ΣΧΕΔΙΑΣΜΟΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Αποστολόπουλος Θεμιστοκλής

p3180013

1η σειρά ασκήσεων

1)

a) Σε ένα ίχνος(track) έχω 256 τομείς(sectors).Ο 1 τομέας έχει χωρητικότητα 4096bytes.Άρα 1 ίχνος έχει χωρητικότητα 256*4096bytes=1.048.576bytes(περίπου 1mb)

1 επιφάνεια έχει 65.536 ίχνη(tracks). 1 track έχει χωρητικότητα 1.048.576 bytes(περίπου 1mb για ευκολία).Άρα μία επιφάνεια έχει χωρητικότητα περίπου 1mb*65.536=65.536mb(ακριβής χωρητικότητα = 68.719.476.736bytes που είναι περίπου 68,71gb)

1 δίσκος έχει 8 πλακέτες **διπλής** όψης.1 όψη της πλακέτας(επιφάνεια) έχει χωρητικότητα 68.719.476.736bytes.Άρα ο δίσκος έχει χωρητικότητα 8*2*68.719.476.736bytes το οποίο είναι περίπου ίσο με 1*10^12 bytes ,δηλαδή 1TB.

b)Τα ίχνη,στοιχισμένα το ένα πάνω στο άλλο «δημιουργούν» έναν κύλινδο.Άρα για να βρώ τον αριθμό των κυλίνδρων αρκέι να γνωρίζω τον αριθμό των tracks σε μία διαφάνεια.Επομένως έχω 65.536 κυλίνδρους.

c)Σε 60sec έχω 7.200 περιστροφές

Σε χsec έχω 0.5 περιστροφή

Μετά από λύση του «συστήματος» βρίσκω ότι η μέση καθυστέρηση περιστροφής είναι χ=30/7200=0,004166sec=4,17msec.

Η μέγιστη καθυστέρηση περιστροφής = χ*2=8,34msec

d)Σε μία περιστροφή μεταφέρεται 1 ίχνος στη RAM.Η περιστροφή αυτή βρήκα ότι χρειάζεται 8,34msec=0,00834sec.Άρα έχω: transfer rate = χωρητικότητα που μεταφέρεται / χρόνος = 1.048.576(χωρητικότητα ίχνους)bytes / 0,00834sec = 125.728.537 bytes/sec.

2)

(block = σελίδα)

Έχω ότι το μέγεθος μίας εγγραφής ισούται με 10+50+30+18+20=128bytes.

1 σελίδα χωράει 1024bytes/128bytes= 8 εγγραφές

Αφού έχω N εγγραφές χρειάζομαι N/8 σελίδες συνολικά για να αποθηκεύσω όλες τις εγγραφές.

Dense:Στον συγκεκριμένο τύπου ευρετηρίου πρέπει να έχω αναφορα **σε όλες** τις εγγραφές κάθε σελίδας. Η αναφορά σε κάποια εγγραφή έχει κόστος 10bytes(a) + 6bytes(pointer) = 16bytes.Έχω N εγγραφές άρα χρειάζομαι N*16bytes να μπουν στο ευρετήριο. Κάθε σελίδα χωράει 1024 bytes, επομένως χρειάζομαι 16*N/1024 = N/64 σελίδες

Sparse: Στον συγκεκριμένο τύπου ευρετηρίου πρέπει να έχω αναφορά στην 1^{η} εγγραφή κάθε σελίδας. Η αναφορά σε κάποια εγγραφή έχει κόστος 10bytes(a) + 6bytes(pointer) = 16bytes και έχω N/8 σελίδες συνολικά, άρα χρειάζομαι 16*N/8=2*N bytes. Κάθε σελίδα χωράει 1024 bytes, επομένως χρειάζομαι

2*N/1024 = N/512 σελίδες.

3)

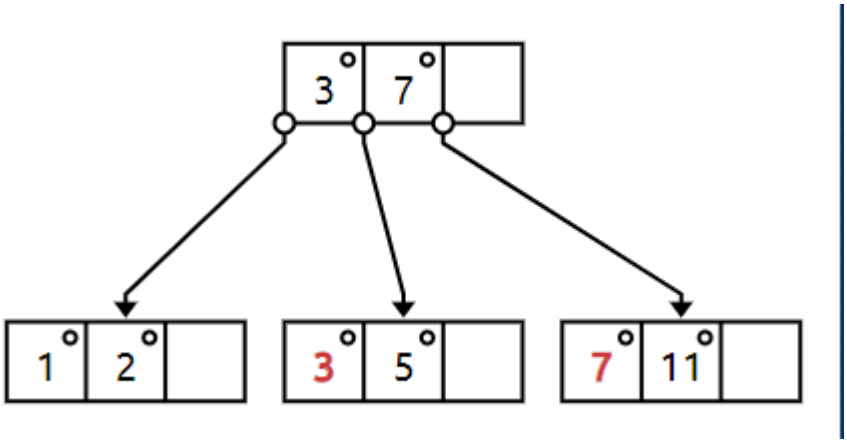
1)Ξεκινάμε από την ρίζα του δέντρου.Βλέπω ότι 41>13 άρα θα πάω στο δεξί παιδί της ρίζας(23,31,43).Τώρα έχω ότι 31<41<43 άρα πηγαίνω στο τρίτο παιδί του συγκεκριμένου κόμβου που είναι φύλο(31,37,41).Εκέι βρίσκω την τιμή 41.

2) Ξεκινάμε από την ρίζα του δέντρου.Βλέπω ότι 40>13 άρα θα πάω στο δεξί παιδί της ρίζας(23,31,43).Τώρα έχω ότι 31<40<43 άρα πηγαίνω στο τρίτο παιδί του συγκεκριμένου κόμβου που είναι φύλο(31,37,41).Εκέι δεν βρίσκω την τιμή 40 άρα δεν υπάρχει στο δέντρο μου.

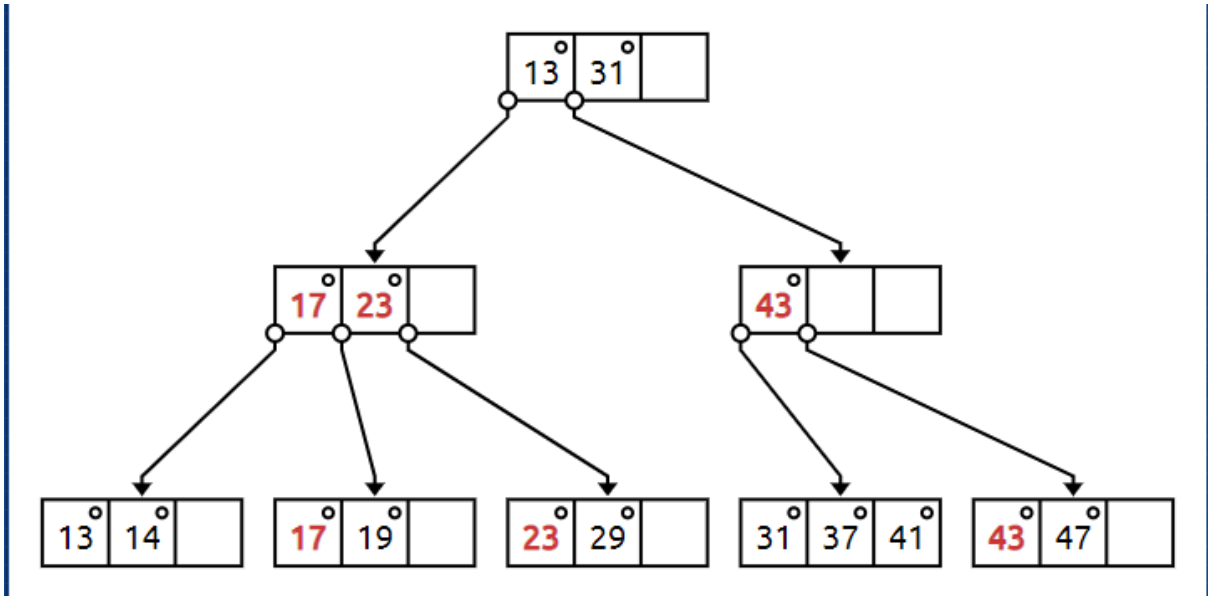
3).Από την ρίζα κατεβαίνω στο αριστερό παιδί και ύστερα πάλι στο αριστερό παιδί .Ξεκινάω να πάω σειριακά από το φύλο που κατέληξα(που είναι το αριστερό φύλο του δέντρου) στα επόμενα φύλα ελέγχοντας αν οι τιμές είναι μικρότερες του 30.εδώ έχω την ακόλουθη ακολουθία:2,3,5,7,11,13,17,19,23,29.

4)Θέλω να βρω το φύλο που θα υπήρχε η τιμή 20.Απο την ρίζα,το 20>13 άρα πάω στο δεξί παιδί(κόμβος με τιμές 23,31,43).Εκεί βλέπω πως 20<31 άρα πάω στο πρώτο παιδί του κόμβου.Έχω φτάσει στο φύλο όπου αν υπήρχε η τιμή 20 θα την έβλεπα εκεί.Παρατηρώ ότι το 20>=13,17,19 άρα πάω στο επόμενο φύλο.Από εδώ και πέρα κάνω σειριακή αναζήτηση με την παράμετρο τιμή<=35.Όταν δεν ισχυεί η συνθήκη, τερματίζω την αναζήτηση.(εδώ έχουμε τα εξής αποτελέσματα:23,29,31

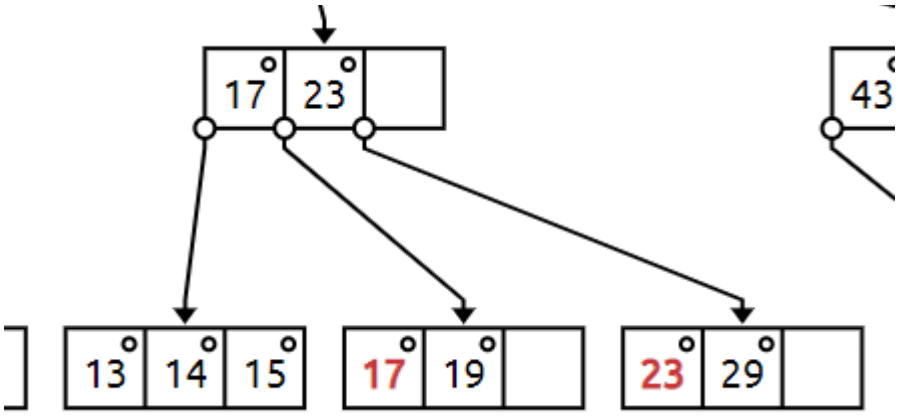
5)Πραγματοποιώ αναζήτηση της τιμής 1 που θέλω να εισάγω στο δέντρο.Έχω ότι 1<13 άρα πάω στο αριστερό παιδί της ρίζας.επείτα έχω ότι 1<7 άρα φτάνω στο αριστερό δέντρο του φύλου.η τιμή μου πρέπει να μπεί στο συγκεκριμένο φύλο αλλά ειναί γεματό.σπάω το φύλο σε 2 φύλα: (1,2),(3,5) και βάζω και στον πατέρα την τιμή 3.φτάνω σε αυτή την μορφή:



6)για την εισαγωγή του 14 πάλι πρέπει να βρώ που πρέπει να μπεί οποτέ θα κάνω αναζήτηση.(14>13 άρα πάω στο δεξί παιδι της ρίζας, επείτα 14<23 άρα φτάνω στο φύλο (13,17,19).Το φύλο αυτό είναι γεμάτο οπότε το σπάω σε: (13, 14), (17,19) και βάζω την τιμή 17 στον πατέρα.Ο πατέρας(23,31,43) είναι γεμάτος οπότε τον σπάω σε : (17, 23),(31,43). Η τιμή 31 φεύγει από τον κόμβο και πάει στην ρίζα.Οι αλλαγές φαίνονται εδώ:

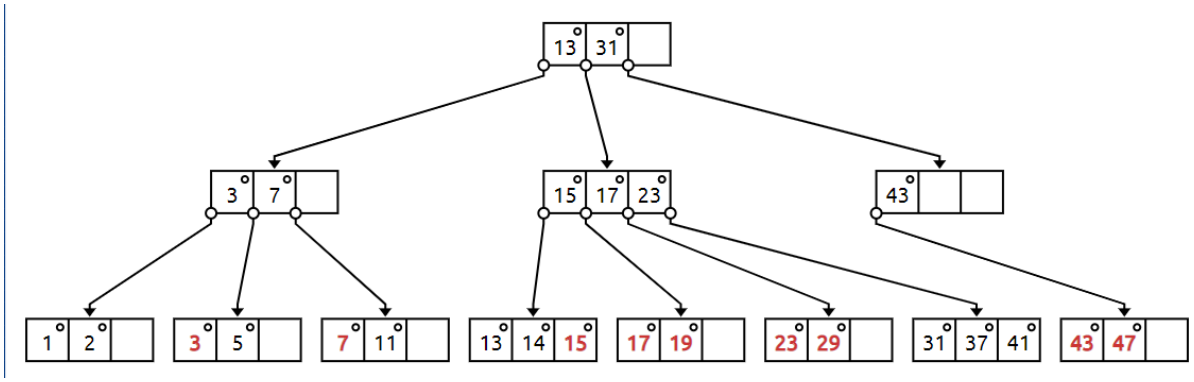


Για γην τιμή 15 κάνω αναζήτηση(13<15<31) πάω στο αριστερό παιδί της ρίζας που με «στέλνει» μεταξύ 13 και 31, επείτα 15<17 πάω στο αριστερό παιδί του κόμβου και φτάνω στο κατάλληλο φύλο).Το 15 χωράει στο φύλο άρα το προσθέτω.Οι αλλαγές :



Για την προσθή του 16 θα βρώ πάλι που πρέπει να μπεί(13<16<31→ 16<17 → 13,14,15).Το φύλο είναι γεμάτο άρα το σπάω σε 2: (13,14), (15,16). Η τιμή 15 πρέπει να προσθεθεί στον πατέρα και ο συγκεκριμένος κόμβος γεμίζει(15,17,23).

ΤΕΛΙΚΗ ΜΟΡΦΉ:



4)

Γνωρίζουμε ότι στα β+ δέντρα ισχύει: μπορώ να έχω n κλειδιά και n+1 pointers σε κάθε κόμβο.Έχουμε ότι το μέγεθος κλειδιού=8bytes και το size του pointer = 12 bytes. Θέλω 8*n + (n+1)*12<=2048 ⇔n<=2036 / 20 ⇔ n<=101,8

Άρα σε έναν κόμβο χωράνε 101 κλειδία.Για να βρω τις εγγραφές πρέπει να πάω στα φύλα. Το πρώτο επίπεδο έχει 101 κλειδιά και 102 pointer.Στο $2^{ο}$ έχω

101*102 κλειδιά (με σκεπτικό ότι αν n =3 στην ρίζα έχω 3 κλειδιά και 4 pointers.άρα στο επόμενο επίπεδο θα έχω 3*4=12keys) και 102^2 pointers(όπως είπα και πριν με το ίδιο σκεπτικό για ν=3 στο $2^{ο}$ επίπεδο θα έχω 4^2 pointers).Τέλος στο $3^{ο}$ επίπεδο που βρίσκονται και οι εγγραφές έχω 102*102*101 keys.Απο εκεί φεύγει ένας pointer για κάθε key σε μία εγγραφή.Άρα χωράνε max 102*102*101 εγγραφές

5)

Θα δείξω τις αλλαγές στα μπακετς όσο γίνονται εισαγωγές.

Εισαγωγή 0000(m=1, i=1, u=1/6)

0000	
0	1

Εισαγωγη 0001(m=1, i=1, u=2/6)

0000	0001

0 1

Εισαγωγη 0001(m=1, i=1, u=3/6)

	0001
0000	0001

0 1

Εισαγωγη 0101(m=1, i=1, u=4/6)

	0101
	0001
0000	0001

0 1

Εισαγωγη 0111(m=1, i=1, u=5/9)

0111

	0101
	0001
0000	0001

0 1

Εισαγωγη 0010(m=1, i=1, u=6/9)

0111

	0101
0010	0001
0000	0001
0	1

Εισαγωγη 0111(m=1, i=1, u=7/9>0.7)

0111
0111

	0101
0010	0001
0000	0001
0	1

Που γίνεται m=10 i=2 u=7/12

0111
0111

	0101	
	0001	
0000	0001	0010
00	*1	10

Εισαγωγή 0010 m=10, i=2 u=8/12

0111
0111

	0101	
	0001	0010
0000	0001	0010
00	*1	10

Εισαγωγή 0011 m=10 i=2 u=9/12>0.7

0011
0111
0111

	0101	
	0001	0010
0000	0001	0010

00 *1 10

Που γίνεται

	0101		0011
	0001	0010	0111
0000	0001	0010	0111

00 01 10 11

	0101		0011	
	0001	0010	0111	
0000	0001	0010	0111	

0000 *01 0010 *11 100

Εισαγωγή 0100 μ=100 i=3 u=10/15

	0101		0011	
	0001	0010	0111	
0000	0001	0010	0111	0100

0000 *01 0010 *11 100

6)

A)

Έχω ότι το ολικό βάθος του ευρετηρίου είναι 10.Αυτό σημαίνει ότι χρησιμοποιούνται 10 bytes για την δημιουργία εγγραφών στο ευρετήριο(δηλαδή έχω 2^10 συνδυασμούς).Άρα το ευρετήριο έχει 2^10 εγγραφές.

B)

Έχω ότι κάθε εγγραφή έχει μέγεθος 4 bytes. Επομένως η συνολική χωρητικότητα του ευρετηρίου θα είναι 4*2^10 bytes.

C)

Έχω ότι το μέγεθος ενός κάδου είναι 2400 bytes.Η εγγραφή δεδομένων είναι 400bytes άρα 1 bucket χωράει 2400/400=6 εγγραφές

Αφού ψάχνω τον μέγιστο αριθμό εγγραφών που περιέχει το αρχείο δεδομένων, σίγουρα είμαστε στην περίπτωση όπου από κάθε εγγραφή του ευρετηρίου «φεύγει» ένας pointer σε κάποιο,μοναδικό bucket.Άρα έχω 2^10 buckets.Επομένως ο μέγιστος αριθμός εγγραφών που περιέχει το αρχείο δεδομένων είναι 6*2^10.

7)

Διαφωνώ με την συγκεκριμένη τεχνική.Με αυτό τον τρόπο θα έχουμε πολύ πιο γεμάτα τα μεσαία διαστήματα ενώ τα αρχικά και τα τελευταία θα είναι σχετικά άδεια.Για παράδειγμα αν θεωρήσουμε ότι ο μέσος μισθός για την συγκεκριμένη επιχειρήση είναι το 4500 θα έχουμε τα περισσοτέρα δεδομένα στο διάστημα [4000,5000) και στα «δίπλα» διαστήματα του ενώ τα άλλα διαστήματα θα είναι σχετικά άδεια.Έτσι δεν θα έχω «ομοιόμορφη» κατανομή των εγγραφών,με αποτέλεσμα άδεια buckets και σπατάλη μνήμης.