ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΛΥΣΗ ΣΤΗ ΔΕΥΤΕΡΗ ΑΣΚΗΣΗ

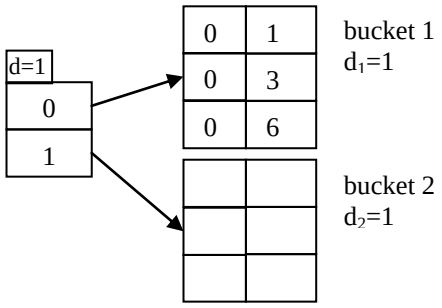
ΜΑΘΗΜΑ **ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ**
ΑΚΑΔ. ΕΤΟΣ 2009-10

ΔΙΔΑΣΚΟΝΤΕΣ Ιωάννης Βασιλείου *Καθηγητής*, Τομέας Πληροφορικής
Τιμολέων Σελλής *Καθηγητής*, Τομέας Πληροφορικής

Ερώτημα 1.
Θεωρείστε επεκτατό κατακερματισμό όπου κάθε κάδος χωρά μέχρι 3 εγγραφές. Χρησιμοποιείστε τα πρώτα bits.
(α) Δώστε το ευρετήριο μετά την εισαγωγή των παρακάτω κλειδιών: 1, 3, 6, 14, 15, 17, 21, 23, 24, 25, 29, 31. Προσδιορίστε το ολικό και το τοπικό βάθος.
(β) Δώστε το ευρετήριο μετά την εισαγωγή των παρακάτω κλειδιών: 17, 15, 14, 6, 3, 1, 31, 29, 25, 24, 23, 21. Προσδιορίστε το ολικό και το τοπικό βάθος.

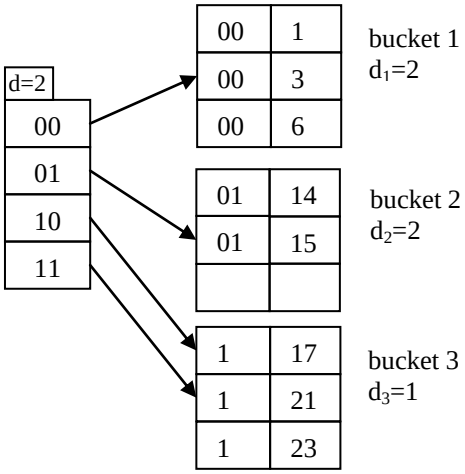
ΛΥΣΗ
(α) Αρχικά d=0 και d_1=0. Ο πίνακας διευθύνσεων και το πρώτο bucket είναι άδεια.
Μετά την εισαγωγή του πρώτου κλειδιού, γίνεται d=1 και $d_1=d_2=1$. Ακολούθως εισάγονται τα 3 και 6.

key	H(key)	Prefix
1	00001	0
3	00011	0
6	00110	0
14	01110	0
15	01111	0
17	10001	1
21	10101	1
23	10111	1
24	11000	1
25	11001	1
29	11101	1
31	11111	1



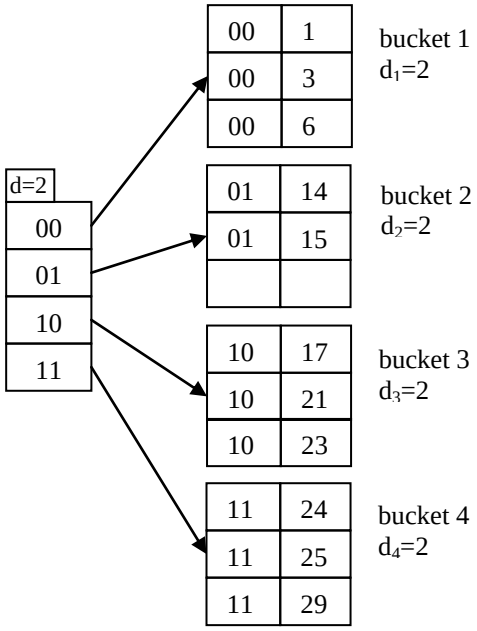
Με τη εισαγωγή του κλειδιού 14, το πρώτο bucket είναι γεμάτο. Γίνεται d=2, ο πρώτος κάδος σπάει σε δύο $d_1=d_2=2$. Ο άλλος κάδος παραμένει με ίδιο τοπικό βάθος d_3=1.

key	H(key)	Prefix
1	00001	00
3	00011	00
6	00110	00
14	01110	01
15	01111	01
17	10001	10
21	10101	10
23	10111	10
24	11000	11
25	11001	11
29	11101	11
31	11111	11



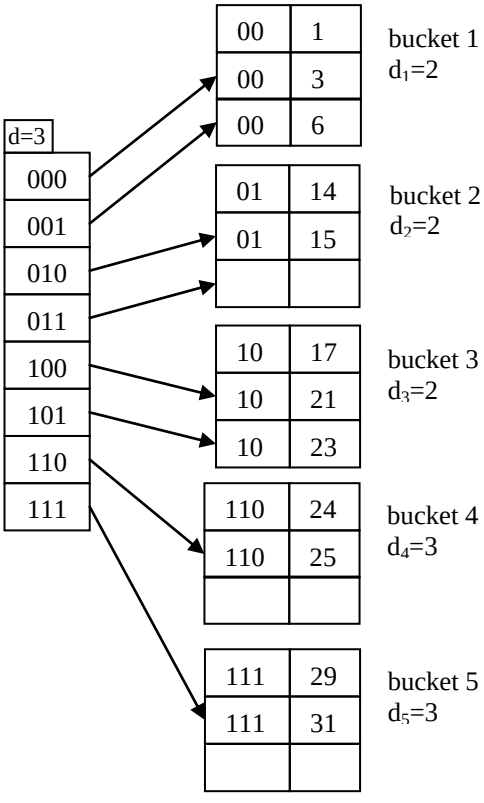
Με τη εισαγωγή του κλειδιού 24, το τελευταίο bucket είναι γεμάτο, και σπάει σε δυο: d_3=2, d_4=2.

key	H(key)	Prefix
1	00001	00
3	00011	00
6	00110	00
14	01110	01
15	01111	01
17	10001	10
21	10101	10
23	10111	10
24	11000	11
25	11001	11
29	11101	11
31	11111	11



Με τη εισαγωγή του κλειδιού 31, το τελευταίο bucket είναι γεμάτο, και γίνεται το d=3, d_4=3, νέο d_5=3.

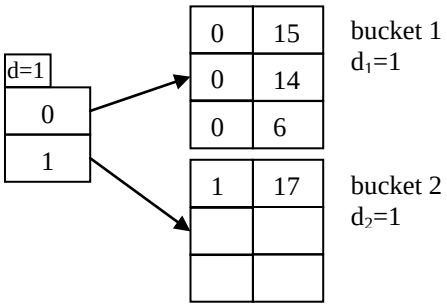
key	H(key)	Prefix
1	00001	000
3	00011	000
6	00110	001
14	01110	011
15	01111	011
17	10001	100
21	10101	101
23	10111	101
24	11000	110
25	11001	110
29	11101	111
31	11111	111



Ολικό βάθος **d=3**. Τοπικά βάθη: **d_1= d_2= d_3=2**, **d_4= d_5=3**.

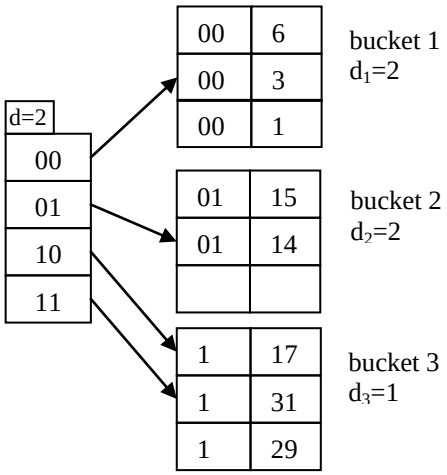
(β) Αρχικά d=0 και d_1=0. Ο πίνακας διευθύνσεων και το πρώτο bucket είναι άδεια.
Μετά την εισαγωγή του πρώτου κλειδιού, γίνεται d=1 και d_1=d_2=1. Ακολούθως εισάγονται τα 15 ως 6.

key	H(key)	Prefix
17	10001	1
15	01111	0
14	01110	0
6	00110	0
3	00011	0
1	00001	0
31	11111	1
29	11101	1
25	11001	1
24	11000	1
23	10111	1
21	10101	1



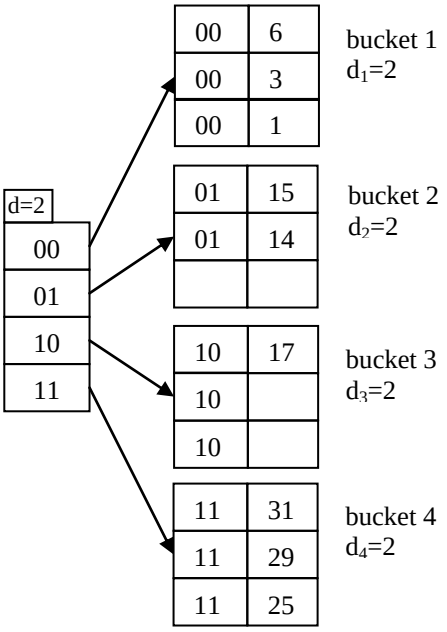
Με τη εισαγωγή του κλειδιού 3, το πρώτο bucket είναι γεμάτο. Γίνεται d=2, ο πρώτος κάδος σπάει σε δύο d_1=d_2=2, ενώ ο άλλος παραμένει με ίδιο τοπικό βάθος d_3=1.

key	H(key)	Prefix
17	10001	10
15	01111	01
14	01110	01
6	00110	00
3	00011	00
1	00001	00
31	11111	11
29	11101	11
25	11001	11
24	11000	11
23	10111	10
21	10101	10



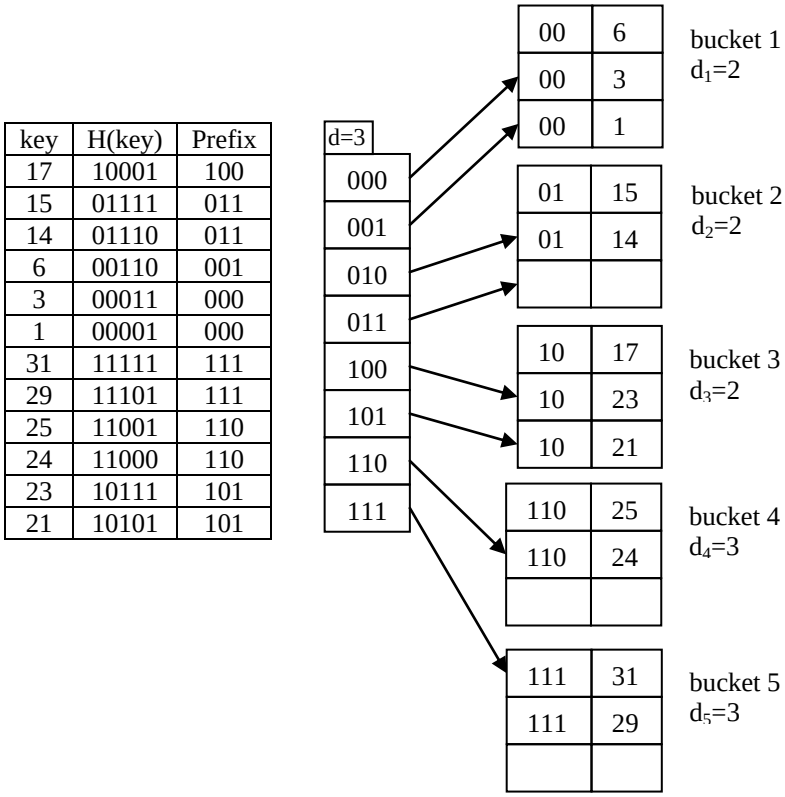
Με τη εισαγωγή του κλειδιού 25, το τελευταίο bucket είναι γεμάτο, και σπάει σε δύο: d_3=2, d_4=2.

key	H(key)	Prefix
17	10001	10
15	01111	01
14	01110	01
6	00110	00
3	00011	00
1	00001	00
31	11111	11
29	11101	11
25	11001	11
24	11000	11
23	10111	10
21	10101	10



Με τη εισαγωγή του κλειδιού 24, το τελευταίο bucket είναι γεμάτο, και σπάει σε δύο. Γίνεται το d=3, d_4=3, νέο d_5=3.

key	H(key)	Prefix
17	10001	100
15	01111	011
14	01110	011
6	00110	001
3	00011	000
1	00001	000
31	11111	111
29	11101	111
25	11001	110
24	11000	110
23	10111	101
21	10101	101



Ολικό βάθος **d=3**. Τοπικά βάθη: **d_1= d_2= d_3=2**, **d_4= d_5=3**.

Ερώτημα 2.

Έστω ένα B^+– δέντρο τάξης 3 αρχικά άδειο. Θεωρείστε ότι τιμή ίση με την τιμή αναζήτησης σε ένα κόμβο αποθηκεύεται στο υποδέντρο στα δεξιά του.

(α) Θεωρείστε ότι εισάγουμε τιμές από το 1 έως το 18 με τη σειρά. Δώστε το δέντρο που προκύπτει. Ποιο είναι το μέγεθος του σε blocks;

(β) Δώστε το δέντρο που προκύπτει αν διαγράψουμε από το δέντρο του ερωτήματος (α) όλους τους πρώτους αριθμούς (από τον μικρότερο στο μεγαλύτερο).

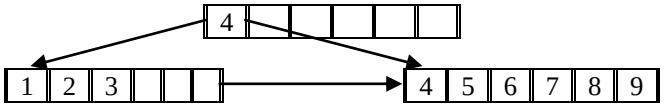
ΛΥΣΗ

(α)

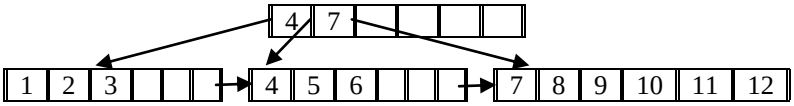
d=3 => Κάθε ενδιάμεσος κόμβος έχει από 3 έως 6 κλειδιά, άρα 4 έως 7 δείκτες σε παιδιά-κόμβους. Η ρίζα μπορεί να έχει από 2 έως 7 παιδιά. Τα φύλλα από 3 έως 6 κλειδιά.

Αρχικά εισάγεται το κλειδί 1, και το δένδρο έχει 1 μόνο κόμβο. Τα κλειδιά 2 έως 6 καταχωρούνται στη ρίζα, εφόσον υπάρχει αρκετός χώρος.

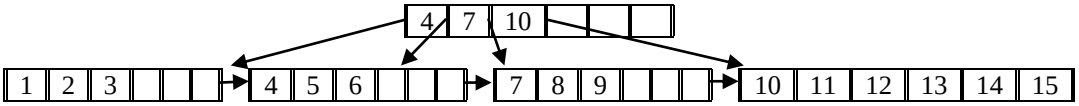
Με την εισαγωγή της τιμής 7, σπάμε σε δύο φύλλα, τα οποία είναι κάτω από τη νέα ρίζα.



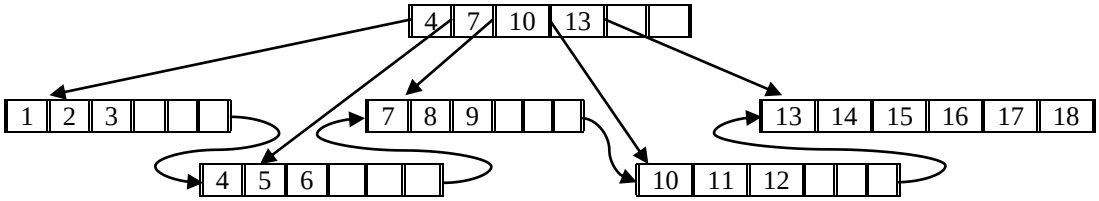
Με την εισαγωγή της τιμής 10 γίνεται διάσπαση του δεύτερου φύλλου, και οι τιμές 11 και 12 εισάγονται στο 2ο φύλλο, όπου υπάρχει χώρος.



Όμοια με την εισαγωγή της τιμής 13 γίνεται διάσπαση του τρίτου φύλλου, και οι τιμές 14 και 15 εισάγονται στο 3ο φύλλο, όπου υπάρχει χώρος.

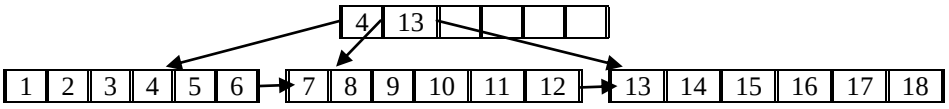


Με την εισαγωγή της τιμής 16 το τέταρτο φύλλο διασπάται, και οι τιμές 17 και 18 εισάγονται στο 4ο φύλλο, όπου υπάρχει χώρος.

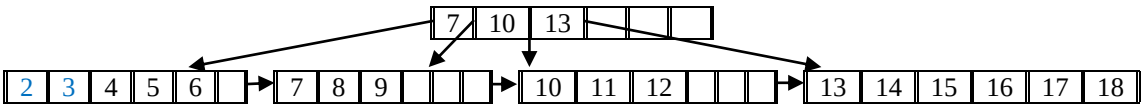


Το μέγεθος του δένδρου είναι **6 blocks**.

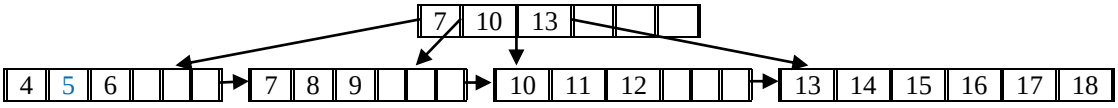
*Αν είχαμε κάνει *bulk loading* το δένδρο που θα προέκυπτε θα ήταν μεγέθους 4 blocks.



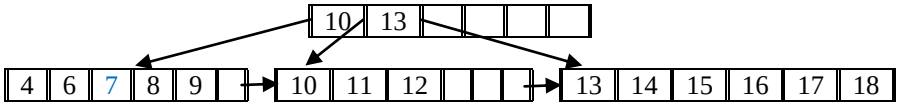
(β) Διαγραφή των αριθμών 1, 2, 3, 5, 7, 11, 13, 17 διαδοχικά.
Με τη διαγραφή της τιμής 1, το 1ο φύλλο γίνεται λιγότερο από 50% γεμάτο. Γίνεται συγχώνευση των 2 πρώτων φύλλων. Δεν υπάρχουν αρκετές τιμές για να γίνει ανακατανομή.



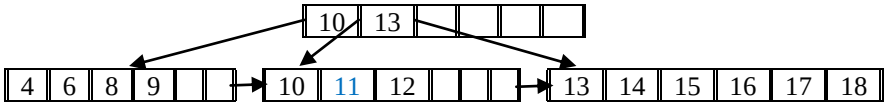
Στη συνέχεια οι τιμές 2 και 3 διαγράφονται από το 1ο φύλλο χωρίς να αλλάξει η δομή του δένδρου.



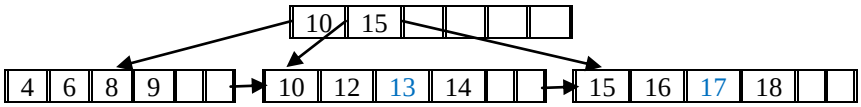
Με τη διαγραφή της τιμής 5, το 1ο φύλλο γίνεται λιγότερο από 50% γεμάτο. Γίνεται και πάλι συγχώνευση των 2 πρώτων φύλλων, εφόσον δεν υπάρχουν αρκετές τιμές στο 2ο φύλλο για αναδιάταξη.



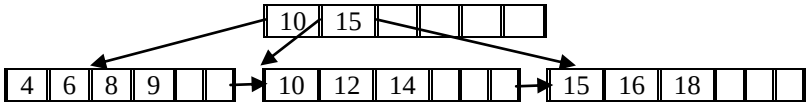
Η τιμή 7 διαγράφεται από το 1ο φύλλο χωρίς να αλλάξει η δομή του δένδρου.



Με τη διαγραφή της τιμής 11, το 2ο φύλλο γίνεται λιγότερο από 50% γεμάτο. Γίνεται ανακατανομή.



Οι τιμές 13 και 17 διαγράφονται χωρίς να επηρεαστεί το δένδρο.



Ερώτημα 3.
Ποιο είναι το μεγαλύτερο μέγεθος αρχείου δεδομένων (σε αριθμό blocks) που μπορεί να δεικτοδοτήσει ένα B^+– δέντρο τάξης d = 28, και που στα φύλλα χωρούν το πολύ 45 ζευγάρια (κλειδί, TID) και ύψους h = 3 στις παρακάτω περιπτώσεις. Θεωρείστε ότι ο παράγοντας ομαδοποίησης (σελιδοποίησης) του αρχείου είναι 35, δηλ. σε κάθε block του αρχείου δεδομένων χωρούν 35 εγγραφές). Κάντε όποιες άλλες υποθέσεις θεωρείτε αναγκαίες.

(α) το αρχείο δεδομένων είναι ταξινομημένο ως προς το πεδίο ευρετηριοποίησης και το πεδίο ευρετηριοποίησης είναι κλειδί
(β) το αρχείο δεδομένων δεν είναι ταξινομημένο ως προς το πεδίο ευρετηριοποίησης και το πεδίο ευρετηριοποίησης είναι κλειδί.
(γ) το αρχείο δεδομένων είναι ταξινομημένο ως προς το πεδίο ευρετηριοποίησης αλλά το πεδίο ευρετηριοποίησης δεν είναι κλειδί και υπάρχουν 100 εγγραφές ανά τιμή του πεδίου.
(δ) το αρχείο δεδομένων δεν είναι ταξινομημένο ως προς το πεδίο ευρετηριοποίησης, το πεδίο ευρετηριοποίησης δεν είναι κλειδί και υπάρχουν 100 εγγραφές ανά τιμή του πεδίου.

ΛΥΣΗ

Το B^+– δέντρο είναι τάξης d = 28, άρα κάθε εσωτερικός κόμβος έχει το πολύ 2d = 56 κλειδιά, άρα το μέγιστο 57 παιδιά-κόμβους.
Επειδή έχει ύψος 3, ο μέγιστος αριθμός φύλλων που μπορεί να έχει είναι:

$$57^3 = 185.193.$$

Η ρίζα έχει 57 παιδιά, καθένα από τα οποία έχει από 57 παιδιά, καθένα έχει 57 παιδιά-φύλλα.
Επειδή στα φύλλα χωρούν το πολύ 45 ζευγάρια (κλειδί, TID), θα έχουμε συνολικά:

$$45 \cdot 185.193 = 8.333.685 \text{ ζεύγη (κλειδί, TID).}$$

(α) Επειδή οι εγγραφές είναι αποθηκευμένες μέσα στα blocks ταξινομημένες με αύξουσα σειρά ως προς το πεδίο ευρετηριοποίησης, μπορούμε να υποθέσουμε ότι στα φύλλα αποθηκεύουμε μόνο την $1^η$ εγγραφή από κάθε block. Οι υπόλοιπες εγγραφές βρίσκονται διατρέχοντας σειριακά τις εγγραφές μέσα στο block, ξεκινώντας από την $1^η$, οποία περιέχεται στο κατάλληλο φύλλο.
Με αυτόν τον τρόπο το B^+-δένδρο με 8.333.685 ζεύγη (κλειδί, TID) μπορεί να δεικτοδοτήσει αρχείο μέγιστου μεγέθους **8.333.685 blocks**.

(β) Το αρχείο δεδομένων δεν είναι ταξινομημένο ως προς το πεδίο ευρετηριοποίησης και το πεδίο ευρετηριοποίησης είναι κλειδί. Συνεπώς, κάθε εγγραφή του αρχείου θα πρέπει να περιέχεται στα φύλλα του δένδρου. Άρα το αρχείο θα πρέπει να έχει το πολύ 8.333.685 εγγραφές. Το μέγιστο μέγεθός του σε blocks είναι: |_8.333.685 / 35_| = **238.105 blocks.** [1]

(γ) Το αρχείο δεδομένων είναι ταξινομημένο ως προς το πεδίο ευρετηριοποίησης, αλλά το πεδίο ευρετηριοποίησης δεν είναι κλειδί και υπάρχουν 100 εγγραφές ανά τιμή του πεδίου.
Αυτό σημαίνει ότι οι εγγραφές του αρχείου είναι ταξινομημένες ανά 100 με ίδια τιμή πεδίο ευρετηριοποίησης. Μπορούμε να κάνουμε ότι στο (α), αλλά επειδή υπάρχει επανάληψη τιμών, θα μπορούσαμε να εισάγουμε μια τιμή στο ευρετήριο ανά *διακριτή* (distinct) τιμή του αρχείου δεδομένων, συν ένα pointer στην πρώτη της εμφάνιση. Οπότε το αρχείο μπορεί να έχει έως 8.333.685*100 εγγραφές. Το μέγιστο μέγεθος αρχείου είναι: |_833.368.500 / 35_| = **23.810.528 blocks.**

[1] Στρογγυλοποιούμε προς τα κάτω, γιατί αν ήταν 238.106 γεμάτα blocks θα είχε 238.106 * 35 = 8.333.710 εγγραφές, οι οποίες δεν θα χωρούσαν στο B^+-δένδρο.

(δ) Το αρχείο δεδομένων δεν είναι ταξινομημένο ως προς το πεδίο ευρετηριοποίησης, το πεδίο ευρετηριοποίησης δεν είναι κλειδί και υπάρχουν 100 εγγραφές ανά τιμή του πεδίου. Υπάρχουν πολλές επιλογές. Για παράδειγμα, μπορούμε να κάνουμε ότι και στο (β), δηλαδή πυκνό ευρετήριο. Άρα το αρχείο θα πρέπει να έχει το πολύ 8.333.685 εγγραφές. Το μέγιστο μέγεθός του σε blocks είναι: |_8.333.685 / 35_| = **238.105 blocks**.
Μια άλλη επιλογή είναι να έχουμε ενδιάμεσο επίπεδο: εισάγουμε μόνο τις *διακριτές τιμές* του αρχείου στο ευρετήριο και ένα δείκτη σε ένα ενδιάμεσο ευρετήριο όπου "μαζεύουμε" με τη σειρά δείκτες στα blocks του αρχείου όπου εμφανίζεται αυτή η τιμή. Με το ενδιάμεσο ευρετήριο, μπορούμε να έχουμε μέγιστο μέγεθος: |_8.333.685*100 / 35_| = **23.810.528 blocks** όπως πριν.

Ερώτημα 4.
i) Θεωρήστε μία σχέση R (A, B, C) που περιέχει 100.000 εγγραφές και με κλειδί το πεδίο A που παίρνει τιμές από 0 έως 99.999.
Έχουμε δύο εναλλακτικές να οργανώσουμε τις εγγραφές της R στο δίσκο:

(α) Σαν ένα κατακερματισμένο αρχείο, το οποίο να έχει 500 κάδους (buckets). Κάθε block στο δίσκο μπορεί να αποθηκεύσει μέχρι 10 εγγραφές – συμπεριλαμβανομένου και ενός δείκτη προς το block υπερχείλισης (εφόσον χρειαστεί). Θεωρήστε επίσης ότι κάθε block περιέχει εγγραφές από ένα μόνο κάδο.

(β) Σαν ένα B^+– δέντρο στο δίσκο. Τα φύλλα του δέντρου περιέχουν τις ίδιες τις εγγραφές της σχέσης και όχι δείκτες προς αυτές. Κάθε φύλλο μπορεί να αποθηκεύσει μέχρι 10 εγγραφές – συμπεριλαμβανομένου και ενός δείκτη προς το επόμενο block στα φύλλα, ενώ κάθε εσωτερικός κόμβος μπορεί να αποθηκεύσει μέχρι 100 κλειδιά.

Για κάθε ένα από τα παρακάτω αιτήματα, προσδιορίστε πόσες λειτουργίες I/O απαιτούνται για τον καθένα από τους παραπάνω τρόπους οργάνωσης για την επεξεργασία του ερωτήματος.
1. Δώσε όλες τις πλειάδες της R.
2. Βρες όλες τις πλειάδες της R ώστε A < 50.
3. Βρες όλες τις πλειάδες της R ώστε A = 50.
4. Βρες όλες τις πλειάδες της R. ώστε A > 50 και A < 100.

ii) Πώς θα άλλαζαν οι απαντήσεις σας στις παραπάνω ερωτήσεις αν το γνώρισμα A **δεν ήταν** κλειδί της R;

ΛΥΣΗ
i) Το πεδίο A είναι **κλειδί** και παίρνει *όλες* τις τιμές από 1 έως 99.999, επειδή υπάρχουν ακριβώς 100.000 εγγραφές.

(α) Κατακερματισμένο αρχείο:
Η συνάρτηση κατακερματισμού μπορεί να επιλεγεί έτσι ώστε οι εγγραφές να ισομοιραστούν στους 500 κάδους [π.χ. h(A) = A mod 500]. Επομένως αποθηκεύονται:
100.000 / 500 = 200 εγγραφές ανά κάδο.
Επειδή κάθε block «χωράει» μέχρι 10 εγγραφές, κάθε κάδος θα χρειαστεί:
200 / 10 = 20 blocks συνολικά ανά κάδο.
Σε κάθε κάδο θα υπάρχουν 19 blocks υπερχείλισης.
Ολόκληρο το κατακερματισμένο αρχείο θα έχει συνολικά:
500 · 20 = 10.000 blocks.

1. Δώσε όλες τις πλειάδες της R.
Για να απαντηθεί το ερώτημα αυτό θα πρέπει να διαβαστούν όλα τα blocks του αρχείου.
Άρα θα χρειαστούν **10.000** λειτουργίες I/O.

2. Βρες όλες τις πλειάδες της R ώστε A < 50.
Καθεμία από τις ζητούμενες πλειάδες βρίσκεται στο 1^{ο} block, καθενός από τους πρώτους 50 κάδους (από 0 έως 49), αν χρησιμοποιηθεί η συνάρτηση h(A) = A mod 500. Επομένως θα διαβαστούν 50 blocks από το αρχείο. Άρα θα χρειαστούν **50** λειτουργίες I/O.

3. Βρες όλες τις πλειάδες της R ώστε A = 50.
Η ζητούμενη πλειάδα βρίσκεται στο 1^{ο} block του 51^{ου} κάδου (με τιμή h(A) = 50), αν χρησιμοποιηθεί η συνάρτηση h(A) = A mod 500. Επομένως θα διαβαστεί 1 block από το αρχείο.
Άρα θα χρειαστεί **1** λειτουργία I/O.

4. Βρες όλες τις πλειάδες της R. ώστε A > 50 και A < 100.
Καθεμία από τις ζητούμενες πλειάδες βρίσκεται στο 1^{ο} block, των κάδων από τον 52^{ο} ως και τον 100^{ο} (τιμές της h(A) από 51 έως 99), αν χρησιμοποιηθεί η συνάρτηση h(A) = A mod 500. Επομένως θα διαβαστούν 49 blocks από το αρχείο. Άρα θα χρειαστούν **49** λειτουργίες I/O.

(β) B^{+}– δέντρο στο δίσκο:
Θεωρούμε ότι το δένδρο είναι γεμάτο.
Κάθε φύλλο μπορεί να αποθηκεύσει μέχρι 10 εγγραφές. Επομένως χρειάζονται τουλάχιστον:
100.000 / 10 = 10.000 φύλλα (γεμάτα).
Τα φύλλα αυτά περιέχουν τις ίδιες τις εγγραφές. Επειδή κάθε εσωτερικός κόμβος έχει το πολύ 100 κλειδιά, θα έχει το πολύ 101 δείκτες σε παιδιά-κόμβους. Το επίπεδο πάνω από τα φύλλα θα έχει:
|¯10.000 / 101¯| = 100 κόμβους.
Αρκεί 1 κόμβος για να δείξει σε αυτούς τους 100, ο οποίος είναι η ρίζα του δένδρου.
Επομένως το δένδρο έχει 3 επίπεδα. Συνολικά αποτελείται από:
10.000 + 100 + 1 = 10.101 blocks.

1. Δώσε όλες τις πλειάδες της R.
Θα πρέπει να διαβαστεί η ρίζα του δένδρου (1 block), στη συνέχεια το 1^{ο} παιδί της ρίζας (1 block) και ακολούθως το 1^{ο} φύλλο. Από το κάθε φύλλο μπορεί να διαβαστεί το επόμενο με τη σειρά χρησιμοποιώντας τους δείκτες των φύλλων. Επομένως διαβάζονται με τη σειρά όλα τα 10.000 blocks των φύλλων. Συνολικά θα διαβαστούν: 1 + 1 + 10.000 = 10.002 blocks.
Άρα θα χρειαστούν **10.002** λειτουργίες I/O.

2. Βρες όλες τις πλειάδες της R ώστε A < 50.
Το 1^{ο} φύλλο έχει εγγραφές για A από 0 έως 9, το 2^{ο} από 10 ως 19, κτλ. Η τιμή A=50 είναι στο 6^{ο} φύλλο. Επομένως θα πρέπει να διαβαστεί η ρίζα του δένδρου, στη συνέχεια το 1^{ο} παιδί της ρίζας και ακολούθως τα 5 πρώτα φύλλα. Συνολικά θα διαβαστούν: 1 + 1 + 5 = 7 blocks.
Άρα θα χρειαστούν **7** λειτουργίες I/O.

3. Βρες όλες τις πλειάδες της R ώστε A = 50.
Θα πρέπει να διαβαστεί η ρίζα του δένδρου, στη συνέχεια κατάλληλο το παιδί της ρίζας και ακολούθως το φύλλο που περιέχει την εγγραφή με A = 50. Συνολικά θα διαβαστούν 3 blocks .
Άρα θα χρειαστούν **3** λειτουργίες I/O (height +1).

4. Βρες όλες τις πλειάδες της R. ώστε A > 50 και A < 100.
Το 6^{ο} φύλλο έχει εγγραφές για A από 50 έως 59, το 7^{ο} από 60 ως 69, κτλ. Το 10^{ο} φύλλο έχει εγγραφές για A από 90 έως 99. Επίσης το 1^{ο} παιδί της ρίζας δείχνει προς τα πρώτα 101 φύλλα. Συνεπώς, θα διαβαστεί η ρίζα του δένδρου, στη συνέχεια το 1^{ο} παιδί της ρίζας και ακολούθως τα φύλλα 6^{ο} έως 10^{ο}. Συνολικά θα διαβαστούν: 1 + 1 + 5 = 7 blocks.
Άρα θα χρειαστούν **7** λειτουργίες I/O.

ii) Το γνώρισμα A **δεν είναι** κλειδί της R. Επομένως παίρνει κάποιες από τις τιμές μεταξύ 0 και 99.999, ενώ περισσότερες από μια εγγραφές μπορούν να έχουν την ίδια τιμή για το A.

(α) Κατακερματισμένο αρχείο:
Η συνάρτηση h(A) = A mod 500 δεν μας εγγυάται την ίση κατανομή των εγγραφών στους κάδους. Κάποιοι μπορεί να έχουν περισσότερα blocks υπερχείλισης από άλλους.

1. Δώσε όλες τις πλειάδες της R.
Για να απαντηθεί το ερώτημα αυτό θα πρέπει να διαβαστούν όλα τα blocks του αρχείου.
Στην *καλύτερη περίπτωση*, οι εγγραφές καταλαμβάνουν και πάλι 10.000 blocks συνολικά.
Στην *χειρότερη περίπτωση*, έχει αποθηκευτεί από 1 εγγραφή σε 499 κάδους. Οι υπόλοιπες 100.000 – 499 = 99.501 εγγραφές αποθηκεύονται σε ένα μόνο κάδο. Ο τελευταίος θα έχει |¯99.501/10¯| = 9.951

blocks (9.950 blocks υπερχείλισης). Επομένως, το αρχείο θα έχει 499 + 9.951 = 10.450 blocks συνολικά.
Άρα θα χρειαστούν από **10.000** (καλύτερη περίπτωση) έως **10.450** (χειρότερη περίπτωση) λειτουργίες I/O.

2. Βρες όλες τις πλειάδες της R ώστε A < 50.
Στην *καλύτερη περίπτωση*, καθεμία από τις ζητούμενες πλειάδες βρίσκεται στο 1^{ο} block, καθενός από τους πρώτους 50 κάδους (από 0 έως 49), αν χρησιμοποιηθεί η συνάρτηση h(A) = A mod 500. Επομένως θα διαβαστούν 50 blocks από το αρχείο, όπως στην περίπτωση (i). Τότε θα χρειαστούν **50** λειτουργίες I/O.
Διαφορετικά, κάποιες εγγραφές βρίσκονται σε blocks υπερχείλισης. Δεν γνωρίζουμε πόσες ακριβώς είναι αυτές, άρα και πόσα επιπλέον blocks θα πρέπει να διαβαστούν. Έστω ότι είναι **ν** blocks υπερχείλισης συνολικά. Τότε θα χρειαστούν **50+ν** λειτουργίες I/O.
Στην *χειρότερη περίπτωση*, όλες οι πλειάδες της R ικανοποιούν τη συνθήκη A < 50, και βρίσκονται αποθηκευμένες σε 50 κάδους ως εξής: 49 κάδοι έχουν από 1 εγγραφή και οι υπόλοιπες 100.000 – 49 = 99.951 εγγραφές βρίσκονται σε ένα κάδο με $\lceil 99.951 / 10 \rceil$ = 9.996 blocks. Συνολικά το αρχείο καταλαμβάνει 48 + 9.996 = 10.044 blocks.
Άρα θα χρειαστούν από **50** (καλύτερη περίπτωση) έως **10.044** (χειρότερη περίπτωση) λειτουργίες I/O.

3. Βρες όλες τις πλειάδες της R ώστε A = 50.
Στην *καλύτερη περίπτωση*, όλες οι ζητούμενες πλειάδες βρίσκονται στο 1^{ο} block του 51^{ου} κάδου (με τιμή h(A) = 50), αν χρησιμοποιηθεί η συνάρτηση h(A) = A mod 500. Επομένως θα διαβαστεί 1 block από το αρχείο, όπως στην περίπτωση (i). Τότε θα χρειαστεί **1** λειτουργία I/O.
Διαφορετικά, κάποιες εγγραφές βρίσκονται σε blocks υπερχείλισης του 51^{ου} κάδου. Δεν γνωρίζουμε πόσες είναι αυτές, άρα και πόσα επιπλέον blocks θα πρέπει να διαβαστούν. Έστω ότι είναι **ν΄** blocks υπερχείλισης συνολικά. Τότε θα χρειαστούν **1+ν΄** λειτουργίες I/O.
Στην *χειρότερη περίπτωση*, όλες οι πλειάδες της R ικανοποιούν τη συνθήκη A = 50, βρίσκονται *όλες στον ίδιο κάδο* και καταλαμβάνουν 10.000 blocks.
Άρα θα χρειαστούν από **1** (καλύτερη περίπτωση) έως **10.000** (χειρότερη περίπτωση) λειτουργίες I/O.

4. Βρες όλες τις πλειάδες της R. ώστε A > 50 και A < 100.
Στην *καλύτερη περίπτωση*, καθεμία από τις ζητούμενες πλειάδες βρίσκεται στο 1^{ο} block, των κάδων από τον 52^{ο} ως και τον 100^{ο}, αν χρησιμοποιηθεί η συνάρτηση h(A) = A mod 500. Επομένως θα διαβαστούν 49 blocks από το αρχείο, όπως στην περίπτωση (i) και θα χρειαστούν **49** λειτουργίες I/O.
Διαφορετικά, κάποιες εγγραφές βρίσκονται σε blocks υπερχείλισης ορισμένων κάδων από τον 52^{ο} ως τον 100^{ο}. Δεν γνωρίζουμε πόσες είναι αυτές, άρα και πόσα επιπλέον blocks θα πρέπει να διαβαστούν. Έστω ότι είναι **ν΄΄** blocks υπερχείλισης συνολικά. Τότε θα χρειαστούν **49+ν΄΄** λειτουργίες I/O.
Στην *χειρότερη περίπτωση*, όλες οι πλειάδες της R ικανοποιούν τη συνθήκη A > 50 και A < 100, και βρίσκονται αποθηκευμένες σε 49 κάδους ως εξής: 48 κάδοι έχουν από 1 εγγραφή και οι υπόλοιπες 100.000 – 48 = 99.952 εγγραφές βρίσκονται σε ένα κάδο με $\lceil 99.952 / 10 \rceil$ = 9.996 blocks. Συνολικά το αρχείο καταλαμβάνει 48 + 9.996 = 10.044 blocks.
Άρα θα χρειαστούν από **50** (καλύτερη περίπτωση) έως **10.044** (χειρότερη περίπτωση) λειτουργίες I/O.

(β) B^{+}– δέντρο στο δίσκο:
Θεωρούμε ότι το δένδρο είναι γεμάτο, επομένως και πάλι χρειάζονται 10.000 φύλλα 100 ενδιάμεσοι κόμβοι και η ρίζα.

1. Δώσε όλες τις πλειάδες της R.
Θα πρέπει να διαβαστεί η ρίζα του δένδρου, στη συνέχεια το 1^{ο} παιδί της ρίζας και ακολούθως το 1^{ο} φύλλο. Από το κάθε φύλλο μπορεί να διαβαστεί το επόμενο με τη σειρά χρησιμοποιώντας τους δείκτες των φύλλων. Επομένως διαβάζονται με τη σειρά όλα τα 10.000 blocks των φύλλων. Συνολικά θα διαβαστούν: 1 + 1 + 10.000 = 10.002 blocks.
Άρα θα χρειαστούν **10.002** λειτουργίες I/O.

2. Βρες όλες τις πλειάδες της R ώστε A < 50.
Θα πρέπει να διαβαστεί η ρίζα του δένδρου, στη συνέχεια το 1^{ο} παιδί της ρίζας και ακολούθως τα πρώτα φύλλα που περιέχουν εγγραφές που ικανοποιούν τη συνθήκη.
Στην *καλύτερη περίπτωση* θα διαβαστούν: 1 + 1 + 1 = 3 blocks. Αυτό ισχύει αν είναι λίγες οι εγγραφές με A < 50 και χωρούν στο 1^{ο} φύλλο. Αυτό θα χρειαστεί **3** λειτουργίες I/O.

Διαφορετικά, θα διαβαστούν τόσα παραπάνω blocks όσα και τα παραπάνω φύλλα που περιέχουν εγγραφές οι οποίες ικανοποιούν τη συνθήκη, $\lceil sel(\varphi) * 10.000 \rceil$. Έστω ότι είναι **k** τα επιπλέον φύλλα. Τότε θα χρειαστούν από **3+k** λειτουργίες I/O.
Στη *χειρότερη περίπτωση*, όλες οι εγγραφές ικανοποιούν τη συνθήκη A < 50. Τότε θα διαβαστούν 10.002 blocks, όπως στο (1.)
Άρα θα χρειαστούν από **3** έως **10.002** λειτουργίες I/O.

3. Βρες όλες τις πλειάδες της R ώστε A = 50.
Θα πρέπει να διαβαστεί η ρίζα του δένδρου, στη συνέχεια κατάλληλο το παιδί της ρίζας και ακολούθως τα φύλλα που περιέχουν τις εγγραφές με A = 50.
Στην *καλύτερη περίπτωση*, θα διαβαστούν: 1 + 1 + 1 = 3 blocks. Αυτό ισχύει αν είναι λίγες οι εγγραφές με A = 50 και χωρούν σε ένα μόνο φύλλο. Τότε θα χρειαστούν από **3** λειτουργίες I/O.
Διαφορετικά, θα διαβαστούν τόσα παραπάνω blocks όσα και τα παραπάνω φύλλα που περιέχουν εγγραφές οι οποίες ικανοποιούν τη συνθήκη A = 50, $\lceil sel(\varphi) * 10.000 \rceil$. Έστω ότι είναι **k΄** τα επιπλέον φύλλα. Τότε θα χρειαστούν από **3+k΄** λειτουργίες I/O.
Στη *χειρότερη περίπτωση*, όλες οι εγγραφές ικανοποιούν τη συνθήκη A = 50. Τότε θα διαβαστούν 10.002 blocks, όπως στο (1.)
Άρα θα χρειαστούν από **3** έως **10.002** λειτουργίες I/O.

4. Βρες όλες τις πλειάδες της R. ώστε A > 50 και A < 100.
Θα πρέπει να διαβαστεί η ρίζα του δένδρου, στη συνέχεια κατάλληλο το παιδί της ρίζας και ακολούθως τα φύλλα που περιέχουν τις εγγραφές με A > 50 και A < 100.
Στην *καλύτερη περίπτωση*, θα διαβαστούν: 1 + 1 + 1 = 3 blocks. Αυτό ισχύει αν είναι λίγες οι εγγραφές με A > 50 και A < 100 και χωρούν σε ένα μόνο φύλλο. Τότε θα χρειαστούν από **3** λειτουργίες I/O.
Διαφορετικά, θα διαβαστούν τόσα παραπάνω blocks όσα και τα παραπάνω φύλλα που περιέχουν εγγραφές οι οποίες ικανοποιούν τη συνθήκη A > 50 και A < 100, $\lceil sel(\varphi) * 10.000 \rceil$. Έστω ότι είναι **k΄΄** τα επιπλέον φύλλα. Τότε θα χρειαστούν από **3+k΄΄** λειτουργίες I/O.
Στη *χειρότερη περίπτωση*, όλες οι εγγραφές ικανοποιούν τη συνθήκη A > 50 και A < 100. Τότε θα διαβαστούν 10.002 blocks, όπως στο (1.)
Άρα θα χρειαστούν από **3** έως **10.002** λειτουργίες I/O.