s nem üres hosszosztállyal optimális prefix-expanzióval konstruálunk. Az algoritmus futási ideje $O(sW^2)$. □

A hosszosztályok választása – Dinamikus programozás

- Dinamikus programozással megoldhatjuk a következő problémákat optimálisan (vagy majdnem optimálisan):
 - Tekintsünk egy konkrét adatstruktúrát az IP Prefix Lookup problémához (pl. Bináris keresés prefixhossz szerint). Válasszuk a prefixhosszakat a prefix-expanzióhoz s nemüres hosszosztályra úgy, hogy az adatstruktúra tárigénye minimális legyen. (bináris keresésnél prefixhossz szerint a tárigényt a prefixek száma plussz a jelzések száma határozza meg.)
 - Tekintsünk egy konkrét adatstruktúrát az IP Prefix Lookup problémához. Legyen adott az adatstruktúrának megengedett tártelület (pl. az rendelkezésre álló second-level-cache mérete által). Válasszuk a prefixhosszakat a prefix-expanzióban úgy, hogy a megengedett tárterületet ne lépjük túl és a worst-case BMP keresési idő minimális legyen.

Multibit-Trie és prefix-ekpanzió

- A BMP keresése Trie adatstruktúrával $O(W)$ időt igényel (worst-case).
- Egy lehetőség a keresési idő megjavítására: Minden Trie-csomópontban a célcímnek egyszerre több bitjét kezelni.
 - Ha pl. egy Trie-csomópontban 3 bitet egyszerre kezelünk, akkor a Trie-csomópontnak tartalmazni kell egy 2^3 elemű arrayt, melynek elemei egy másik Trie-csomópontra mutatnak vagy egy linket tárolnak.



Multibit-Trie és prefix-expanzió

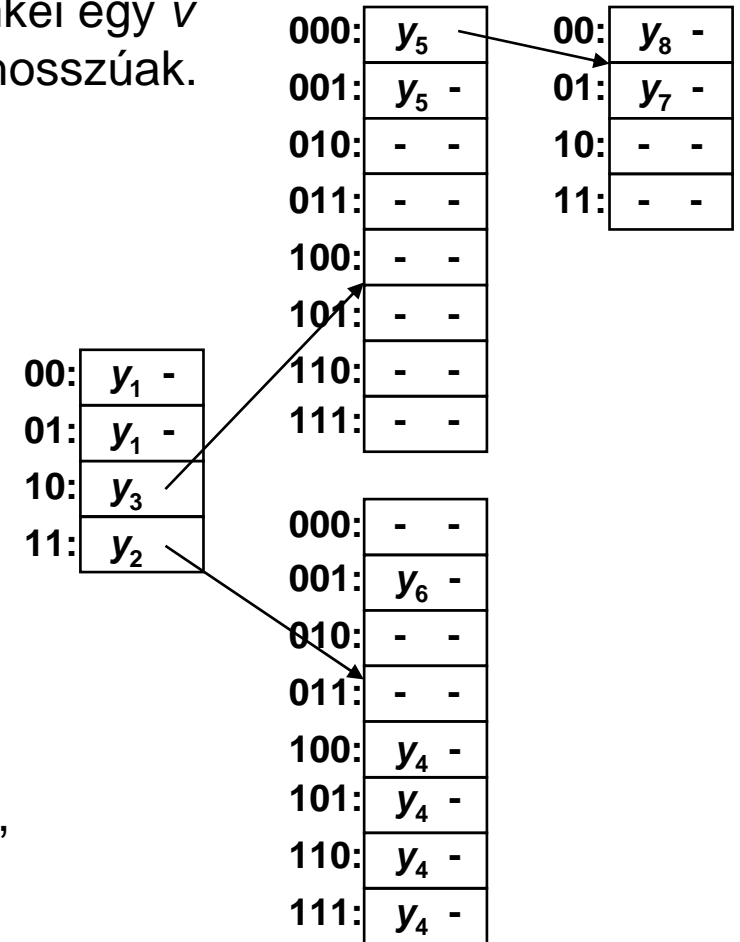
- Egy **Multibit Trie** egy olyan Trie, amiben az élek címkei egy v csomópontból a gyermekeihez mind egyformán k_v hosszúak.
 - A címkek tárolása helyett egy 2^{k_v} elemű arrayt használunk.
 - Az array i . elemében, $0 \leq i < 2^{k_v}$, az áll, hogy
 - létezik-e egy él a Trie-ban i címkével, és ha igen, hová vezet;
 - ahhoz a prefixhez, ami a gyökérből a v -hez vezető úton lévő élek címkeinek és az i sztring konkatenációjaként áll elő, tárolunk-e egy linket.

- Példa:

$$L_2 = \{ (00, y_1), (01, y_1), (10, y_3), (11, y_2) \}$$

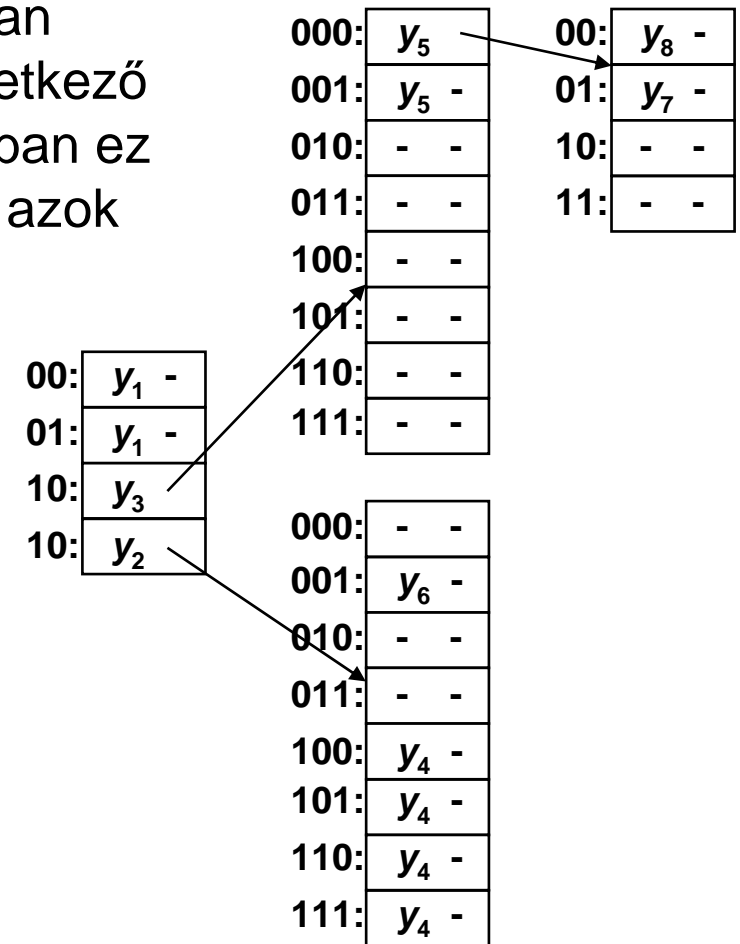
$$L_5 = \{ (11100, y_4), (11101, y_4), (11110, y_4), (11111, y_4), (10000, y_5), (10001, y_5), (11001, y_6) \}$$

$$L_7 = \{ (1000000, y_8), (1000001, y_7) \}$$



Multibit-Trie és prefix-ekpanzió

- A bitek száma b_v , amit egy v Trie-csomópontban kezelünk, egy nemüres hosszosztály és a következő nemüres osztály hosszkülönbsége. A példánkban ez a gyökérben 2, annak gyermekeinél $5-2=3$, és azok gyermekeinél $7-5=2$.
- Egy k célcím keresése:
A w gyökérben tekintjük k első b_w bitjét $k[1..b_w]$ -t. A keresést annál a gyermeknél folytatjuk, amelyre w -ben az array $k[1..b_w]$ -edik eleme mutat, ha ilyen mutató létezik. Amíg ilyen mutató létezik, követjük a megfelelő utat. Az utolsó link y , amit ezen az úton találtunk, az tartozik a BMP-hez.



Multibit-Trie és prefix-expanzió

- A worst-case keresési idő egy T Multibit-Trie-ban lineáris T mélységéhez, azaz $O(s)$, ahol s a nem üres hosszosztályok száma.
- Egy bejegyzés befűzése vagy törlése hatékonyan elvégezhető.

A tárigény minimalizálása a Multibit-Trie-ban

- A Trie minden v csomópontjának tárigénye 2^{b_v} (az array elemeinek száma v -ben).
- A T Trie teljes tárigénye $\sum_{v \in T} 2^{b_v}$.

Optimalizálási feladat: Hogyan kell egy megadott s esetén az s hosszosztályt kiválasztani, hogy a Multibit-Trie tárigénye minimális legyen?

Feltétel: A hosszosztályok és a Trie rétegei között egy-az-egyhez megfelelés van. (Multibit-Trie esetén enélkül a feltétel nélkül is optimalizálhatunk. [Srinivasan, Varghese 99])

- A Trie egy rétege azon Trie-csomópontok halmazát jelenti, melyek a gyökértől ugyanolyan távolságra vannak. Az 1. réteg a gyökérből áll. A 2. réteg a gyökér gyermekeiből, és így tovább...
- Az i . réteg megfelel az L_i hosszosztálynak.
- Következésképpen egy réteg minden csomópontjánál ugyanannyi bitet tekintünk.
- Ezért egy réteg minden csomópontja ugyanakkora tárigényű.

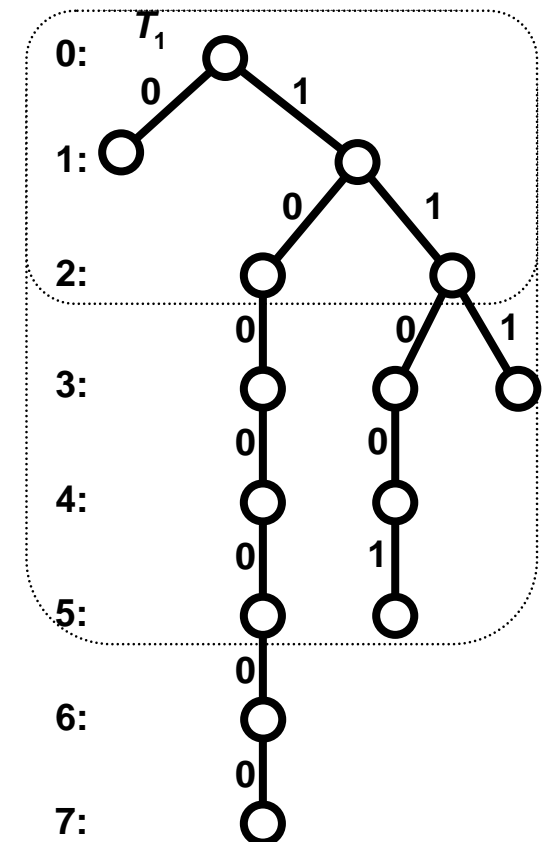
A tárigény minimalizálása a Multibit-Trie-ban

- Jelölje $A[j,l]$, $0 \leq j < l \leq m$, a nemüres L_l hosszosztály minimális tárigényét, ha a tőle rövidebb leghosszabb nemüres hosszosztály L_j , $j < l$.

- $A[j,l]$, $0 \leq j < l \leq m$ kiszámítása:

- Felépítünk egy 1-Bit-Trie-t T_1 -t az eredeti prefixekhez úttömörítés nélkül.
- Legyen $n(i)$ azon csomópontok száma T_1 -ben, amelyek távolsága a gyöktől i és legalább egy gyermekük van.
- Teljesül: $A[j,l] = 2^{l-j} n(j)$.

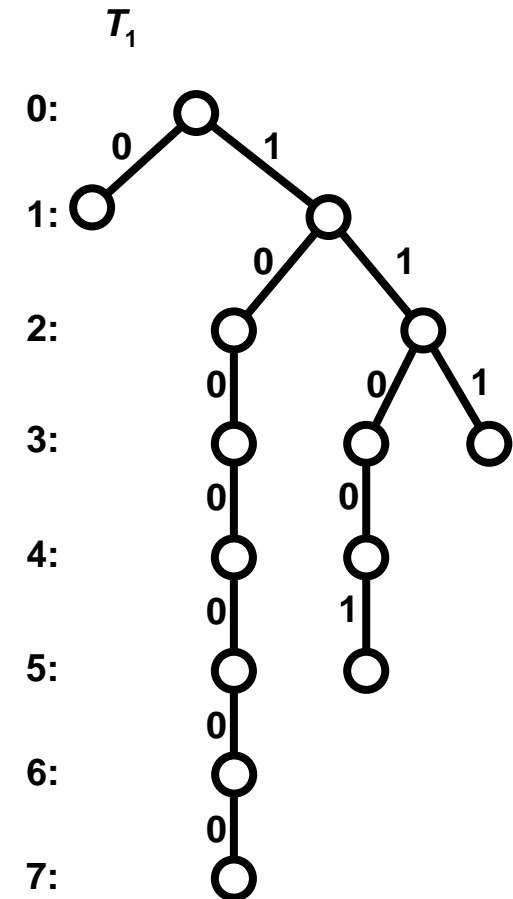
A Multibit-Trie L_j -nek megfelelő rétegének minden csomópontjában az array minden eleme, amely T_1 -ben egy olyan csomópontnak felel meg, aminek legalább egy gyermeke van, L_l -nek megfelelő réteg csomópontjára mutat. Tehát az L_l -nek megfelelő rétegnek pontosan $n(j)$ csomópontot kell tartalmaznia, amelyek mindegyike 2^{l-j} elemet tartalmaz. Minden $0 \leq j < l \leq m$ -hez $A[j,l]$ kiszámítható T_1 segítségével $O(m^2) = O(W^2)$ idő alatt.



A tárigény minimalizálása a Multibit-Trie-ban

Példa:

- $L_1 = \{(0, y_1), (1, y_2)\}$
 - $L_2 = \{(10, y_3)\}$
 - $L_3 = \{(111, y_4)\}$
 - $L_4 = \{(1000, y_5)\}$
 - $L_5 = \{(11001, y_6)\}$
 - $L_6 = \{(100000), y_7\}$
 - $L_7 = \{(1000000), y_8\}$
-
- $A[0,2] = 2^{2-0} n(0) = 4 \cdot 1 = 4,$
 - $A[2,5] = 2^{5-2} n(2) = 8 \cdot 2 = 16,$
 - $A[5,7] = 2^{7-5} n(5) = 4 \cdot 1 = 4.$
-
- Ez megadja egy 3 szintű Trie társzükségletét L_2 , L_5 és L_7 hosszosztályokkal.



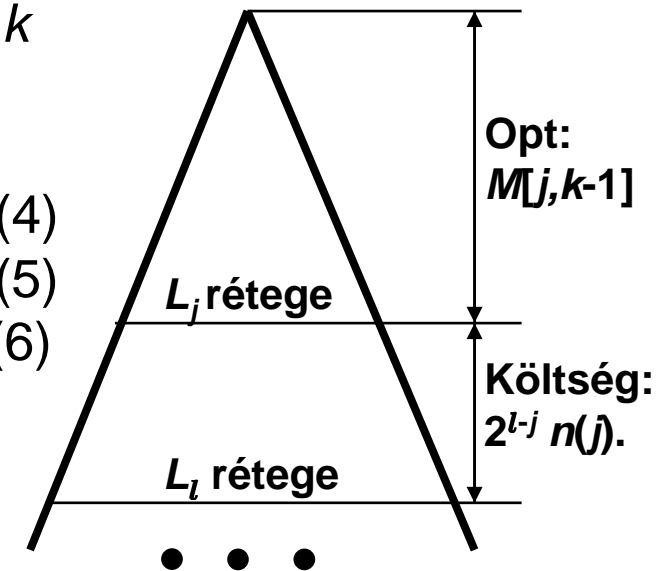
A tárigény minimalizálása a Multibit-Trie-ban

- Tekintsük azt a Multibit-Triet, ami akkor keletkezik, ha az eredeti L_1, \dots, L_l hosszosztályokat úgy alakítjuk át prefix expanzióval, hogy
 - legfeljebb k nem üres hosszosztály marad és
 - L_l a legnagyobb hosszt tartalmazó hosszosztály.
- Legyen $M[l, k]$ a minimális tárigénye a Trie rétegeinek a gyökértől addig a rétegig, ami L_l -nek felel meg, ha k hosszosztályt engedünk meg. Ekkor érvényes:

$$M[l, 1] = A[0, l] \quad \forall l \geq 1 \quad (4)$$

$$M[0, k] = 0 \quad \forall k \geq 1 \quad (5)$$

$$M[l, k] = \min_{0 \leq j < l} \{ M[j, k-1] + A[j, l] \} \quad \text{ha } l > 0, k > 1 \quad (6)$$



A tárigény minimalizálása a Multibit-Trie-ban

- (4) ($M[l,1] = A[0,l]$, $\forall l \geq 1$) igaz, mert a gyökér társzükséglete pontosan $A[0,l] = 2^l n(0) = 2^l$, ha L_l az a nemüres hosszosztály, ami a legrövidebb sztringeket tartalmazza.
- (5) ($M[0,k] = 0$ für $\forall k \geq 1$) csak annyit mond, hogy egy üres Multibit-Trie nem igényel tárt.
- (6) ($M[l,k] = \min_{0 \leq j < l} \{ M[j,k-1] + A[j,l] \}$ für $l > 0, k > 1$) igaz, mert az L_l -t megelőző leghosszabb hosszosztályra, L_j -re minden lehetőséget megvizsgálunk és ekkor a tárigény a következő két érték összege:
- Az optimális tárigény a Multibit-Trie rétegekhez, amelyek a hosszosztályoknak L_j -ig felelnek meg és legfeljebb $k-1$ nem üres hosszosztályt tartalmaznak (ezt $M[j,k-1]$ írja le) és
 - Az L_l hosszosztály rétegének a tárigénye (ezt $A[j,l]$ írja le).

A tárigény minimalizálása a Multibit-Trie-ban

- Dinamikus programozással hatékonyan ki tudjuk számítani $M[l,k]$ minden értékét.
- $M[m,s]$ megadja a minimális tárigényt a Multibit-Trie-hoz, ami s réteget tartalmaz.
- Legyen $P[l,k]$ az a j érték, amire $M[l,k] = \min_{0 \leq j < l} \{ M[j,k-1] + A[j,l] \}$ a minimumot veszi fel. A számítás végén
 $m, P[m,s], P[P[m,s],s-1], \dots$
adja meg a prefix-hosszakat az s hosszosztályban.

A tárigény minimalizálása a Multibit-Trie-ban

Algoritmus Optimalis_Multibit_Trie

Input: $A[j,l]$, $\forall 0 \leq j < l \leq m$; A hosszosztályok száma s

Output: $M[l,k]$ és $P[l,k]$, $\forall 0 \leq l \leq m$, $1 \leq k \leq s$

1. for $l = 0$ to m do
2. for $k = 1$ to s do
3. if $l = 0$ then $M[l,k] := 0$; $P[l,k] := 0$;
4. else if $k = 1$ then $M[l,k] := A[0,l]$; $P[l,k] := 0$;
5. else
6. $M[l,k] := \min_{0 \leq j < l} \{ M[j,k-1] + A[j,l] \}$;
7. $P[l,k] :=$ az a j , amire az előző sor a minimumot adja;
8. fi;
9. od;
10. od;

Az optimális Multibit-Trie kiszámításának időanalízise

- Az 1-Bit-Trie felépítése és az $n(i)$ und $A[j,l]$ értékek kiszámítása $O(N \cdot W + W^2)$ időt igényel.
- $M[l,k]$ értékeinek kiszámítása az Optimalis-Multibit-Trie algoritmussal $O(s \cdot W^2)$ időt igényel.
- Összesen: $O(N \cdot W + s \cdot W^2)$ idő.

Tétel 2: Legyenek egy routing-tábla bejegyzései és a kívánt nem üres hosszosztályok száma s adva. Az s nemüres hosszosztály kiválasztása, amely a routing-táblához egy s rétegű Multibit-Trie tárigényét minimalizálja, $O(N \cdot W + s \cdot W^2)$ idő alatt kiszámítható. □

Irodalom

- V. Srinivasan, G. Varghese: **Fast Address Lookups using Controlled Prefix Expansion**. *ACM Transactions on Computer Systems*, Vol. 17(1), 1-40, 1999.