Κινητός και Διάχυτος Υπολογισμός (Mobile & Pervasive Computing)

Δημήτριος Κατσαρός

Χειμώνας 2015

Διάλεξη 7η

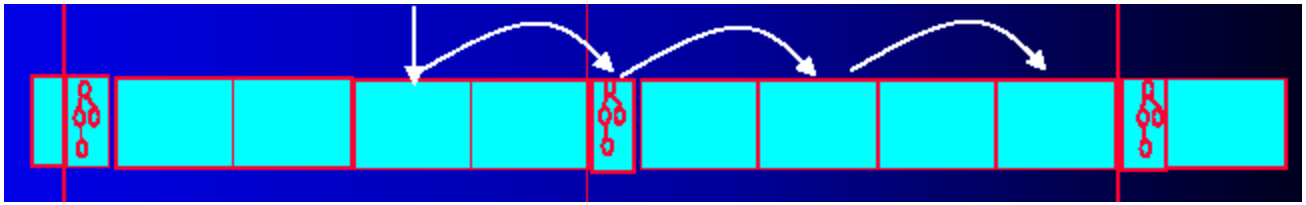
Περιεχόμενα

- **Ευρετήρια**

Παράμετροι ενδιαφέροντος (1/2)

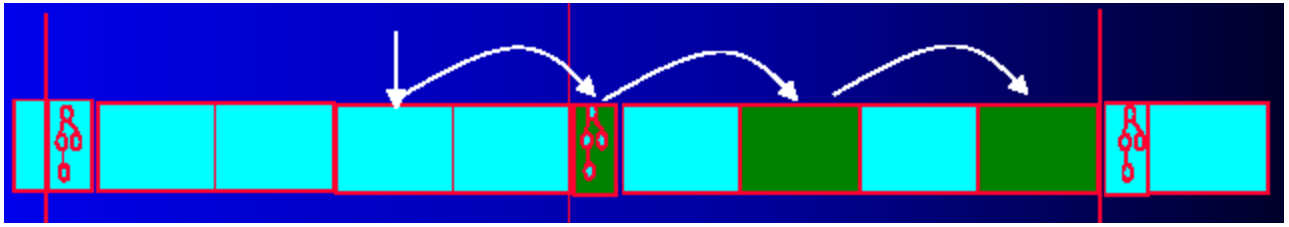
- **Tuning time**: Ο χρόνος που ο κινητός πελάτης δαπανά "ακούγοντας" το κανάλι. Προσδιορίζει την κατανάλωση ενέργειας για την απόκτηση των δεδομένων
- **Latency (Access time)**: Ο χρόνος που περνάει (κατά μέσο όρο) από τη στιγμή που ο κινητός πελάτης "κάνει αίτηση" για κάποια δεδομένα μέχρι τη στιγμή που τα δεδομένα αυτά έρχονται στην κατοχή του πελάτη
  - **Probe wait**: Ο μέσος χρόνος από τη στιγμή συντονισμού στο κανάλι μέχρι να βρει τον δείκτη (pointer) για τον επόμενο index. Είναι ίσος με το μισό της απόστασης μεταξύ δυο τμημάτων index.
  - **Bcast wait**: Ο μέσος χρόνος από τη στιγμή που βρίσκεται ο πρώτος index μέχρι να "κατεβούν" όλα τα δεδομένα

Παράμετροι ενδιαφέροντος (2/2)

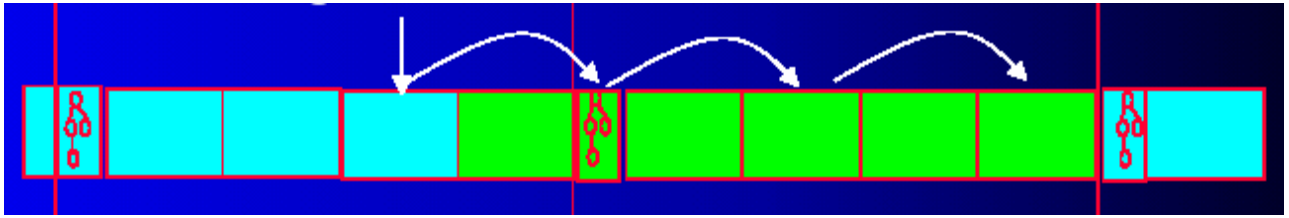


Querying

Tuning time

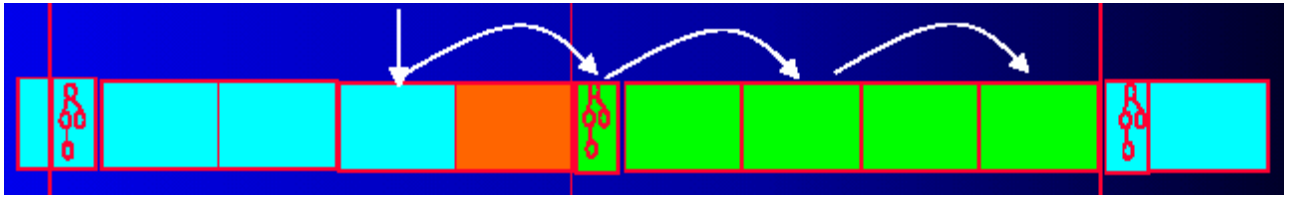


Latency



Probe wait

Bcast wait



Οργάνωση του καναλιού εκπομπής

- **Packet**: η βασική (μικρότερη) μονάδα μεταφοράς μηνυμάτων στα δίκτυα
- **Bucket**: η μικρότερη λογική μονάδα εκπομπής. Αποτελείται από σταθερό αριθμό packets. Όλα τα buckets έχουν το ίδιο μέγεθος
  - Index buckets
  - Data buckets
- **Index Segment**: σύνολο συνεχόμενων index buckets
- **Data Segment**: σύνολο συνεχόμενων data buckets

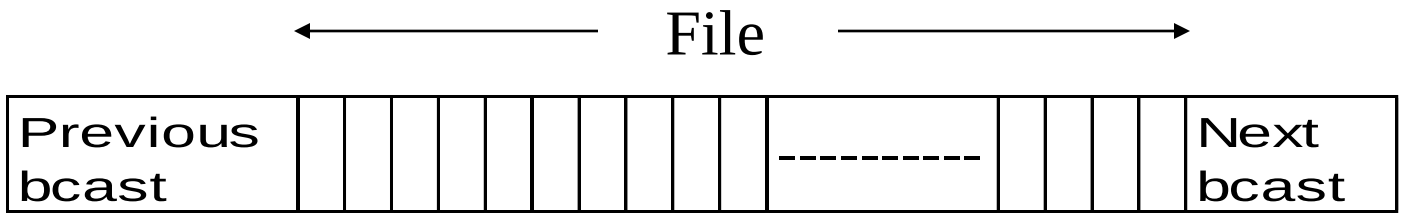
Οργάνωση του καναλιού εκπομπής

- Περιεχόμενα bucket
  - **Bucket_id**: το offset του bucket από την αρχή του κύκλου εκπομπής
  - **Bcast_pointer**: το offset μέχρι την αρχή του επόμενου κύκλου εκπομπής
  - **Index_pointer**: το offset μέχρι την αρχή του επόμενου index segment
  - **Bucket_type**: data bucket ή index bucket
- Index bucket: είναι μια ακολουθία της μορφής:
  - (**attibute_value, offset**): offset είναι ένας δείκτης στο bucket που περιέχει εγγραφή που προσδιορίζεται από την attribute_value

Clustering index

- **Clustering index**: ένα ευρετήριο (index) είναι clustered πάνω σε ένα attribute, εάν όλες οι εγγραφές με την ίδια τιμή για το attribute αυτό, εμφανίζονται συνεχόμενες σε ένα “αρχείο”
- Τα δυο άκρα στην βελτιστοποίηση Tuning time και Acceess time
  - Latency_opt
  - Tune_opt

Latency OPT

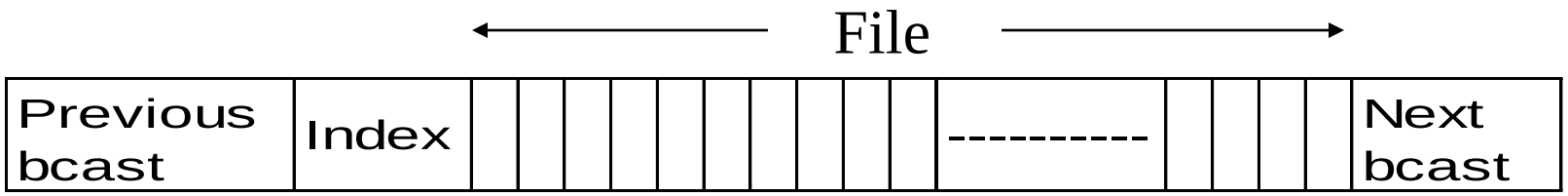


Latency είναι η βέλτιστη: Δεν υπάρχει επιβάρυνση για το index

Latency = Data/2 + C

Tuning time = Data/2 + C

Tuning OPT



Tuning time είναι ο βέλτιστος:

Latency = (Data + Index) / 2 + (Data + Index) / 2 + C

= Data + Index + C

Tuning time = k + C

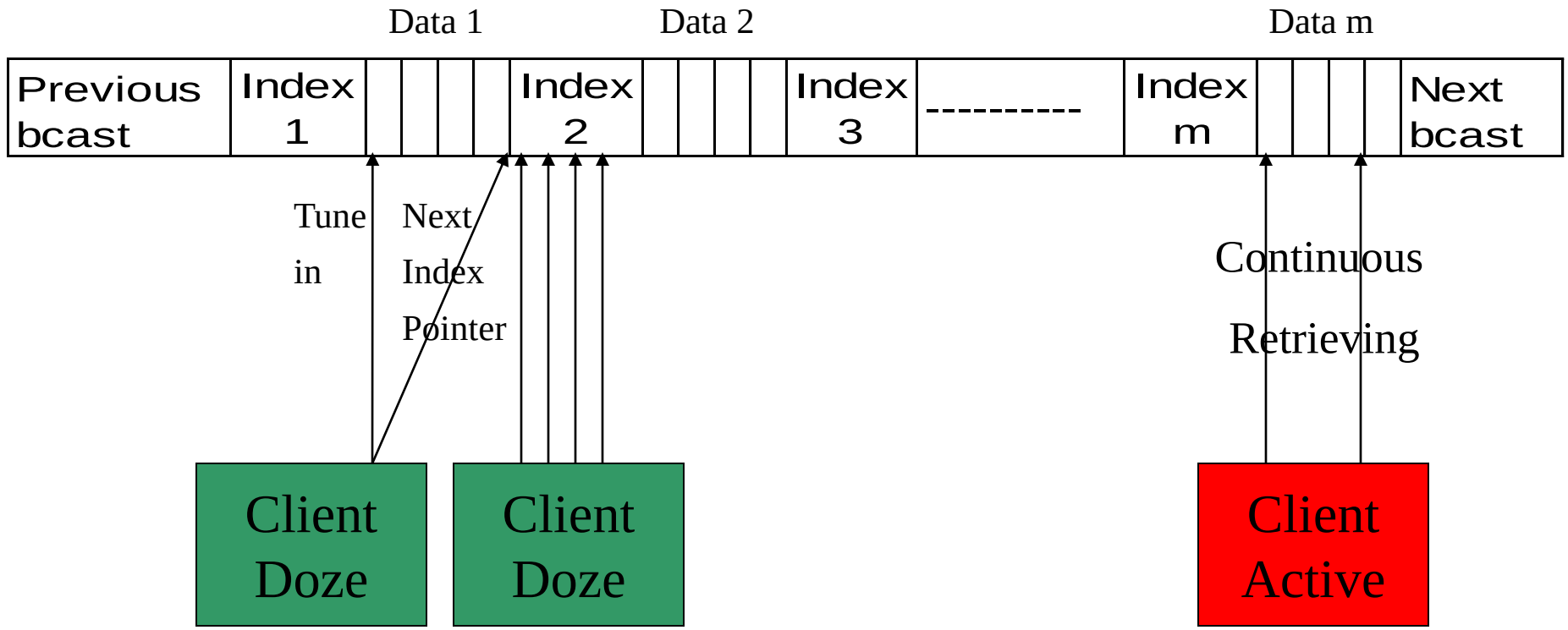
k: number of levels in the index tree

(1, m) Indexing

Το ευρετήριο (index) εκπεμπεται m φορές κατά τη διάρκεια μιας εκπομπής του αρχείου.

Πολυεπίπεδο ευρετήριο (index), δηλαδή δένδρο

Ολόκληρος ο index εκπέμπεται πριν από το 1/m κομμάτι του αρχείου.



Ανάλυση (1, m) Indexing

- Η κατανομή πιθανότητας του initial probe για τους πελάτες είναι ομοιόμορφη
- Data: το μέσο μέγεθος του αρχείου
- C: η coarseness του index attribute
- Ο index έχει δείκτες μόνο στην πρώτη εμφάνιση εγγραφής με συγκεκριμένη τιμή για το attribute
- Άρα, κατασκευάζουμε ευρετήριο μόνο για (Data/C) data buckets
- n, η χωρητικότητα ενός bucket, δηλ., ο αριθμός των ζευγών (attribute_value,offset) που μπορεί να στεγάσει
- k: ο αριθμός των επιπέδων του ευρετηρίου
- Index: ο αριθμός των buckets του

Ανάλυση (1, m) Indexing

Όταν το δένδρο είναι πλήρως ισοζυγισμένο

$$k = \left\lceil \log_n \left(\frac{Data}{C} \right) \right\rceil \qquad Index = \sum_{i=0}^{k-1} n^i$$

Latency: The *probe wait* is $\frac{1}{2} * (Index + \frac{Data}{m})$ and the *bcast wait* is $\frac{1}{2} * ((m * Index) + Data) + C$. Hence, the latency is

$$\frac{1}{2} * \left(Index + \frac{Data}{m} \right) + \frac{1}{2} * ((m * Index) + Data) + C,$$

i.e.,

$$\frac{1}{2} * \left((m+1) * Index + \left(\frac{1}{m} + 1 \right) * Data \right) + C.$$

Ανάλυση (1, m) Indexing

Tuning time: $1 + k + C$

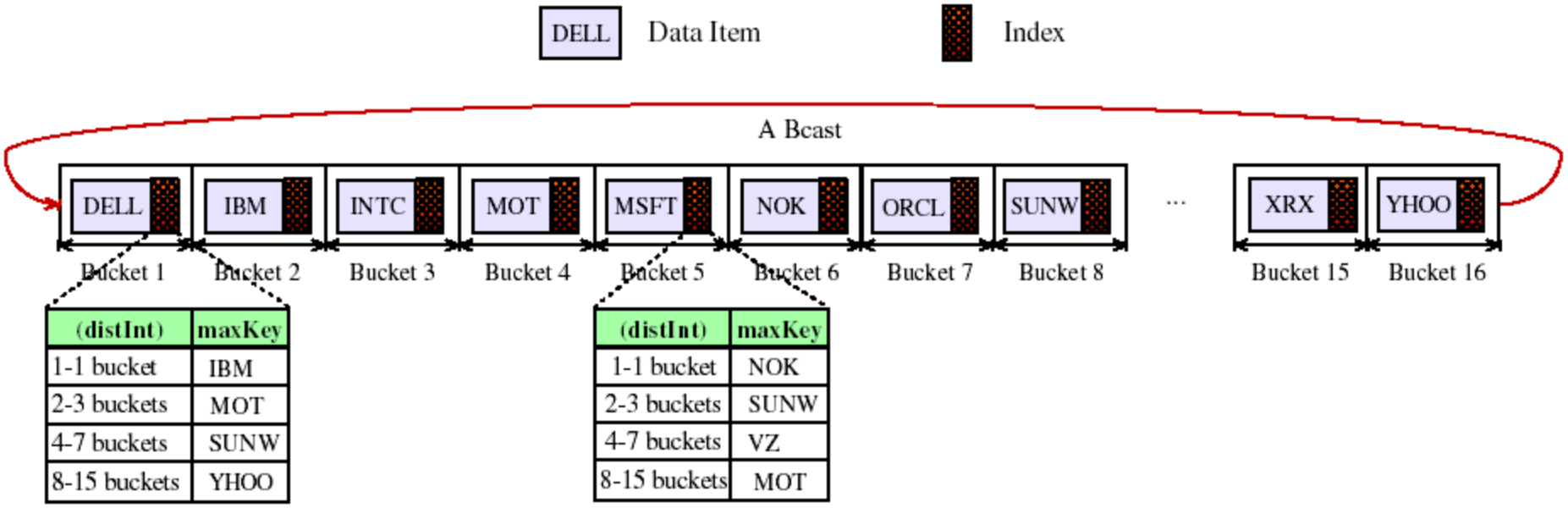
- **Βέλτιστο m για ελαχιστοποίηση Latency:**
  - Παραγώγιση την εξίσωση της Latency ως προς m
  - Εξίσωση με 0
  - Επίλυση ως προς m

$$m^* = \sqrt{\frac{Data}{Index}}$$

Exponential Index (1/2)

- Data buckets
  - Data part
  - Index table
    - Index entries:
      - κάθε entry indexes ένα segment από buckets και έχει τη μορφή **{distInt, maxKey}**
        - » distInt: καθορίζει την απόσταση των buckets από το τρέχον bucket (μετρημένο σε αριθμό buckets)
        - » maxKey: είναι η τιμή του μέγιστου κλειδιού αυτών των buckets
      - Τα μεγέθη των segments αυξάνουν εκθετικά (δυνάμεις του 2). Η i-οστή entry περιγράφει το segment των buckets τα οποία είναι σε απόσταση 2^{i-1} έως $2^{i}-1$
      - Προφανώς οι τιμές distInt δεν χρειάζονται αφού μπορούν εύκολα να συναχθούν

Exponential Index (2/2)



(distInt)	maxKey
1-1 bucket	IBM
2-3 buckets	MOT
4-7 buckets	SUNW
8-15 buckets	YHOO

(distInt)	maxKey
1-1 bucket	NOK
2-3 buckets	SUNW
4-7 buckets	VZ
8-15 buckets	MOT

Κινητός και Διάχυτος Υπολογισμός (Mobile & Pervasive Computing)

Δημήτριος Κατσαρός

Χειμώνας 2018

Διάλεξη 8η

Περιεχόμενα

- **Ευρετήρια**

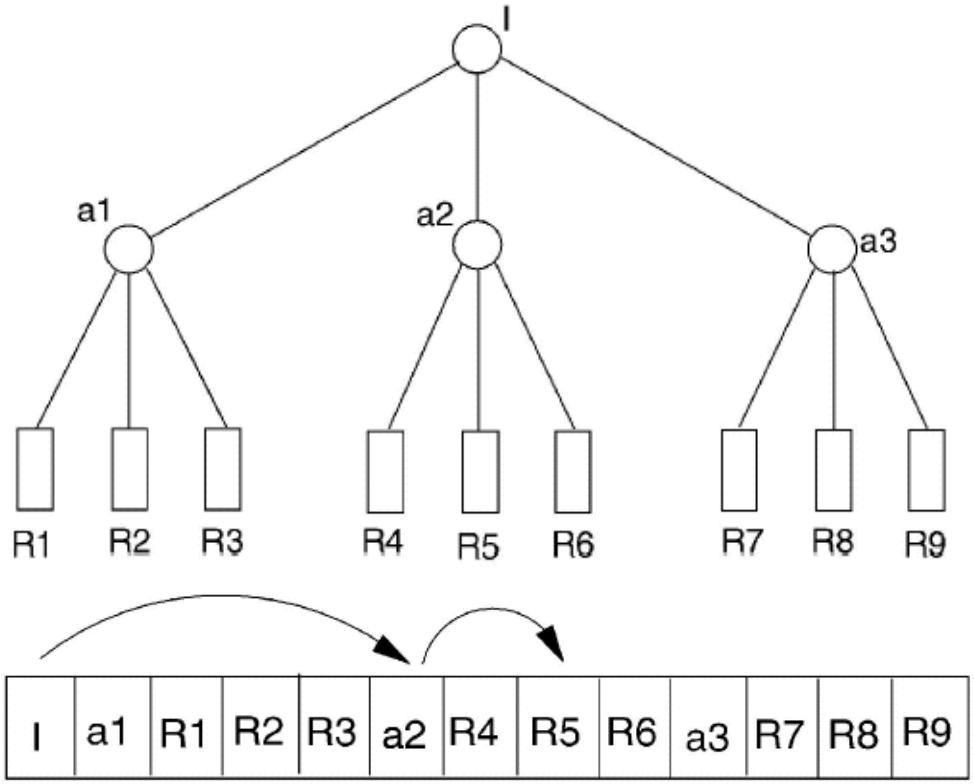
Κυρτά πρότυπα προσπέλασης

- Τι γίνεται όταν το πρότυπο προσπέλασης στις εγγραφές δεν είναι ομοιόμορφο?
- Τα ευρετήρια που παρουσιάσαμε είναι κατάλληλα για ομοιόμορφα πρότυπα προσπέλασης
- n: αριθμός εγγραφών
- R_i: i-οστή εγγραφή
- $Pr(R_i)$: πιθανότητα προσπέλασης εγγραφής R_i
- $I_{pb}(R_i)$: αριθμός index nodes μέχρι να φτάσουμε στην εγγραφή ή index node R_i
- a_i: αναπαριστά έναν index node
- $d(a_i)$: fanout του κόμβου a_i
- $Path(R_i)$: σύνολο των index nodes από τη ρίζα μέχρι την εγγραφή R_i
- $f(a_i)$: κόστος "ενεργοποίησης" index node a_i. Συνήθως $f(a_i)=d(a_i)$.

Ισοζυγισμένα δενδρικά ευρετήρια

Έστω ότι το μέσο κόστος εντοπισμού μιας εγγραφής εκφάζεται ως:

$$\sum_{1 \le i \le n} Pr(R_i) \sum_{a_j \in Path(R_i)} f(a_j)$$

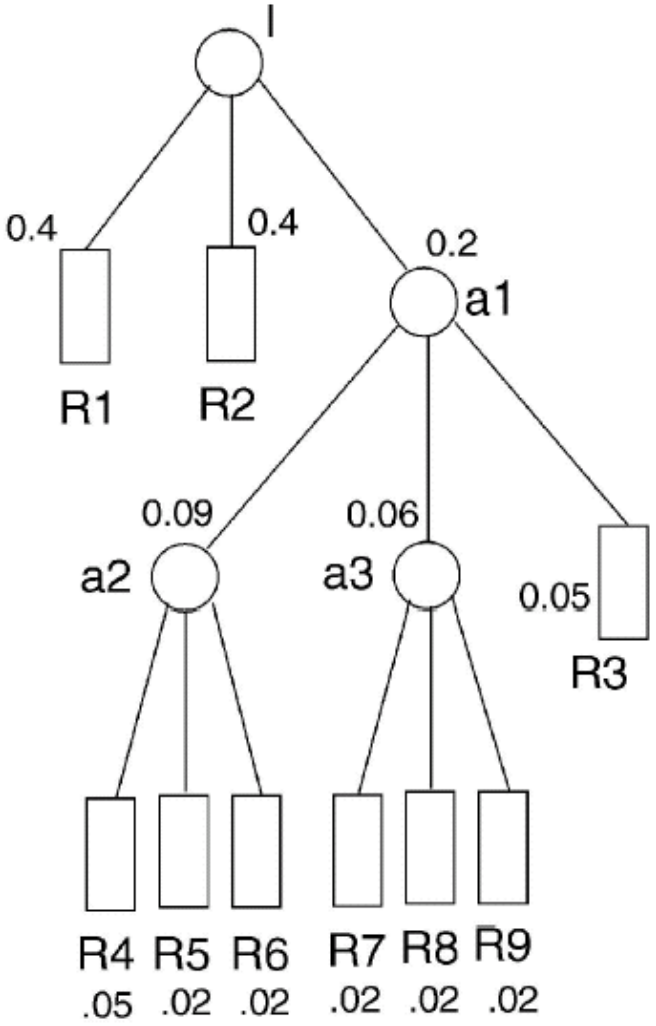


Το διπλανό ευρετήριο είναι βέλτιστο για ομοιόμορφη πιθανότητα προσπέλασης.

Μέσο κόστος: $C(T^B_{d=3}) = 6$

Μη ισοζυγισμένα δενδρικά ευρετήρια σταθερού fanout [π.χ., d=3]

Data record	R1	R2	R3	R4	R5	R6	R7	R8	R9
$Pr(R_i)$	0.4	0.4	0.05	0.05	0.02	0.02	0.02	0.02	0.02
$I_{pb}(R_i)$ in $T^B_{d=3}$	2	2	2	2	2	2	2	2	2
$I_{pb}(R_i)$ in $T^I_{d=3}$	1	1	2	3	3	3	3	3	3



$C(T^I_{d=3})$ = 0.4*3+0.4*3+ **[για R_1 και R_2]**

0.05*(3+3)+ **[για R_3]**

0.05*(3+3+3)+ **[για R_4]**

5*0.02*(3+3+3) **[για υπόλοιπες]**

= 0.4*6+0.05*15+45*0.02 = **4.28**

Σειρά εκπομπής στο ασύρματο κανάλι

I	R1	R2	a1	R3	a2	R4	R5	R6	a3	R7	R8	R9

Αλγόριθμος κατασκευής $T^I_{d=j}$

Algorithm CF: Use access frequencies to build an index tree with a fixed fanout d.

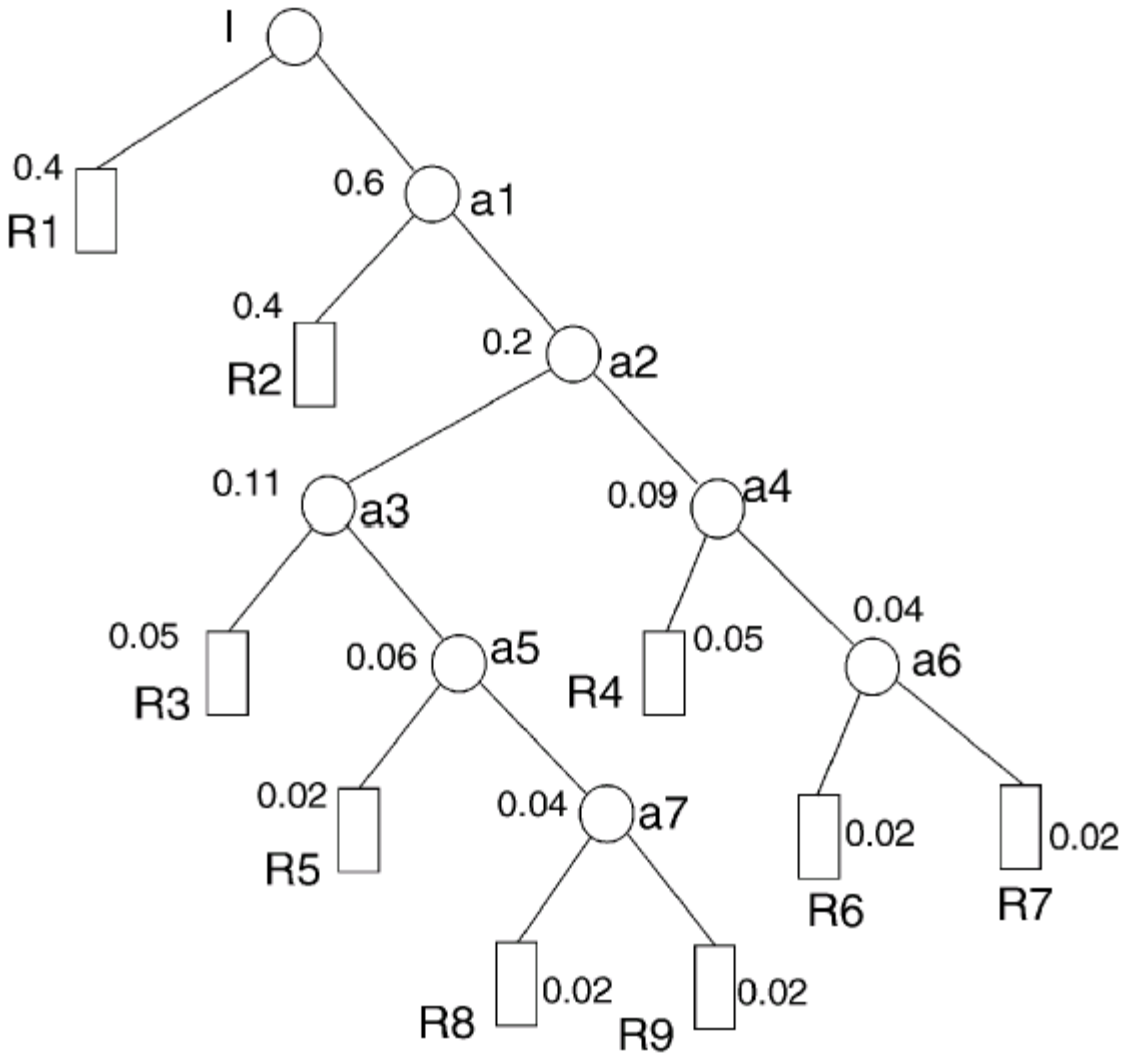
Step 1: Initially, we have a forest of n subtrees, each of which is a single node labeled with the corresponding access frequency.

Step 2: Attach the d subtrees with the smallest labels to a new node. Label the resulting subtree with the sum of all labels from its d child subtrees.

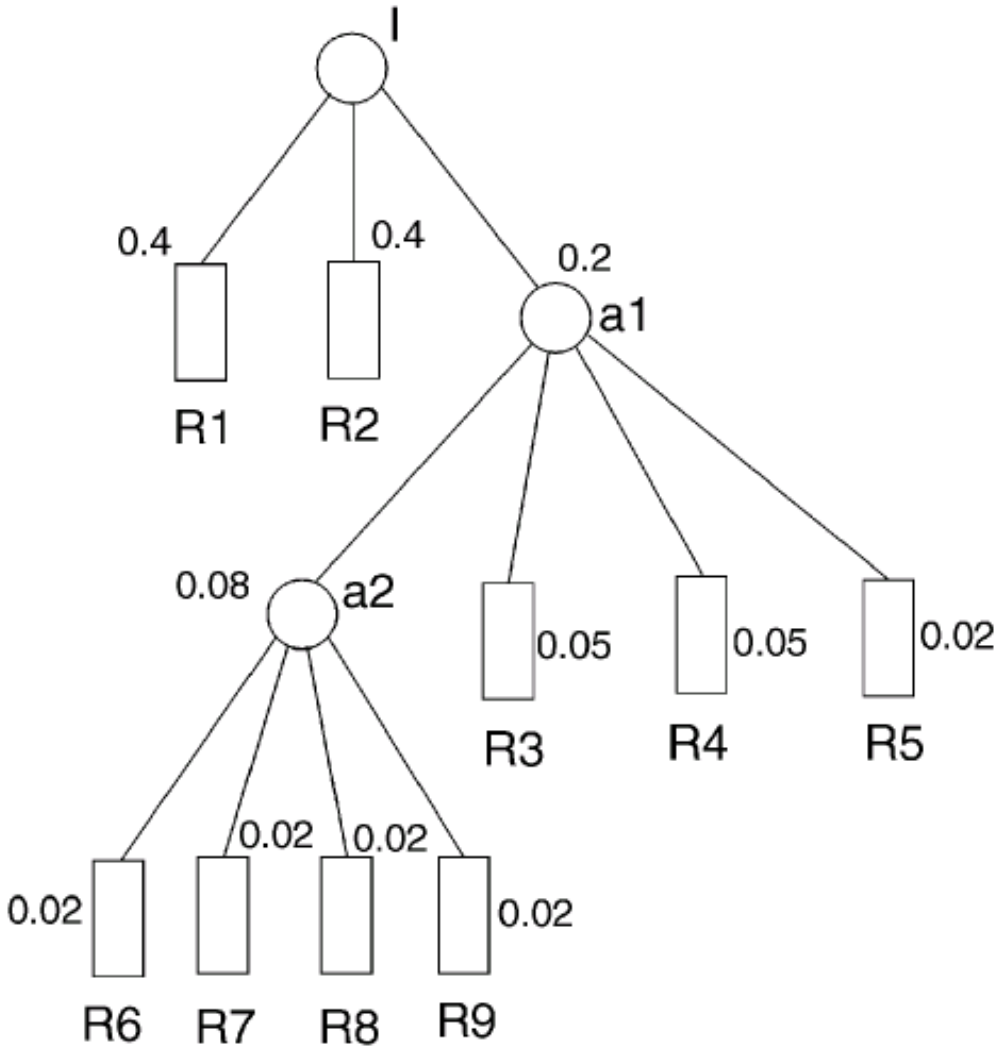
Step 3: $n = n - d + 1$. (Then, n is the number of remaining subtrees.) If $n = 1$ stop else goto Step 2.

Κατασκευάζει ένα Huffman δένδρο με fanout = d.
Στο βασικό Huffman δένδρο, fanout = 2.

Μη ισοζυγισμένα δενδρικά ευρετήρια σταθερού fanout [π.χ., d=2]



Μη ισοζυγισμένα δενδρικά ευρετήρια σταθερού fanout [π.χ., d=4]



Μη ισοζυγισμένα δένδρα σταθερού fanout

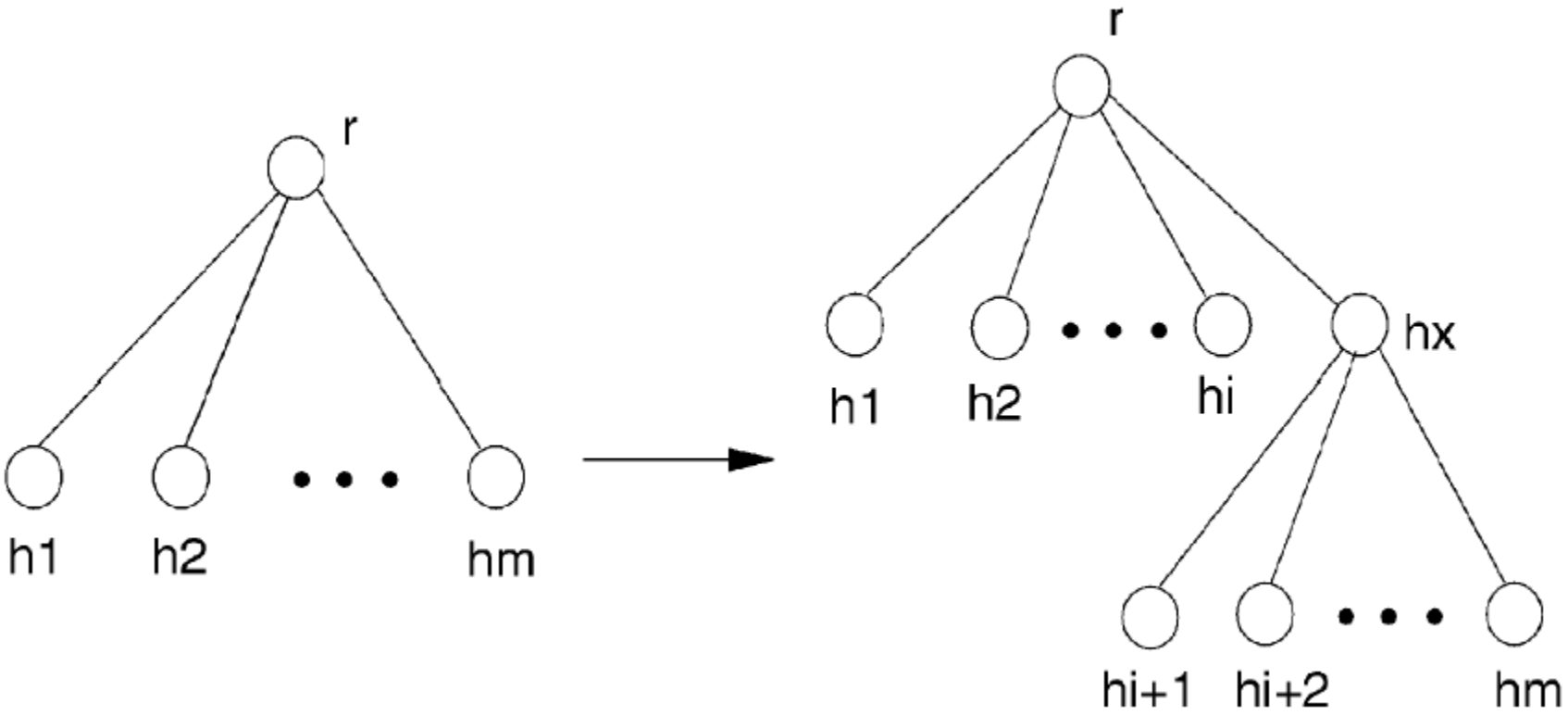
- Δεν υπάρχει **μονοτονική σχέση**, ούτε αύξουσα ούτε φθίνουσα, για τις τιμές κόστους $C(T^I_{d=j})$ σε σχέση με το επιτρεπτό fanout j.
- Αυτό το γεγονός, σε συνδυασμό με την ιεραρχική φύση της κατασκευής, υπονοεί ότι, επιτρέποντας μεταβλητά fanouts στους κόμβους του δένδρου, μπορούμε να ελαττώσουμε ακόμα περισσότερο το μέσο κόστος εντοπισμού των εγγραφών

Βασική ιδέα (1/2)

Lemma 1. *Suppose that node r has m child nodes, h_1, h_2, ..., and h_m, which are sorted according to descending order of $Pr(h_j)$, $1 \leq j \leq m$, i.e., $Pr(h_j) \geq Pr(h_k)$ if and only if $j \leq k$. Then, the average cost of index probes can be reduced by grouping nodes h_{i+1}, h_{i+2}, ..., and h_m and attaching them under a new child node if and only if*

$$(m - i - 1) \sum_{1 \leq j \leq i} Pr(h_j) > \sum_{i+1 \leq j \leq m} Pr(h_j).$$

Βασική ιδέα (2/2)



Μη ισοζυγισμένα δένδρα μεταβλητού fanout T^I_V : Αλγόριθμος κατασκευής

Algorithm VF:

Step 1: Assume that R_1, R_2, ..., and R_n have been sorted according to descending order of $Pr(R_j)$, $1 \leq j \leq n$, i.e., $Pr(R_j) \geq Pr(R_k)$ iff. $j \leq k$.

Step 2: *Partition* $(R_1, R_2, \ldots, R_n)$.

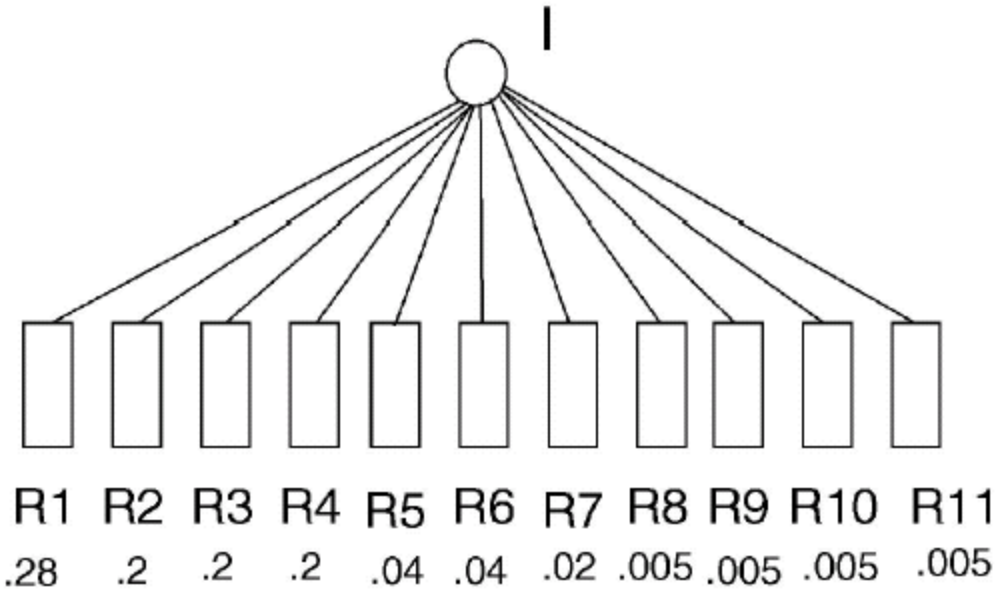
Step 3: Report the resulting index tree.

Μη ισοζυγισμένα δένδρα μεταβλητού fanout T^I_V : Αλγόριθμος κατασκευής

Procedure *Partition* $(h_1, h_2, \ldots, h_m)$:

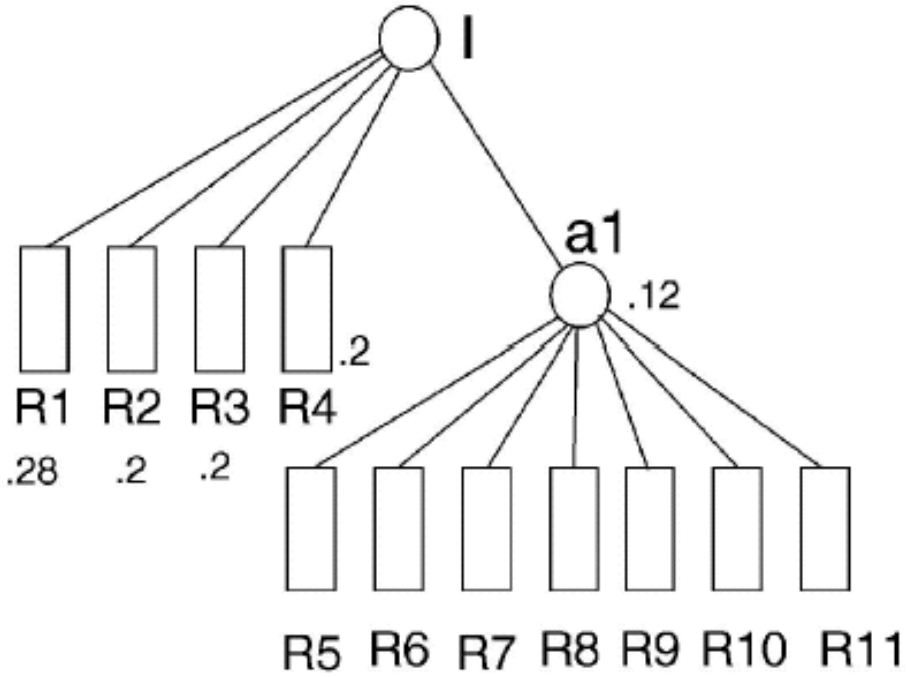
1. Let $y(i) = (m-i-1)\sum_{1\le j\le i} Pr(h_j) - \sum_{i+1\le j\le m} Pr(h_j)$. Determine i^* such that $y(i^*) = \max_{\forall i\in\{1,m-2\}} \{y(i)\}$.
2. If $y(i^*) \le 0$, then return.
3. Attach nodes $h_{i^*+1}, h_{i^*+2}, \ldots, h_m$ under a new index node hx in the index tree.
4. *Partition* $(h_{i^*+1}, h_{i^*+2}, \ldots, h_m)$.
5. Insert hx into the ordered list $(h_1, h_2, \ldots, h_{i^*})$ and relabel them as $(h_1, h_2, \ldots, h_{i^*+1})$ according to descending order of $Pr(h_j)$, $1 \le j \le i^*+1$.
6. *Partition* $(h_1, h_2, \ldots, h_{i^*+1})$.
7. Return.

Παράδειγμα κατασκευής T^I_V (1/6)



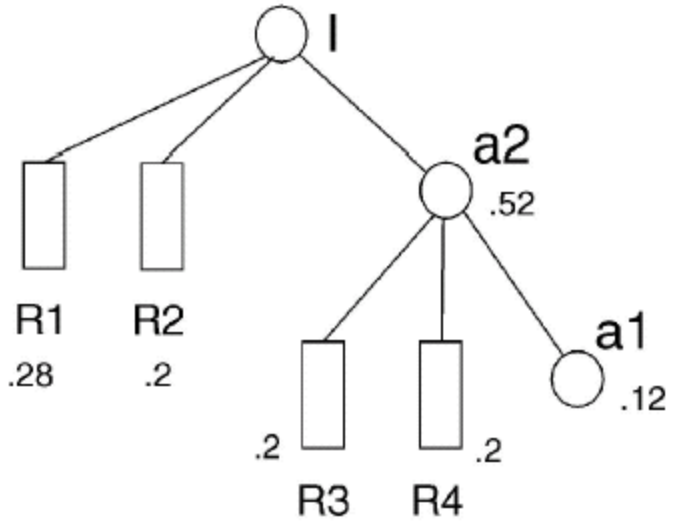
i	1	2	3	4	5	6	7	8	9
$(10-i)\sum_{1<j<i} Pr(h_j)$	9*.208	8*0.48	7*0.68	6*0.88	5*0.92	4*0.96	3*0.98	2*0.985	0.99
$\sum_{i+1<j<11} Pr(h_j)$	0.72	0.52	0.32	0.12	0.08	0.04	0.02	0.015	0.01
$y(i)$	1.8	3.32	4.44	5.16	4.52	3.8	2.92	1.955	0.98

Παράδειγμα κατασκευής T^I_V (2/6)

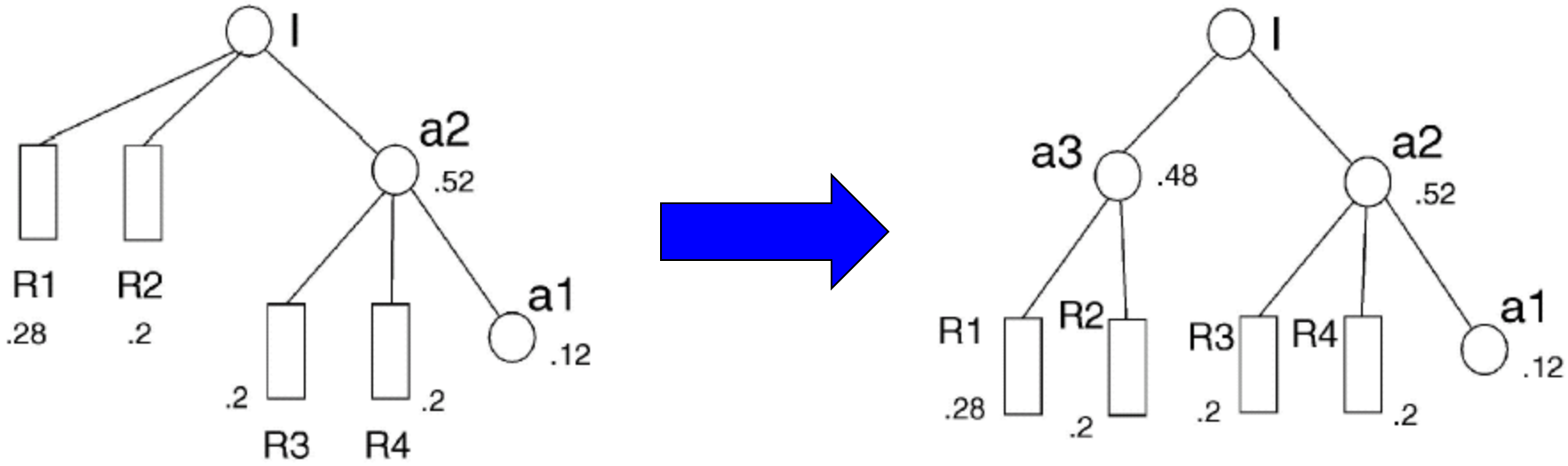


Παράδειγμα κατασκευής T^I_V (3/6)

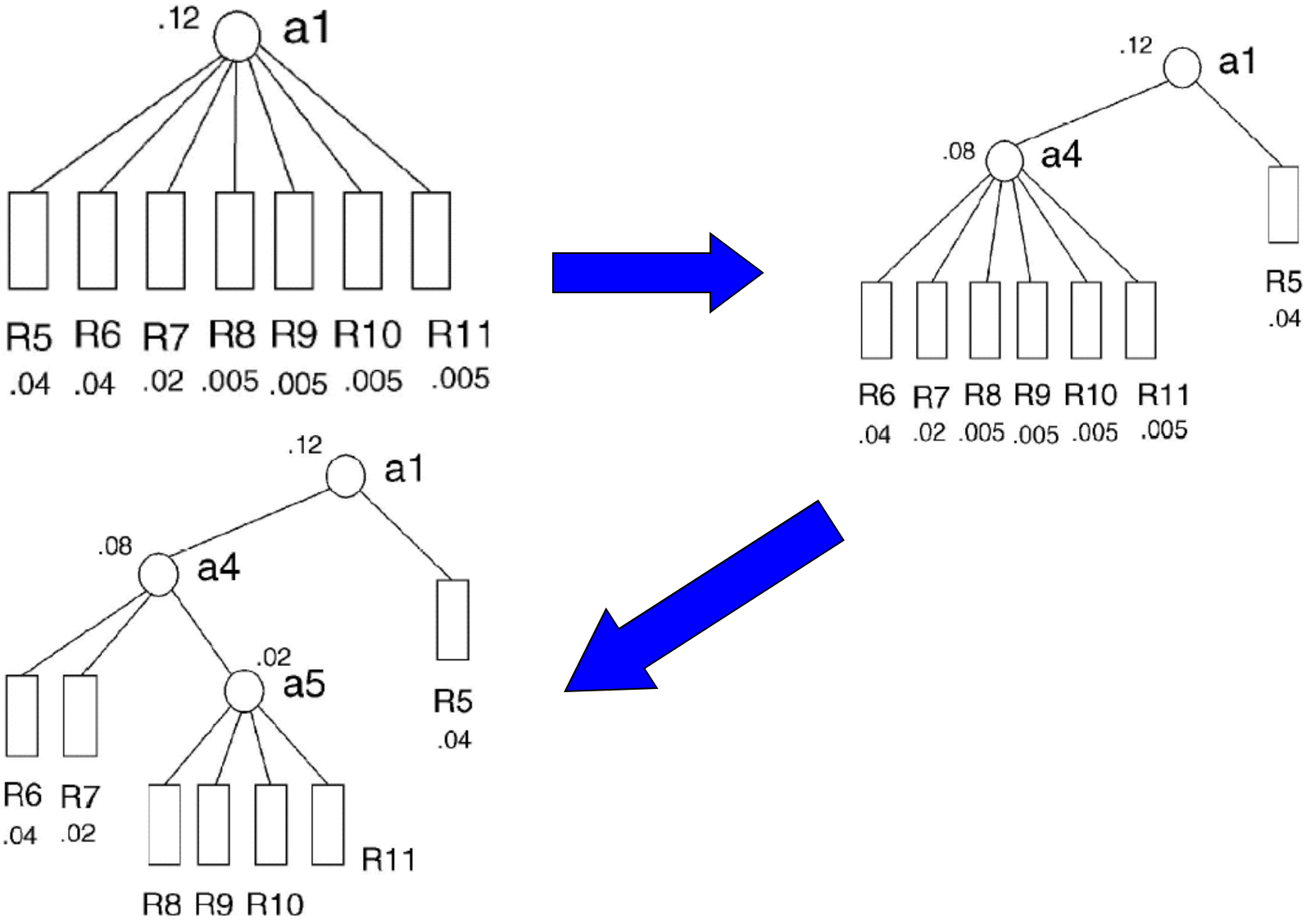
i	1	2	3
$(4-i)\sum_{1<j<i} Pr(h_j)$	3*0.28	2*0.48	0.68
$\sum_{i+1\leq j\leq 5} Pr(h_j)$	0.72	0.52	0.32
$y(i)$	0.12	0.44	0.32



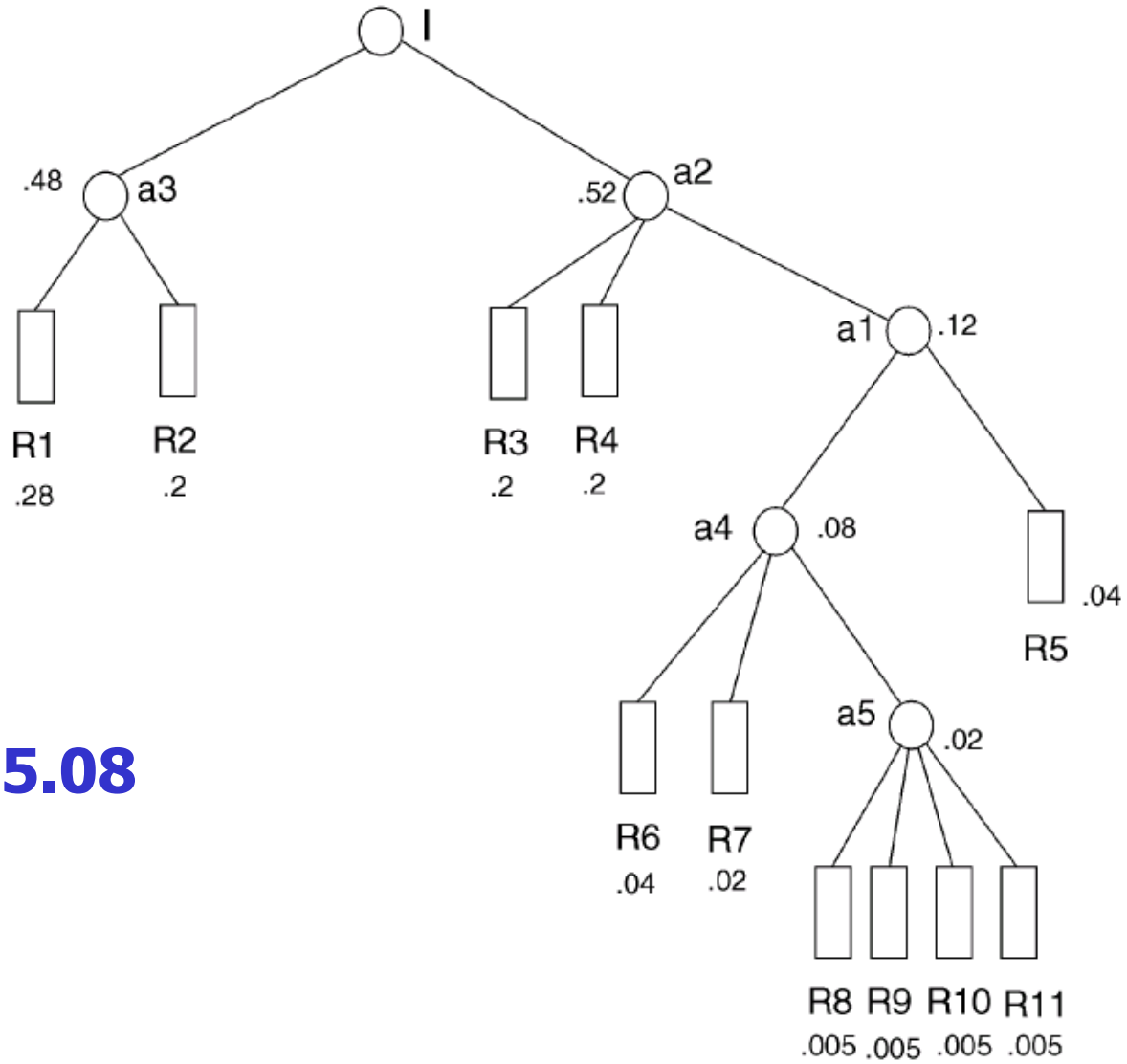
Παράδειγμα κατασκευής T^{I}_{V} (4/6)



Παράδειγμα κατασκευής T^I_V (5/6)

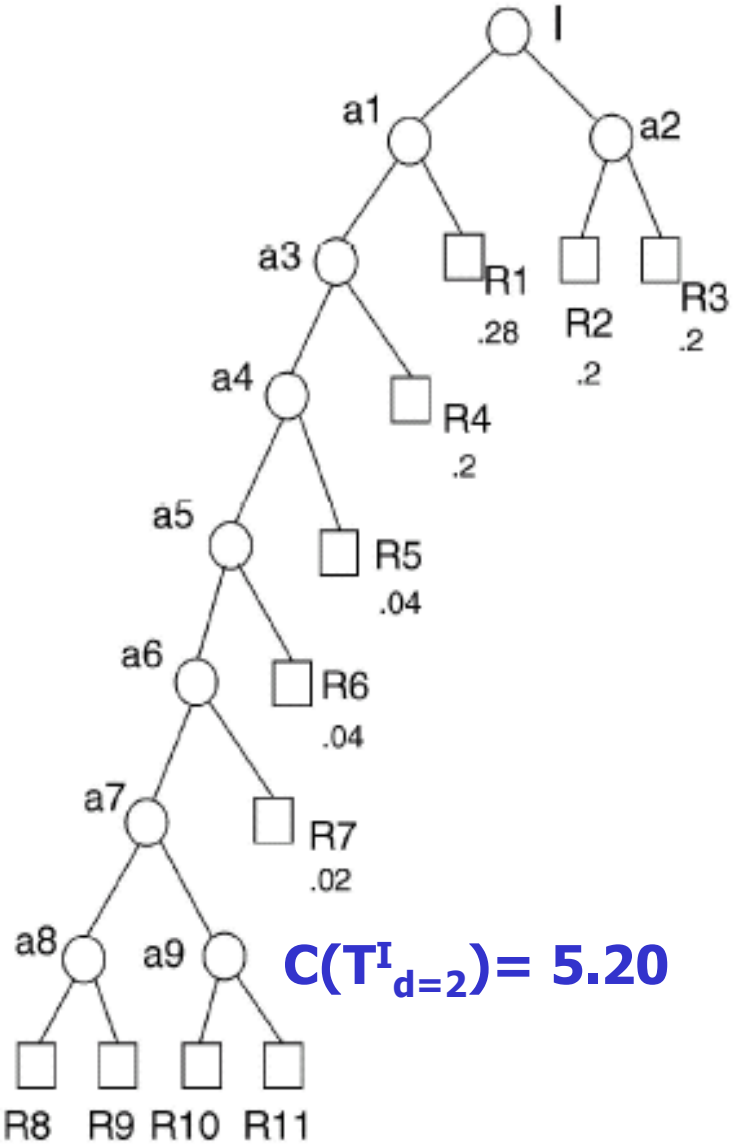


Παράδειγμα κατασκευής T^I_V (6/6)

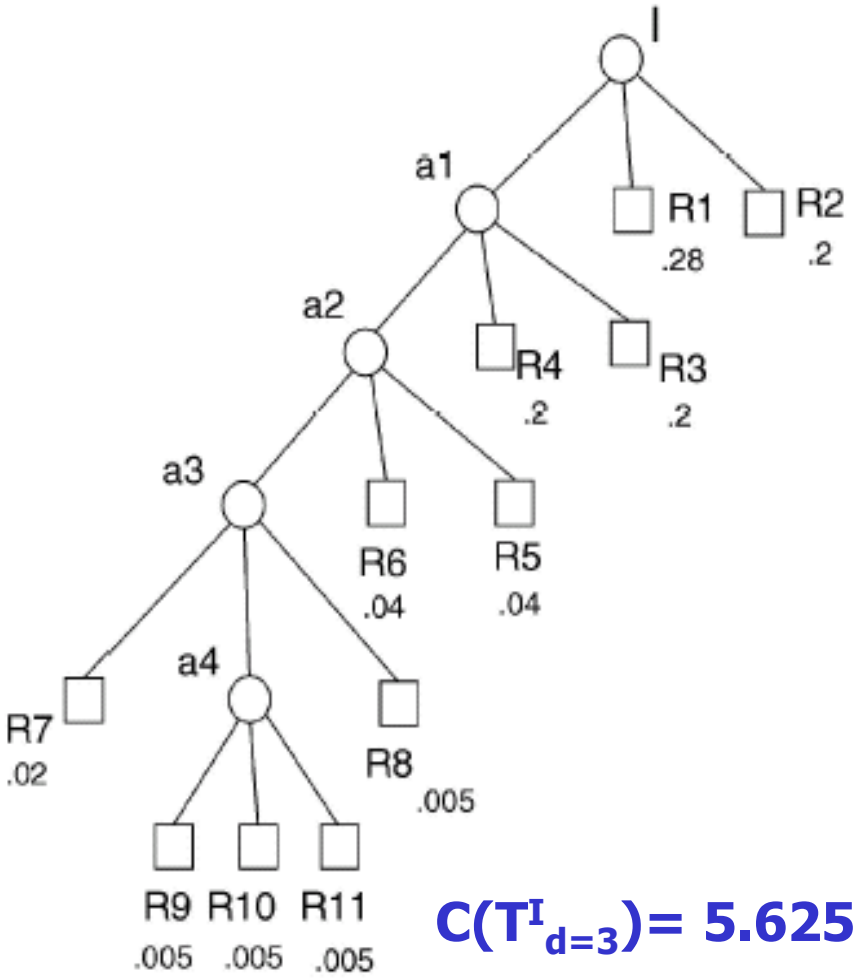


$C(T^I_V)$= 5.08

Αντιδιαστολή T^I_V με $T^I_{d=2}$ και $T^I_{d=3}$



$C(T^I_{d=2}) = 5.20$

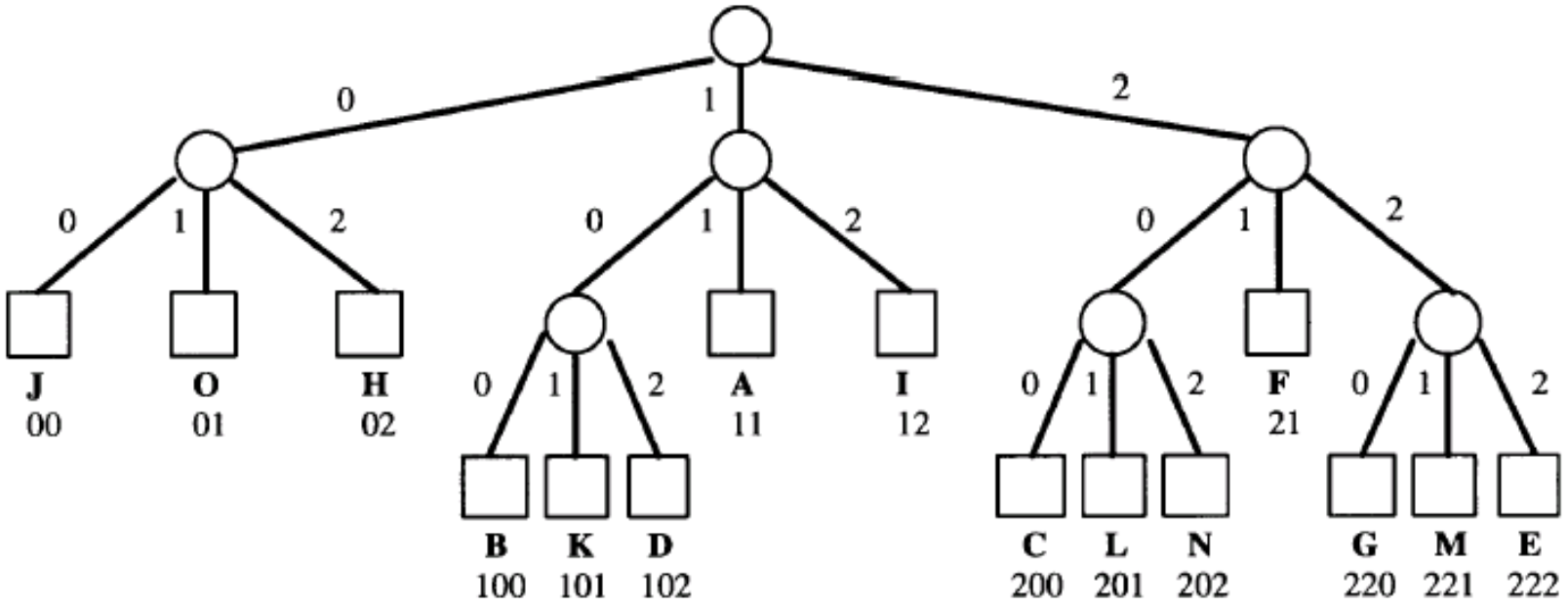


$C(T^I_{d=3}) = 5.625$

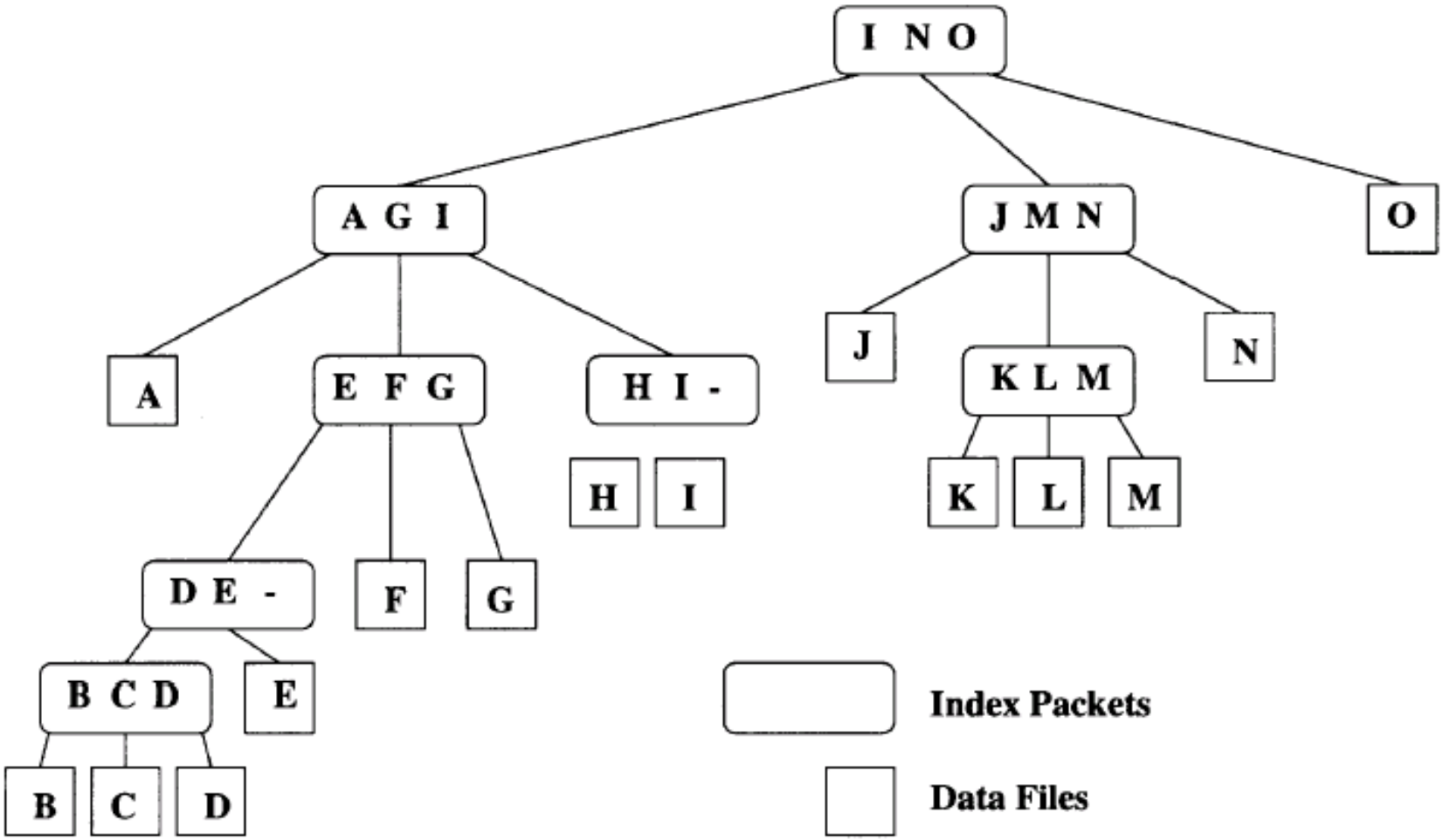
"Αλφαβητικά" δενδρικά ευρετήρια

- Ομοιόμορφη προσπέλαση & ταξινομημένα κλειδιά
  - (1,m) indexing
  - Distributed indexing
  - Exponential indexing
- Μη ομοιόμορφη προσπέλαση & μη ταξινομημένα κλειδιά
  - $T^I_{d=j}$
  - T^I_V
- **Μη ομοιόμορφη προσπέλαση & ταξινομημένα κλειδιά ????**

Το πρόβλημα των Huffman δένδρων



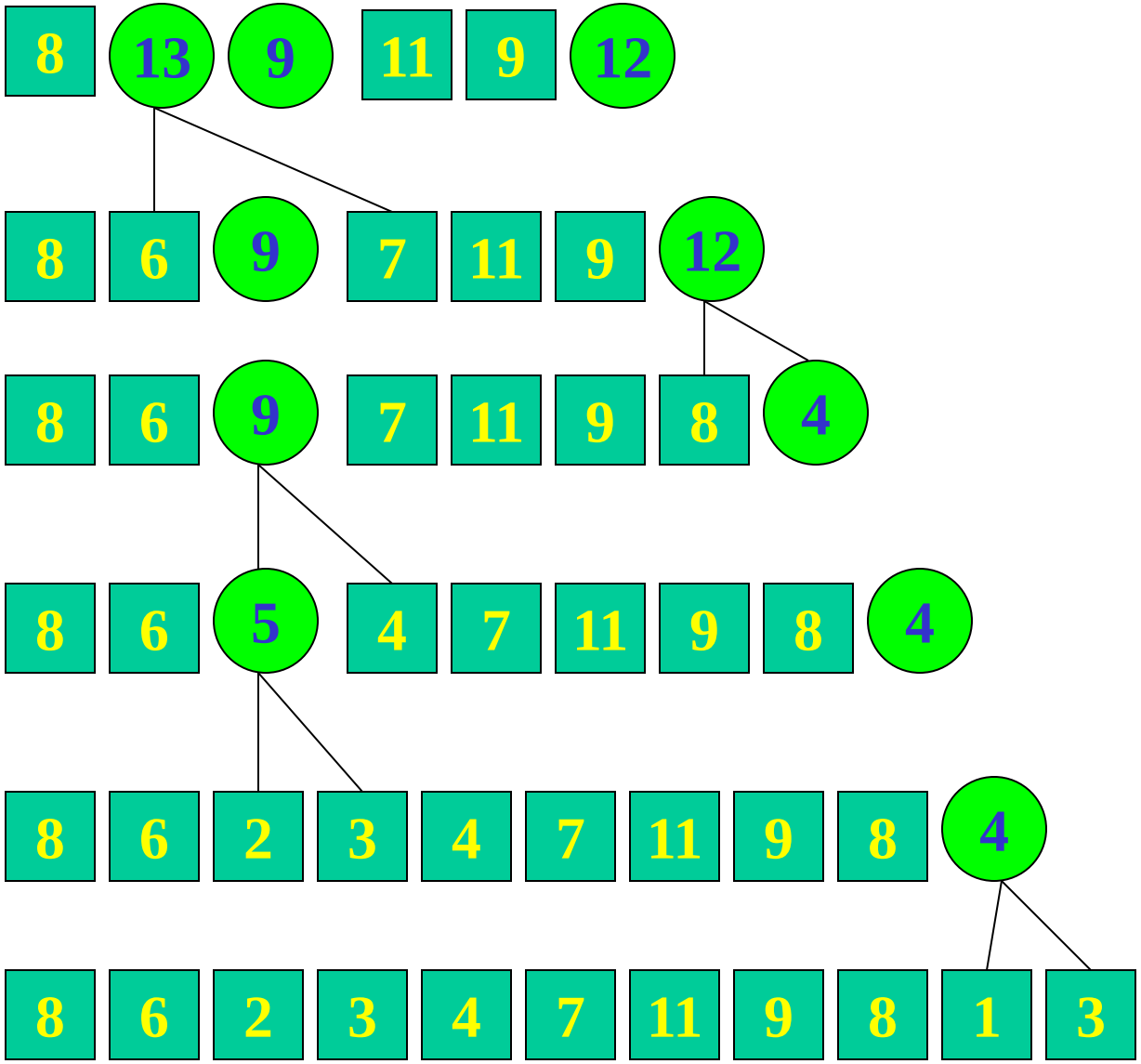
Τα Alphabetic δένδρα



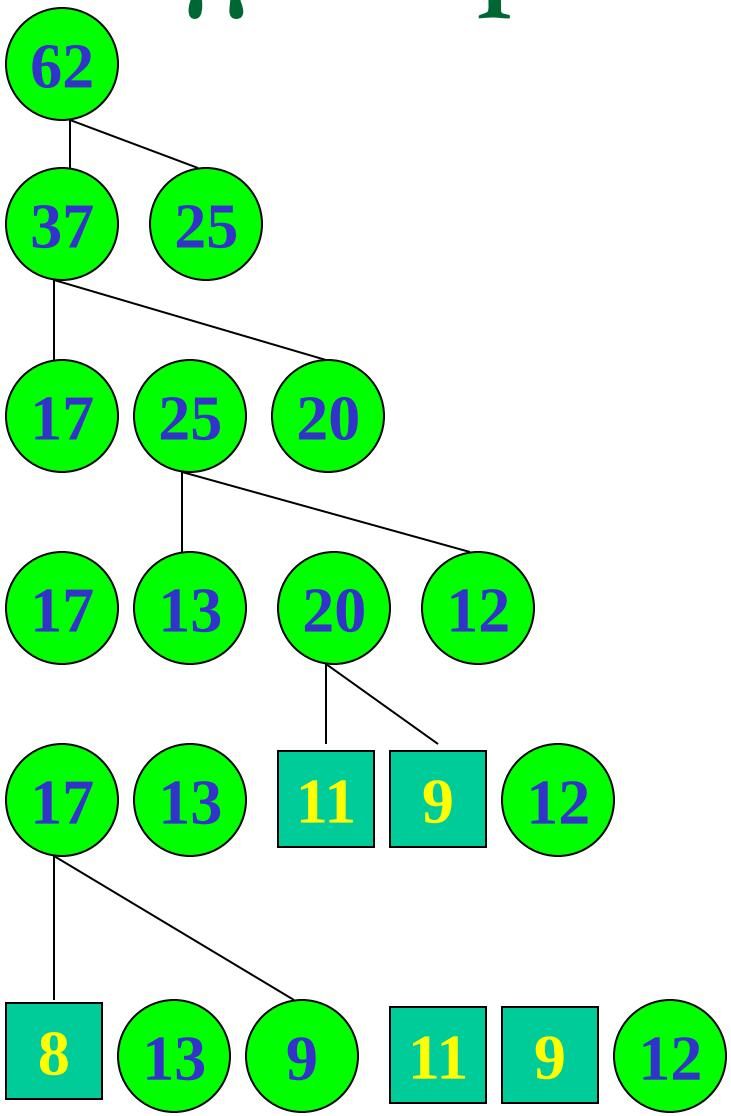
Παράδειγμα κατασκευής δυαδικού αλφαβητικού δένδρου

A	B	C	D	E	F	G	H	I	J	K
8	6	2	3	4	7	11	9	8	1	3

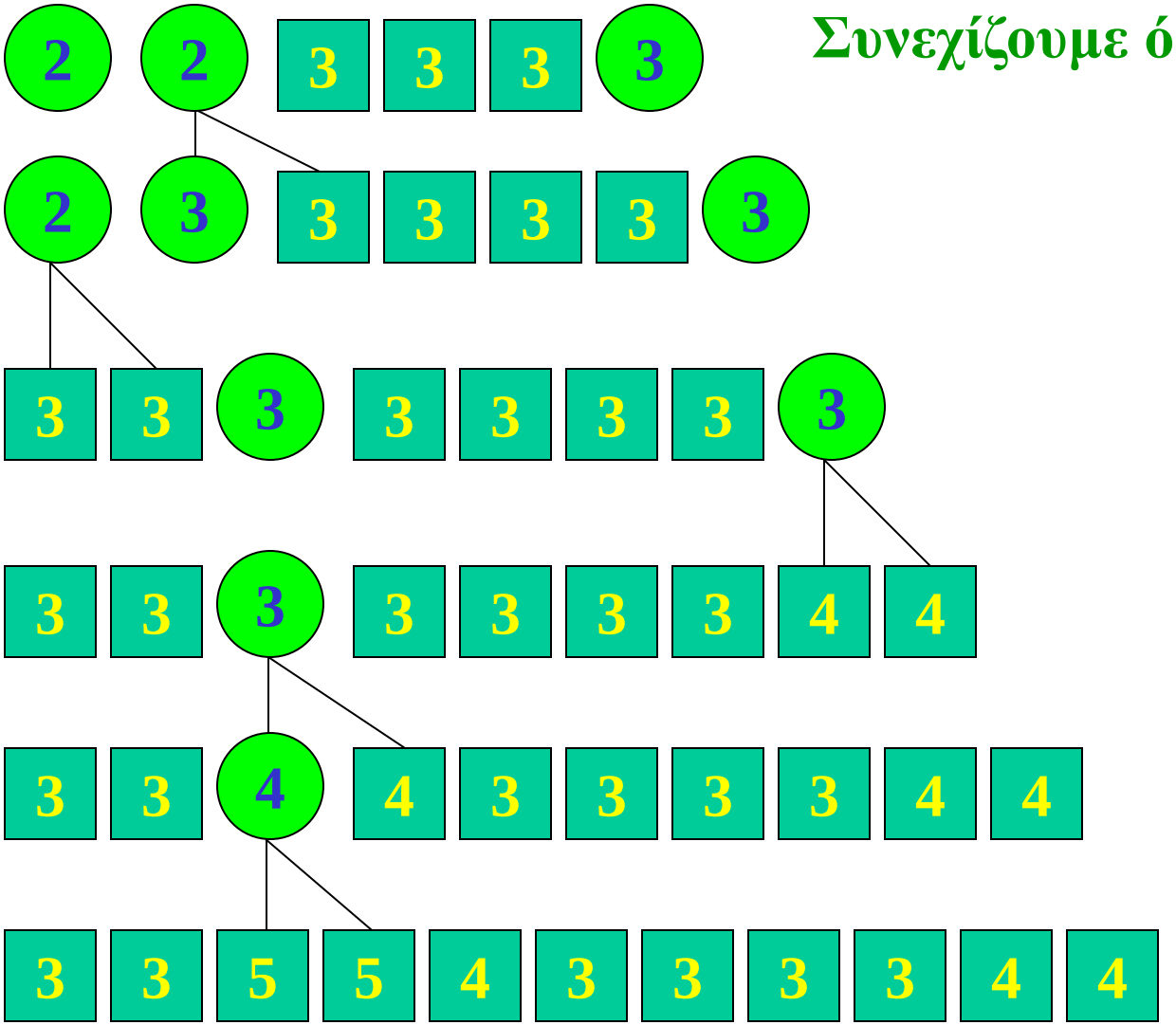
Παράδειγμα Alphabetic tree (1/4)



Παράδειγμα Alphabetic tree (2/4)



Παράδειγμα Alphabetic tree (3/4)



Συνεχίζουμε όμοια. Τελικά …

Ακολουθία επιπέδων

Alphabetic tree (4/4)