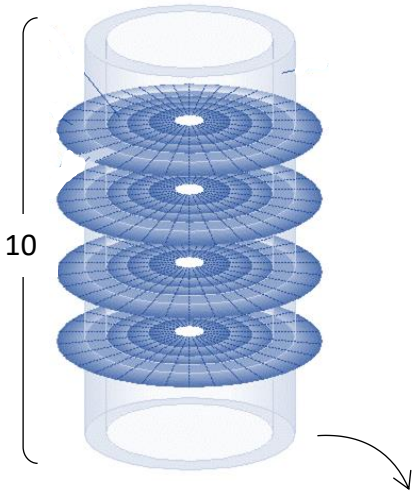
Συστήματα Διαχείρισης & Ανάλυσης Δεδομένων
1η Ομάδα Ασκήσεων

Αλβιόνα Μάντσο
3200098



Άσκηση 1

1) Εφόσον ο δίσκος έχει 10.000 cylinders, θα υπάρχουν 10.000 track/surface.

Συνεπώς:

$$10\ surface * 10.000 \frac{track}{surface} * 1.000 \frac{sector}{track} * 512 \frac{bytes}{sector}$$
$$= 51.200.000.000\ bytes = \boxed{50.000.000\ KB\ \sim\ 47{,}68\ GB}$$

---

2) Η ταχύτητα περιστροφής είναι 10.000 rpm.

$$\frac{10.000}{60} = \frac{1}{x} \Leftrightarrow x = \frac{60}{10.000} \Leftrightarrow x = 0{,}006\ \text{sec} = 6\ msec$$

...είναι ο χρόνος που απαιτεί μια περιστροφή.

Άρα Average Rotation Latency = $\frac{1}{2} * 6 = \boxed{3\ msec}$

---

3) Εφόσον 1 sector είναι 512 bytes, 1 block αποτελείται από

$$\frac{4096}{512} = 8\ sectors$$

Σύμφωνα με το παραπάνω αποτέλεσμα (2), σε 6 msec η κεφαλή περνάει από 1.000 sectors άρα για 1 block απαιτούνται

$$\frac{6}{1.000} * 8 = 0{,}006 * 8 = 0{,}048\ msec = \boxed{48\ \mu sec}$$

---

4) Έχουμε 1.000.000 records με το καθένα να καταλαμβάνει 1 block, επομένως απαιτούνται 1.000.000 blocks για το αρχείο.
T = Average Seek Time + Average Rotation Latency + Transfer Time =
8 + 3 + 1.000.000 * 0,048 = 48.011 msec = $\boxed{48{,}011\ sec}$

Σημείωση: Χρησιμοποιούνται οι μέσοι χρόνοι γιατί δεν προσδιορίζεται συγκεκριμένα blocks (κι ούτε αναφέρονται οι σχετικές τους θέσεις).

5) Το ευρετήριο δημιουργείται στο πρωτεύον κλειδί του πίνακα, συνεπώς δεν θα υπάρχουν διπλοεγγραφές (εγγραφές όπου επαναλαμβάνεται η τιμή του index key).
Υποθέτουμε ότι το ευρετηρίου δεν περιλαμβάνει τα blocks δεδομένων, παρά μόνο pointers προς αυτά οπότε θα χρειάζεται πάντα μια επιπλέον ανάγνωση block μέσω του pointer που υποδεικνύει το ευρετήριο.

Για την ανάκτηση N τυχαίων εγγραφών απαιτείται χρόνος:

$$N * (3 + 1) * (avg.\ Seek\ Time + avg.Rotation\ latency + Transfer\ Time) =$$

$$= N * 4 * (8 + 3 + 0{,}048) = \boxed{44{,}192N\ msec}$$

Για τη διάσχιση του B+ Tree (3 επίπεδα άρα ανάγνωση τριών blocks)

(δεν χρειάζεται άλλη ανάγνωση π.χ. μέσω της συνδεδεμένης λίστας που δημιουργούν τα φύλλα του δέντρου, εφόσον το index key είναι primary key στη σχέση και άρα σίγουρα δεν θα υπήρχαν διπλές τιμές)

Για την ανάγνωση του block/σελίδας που περιέχει την εγγραφή

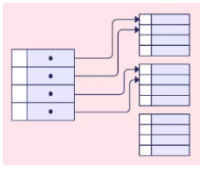
(θεωρούμε ότι τα φύλλα του δέντρου περιέχουν αναφορές προς τις σελίδες των εγγραφών)

Σημείωση: Χρησιμοποιείται το Average Seek Time γιατί πληροφορούμαστε ότι τα blocks δεν είναι συνεχόμενα στον δίσκο και συνεπώς θα γίνουν τυχαίες προσπελάσεις.

Άσκηση 2

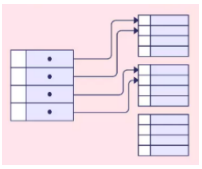
A.

Clustered
Index on **empid**



B.

Non-clustered
Index on **age**



```
1. UPDATE Employees
   SET age = age + 1
```

A. Το ευρετήριο έχει index key το πεδίο empid και συνεπώς δεν χρησιμοποιείται στο ερώτημα αυτό, το οποίο ενημερώνει τις ηλικίες όλων των εγγραφών της σχέσης Employees (χωρίς κάποια συνθήκη). Το γεγονός ότι το ευρετήριο είναι clustered καθορίζει μεν τη φυσική θέση των εγγραφών στον δίσκο αλλά εφόσον ενημερώνονται όλες οι εγγραφές της σχέσης αυτό δεν επηρεάζει την εκτέλεση του ερωτήματος. Συνεπώς η ύπαρξη του ευρετηρίου Α δεν επηρεάζει την εκτέλεση με κανέναν τρόπο.

B. Το ευρετήριο έχει index key που ταυτίζεται με το πεδίο που ενημερώνεται (age). Για την αποφυγή του Halloween Problem ο optimizer δεν θα χρησιμοποιήσει το ευρετήριο αυτό κατά την εκτέλεση του ερωτήματος. Ωστόσο θα πρέπει τελικά το ευρετήριο να τροποποιηθεί για να περιέχει τις νέες τιμές age που προκύπτουν από την ενημέρωση. Συνεπώς η ύπαρξη του ευρετηρίου B επιβραδύνει την εκτέλεση (λόγω της ανάγκης ενημέρωσης και του ευρετηρίου).

2.
```
UPDATE Employees
SET salary = salary * 1.10
WHERE empid >=1 and empid <=100
```

A. Το ευρετήριο έχει index key το πεδίο empid και συνεπώς θα αξιοποιηθεί στο ερώτημα αυτό που ενημερώνει τους μισθούς μόνον όσων εγγραφών ικανοποιούν κάποια συνθήκη που αφορά το πεδίο empid (`empid >=1 and empid <=100`). Για να μπορέσει να συμβεί αυτό θα πρέπει το είδος του ευρετηρίου να προσφέρεται για range queries (π.χ. ένα B+ δέντρο θα ήταν βοηθητικό). Το γεγονός ότι το ευρετήριο είναι clustered καθορίζει και τη φυσική θέση των εγγραφών στον δίσκο ώστε εγγραφές με διαδοχικά empid-s θα είναι και διαδοχικά γραμμένες στο δίσκο (εάν το είδος του ευρετηρίου είναι όπως περιεγράφηκε). Συνεπώς η ύπαρξη του ευρετηρίου A επιταχύνει την εκτέλεση.

B. Το ευρετήριο έχει index key το πεδίο age και συνεπώς δεν χρησιμοποιείται στο ερώτημα αυτό, ούτε επηρεάζει τη φυσική θέση των εγγραφών στο δίσκο (αφού είναι non-clustered) και βέβαια ούτε ενημερώνεται το ίδιο το ευρετήριο. Άρα η ύπαρξη του ευρετηρίου B δεν επηρεάζει την εκτέλεση με κανέναν τρόπο.

3.
```
UPDATE Employees
SET salary=salary * 1.10
WHERE departmentid = 10
```

A. Το ευρετήριο έχει index key το πεδίο empid και συνεπώς δεν χρησιμοποιείται στο ερώτημα αυτό που ενημερώνει τους μισθούς όσων εγγραφών ικανοποιούν μεν κάποια συνθήκη αλλά η συνθήκη δεν αφορά το πεδίο empid (`departmentid = 10`). Το γεγονός ότι το ευρετήριο είναι clustered καθορίζει και τη φυσική θέση των εγγραφών στον δίσκο. Από αυτήν την οπτική, η διάταξη που έχει διαμορφωθεί για τις εγγραφές λόγω του ευρετηρίου μπορεί να είναι ή να μην είναι ευνοϊκή για το ερώτημα και δεν μπορούμε να το γνωρίζουμε αυτό. Συνεπώς η ύπαρξη του ευρετηρίου A μπορεί να επιταχύνει, να επιβραδύνει ή να μην επηρεάζει την εκτέλεση.

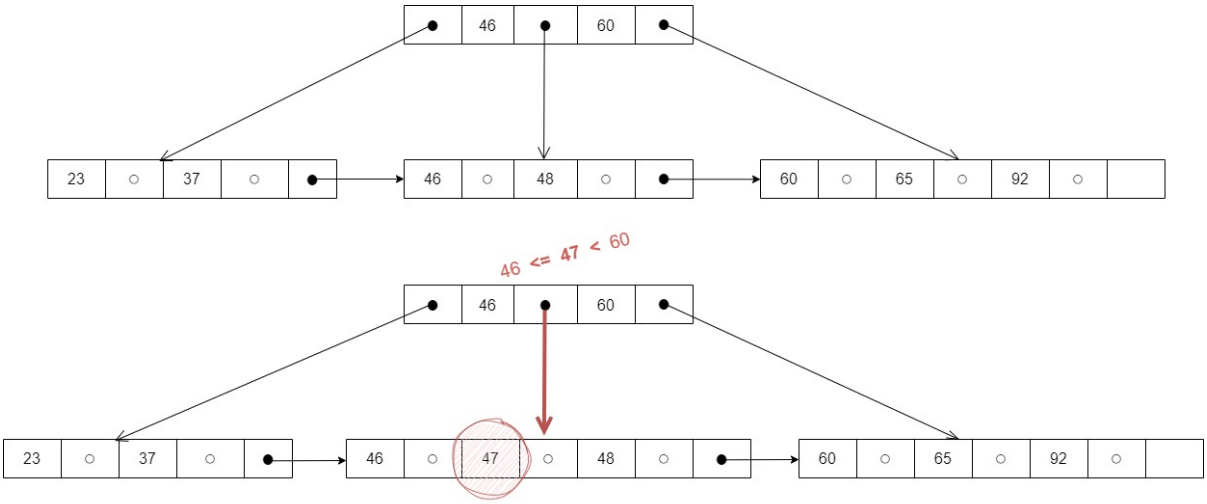
B. Το ευρετήριο έχει index key το πεδίο age και συνεπώς δεν χρησιμοποιείται στο ερώτημα αυτό, ούτε επηρεάζει τη φυσική θέση των εγγραφών στο δίσκο (αφού είναι non-clustered) και βέβαια ούτε ενημερώνεται το ίδιο το ευρετήριο. Άρα η ύπαρξη του ευρετηρίου B δεν επηρεάζει την εκτέλεση με κανέναν τρόπο.

Άσκηση 3

1)

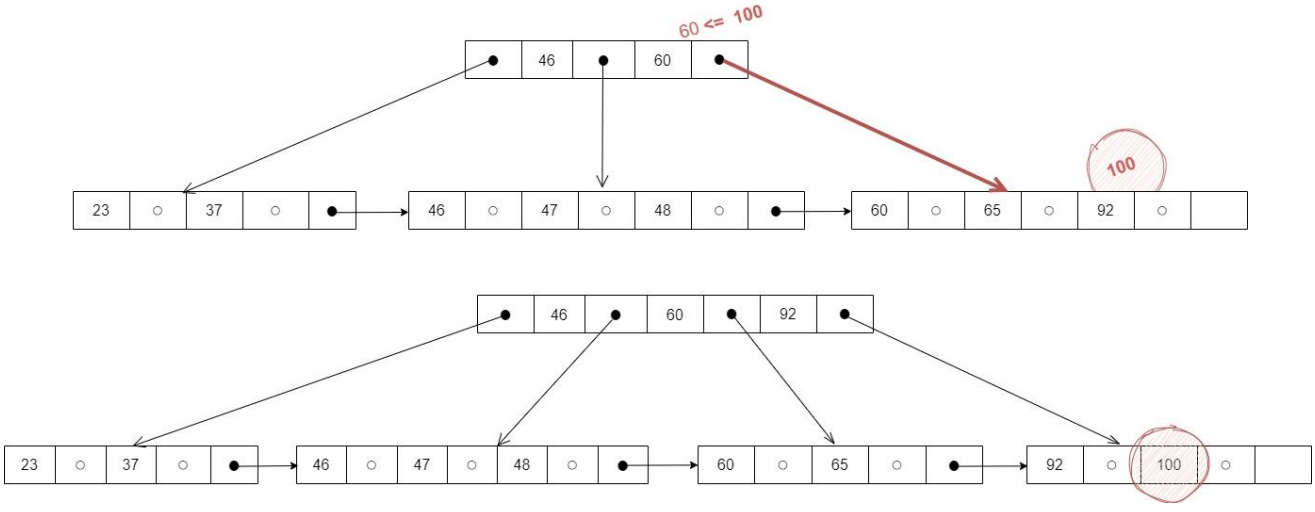
Εισαγωγή 47

- Αναζήτηση με τη τιμή 47 που θέλουμε να εισαγάγουμε.
- Εισάγουμε την τιμή και τον δείκτη στο φύλλο που καταλήξαμε. Βρισκόμαστε στην απλή περίπτωση όπου η τιμή χωράει στο φύλλο (n=3).



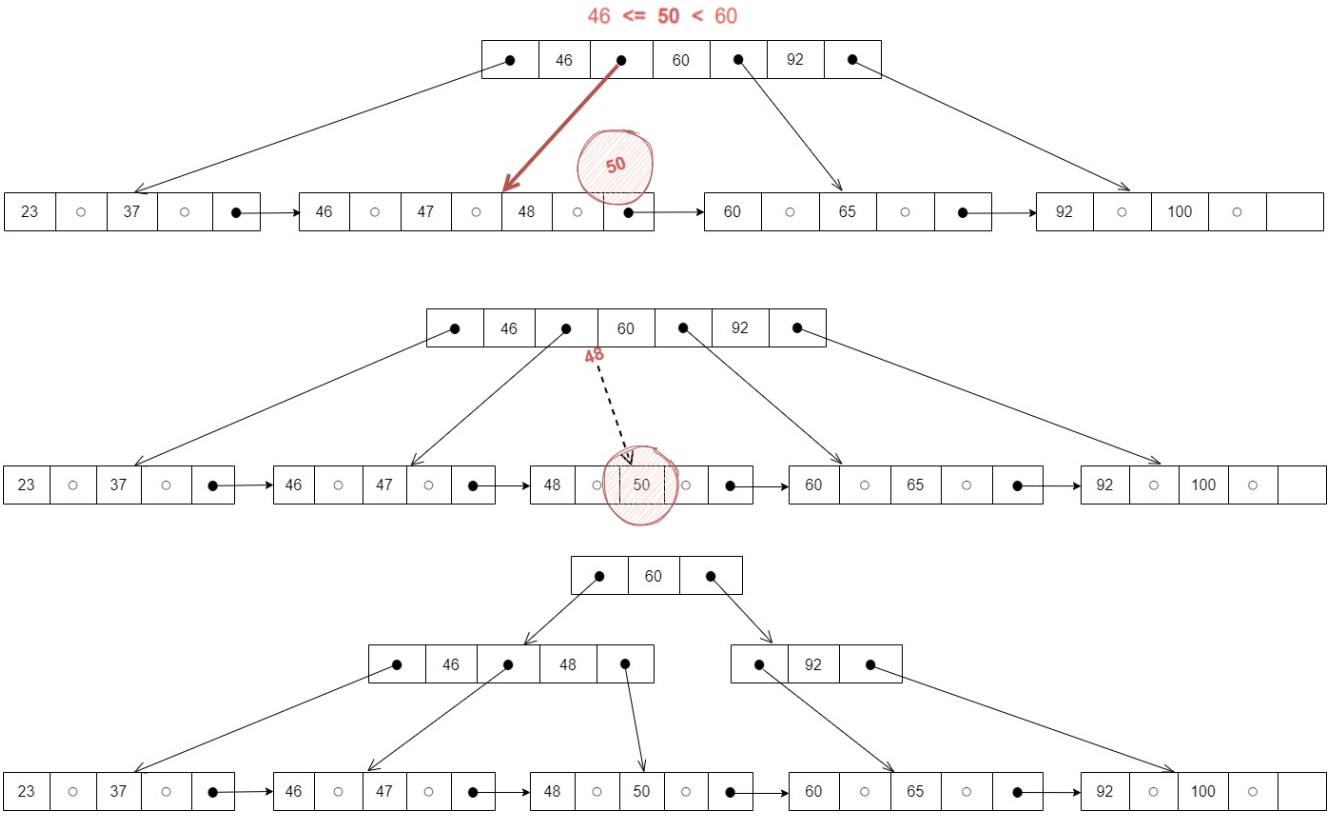
Εισαγωγή 100

- Αναζήτηση με την τιμή 100 που θέλουμε να εισαγάγουμε.
- Βρισκόμαστε στην περίπτωση όπου η τιμή δεν χωράει στο φύλλο (n=3) - leaf overflow. Δημιουργούμε νέο κόμβο-φύλλο. Τοποθετούμε στον κόμβο του αμέσως παραπάνω επιπέδου τον median (92) ως key καθώς και έναν επιπλέον pointer που θα δείχνει στο νέο κόμβο-φύλλο. Μοιράζουμε τα στοιχεία μεταξύ των δύο κόμβων-φύλλων με τον έναν να περιέχει τα <92 (αριστερά) και τον άλλον τα >=92 (δεξιά).

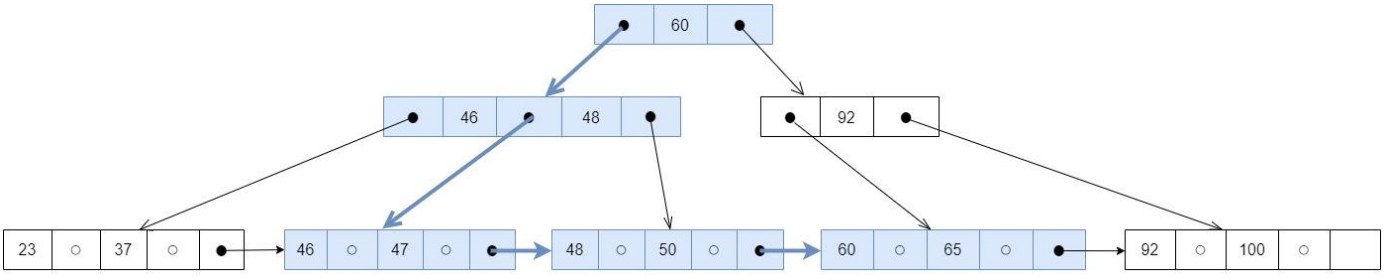


Εισαγωγή 50

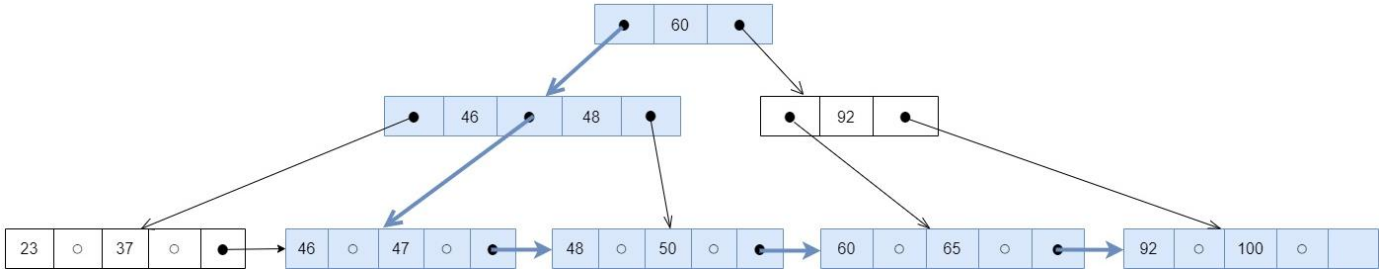
- Αναζήτηση με την τιμή 50 που θέλουμε να εισαγάγουμε.
- Βρισκόμαστε στην περίπτωση όπου η τιμή δεν χωράει στο φύλλο (n=3) οπότε δημιουργούμε νέο κόμβο-φύλλο. Τοποθετούμε στον κόμβο του αμέσως παραπάνω επιπέδου τον median (48) ως key καθώς και έναν επιπλέον pointer που θα δείχνει στο νέο κόμβο-φύλλο. Μοιράζουμε τα στοιχεία μεταξύ των δύο κόμβων-φύλλων με τον έναν να περιέχει τα <48 (αριστερά) και τον άλλον τα >=48 (δεξιά).
- Αναδρομικά διασπώνται όλοι οι κόμβοι του μονοπατιού μέχρι τη ρίζα και έχουμε τη δημιουργία νέας ρίζας - new root. Το ύψος του δέντρου αυξάνεται κατά 1.



2)



α) *Όταν το κλειδί αναζήτησης είναι μοναδικό:* Αναζητούμε την πρώτη τιμή του εύρους (46), και προσπελαύνουμε έτσι τους κόμβους του δέντρου από τη ρίζα προς τα κάτω (3 κόμβοι). Στη συνέχεια μέσω των δεικτών μετακινούμαστε δεξιά για όσο δεν βρίσκουμε αναφορά σε εγγραφή με τιμή στο κλειδί αναζήτησης >=65 (+2 κόμβοι).
Μόλις εντοπίσουμε την τιμή 65 (η τιμή 65 είναι το τέλος του εύρους) μπορούμε να σταματήσουμε τη διάσχιση αφού είναι σίγουρο πως δεν θα υπάρχει άλλη εγγραφή μετην ίδια τιμή (εφόσον το κλειδί αναζήτησης είναι μοναδικό!).
Προσπελαύνουμε λοιπόν 5 κόμβους, που φαίνονται σημειωμένοι παραπάνω.



β) *Όταν το κλειδί αναζήτησης δεν είναι μοναδικό*: Όπως παραπάνω, όμως τώρα δεν σταματάμε μόλις εντοπίσουμε την τιμή 65. Ο λόγος είναι ότι στον επόμενο κόμβο/σελίδα θα μπορούσε να υπάρχει και πάλι η τιμή 65 δλδ θα μπορούσανε να υφίστανται και άλλη/άλλες εγγραφή/εγγραφές με τιμή 65 στο πεδίο αυτό (εφόσον δεν είναι μοναδικό). Στην προκειμένη λοιπόν περίπτωση, προσπελαύνουμε έναν ακόμη κόμβο, παρατηρούμε ότι η πρώτη-μικρότερη τιμή του είναι 92 και τότε σταματάμε. Συνεπώς προσπελαύνουμε 6 κόμβους (που ξανά φαίνονται σημειωμένοι στο σχήμα).

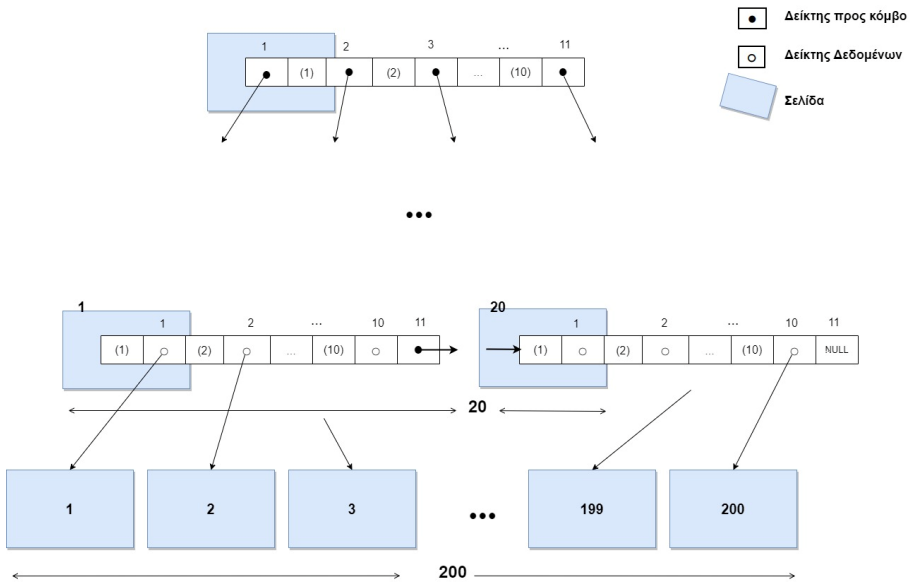
Άσκηση 4

- Αρχικά, εφόσον κάθε block του δίσκου χωράει 5 εγγραφές, θα χρειαστούν $\frac{1000}{5} = \mathbf{200}$ **blocks** για την αποθήκευση της σχέσης R.

Στη συνέχεια σκεφτόμαστε ως εξής:

- Ένας κόμβος του B+ δέντρου αντιστοιχεί σε ένα block και γνωρίζουμε ότι το block χωράει 10 keys και 11 pointers. Άρα οι κόμβοι του B+ δέντρου θα έχουν 10 keys και 11 pointers (δλδ n=10).
  Τα φύλλα του B+ δέντρου χρησιμοποιούν 10 pointers προς block δεδομένων και 1 pointer προς το επόμενο φύλλο (αφού το δέντρο είναι B+). Άρα απαιτούνται το λιγότερο $\frac{200}{10} = \mathbf{20}$ κόμβοι-φύλλα.

Το σχήμα έχει ως εξής:



Προχωρώντας στους ενδιάμεσους κόμβους του δέντρου, θα χρειαστούμε το λιγότερο **2** κόμβους που θα δείχνουν στα 20 φύλλα (π.χ. θα χρησιμοποιεί 11 pointers ο ένας και 9 ο άλλος, πλήθη συμβατά με τον κανόνα $\left\lceil \frac{n+1}{2} \right\rceil$).

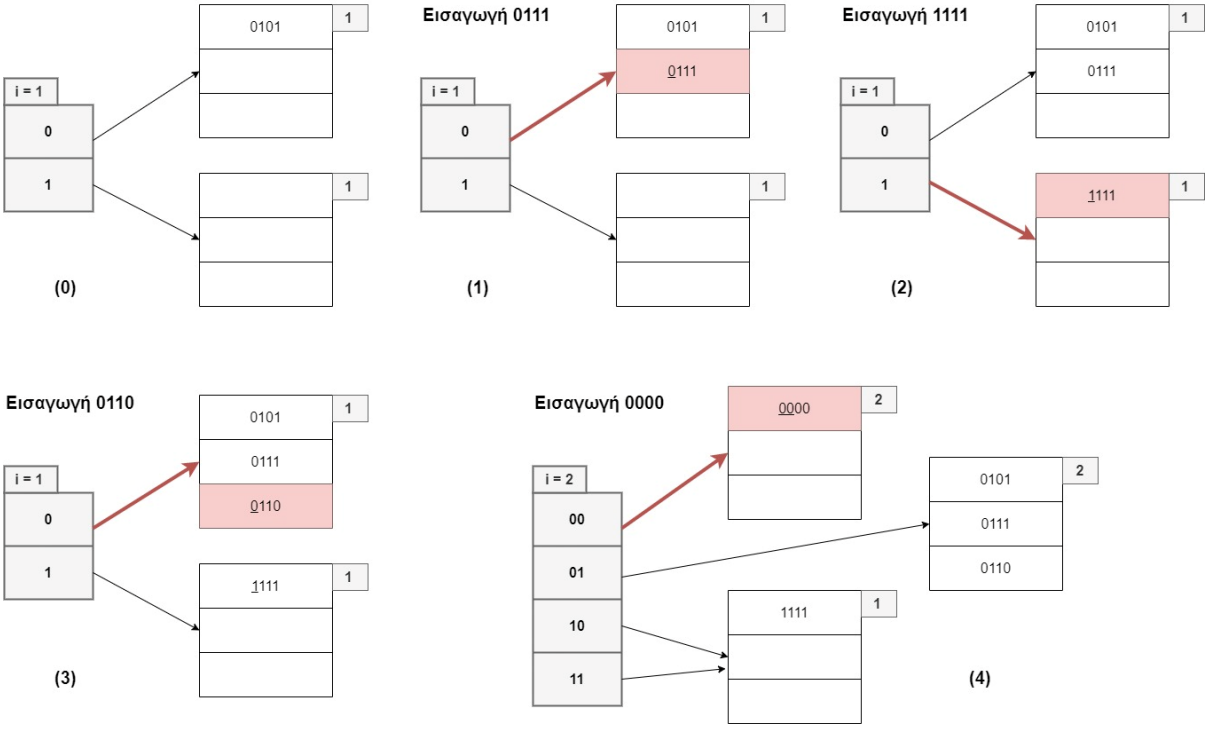
Τέλος αρκεί **1** ακόμη κόμβος-ρίζα που θα δείχνει στους 2 αυτούς κόμβους (θα χρησιμοποιεί δλδ τους 2 pointers, πλήθος που δεν μας απασχολεί γιατί η ρίζα εξαιρείται από τον κανόνα $\left\lceil \frac{n+1}{2} \right\rceil$).

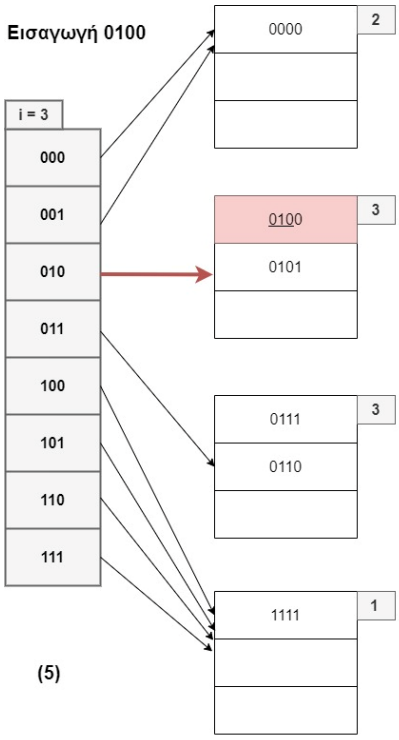
Άρα για την αποθήκευση του ευρετηρίου θα χρειαστούν $20 + 2 + 1 =$ **23 blocks.**

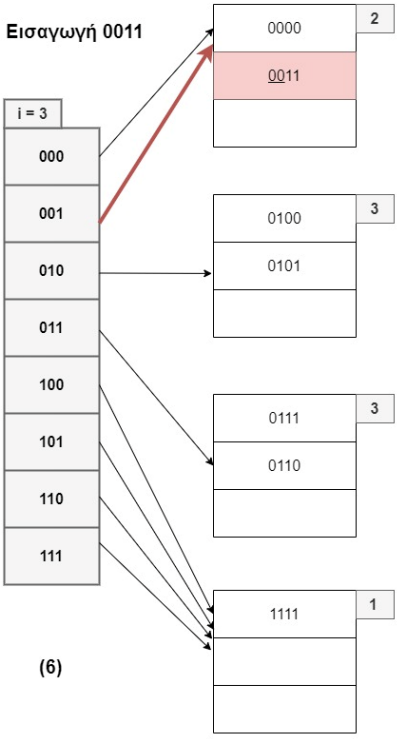
Συνεπώς συνολικά απαιτούνται $200 + 23 =$ **223 blocks**.

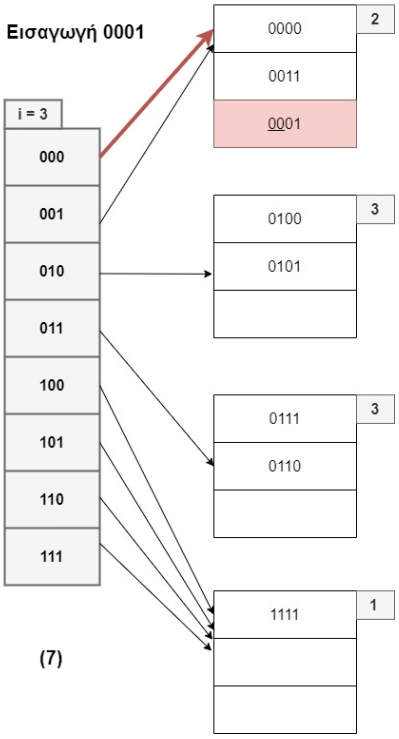
Άσκηση 5

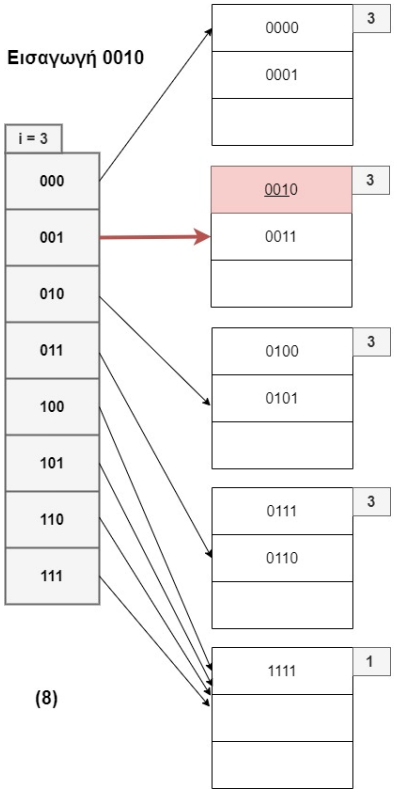
1)

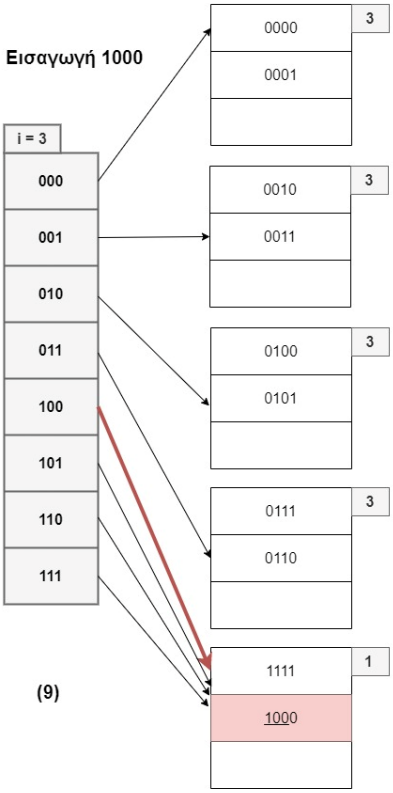


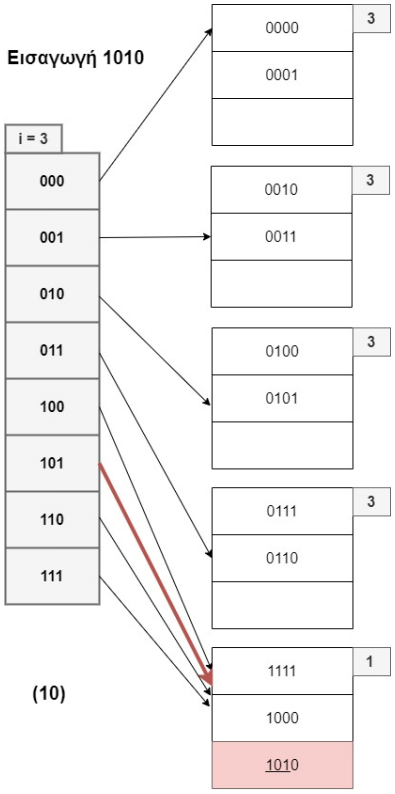

Εισαγωγή 0100
i = 3
000
001
010
011
100
101
110
111
0000
2
0100
3
0101
0111
3
0110
1111
1
(5)


Εισαγωγή 0011
i = 3
000
001
010
011
100
101
110
111
0000
2
0011
0100
3
0101
0111
3
0110
1111
1
(6)


Εισαγωγή 0001
i = 3
000
001
010
011
100
101
110
111
0000
2
0011
0001
0100
3
0101
0111
3
0110
1111
1
(7)


Εισαγωγή 0010
i = 3
000
001
010
011
100
101
110
111
0000
3
0001
0010
3
0011
0100
3
0101
0111
3
0110
1111
1
(8)

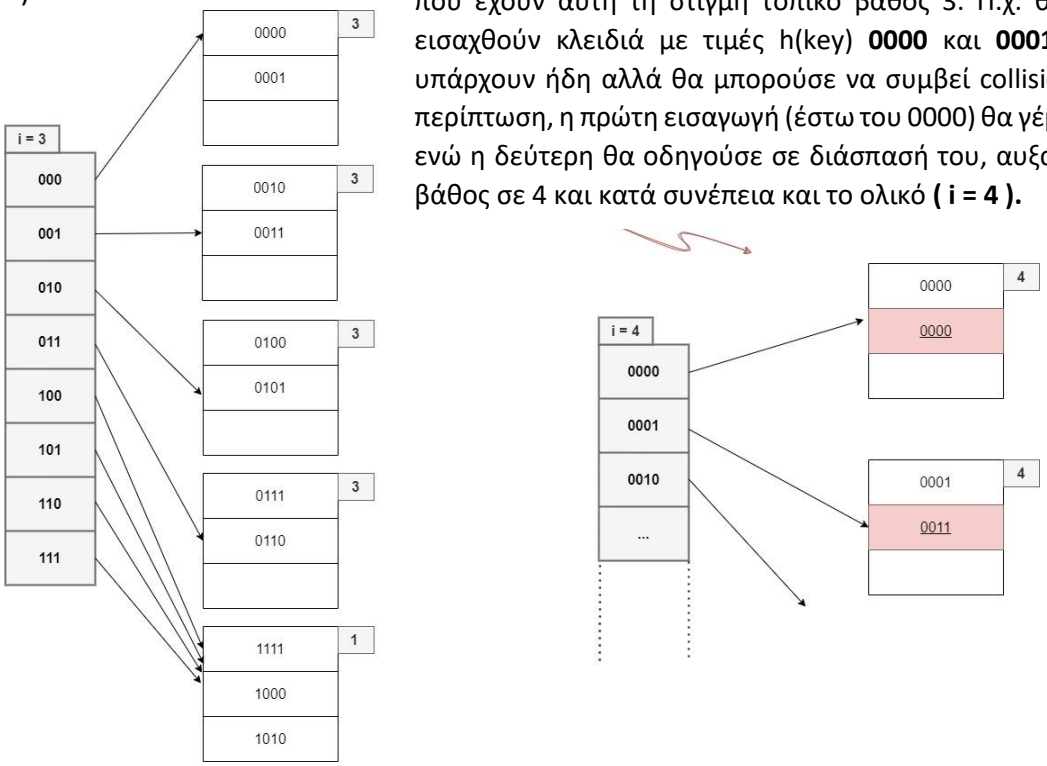

Εισαγωγή 1000
i = 3
000
001
010
011
100
101
110
111
0000
3
0001
0010
3
0011
0100
3
0101
0111
3
0110
1111
1
1000
(9)


Εισαγωγή 1010
i = 3
000
001
010
011
100
101
110
111
0000
3
0001
0010
3
0011
0100
3
0101
0111
3
0110
1111
1
1000
1010
(10)

Για να διπλασιαστεί το ευρετήριο, θα πρέπει να αυξηθεί κάποιο από τα τοπικά βάθη και να ξεπεράσει το ολικό βάθος (το οποίο είναι i=3).

Αρκεί λοιπόν να εισαχθούν 2 τιμές στο ίδιο bucket μεταξύ των buckets που έχουν αυτή τη στιγμή τοπικό βάθος 3. Π.χ. θα μπορούσαν να εισαχθούν κλειδιά με τιμές h(key) **0000** και **0001** (οι τιμές αυτές υπάρχουν ήδη αλλά θα μπορούσε να συμβεί collision). Σε αυτήν την περίπτωση, η πρώτη εισαγωγή (έστω του 0000) θα γέμιζε το **bucket 000** ενώ η δεύτερη θα οδηγούσε σε διάσπασή του, αυξάνοντας το τοπικό βάθος σε 4 και κατά συνέπεια και το ολικό **(i = 4).**

2)



Άσκηση 6

Έστω ότι το τοπικό βάθος του αρχικού υπερχειλισμένου bucket είναι i_{local}. Εφόσον το bucket υπερχείλισε, συμπεραίνουμε ότι περιείχε N εγγραφές και επιχειρήθηκε να εισαχθεί 1 επιπλέον και μάλιστα όλες οι N+1 αυτές εγγραφές έχουν κοινά τα i_{local} πρώτα bits της τιμής της συνάρτησης κατακερματισμού (έστω x αυτή η ακολουθία bits). Λόγω της υπερχείλισης δημιουργούνται δύο buckets, τα x0 και x1 με τοπικά βάθη $i_{local}'=i_{local} + 1$. Για να εισαχθούν όλες οι εγγραφές (N+1) στο ένα μόνο από τα δύο bucket θα πρέπει το το νέο bit που κοιτάζουμε (το i_{local}'-οστό) να είναι ίδιο για όλες τις N+1 εγγραφές.

Η πιθανότητα λοιπόν είναι:

$P(\text{«Το } i_{local}\text{'-οστό bit των N+1 εγγραφών είναι 0»} \cup \text{«Το } i_{local}\text{'-οστό bit των N+1 εγγραφών είναι 1»}) =$

$P(\text{«Το } i_{local}\text{'-οστό bit των N+1 εγγραφών είναι 0»}) + P(\text{«Το } i_{local}\text{'-οστό bit των N+1 εγγραφών είναι 1»}) =$

N+1 φορές

$$\underbrace{\tfrac{1}{2} * \tfrac{1}{2} * \tfrac{1}{2} * \ldots * \tfrac{1}{2}}_{N+1} = \left(\tfrac{1}{2}\right)^{N+1} + \left(\tfrac{1}{2}\right)^{N+1} = 2*\left(\tfrac{1}{2}\right)^{N+1}$$

Η πιθανότητα η πρώτη εγγραφή να έχει bit 0 στην i_{local}' θέση του $h(key_1)$ είναι ½ γιατί υπάρχουν 2 δυνατές τιμές για το bit (0 και 1)

Η πιθανότητα η δεύτερη εγγραφή να έχει bit 0 στην i_{local}' θέση του $h(key_2)$ είναι ½ γιατί υπάρχουν 2 δυνατές τιμές για το bit (0 και 1)

Όπως αναμέναμε, η πιθανότητα αυτή μειώνεται όσο αυξάνεται το μέγεθος των buckets (N).