
Database Systems

Instructors: Hao-Hua Chu
Winston Hsu
Fall Semester, 2007

Assignment 4 Solutions

Questions

1. Consider the page format for variable-length records that uses a slot directory.
 - a. One approach to managing the slot directory is to use a maximum size (i.e., a maximum number of slots) and allocate the directory array when the page is created. Discuss the pros and cons of this approach with respect to the approach discussed in the text.
 - b. Suggest a modification to this page format that would allow us to sort records (according to the value in some field) without moving records and without changing the record ids.

Ans:

- a. This approach is simpler, but less flexible. We can easily either allocate too much space for the slot directory or too little, since record lengths are variable and it is hard to estimate how many records are likely to fit on a given page.
 - b. One modification that would allow records to be sorted by a particular field is to store slot entries as <logical record number within page, offset> pairs and sort these based on the record's field value.
-
2. Modern disk drives store more sectors on the outer tracks than the inner tracks. Since the rotation speed is constant, the sequential data transfer rate is also higher on the outer tracks. The seek time and rotational delay are unchanged. Given this information, explain good strategies for placing files with the following kinds of access patterns:
 - a. Frequent, random accesses to a small file (e.g., catalog relations).
 - b. Sequential scans of a large file (e.g., selection from a relation with no index).
 - c. Random accesses to a large file via an index (e.g., selection from a relation via the index).
 - d. Sequential scans of a small file.\

Ans:

- a. Place the file in the middle tracks. Sequential speed is not an issue due to the small size of the file, and the seek time is minimized by placing files in the center.

- b. Place the file in the outer tracks. Sequential speed is most important and outer tracks maximize it.
 - c. Place the file and index on the inner tracks. The DBMS will alternately access pages of the index and of the file, and so the two should reside in close proximity to reduce seek times. By placing the file and the index on the inner tracks we also save valuable space on the faster (outer) tracks for other files that are accessed sequentially.
 - d. Place small files in the inner half of the disk. A scan of a small file is effectively random I/O because the cost is dominated by the cost of the initial seek to the beginning of the file.
3. Consider the B+ tree index of order $d = 2$ shown in the figure below.
- a. Show the tree that would result from inserting a data entry with key 9 into this tree.
 - b. Show the B+ tree that would result from inserting a data entry with key 3 into the original tree. How many page reads and page writes does the insertion require?
 - c. Show the B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the left sibling is checked for possible redistribution.
 - d. Show the B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the right sibling is checked for possible redistribution.
 - e. Show the B+ tree that would result from starting with the original tree, inserting a data entry with key 46 and then deleting the data entry with key 52.
 - f. Show the B+ tree that would result from deleting the data entry with key 91 from the original tree.
 - g. Show the B+ tree that would result from starting with the original tree, inserting a data entry with key 59, and then deleting the data entry with key 91.
 - h. Show the B+ tree that would result from successively deleting the data entries with keys 32, 39, 41, 45, and 73 from the original tree.

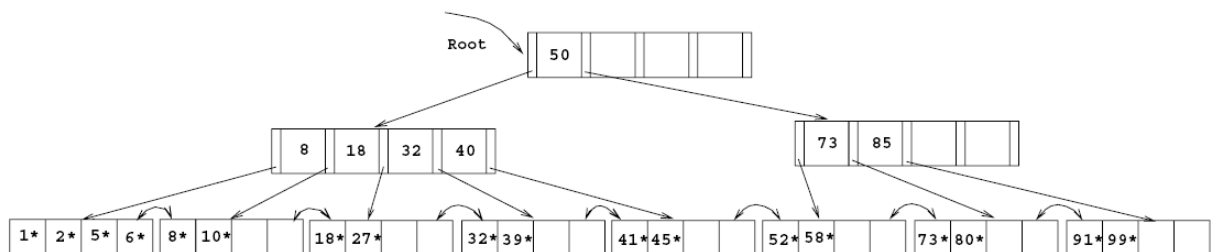
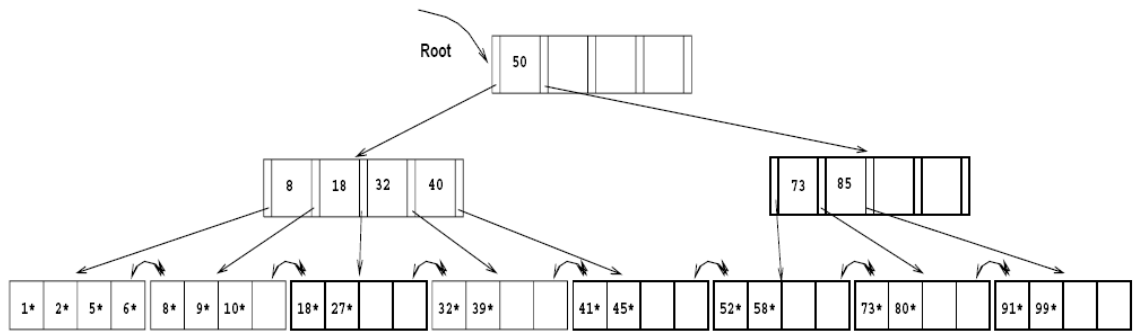


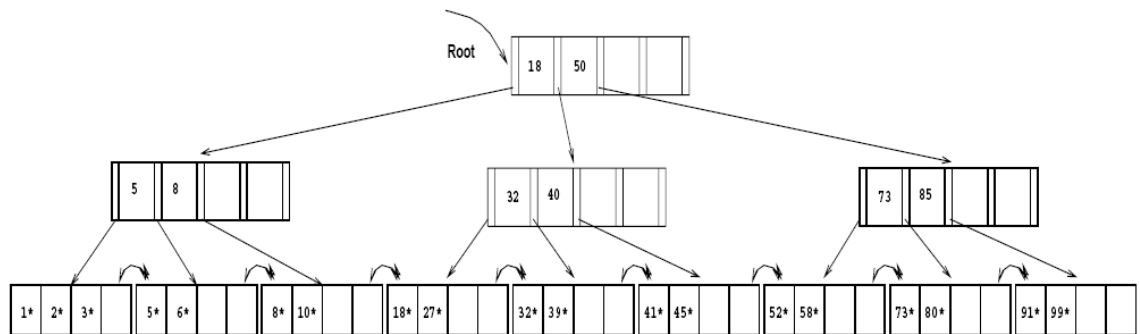
Figure for Question 3

Ans:

a.

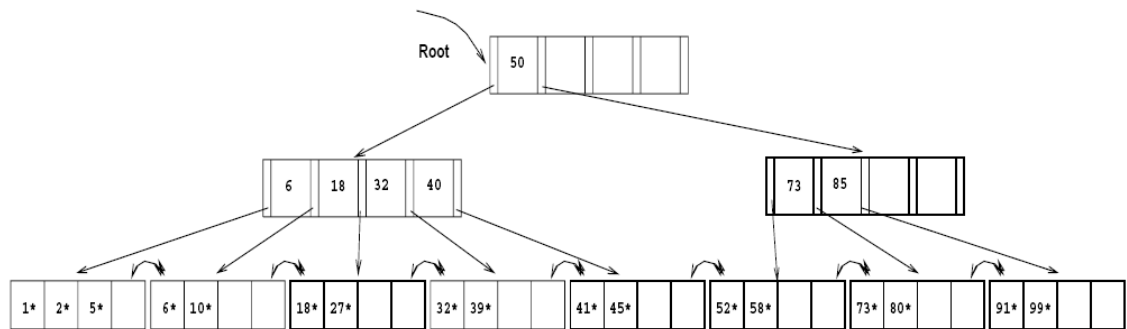


b.

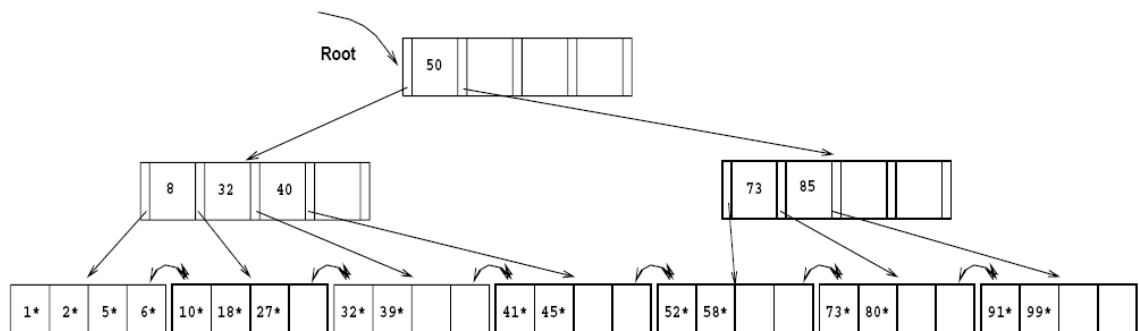


The insertion will require 5 page writes, 4 page reads and allocation of 2 new pages.

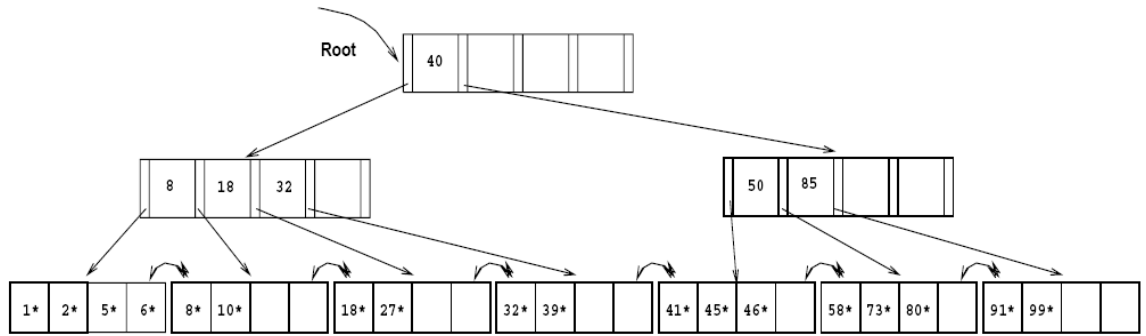
c.



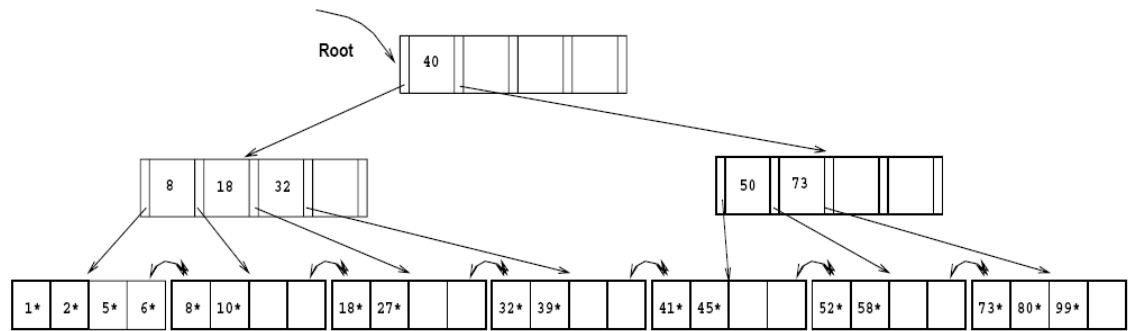
d.



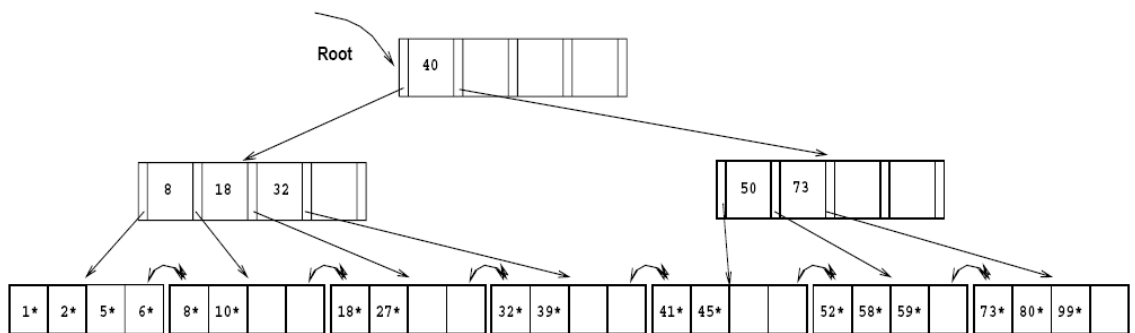
e.



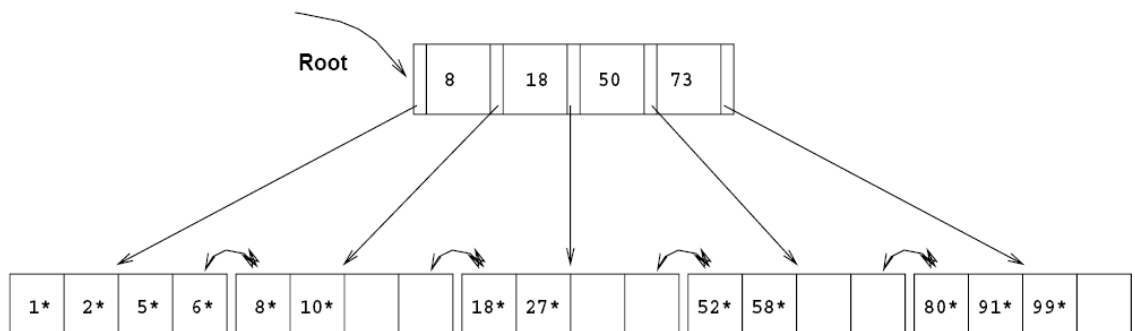
f.



g.



h.



4. Consider the B+ tree index shown in the figure below, which uses Alternative (1) for data entries. Each intermediate node can hold up to five pointers and four key values. Each leaf can hold up to four records, and leaf nodes are doubly linked as usual, although these links are not shown in the figure. Answer the following questions.
- Name all the tree nodes that must be fetched to answer the following query: "Get all records with search key greater than 38."
 - Insert a record with search key 109 into the tree.
 - Delete the record with search key 81 from the (original) tree.
 - Name a search key value such that inserting it into the (original) tree would cause an increase in the height of the tree.
 - Note that subtrees A, B, and C are not fully specified. Nonetheless, what can you infer about the contents and the shape of these trees?
 - How would your answers to the preceding questions change if this were an ISAM index?
 - Suppose that this is an ISAM index. What is the minimum number of insertions needed to create a chain of three overflow pages?

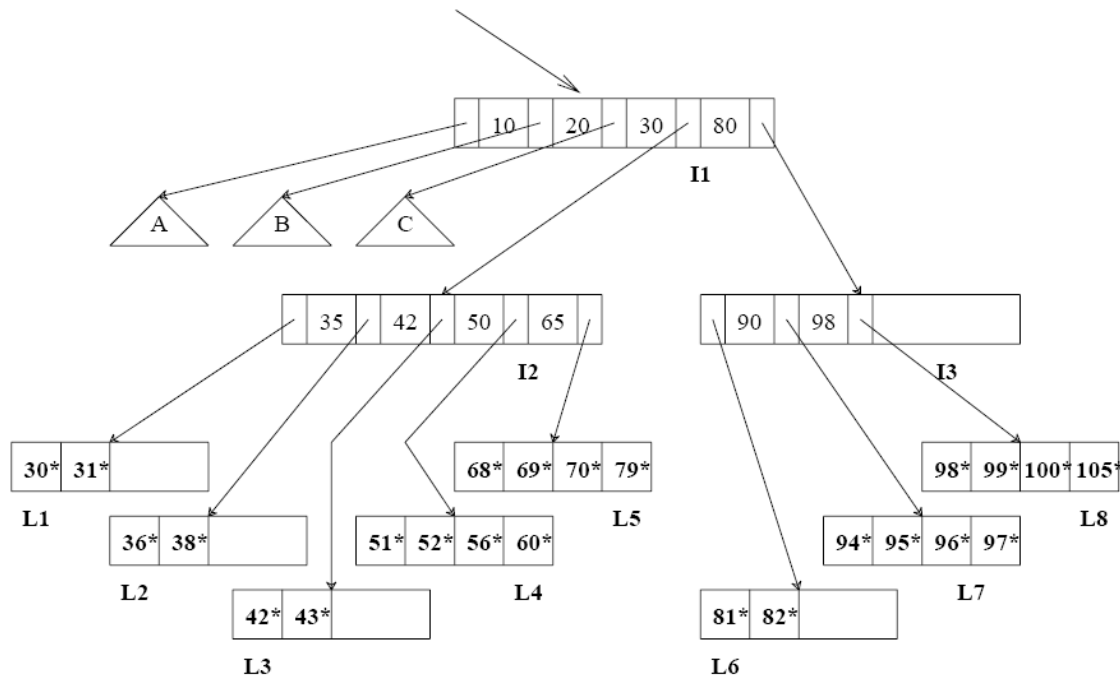
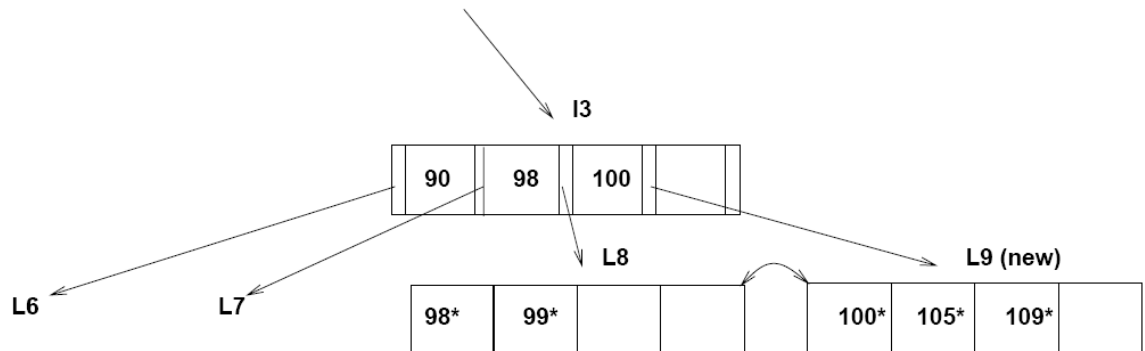


Figure for Question 4

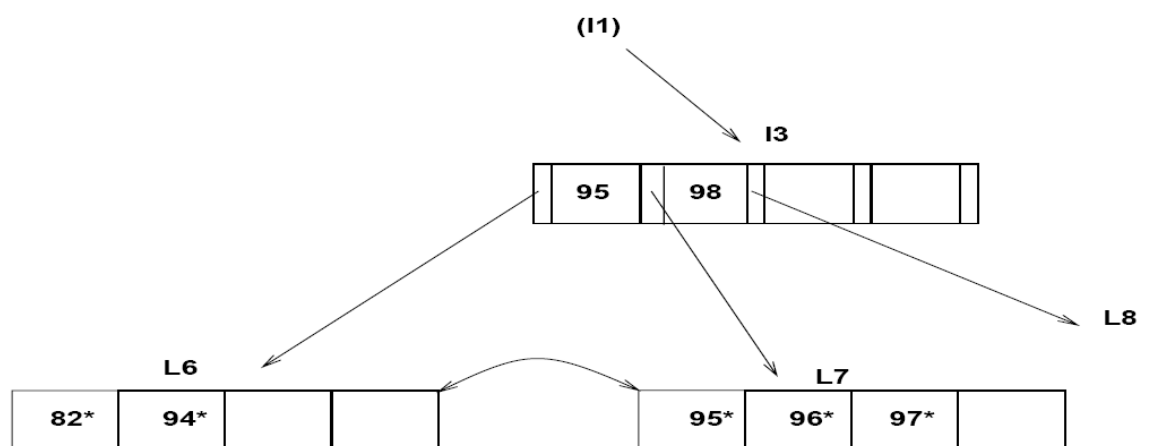
Ans:

a. I1, I2, and everything in the range [L2..L8].

b.



c.



d. There are many search keys X such that inserting X would increase the height of the tree. Any search key in the range $[65..79]$ would suffice. A key in this range would go in L5 if there were room for it, but since L5 is full already and since it can't redistribute any data entries over to L4 (L4 is full also), it must split; this in turn causes I2 to split, which causes I1 to split, and assuming I1 is the root node, a new root is created and the tree becomes taller.

e. We can infer several things about subtrees A, B, and C. First of all, they each must have height one, since their "sibling" trees (those rooted at I2 and I3) have height one. Also, we know the ranges of these trees (assuming duplicates fit on the same leaf): subtree A holds search keys less than 10, B contains keys ≥ 10 and < 20 , and C has keys ≥ 20 and < 30 . In addition, each intermediate node has at least 2 key values and 3 pointers.

- f. The answers for the questions above would change as follows if we were dealing with ISAM trees instead of B+ trees.
- (a) This is only a search, so the answer is the same. (The tree structure is not modified.)
 - (b) Because we can never split a node in ISAM, we must create an overflow page to hold inserted key 109.
 - (c) Search key 81 would simply be erased from L6; no redistribution would occur (ISAM has no minimum occupation requirements).
 - (d) Being a *static* tree structure, an ISAM tree will never change height in normal operation, so there are no search keys which when inserted will increase the tree's height. (If we inserted an X in [65..79] we would have to create an overflow page for L5.)
 - (e) We can infer several things about subtrees A, B, and C. First of all, they each must have height one, since their "sibling" trees (those rooted at I2 and I3) have height one. Here we suppose that we create a balanced ISAM tree. Also, we know the ranges of these trees (assuming duplicates fit on the same leaf): subtree A holds search keys less than 10, B contains keys ≥ 10 and < 20 , and C has keys ≥ 20 and < 30 . Additionally, each of A, B, and C contains five leaf nodes (which may be of arbitrary fullness), and these nodes are the first 15 consecutive pages prior to L1.
- g. If this is an ISAM tree, we would have to insert at least nine search keys in order to develop an overflow chain of length three. These keys could be any that would map to L4, L5, L7, or L8, all of which are full and thus would need overflow pages on the next insertion. The first insert to one of these pages would create the first overflow page, the fifth insert would create the second overflow page, and the ninth insert would create the third overflow page (for a total of one leaf and three overflow pages).

1. In a database system, there are 4 frames available in the memory buffer.

Frame1 pin_count=0 dirty=1 Last Access: 4 th ms P ₁	Frame2 pin_count=1 dirty=0 Last Access: 1 st ms P ₂
Frame3 pin_count=0 dirty=1 Last Access: 3 rd ms P ₃	Frame4 pin_count=0 dirty=0 Last Access: 2 nd ms P ₄

Three operations, namely read, modify, and release can be performed on the pages. The read operation assumes that a new user is accessing a page (either already in the buffer or not) and reading the page without changing it. A release operation does not access the page but indicates that one of the previous users does not need it anymore. A modify operation assumes that a new user is modifying the page if it is already in the buffer, or reading a new page from the disk and modifying it. Please show how read, modify, and release operations are changing the last access time, pin_count and dirty attributes of the frame. Consider that the following operations are performed:

5	6	7	8	9	10	11	12
<i>read</i> (P ₁)	<i>release</i> (P ₂)	<i>modify</i> (P ₄)	<i>release</i> (P ₁)	<i>read</i> (P ₅)	<i>read</i> (P ₆)	<i>release</i> (P ₄)	<i>read</i> (P ₇)

- a) Show the contents of the memory buffer after the 12th millisecond if Least Recently Used (LRU) replacement policy is applied.

Frame1 pin_count = 1 dirty= 0 Last Access=12 Page= P7	Frame2 pin_count = 1 dirty= 0 Last Access= 9 Page= P5
Frame3 pin_count = 1 dirty= 0 Last Access= 10 Page= P6	Frame4 pin_count = 0 dirty= 1 Last Access= 7 Page= P4

- b) Show the contents of the memory buffer after the 12th millisecond if Most Recently Used (MRU) replacement policy is applied.

Frame1 pin_count = 1 dirty= 0 Last Access= 9 Page= P5	Frame2 pin_count = 0 dirty= 0 Last Access= 1 Page= P2
Frame3 pin_count = 1 dirty= 0 Last Access= 10 Page= P6	Frame4 pin_count = 1 dirty= 0 Last Access= 12 Page= P7

- c) How many disk accesses are needed if LRU is used? Indicate which pages are written to and read from the disk.

3 read operations (P5, P6, P7) + 2 write operations (P1, P3) = 5 disk accesses

- d) How many disk accesses are needed if MRU is used? Indicate which pages are written to and read from the disk.

3 read operations (P5, P6, P7) + 3 write operations (P1, P3, P4) = 6 disk accesses

2. We have got two tables stored in two different files with the given schemas: Students (sid: string, name: string, departmentid: integer) and Departments(departmentid: integer, departmentname: string). Students file is 4 GB and departments file is 1 MB. Both tables consist of fixed length records of 64 bytes. The system has got a block size of 1KB bytes.

Each index entry stores a search key of 4 bytes and 4 bytes of pointer. (1 GB = 2^{30} , 1 MB = 2^{20} , 1 KB = 2^{10})

- a) How many disk accesses in the worst case is needed to find the student who has student id 20122015 if a heap file is used for the Students file?

$$2^{32}/2^6 = 2^{26} \text{ records in the file}$$

$$2^{10}/2^6 = 2^4 \text{ records in each data block}$$

$$2^{26}/2^4 = 2^{22} \text{ blocks in the file}$$

2²² disk accesses

- b) How many disk accesses is needed to read the name of the student who has student id 20122015 if a sorted file (according to ids) and a sparse index (built on the student id) is used?

$$2^{10}/2^3 = 2^7 \text{ entries in each index block}$$

$$2^{22}/2^7 = 2^{15} \text{ blocks in the sparse index}$$

$$[\log_2(2^{15})+1] + 1 = 17 \text{ disk accesses}$$

- c) How many disk accesses is needed to find the name of the department of the student with id 20457789 if both files are heap files and using a 2-level index (on student id and department id)?

Students file

$$2^{26}/2^7 = 2^{19} \text{ blocks in the dense index in the first level}$$

$$2^{19}/2^7 = 2^{12} \text{ blocks in the sparse index in the second level}$$

Departments file

$$2^{20}/2^6 = 2^{14} \text{ records in the departments file}$$

$$2^{14}/2^7 = 2^7 \text{ blocks in the dense index in the first level}$$

$$2^7/2^7 = 1 \text{ block in the sparse index in the second level}$$

$$\text{Find department id: } ([\log_2(2^{12})+1] + 1 + 1) + \text{find department name } (1 + 1 + 1) \\ = 18 \text{ disk accesses}$$

- d) How many disk accesses is needed to find the number of students who are studying 'Computer Science' in the best case if the students file is sorted based on the student id (without an index) and the department file is also sorted based on the department id (without an index)? Note that every department has a different name.

$$2^{22} \text{ blocks in the students file}$$

$$2^{14}/2^4 = 2^{10} \text{ blocks in the departments file}$$

$$1 + 2^{22} \text{ disk accesses}$$

Hash-Based Indexing

The least significant (rightmost) bits of the key values is used as the hash key. Each bucket can store up to 4 entries. Consider that you insert the following keys in order.

key	binary key
5	101
64	1000000
9	1001
25	11001
31	11111
15	1111
10	1010
7	111
3	11
44	101100

Linear hashing

- a. (5 pts) Does linear hashing always split the bucket that overflows? Explain briefly.

SOLUTION

No. In case of an insertion into the overflow bucket, next bucket is split. Next bucket is not necessarily the bucket that overflows.

- b. (5 pts) Display the final structure with the decimal key values and the next pointer.

SOLUTION

Next →	00	64	44		
	01	5	9	25	
	10	10			
	11	31	15	7	3

- c. (5 pts) What is the largest key (in decimal) less than 25 whose insertion cause a split?

SOLUTION

23

Extendible Hashing

- a. (5 pts) Display the final structure with the decimal key values. Indicate the global and local depths.

SOLUTION

	2
00	B1
01	B2
10	B1
11	B3

B1 / 1	64	10	44	
B2 / 2	5	9	25	
B3 / 2	31	15	7	3

- b. (5 pts) Give an example key value whose insertion cause a split but no change in the directory?

SOLUTION

The insertion of a key either does not cause a split (if inserted in B1 and B2) or splits and doubles the directory (if inserted in B3).

- c. (5 pts) What is the largest key (in decimal) less than 44 whose insertion cause the directory to double in size?

SOLUTION

43

Tree-Based Indexing

In the following questions, while explaining your solutions please recall that level 0 refers to root level.

B-Tree Indexing

Consider that you are going to create a B tree index for a file of 3000 records with unique keys. The order of buckets is set to 9.

- a. (5 pts) Consider that you are going to create a B tree index for the given set of records. The order of buckets is set to 9. At the worst case, what is be the maximum number of disk page access to locate a record with a given key? Explain briefly.

SOLUTION

In order to find the maximum number of possible disk page accesses, the buckets should be filled with at least order of records but the root may only store a single record. Hence,

Root 1

Level 1 $2 \times 9 = 18$

Level 2 $2 \times 9 \times 10 = 180$

Level 3 $2 \cdot 9 \cdot 10 \cdot 10 = 1800$

Level 4 cannot exist as there are not enough records to accommodate with the order in each bucket. At the maximum, the search will visit all levels and thus 4 I/O.

- b. (5pts) What is the minimum number of levels in the B tree to accommodate such number of records? Explain briefly.

333

SOLUTION

In order to have the minimum number of levels, the buckets should be filled with the double of the order of records including the root.

Root 18

Level 1 $18 \cdot 19 = 342$

Level 2 $18 \cdot 19 \cdot 19 = 6498$

As it can be seen from the above storage allocation, B tree with Level 1 cannot accommodate all records but Level 2.

B+ Tree Indexing

Consider that you are going to define a B+ tree index for the same file of 3000 records with unique keys. The order of the data buckets is 9 but the order of internal nodes is 200.

- c. (5 pts) At the worst case, what is be the maximum number of disk page access to locate a record with a given key? Explain briefly.

SOLUTION

Root 1 bucket $\Rightarrow \lfloor 333/200 \rfloor = 1$

Data level each bucket store order $\Rightarrow \lfloor 3000/9 \rfloor = 333$

2 disk accesses is required.

- d. (5 pts) What is the minimum number of levels in the B+ tree to index such number of records? Explain briefly.

SOLUTION

Root 1 bucket $\Rightarrow \lceil 167/400 \rceil = 1$

Data level each bucket store order $\Rightarrow \lceil 3000/18 \rceil = 167$

2 level index including the root is required.

ID:

NAME:

BBM 371 – Data Management
Quiz 2

1. True/False Question (20 points)

For each of the following questions, circle T (true) or F (false).

T	F	Considering a file that has variable length records, when a field updated to larger size than the subsequent fields must be shifted.
T	F	In a heap file data records are in sorted order.
T	F	In range search, at the worst case $N/2bf$ blocks are accessed where N and bf denote the number of records and the blocking factor, respectively.
T	F	Equality search in a sorted file without overflow pages is faster than the one with overflow pages.
T	F	Time to update a record of fixed length is equal to time to delete the record and time to add the updated record.
T	F	Deletion of a record does not always incur a shift in the file.
T	F	Index files stores search keys and pointers to data-file records.
T	F	Index size is always smaller than the file size.
T	F	If blocking factor is one than sparse index is equivalent to dense index.
T	F	In multi-level indexing, every level is a sparse index.

2. File Structures and Indexing (80pts)

Consider two files F1 and F2 such that F1 contains 2^{24} records of size 16 bytes and F2 contains 2^{32} records of size 1024 bytes. Each file shares the same key of size 4 bytes. F1 is a sorted file and F2 has a 2 level file index where data entry pointers are of size 4 bytes. Note that file indices are stored in the disk. These files are stored on a single disk with the following specifications:

- Block size: 1024 bytes,
- Speed: 7500 RPM,
- Average seek time: 4ms.,
- Average transfer time: 1ms.

- a. How long does it take to randomly access (find and read) a single record with a given key value in F1?

$$\text{Blocking factor} = 1024/16 = 64 = 2^6$$

$$\# \text{ of blocks} = 2^{24} / 2^6 = 2^{18}$$

$$\text{Rotation Time} = 60000\text{ms} / 7500 = 8 \text{ ms}$$

$$\text{Random Access Time} = s + r + t = (4 + 4 + 1) = 9 \text{ ms}$$

$$\text{Assuming binary search will perform at most } \log B \text{ accesses, } D * \log B = 9 * \log 2^{18} = 162\text{ms}$$

- b. How long does it take to randomly access (find and read) a single record with a given key value in F2?

$$\text{Blocking factor} = 1024/1024 = 1$$

ID:

NAME:

of blocks = 2^{32} 1. level index (As bkfr = 1, dense index in any case) $\rightarrow 1024/8 = 2^7$ records/block $\Rightarrow 2^{32} / 2^7 = 2^{25}$ blocks2. level index(sparse index) $\rightarrow 2^{25} / 2^7 = 2^{18}$ Assuming binary search will perform at most $\log B$ accesses, $D * (2 + \log B) = 9 * (2 + \log 2^{18}) = 180\text{ms}$

c. How long does it take to perform a range search in F2 at the worst case?

 $D * B = 9 * 2^{32} \text{ ms.}$

d. How long does it take to find the common keys in F1 and F2?

As F1 is sorted and 1. level of F2 is sorted, reading blocks from F1 and F2 in order and comparing the keys will be enough to find the common ones. Hence,

 $9 * (2^{18} + 2^{25}) = 1152 * 2^{18}$

Name:

ID:

BBM 371 – DATA MANAGEMENT
QUIZ4
27.12.2018

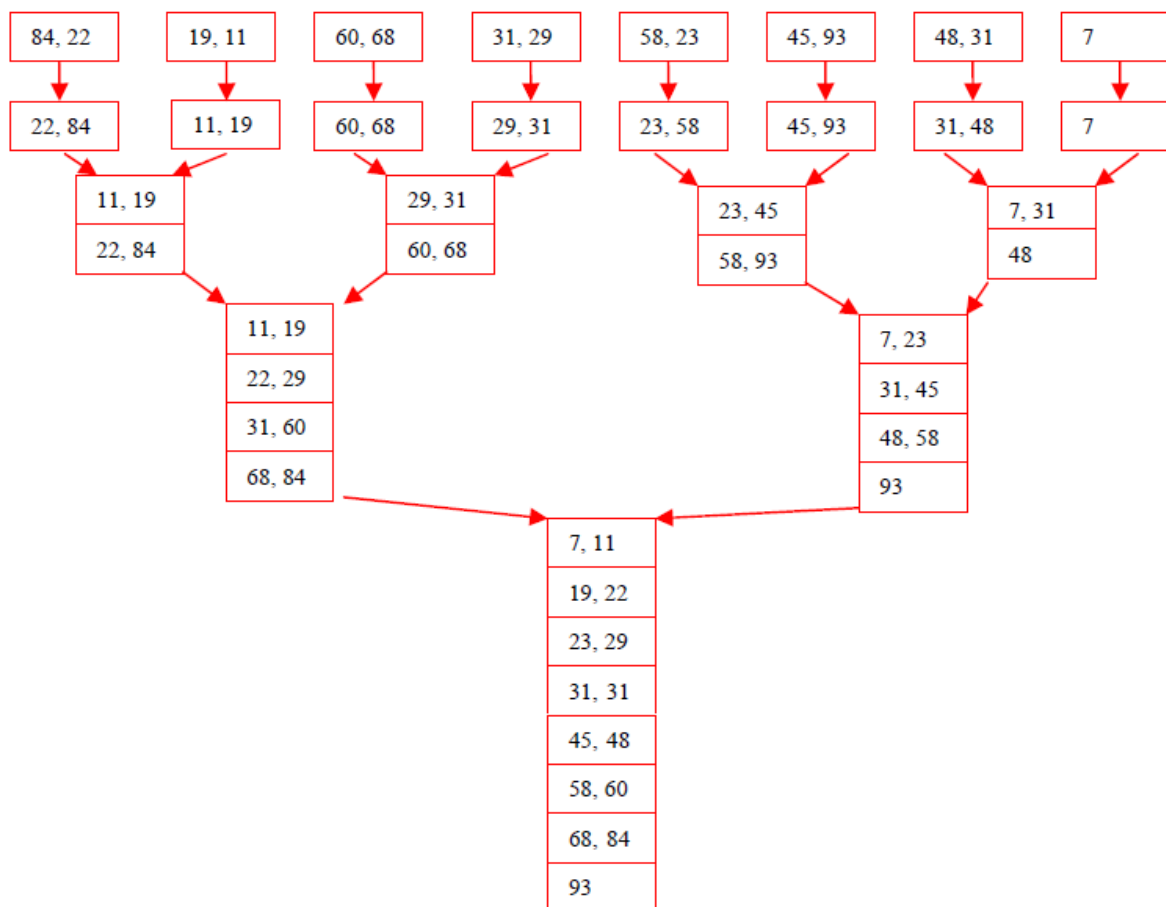
1. Suppose we wish to sort the following values:

84, 22, 19, 11, 60, 68, 31, 29, 58, 23, 45, 93, 48, 31, 7

Assume that:

- you have three pages of memory for sorting
- you will use an external sorting algorithm with a 2-way merge
- a page only holds two values

For each sorting pass, show the contents of all temporary files.



2. If you have 100 pages of data, and 10 pages of memory, what is the minimum number of passes required to sort the data.

$$\# \text{ of passes} = 1 + \lceil \log_{B-1} [N/B] \rceil = 1 + \lceil \log_9 [100/10] \rceil = 1 + \lceil \log_9 10 \rceil = 1 + 2 = 3$$

3. If you have 500,000 pages of data, what is the minimum number of pages of memory required to sort the data in 3 passes?

$$\# \text{ of passes} = 1 + \lceil \log_{B-1} [N/B] \rceil = 3 \text{ so}$$

$$\lceil \log_{B-1} [N/B] \rceil = 2 \Rightarrow [N/B] \leq (B-1)^2 \Rightarrow N \leq B(B-1)^2 \Rightarrow B = 81$$

You can also simplify this to $N \leq B^3$, in which case $B = 80$

BİL353 KÜTÜK DÜZENLEME VE ERİŞİMİ

1. Vize / 87 Puan

18/11/2000

100 dk.

Adı Soyadı :

Numara :

Not Çizelgesi

1	2	3	4	5	Toplam
---	---	---	---	---	--------

Sınav sırasında soru sorulmayacaktır. Sorulara ait sorularınız bir defa sınav başında cevaplandırılacaktır.

Başarılar

Ebru A. Sezer

(15 Puan)

1. Birkaç cümle ile aşağıdaki soruları cevaplayınız.

- A) Veri tabanı yönetim sistemlerinde (DBMS) fiziksel bağımsızlık (*physical independence*) nedir? Neden önemlidir?
- B) Yastık havuzunda (*buffer pool*) yer alan bir sayfanın **pin** ve **count** değişkenleri ne ifade eder? Sayfa yönetimini nasıl etkiler?
- C) Değişken uzunluklu kayıtların (*variable length records*) sayfa içine yerleştirilmelerinde ekleme/silme işlemlerine rağmen adreslerinin değişmemesi nasıl sağlamıştır?
- D) İyi bir anahtarlama fonksiyonunun (*hashing function*) özellikleri nelerdir?
- E) Extendible hashing yönteminde komşu buketlerin (*adjacent buckets*) belli bir eleman sayısının altında eleman taşımaları sonucu birleşmeleri sözkonusudur. İlgili yöntemde boş buket oluşmasına sebep olabilecek iki farklı durum önerebilir misiniz? Nelerdir?

(20 Puan)

2. Sadece doğru, yanlış kelimelerini kullanarak aşağıdaki bilgiler için yargınızı bildiriniz. Yargınızı iki üç cümle ile pekiştiriniz.

- A) Yastık havuzunda (*buffer pool*) yer alan sayfaların **pin** değişkenlerinin güncellenmesi disk yönetim (*disk space management*) sisteminin görevidir.
- B) Cluster, mantıksal blok (*logical block*) olarak da kullanılabilir/düşünülebilir.
- C) Dizin (*index*) kurmada kullanılan arama anahtarının (*search key*) birincil anahtar (*primary key*) olması gerekir.
- D) Seyrek dizinde (*sparse index*) her arama anahtarı ile bir kayıt adreslenirken, sıkışık dizinde (*dense index*) bir arama anahtarı ile bir grup kayıt adreslenebilir.
- E) Bir yığın kütüğe (*heap file*) ait sayfaların yönetiminde; kütüğe ait sayfalar, veri içeren sayfalar listesi ve kütüğe atanmış boş sayfalar listesi olmak üzere ikiye ayrılır. Kütüğe ekleme yapılacaksa boş sayfaları taşıyan listeden bir sayfa seçilir. Ekleme işlemi gerçekleştirildikten sonra sayfa veri içeren sayfaların bulunduğu listeye eklenir.

- F) Her iz (*track*), sektör (*sector*) adı verilen yaylara bölünmüştür. Diske erişim sektör tabanlıdır.
- G) Extendible hashing yöntemi ile oluşturulan dizin sayesinde; verilen bir arama anahtarı değerine karşılık gelen kayıda 1 erişimde ulaşılır. (İstenen kayıt, kütükte kayıtlıdır.)
- H) Extendible hashing yönteminde komşu buketler (*adjacent buckets*) birleştirildikten sonra dizin (*index*) küçültme işlemine geçilir.
- I) Diskte okuma yazma sırasında dönen, disk yüzeyleridir (*platters*).

(27 puan)

3. Elinizde bulunan diskin her sektörü (*sector*) 512 byte'tır. Her yüzeyde (*platter*) 2000 iz (*track*) vardır. Her izde 50 sektör bulunur. Diskte toplam 5 tane çift yüzü (*double sided*) yüzey bulunmaktadır. Ortalama arama zamanı (*average seek time*) 10 msec'tir.

- A) Bir iz ve bir yüzey kapasitesini hesaplayınız.
- B) Diskin toplam kapasitesini hesaplayınız.
- C) Diskte kaç silindir bulunmaktadır, diskler üzerinde tanımlanmış silindir (*cylinder*) kavramının önemini açıklayınız.
- D) Geçerli bir mantıksal blok boyu (*logical block size*) öneriniz. Örneğin 256, 2048, 51200 değerleri sizin için geçerli midir, tartışınız.
- E) Diskin dönüş hızı 54000 rpm (*revolution per minute*) ise maksimum dönüş gecikmesi (*rotational delay*) değerini hesaplayınız.
- F) 54000rpm ile önerdiğiniz blok boyuna göre o bir bloğu okumak için gereken aktarım zamanı (*transfer time*) ve toplam erişim süresini (*access time*) hesaplayınız.

Elinizde 100.000 kayıt içeren bir yığın (*heap*) kütük olsun. Bir kayıt uzunluğu 100 byte'tır ve bir kaydın iki bloğa yayılmasına izin verilmemektedir. Elinizdeki diski dikkate alarak:

- G) D şıkkında önerdiğiniz blok boyuna kaç kayıt sığdığını hesaplayınız.
- H) Tüm kütüğü saklamak için kaç blok gerektiğini hesaplayınız.
- I) Tüm kütüğün kaç yüzey kullanılarak saklanabileceğini hesaplayınız.
- J) Tüm kütüğü okumak için geçen süreyi hesaplayınız.
- K) Tüm kafaların aynı anda okuma yapabildiğini varsayarak; sektörlerin, izlerin diske yerleşiminin nasıl olmasını önerirsiniz, öneriniz erişim süresini nasıl etkiler, açıklayınız.

(10 Puan)

4. Anahtarlama fonksiyonu (*hashing function*) ile bir kütük üzerinde erişim, okuma, yazma işlemleri yapılmaktadır.

M : Kütüğe atanmış fiziksel disk alanı sığası

R : Elde var olan kayıt sayısı

N : Anahtarlama fonksiyonu ile üretilen ayrık ev adresi (*home address*) sayısı

K : Beş tane büyük harften oluşan anahtar

H(K) : 0 ile M-1 arası değer üreten anahtarlama fonksiyonu olsun.

Bu bilgiler ışığında aşağıdaki sorulara cevap veriniz.

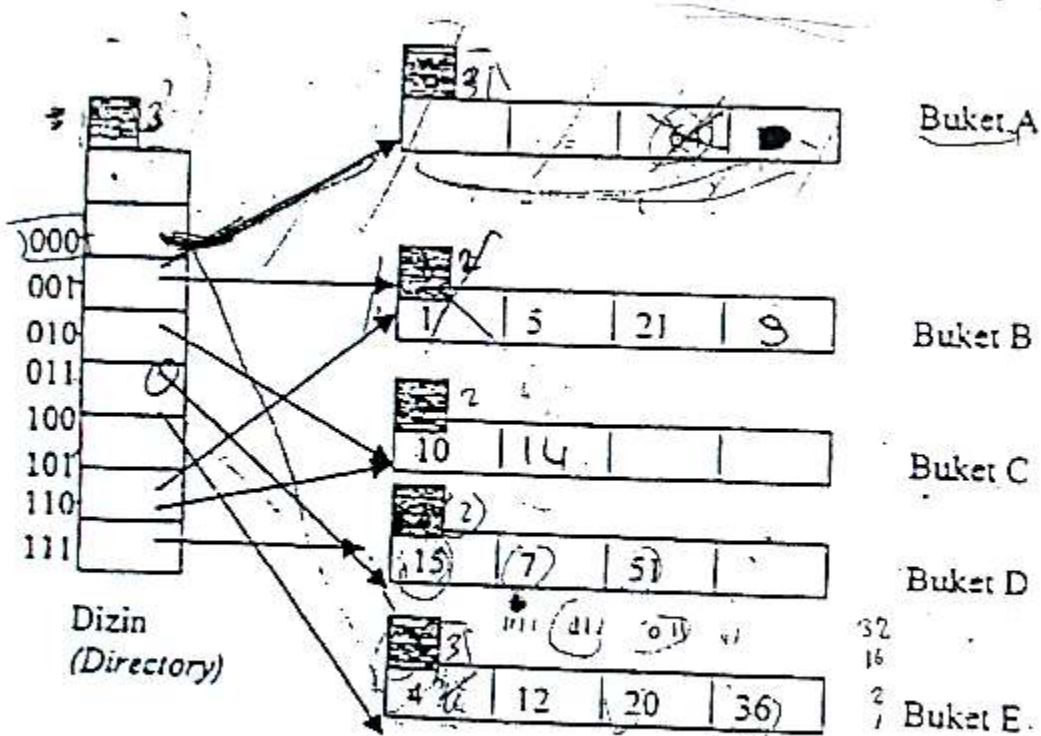
- A) N ve R arasındaki ilişki nasıldır açıklayınız.
- B) R ve M arasındaki ilişki nasıldır açıklayınız.
- C) Eğer H(K) optimum anahtarlama fonksiyonu ise N,R ve M,R arasındaki ilişki nasıl olacaktır, açıklayınız.

(Optimum anahtarlama fonksiyonu, her kayıt için $[0, M-1]$ aralığında ayırık ev adresi (home address) üretmeyi garantiler)

(15 Puan)

5. Çizim 1'de verilen extendible hashing index üzerinde aşağıda belirtilen işlemleri gerçekleştiriniz. Her bir adımı ayrı bir şekil ile ifade edip çözüm yolunuzu kısaca yazınız.

- A) Dizin (directory) üzerinde bölünmeye sebep olmuş en son eleman sizce hangisi Ya da hangileridir.
- B) Arama anahtarı 64 olan kaydı siliniz.
- C) B şikkında oluşan dizin üzerine arama anahtarı 9 ve 14 olan kayıtları ekleyiniz.



izim 1. Soru 6 için örnek

BİL353 KÜTÜK DÜZENLEME VE ERİŞİMİ

26 Kasım 2002

1. Vize

100 Puan / 100 dakika

SORULAR

(25 Puan)

1. Aşağıdaki soruları kısaca cevaplayınız.
 - a. Kütük düzenleme yöntemlerine ihtiyaç duyulma sebepleri nelerdir?
 - b. Anahtarlama yöntemi ile kütük düzenleme hangi ihtiyaç üzerine geliştirilmiştir? Yöntemin hedefi nedir?
 - c. Diske veri yerleştirmede silindir yaklaşımının ortaya çıkma sebebi ve avantajı nedir?
 - d. İdeal olanı bir dizin kütüğünün tamamı ile bellekte tutulmasıdır. Ancak bunun tersi durumlar için önereceğiniz dizin oluşturma biçimleri nelerdir?
 - e. Bildiğiniz tüm birincil ve ikincil bellek türlerini hızlarına göre listeleyiniz.

(23 Puan)

2. Veri Tabanı Yönetim Sistemleri'nin yazılımsal olarak en alt 3 katmanını oluşturan disk yönetim (disk manager), bellek yönetim (buffer manager) ve kütük yönetim (file manager) sistemlerinin görevlerini ve ihtiyaç duyulan bir kaydın belleğe çekilmesi için birbirleri ile kurdukları ilişkiyi açıklayınız.

(25 Puan)

3. Aşağıdaki ifadelerden her biri için doğru ya da yanlış yargılarından birini bir kaç cümle ile birlikte açıklayınız.
 - a. VTYS'de tutulan bütün kayıt türlerinin değişken uzunluklu olması bellek ve hız kıstaslarınca tavsiye edilir.
 - b. Bellek yönetim sisteminde yastık havuzunda öncelikle yığın kütükler tutulur.
 - c. Blok ve sayfa aynı kavramlardır.
 - d. Sıralı kütükler için seyrek, yığın kütükler için yoğun dizin kullanmak anlamlıdır.
 - e. Anahtarlama ile oluşturulan bir dizinleme arama anahtarı üzerinden sıralı liste ile dökme işlemi, arama anahtarı olmayan herhangi bir alan üzerinde sıralı liste dökme işlemine göre en az iki kat hızlı çalışır.

2007-2008 ARA SINAV 1 - EBRU SEZER

- Dizin çeşitleri
 - Anahtar tabanlı
 - Ağaç tabanlı
- Anahtar seçiminde dikkat edilecek hususlar
- yoğun dizin dinamik dizin midir
- B+ agacında verileri sıralı yazdırma en kötü maliyeti,
- dizin...(?)..olmalıdır.Çünkü..(?)..
- external sorting run1 çıktısı
- linear mi extendible mi tercih edersiniz
- B ağacının her zaman dengeli olması nasıl garanti ediliyor

Doç. Dr. Ebru A. Sezer

1-10 Aralığındaki Soruların Çözümleri*1 - 10 aralığındaki sorular için ortak açıklama yapılmıştır*

Elinizde yığın (heap) kütük biçiminde düzenlenmiş ve saklanmış 100.000 adet kayıt bulunmaktadır. Her bir k aydın uzunluğu 250 byte'dır. Bu kütüğün saklandığı diskin fiziksel özellikleri şu biçimdedir: 512 byte/ sektör, 12sektör/iz, 4 sektör/öbek, 1000 iz / yüzey, tek taraflı toplam 5 adet yüzey, dönüş hızı 7200 rpm ve ortalama yatayda arama zamanı 6msn'dir. Kütüğünüzü kullanacağınız sistemde en fazla 20 adet ana bellek sayfasıkullanabilirsiniz. Kayıtlar sektörler arasına yayılmamaktadır ve 64 bitlik mimari kullanılmaktadır . Dizin için kullanılacak anahtar alanın boyu 10 byte'dır. Çift anahtar problemi yoktur.

1. Yığın kütüğünüzün bloklama faktörü

$$512 / 250 = 2.05 \text{ kayıtlar sektörler arasına dağılamadığı için } 2' \text{ dir ve } 2 * 4 = 8$$

2. Silindir öncelikli yerleşimde yığın kütüğü saklamak için gerekli silindir sayısı

$$\text{Bir ize yerleşen kayıt sayısı } 8 * 3 = 24 \text{ ve bir silindire yerleşen kayıt sayısı } 24 * 5 = 120$$

$$100000 / 120 = 833.3 \text{ yani } 834$$

3. Kütüğü baştan sona blok-blok okumanın en kötü maliyeti

$$100000 / 8 = 12.500 \text{ okunması gereken blok sayısı } O(12500)$$

4. Kütükten tek bir blok okumak için gereken süre

$$\text{Ortalama arama zamanı } 6\text{msn}$$

$$\text{Blok başı bulma ortalama süresi: } (60 \text{ sn} / 7200 \text{ rev}) * (1/2) = 4.17 \text{ msn}$$

$$\text{Blok aktarım süresi} = (60 / 7200) * (4/12) = 2.78 \text{ msn}$$

$$\text{Bir blok için toplam süre} = 6 + 4.17 + 2.08 = 12,94 \text{ msn}$$

5. Oluşturulacak yoğun dizinin bloklama faktörü

$$\text{Yoğun dizin içinde anahtar ve gösterge saklanacaktır : } 10 \text{ (anahtar boyu)} + 8 \text{ (gösterge boyu)} = 18$$

$$512 / 18 = 28.4 \text{ kayıtlar sektörler arasına dağılamadığı için } 28' \text{ dir ve } 28 * 4 = 112$$

6. Yoğun dizinin ikincil bellekte saklanması halinde tüketeceği blok sayısı

$$100.000/112 = 892,8 \text{ yani } 893$$

7. Madde 6'daki yoğun dizin üstüne kurulacak seyrek dizini ana bellekte saklamak için gerekli sayfa sayısı

$$\text{Seyrek dizinde toplam } 893 \text{ sayfa için birer kayıt saklanacaktır. Seyrek dizinin bloklama faktöründe } 112 \text{ olacaktır. } 893 / 112 = 7.97 \text{ yani } 8 \text{ sayfa gereklidir.}$$

8. Madde 7 ile oluşturulan çok katmanlı dizin ile bir yığın kütükte mevcut olan bir kaydı okumak için geçen süre

Seyrek dizin anabelleğe sığmaktadır ancak yoğun dizin disk üzerinde saklanmaktadır. Veri bloğu ile birlikte t

oplam 2 blok okuması yapılacaktır. Tek blok için soru 4'te hesaplanan 12,25msn'in iki kere gerçekleşmesiger ekecektir, $12.25 * 2 = 25.89$ msn.

9.Verilerin yığılma olmaksızın (homojen) dağılımı söz konusu ise; yani ideal koşullar altında yığın kütük üstün e kurulacak bir genişleyebilir anahtarlama dizine ait "directory" oluşturmak için gerekli en çok blok sayısı:

Bir sektörde tutulacak anahtar ve gösterge sayısı = $512 / (10 + 8) = 28.4$ yani 28

Bir bukette tutulacak anahtar ve gösterge sayısı = $28 * 4 = 112$ ve toplam $100000 / 112 = 893$ buket gereklidir.

Bir sektörün adresleme kapasitesi $512 / 8 = 64$ ve bir bloğun adresleme kapasitesi $64 * 4 = 256$ ise

$893 / 256 = 4$ blok gereklidir

10. Madde 9 ile oluşturulan genişleyebilir anahtarlama dizinde "genel derinlik" değeri

893 buket $2^{10} = 1024$ olmak üzere son 10 bit ile adreslenir

(20 Puan) Soru-1: Bir veritabanı sistemi hafızada toplam 8 frame saklayacak şekilde arabellek (memory buffer) kullanmaktadır. Bu arabellek takip etmeye başladıktan 8 zaman birim (mili saniye gibi düşünebilirsiniz) sonra aşağıdaki gibi bir içeriğe sahip olmuştur. Bu tabloda 1. Frame'de Sayfa 1 (Page) bulunmaktadır. Bu sayfaya en son 2. Zaman biriminde erişilmiştir, yani en son bu sayfa için okuma veya yazma isteği 2. milisaniyede gerçekleşmiştir. Bu sayfa için arabellek yönetiminde kullanılan pin_count ve dirty değerleri de 1 olarak verilmiştir.

pin_count=1 dirty=1 Son Erişim=2 S_1	pin_count=3 dirty=1 Son Erişim=8 S_2	pin_count=0 dirty=0 Son Erişim=1 S_3	pin_count=0 dirty=0 Son Erişim=4 S_4
pin_count=1 dirty=1 Son Erişim=3 S_5	pin_count=0 dirty=0 Son Erişim=5 S_6	pin_count=1 dirty=0 Son Erişim=6 S_7	pin_count=0 dirty=1 Son Erişim=7 S_8

Sistemde oku ve bırak operasyonları devam etmektedir. Bir oku operasyonu, belirtilen sayfayı yeni bir kullanıcı tarafından okumak için kullanılmaktadır. Okuma işlemi sayfada herhangi bir değişiklik yapmamaktadır. bırak operasyonu ise daha önce belirtilen sayfaya erişmiş bir kullanıcının seansını bitirdiğini ve sayfayı serbest bıraktığını belirtmek için kullanılmaktadır. Aşağıdaki operasyonlar, belirtilen sayfalar için belirtilen zamanlarda gerçekleştirilmektedir.

9	10	11	12	13	14	15	16	17
<i>oku(S_1)</i>	<i>oku(S_2)</i>	<i>bırak(S_1)</i>	<i>bırak(S_2)</i>	<i>bırak(S_5)</i>	<i>bırak(S_7)</i>	<i>oku(S_9)</i>	<i>oku(S_8)</i>	<i>oku(S_{10})</i>

(a) Sistemin arabellek yönetimi için Least Recently Used (LRU) algoritmasını kullandığını biliyorsak, bu değişikliklerden sonra (yani 18. zaman biriminde) arabelleğin içeriğinin güncel halini aşağıya yazınız.

<i>pin_count=1 dirty=1 Son Erişim = 9 S_1</i>	<i>pin_count=3 dirty=1 Son Erişim=10 S_2</i>	<i>pin_count=1 dirty=0 Son Erişim=15 S_9</i>	<i>pin_count=0 dirty=0 Son Erişim=4 S_4</i>
<i>pin_count=1 dirty=0 Son Erişim=17 S_{10}</i>	<i>pin_count=0 dirty=0 Son Erişim=5 S_6</i>	<i>pin_count=0 dirty=0 Son Erişim=6 S_7</i>	<i>pin_count=1 dirty=1 Son Erişim=16 S_8</i>

NOTLANDIRMA: Her bir kutunun doğru olması için 1 puan. Eğer son erişim, dirty veya pin_count değeri yanlış değerlerinden biri yanlış ise 0.5 puan, iki tanesi yanlış ise 0 puan. S10 ve S9'u doğru yere yerleştirmek için 2şer puan. Toplam 12 puan.

(b) Bu işlemler sonrasında toplam kaç tane disk operasyonu olacaktır. Disk operasyonlarını okuma ve yazma olarak belirtiniz.

Veri diskten sadece S_9 ve S_{10} sayfalarının okunması için yapılacaktır. Ayrıca S_{10} 'u yerleştirmek için dirty=1 olan S_5 çıkartılmıştır. Bu sayfanın da diske yazılması gerekmektedir. Dolayısıyla toplam 2 okuma ve 1 yazma işlemi olacak şekilde toplam 3 disk işlemi gerçekleşmiştir.

NOTLANDIRMA: S9 ve s10 için 3er puan. S5 için 2 puan. Bunların dışındaki her bir okuma/yazma için -1 puan. Toplam 8 puan.

(10 Puan) Soru 2: Aşağıdaki dizin yapıları ve veri dosyası yapılarından hangileri kullanılamaz, net bir şekilde kullanılabilir veya kullanılamaz olarak işaretleyiniz.

- (a) Yoğun dizin (Dense index) – Sıralı veri dosyası (Sorted data file) TRUE
- (b) Seyrek dizin (Sparse index) – Yığma veri dosyası (Heap data file) FALSE
Bu sistem kullanılamaz çünkü Seyrek dizinin sayfadaki diğer kayıtları bulabilmesi için sıralı tutması gerekir.
- (c) İki Seviyeli Dizin: Seyrek dizin – Seyrek dizin – Sıralı veri dosyası TRUE
- (d) İki Seviyeli Dizin: Yoğun dizin – Yoğun dizin – Yığma veri dosyası FALSE
- (e) İki Seviyeli Dizin: Yoğun dizin – Seyrek dizin – Sıralı veri dosyası FALSE

Bu iki seçenekte de 2. Seviyede yoğun dizin kullanılmıştır. 1. Seviye dışında yoğun dizin kullanılırsa, zaten dizin aynı boyutta olacağı için kullanmanın anlamı yoktur.

Notlandırma: Her doğru şık için 2 puan.

BBM371 VERİ YÖNETİMİ, SINAMA-3
8.12.2015/30 Puan

Soru	İşlem	Cevap
Blok boyu 4096byte ve adresleme için 8 byte kullanılan bir ortamda anahtar boyu 24 byte olmak üzere; B ağacının bir düğümünde saklanabilecek en çok anahtar (2d) sayısı kaçtır?	$(2d \cdot 24) + (2d \cdot 8) + ((2d+1) \cdot 8) \leq 4096$ $d=51$	102
Blok boyu 4096byte ve adresleme için 8 byte kullanılan bir ortamda anahtar boyu 24 byte olmak üzere; B+ ağacının bir düğümünde saklanabilecek en çok anahtar (2d) sayısı kaçtır?	$(2d \cdot 24) + ((2d+1) \cdot 8) \leq 4096$ $d=63$	126
3 düzeyli bir B+ ağacının sadece kökü ana bellekte saklanıyor. Aralık sorgusunun cevabını oluşturan kayıtlar toplam 80 blok içine yayılmış ise, bu kayıtlara <u>sadece dizin üzerinde</u> ulaşmak için gereken <u>en çok</u> disk erişim sayısı nedir?	0 (kök) + 1(2.düzye) + 1 (3.düzye ve ilk blok) + 79 (aralığa giren değerleri okuma)	81
3 düzeyli ve bir B ağacının sadece kökü ana bellekte saklanıyor. Aralık sorgusunun cevabını oluşturan kayıtlar toplam 80 bloğun içine yayılmıştır. Bu blokların birtanesi 2. ve 79'u 3. düzeyde olacak biçimde yerleşmişlerdir. Bu kayıtlara <u>sadece dizin üzerinde</u> ulaşmak için gereken <u>en çok</u> disk erişim sayısı nedir?	0 (kök) + 1(2.düzye) + 1 (3.düzye ve ilk blok) + 79 (aralığa giren değerleri okuma) + 79 (kök orta dolaşımında ata anahtarı okuma)	160
Her bir düğümünde 10 anahtar tutan bir B ağacının 3 düzeyli olması halinde tutabileceği en çok anahtar sayısı kaçtır?	0.düzye -> 1 düğüm-> 10 anahtar 1.düzye-> 11 düğüm-> 110 anahtar 2.düzye-> 121 düğüm->1210 anahtar Toplam= 1330 anahtar	1330

B+ ağacının 4 özelliğini yazınız?	<ol style="list-style-type: none">1- Kök hariç her düğümü yarı doludur2- Her zaman dengelidir3- Büyüme ve küçülmeler yaprakta köke doğru ele alınır4- Her anahtar değeri yaprakta yer alır ve yapraklar verinin sıralı bağlaçlı listesi biçimindedir.
-----------------------------------	--

Soru-1. Toplam 7 bloktan oluşan bir yığın kütüğün üzerinde sıra ile 2 kere baştan sona okuma yapılacaktır. Bu işlem için toplam 3 boş çerçeve kullanılacaksa, yer değiştirme algoritmasının LRU ve MRU olması halinde gerçekleşecek disk erişim sayısını ayrı ayrı hesaplayınız.

Yöntem	İşlem	Disk erişim sayısı
LRU	LRU (ilk okuma) = >1,2,3 / 4,2,3 / 4,5,3/ 4,5,6 / 7,5,6 (7 okuma). İkincisi de bellekte kalan sayfalara bakıldığında 7 kere kullanılacaktır. Toplam 14 okuma olur.	14
MRU	<p>*MRU (ilk okuma) = >1,2,3 / 1,2,4 / 1,2,5/ 1,2,6 / 1,2,7 (7 okuma)/ 1,2,3/1,2,4. İkincisi de bellekte kalan sayfalara bakıldığında 5 kere kullanılacaktır. Toplam 12 okuma olur. (Belleğe çekme zamanına göre MRU)</p> <p>**MRU (ilk okuma) = >1,2,3 / 1,2,4 / 1,2,5/ 1,2,6 / 1,2,7 (7 okuma)/1,3,7/ 1,4,7,. İkincisi de bellekte kalan sayfalara bakıldığında 5 kere kullanılacaktır. Toplam 12 okuma olur. (Son okuma işlem zamanına göre MRU)</p>	11** ve 12*

Soru 2. Her biri 150byte olan 600.000 kayıt bir yığın (heap) kütük içinde saklanmıştır. Kayıtlarda dizinleme için kullanılan biricik anahtarın boyu 22byte'tır. Bu kütüğün saklandığı diskin fiziksel özellikleri yandaki tabloda verilmiştir. Kütüğün saklandığı sistemde en fazla 10 adet ana bellek sayfası mevcuttur ve gösterge boyu 8 byte'tır.

Disk özelliği:	Değeri
Sektör boyu:	512 byte/sektör
İzdeki sektör sayısı:	20 sektör /iz
Öbekteki sektör sayısı:	5 sektör/öbek
Yüzeydeki iz sayısı:	3000 iz / yüzey
Diskteki yüzey sayısı	10 adet ÇİFT taraflı yüzey/disk
Dönüş hızı:	6000 rpm
Yatayda arama zamanı:	8ms

Bu verilere göre aşağıdaki soruları cevaplayınız?

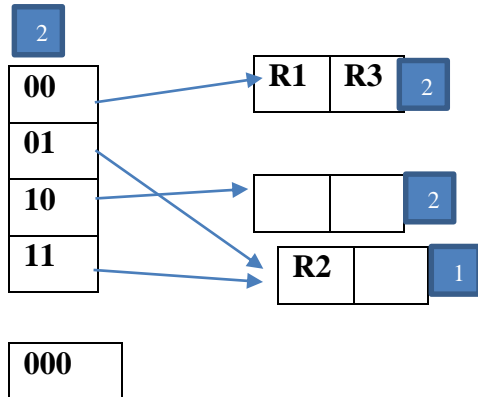
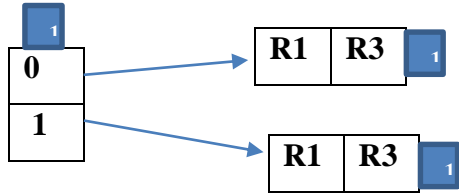
No	Soru	İşlem	Sonuç
1	Yığın kütüğün bloklama faktörü nedir	$512/150 = 3 * 5 = 15$	15
2	Yığın kütüğü saklamak için gereken toplam blok sayısı nedir	$600.000/15 = 40000$	40.000
3	Disk in saklama kapasitesi toplam kaç bloktur	$20/5 = 4$ öbek/iz $4*20 = 80$ öbek/silindir $80*3000 = 240.000$ öbek/disk	240.000
4	Silindir öncelikli yerleşimde yığın kütüğü saklamak için gerekli silindir sayısı kaçtır	$40.000/80 = 500$	500
5	Kütükten tek bir blok okumak için gereken süre nedir	$60.000/6.000 = 10$ msn $8\text{ms} + (10/2 =) 5\text{ms} + (10/4 =) 2.5\text{msn}$ $= 15.5\text{msn}$	15.5 sn
6	Oluşturulacak yoğun dizinin bloklama faktörü kaçtır	$512/(22+8) = 17$ $17 * 5 = 85$	85
7	Yoğun dizinin ikincil bellekte tüketeceği blok sayısı kaçtır	$600.000/85 =$	7059
8	Yoğun dizin üzerine kurulan kaçinci seyrek dizin bellekteki boş sayfalara sığacak büyüklükte olacaktır?	$7059/85 = 84$ blok (1.seyrek) $84/85 = 1$ (2.seyrek)	2. düzey
9	8. soruya göre kütükten bir kayıdı okumak için geçen toplam süre ne olacaktır?	#Disk erişimi= 1.seyrek + yoğun + kütük = 3 blok okuma süresi = 3 *15.5 = 46.5 msn	46.5msn
10	6. sorudaki yoğun dizin üzerine başka bir dizin <u>oluşturulmaz ise</u> , kütükten bir kayıdı okumak için geçen en çok süre ne olur	#Disk erişimi = yoğun dizinde ikili arama + kütük = $(\log_2 7059) + 1 = 14$ ise okuma süresi $14*15,5 = 217$ msn	217msn

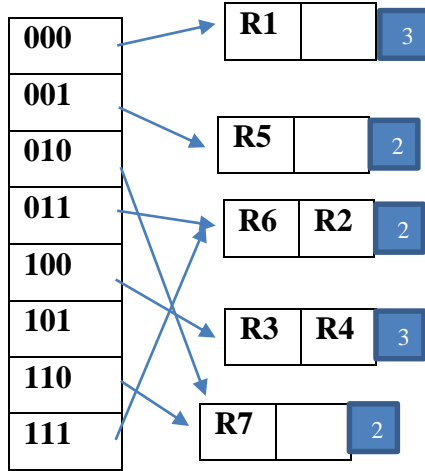
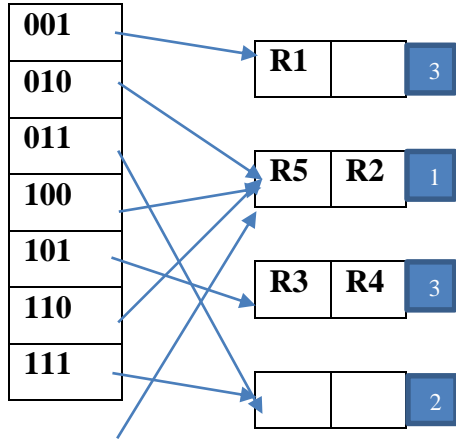
Soru-3. Genişleyebilir anahtarlama yöntemi kullanılarak aşağıda verilen anahtar değerleriyle dizin oluşturulacaktır. Verileri saklamak için kullanılan kapların (bucket) her biri toplam 2 kayıt saklayabilmektedir. Anahtarlama fonksiyonu olarak $H(k) = k \bmod 29$ kullanılmaktadır ve $H(k)$ değerinin en önemsiz bitinden başlayarak anahtarlama yapılmaktadır.

(a) Aşağıda verilen anahtar değerleri için tabloyu doldurunuz.

Kayıt	k	H(k)	Binary
R1	53	24	11000
R2	32	3	00011
R3	70	12	01100
R4	107	20	10100
R5	30	1	00001
R6	65	7	00111
R7	60	2	00010

(b) Yukarıdaki tabloya göre, kayıtlar verildiği sıra ile işlendiğinde oluşacak genişleyebilir anahtarlama yapısını gösteriniz. Her bir kap bölünme işlemini net bir şekilde çiziniz. Rehberin (directory) hangi kapları (bucket) gösterdiğini, yerel derinliği ve genel derinliği belirtiniz.





Soru-4. Bir veritabanı verileri belli bir k alanına (field) göre dizinlemek için genişleyebilir anahtarlama (extendible hashing) yöntemini kullanmaktadır. Söz konusu k alanı 4 byte ile gösterilen rasyonel sayılardır ve değerleri tekrar etmemektedir. Toplam 262.144 kayıt olduğu bilinmektedir. Kap içerisinde anahtar değerine karşılık gelen kaydın yerini gösteren 8 byte boyutunda disk adresi tutulmaktadır. Veritabanı sayfa boyutu (page size) olarak 6 KB kullanılmaktadır. Bu bilgilere göre aşağıdaki soruları cevaplayınız.

(a) Bu veriyi dizinlemek için genel derinliğin alabileceği en düşük değeri hesaplayınız.

6 KB / 12 = 512 entries per bucket

262.144/512 = 512 buckets needed with uniform distribution

$2^d=512 \rightarrow d = 9$

(b) Aşağıdaki 3 soruyu aşağıdaki değişkenler ve yukarıda verilen değerlere göre cevaplayınız.

Cevabınızı mümkün olduğunca sadeleştiriniz.

N : Kullanılan toplam kap sayısı (number of buckets)

R : Bir sayfanın okunma zamanı

D : Directory boyutu (byte olarak)

(i) Ara bellekte sadece directory bulunuyorsa, ' k değeri 20,2 olan kayıdı getir' gibi bir sorguyu işleme zamanını belirtiniz.

1 bucket okumak için +1 kütük okuma = 2 erişim

(ii) Dizin için ara bellekten faydalanılmıyorsa, $k=20,2$ gibi bir sorguyu işleme zamanını yazınız.

*$(\text{ceil}(D/6K) + 1\text{bucket} + 1\text{ kütük}) * R$*

(iii) Rasyonel sayı olan k anahtarı için ' $k > 10$ & $k < 20$ koşullarını sağlayan kayıtların getirilmesi' gibi bir aralık sorgusu verilirse kayıtların bulunma zamanını belirtiniz (Ara bellekte sadece directory bulunuyorsa).

*Read all the buckets. $N * R + M * R$*

Soru-5. Hangi işlem senaryosunda yoğun dizinle birlikte kullanılan sıralı kütük, seyrek dizinle birlikte kullanılan sıralı kütüğe göre daha avantajlı bir kullanım sunar (ipucu: arama zamanındaki disk erişimini azaltır) (10 puan)

Eğer aranan kayıt veri dosyasında bulunmuyorsa/min-max tipi sorgular/count tipi sparse index'de bucket okumak gerekir. Fakat yoğun dizinde sadece dizine bakıp kaydın olmadığını bilebiliriz.

Puan alan diğer cevaplar:

- *Yoğun dizinde her kaydın bucket içinde nerede olduğunu biliyoruz, bu yüzden daha hızlı olur. Disk erişimi sayısı etkilenmiyor. 4 puan*
- *Eğer tekrar eden değerler varsa: Bu durumda eğer Seyrek dizinde derste bahsettiğimiz gibi bir çözüm uygulanmazsa önündeki ve sonrasındaki bucketlerin de okunması gerekebilir. 8 puan.*

Example Questions

1. A database uses a Linear Hashing based index. Answer the following questions if the Next pointer is pointing to bucket 25 (buckets are indexed starting from 0).

- a) What is the minimum number of primary buckets in this Linear Hash?
- b) If the number of primary buckets is as calculated in part (a) how many bucket splits are needed to split the 3rd bucket.
- c) If Next points to 25 and the number of primary buckets is 89 which buckets are accessed for searching the following two keys k_1 and k_2 , clearly show how you calculate the bucket index from the hash values indicating the bits you are using (Use the least significant bits for addressing).

$$H(k_1) = 175$$

$$H(k_2) = 83$$

2. An extendible hash is used to index a string field storing names of students. A bucket stores 100 keys and the global depth of the directory is 8. There are 256 buckets in this extendible hash. If the directory is buffered in the memory, how many disk accesses are needed for the following queries:

- a) Finding if there is a student with name "Ahmet" exists or not?

- b) Finding the number of students whose name starts with the letter 'B'?

3. You have got 1000 records that are fixed length and 100 bytes. The records will be stored on a disk with a sector size of 102 bytes. Each cluster involves 4 sectors and each track involves 60 sectors. The rotational time is 4000 rpm and the average seek time is 6 msec. Answer the following questions.

- a) Blocking factor of the files:

- a) Number of blocks to store the file.

- b) Average time to find a record if the read/write head is at a random position initially and the file is sorted (cylinder based approach is not used):

- c) Time to add a new record into the file if the read/write head is at a random position initially and the file is sorted. Consider the worst case in your solution and assume that the number of blocks in the sorted file will remain the same once the new record is inserted:

- d) We want to index the given file with a dense index. Assume that index stores a search key of 4 bytes and a pointer of 4 bytes. How many disk accesses do we need in order to find a record?

- e) We want to index the given file with a sparse index. Assume that index stores a search key of 4 bytes and a pointer of 4 bytes. How many disk accesses do we need in order to find a record?

- f) Imagine we have 5 available buffer pages and our file is a heap file. We want to sort the file. How many runs will be produced in the first pass?

- g) How many passes will it take to sort the file completely?

- h) What is the total IO cost of sorting the file?

BBM 371 – Data Management - Fall 2018
Midterm Exam
December 13, 2018

Name: _____

Student ID number: _____ Section: _____

Problem	Points	Grade
1	25	
2	25	
3	30	
4	20	
Total	100	

INSTRUCTIONS

- Do not open this exam booklet until you are directed to do so. Read all the instructions first.
- When the exam begins, write your name on every page of this exam booklet.
- The exam contains five multi-part problems. You have **120 minutes** to earn 100 points.
- The exam booklet contains **6 pages** to the exam, including this one.
- This exam is a **closed book and notes exam**. Please write your answers in the space provided on the exam paper.
- Show all work, as partial credit will be given. You will be graded not only on the correctness and efficiency of your answers, but also on your clarity that you express it. Be neat.
- Good luck!

1. Memory Page Buffer (25 pts)

In a database system, there are 4 frames available in the memory buffer.

Frame1 pin_count=1 dirty=0 Last Access: 12 th ms P ₇	Frame2 pin_count=1 dirty=0 Last Access: 9 th ms P ₅
Frame3 pin_count=1 dirty=0 Last Access: 10 th ms P ₆	Frame4 pin_count=0 dirty=1 Last Access: 7 th ms P ₄

Three operations, namely read, modify, and release can be performed on the pages. The read operation assumes that a new user is accessing a page (either already in the buffer or not) and reading the page without changing it. A release operation does not access the page but indicates that one of the previous users does not need it anymore. A modify operation assumes that a new user is modifying the page if it is already in the buffer, or reading a new page from the disk and modifying it. Please show how read, modify, and release operations are changing the last access time, pin_count and dirty attributes of the frame. Consider that the following operations are performed:

5	6	7	8	9	10	11	12
<i>read</i> (P ₁)	<i>release</i> (P ₂)	<i>modify</i> (P ₄)	<i>release</i> (P ₁)	<i>read</i> (P ₅)	<i>read</i> (P ₆)	<i>release</i> (P ₄)	<i>read</i> (P ₇)

- a) (10 pts) Show the contents of the memory buffer after the 12th millisecond if Least Recently Used (LRU) replacement policy is applied.

Frame1 pin_count = dirty= Last Access= Page=	Frame2 pin_count = dirty= Last Access= Page=
Frame3 pin_count = dirty= Last Access= Page=	Frame4 pin_count = dirty= Last Access= Page=

- b) (5 pts) Show the contents of the memory buffer after the 12th millisecond if Most Recently Used (MRU) replacement policy is applied.

Frame1 pin_count = dirty= Last Access Page=	Frame2 pin_count = dirty= Last Access= Page=
Frame3 pin_count = dirty= Last Access= Page=	Frame4 pin_count = dirty= Last Access= Page=

- c) (5 pts) How many disk accesses are needed if LRU is used? Indicate which pages are written and read from the disk. Explain briefly.

--

- d) (5 pts) How many disk accesses are needed if MRU is used? Indicate which pages are written and read from the disk. Explain briefly.

--

2. Disk Cost Model (25 pts)

We have got two tables stored in two different files with the given schemas:

Students (sid: string, name: string, departmentno: integer)

Departments (departmentno: integer, departmentname: string)

Students file is 4 GB and departments file is 1 MB. Both tables consist of fixed length records of 64 bytes. The system has got a block size of 1KB bytes. Each index entry stores a search key of 4 bytes and 4 bytes of pointer. ($1\text{ GB} = 2^{30}$, $1\text{ MB} = 2^{20}$, $1\text{ KB} = 2^{10}$)

- a) (5 pts) How many disk accesses is needed to find the student who has student id 20122015 if a heap file is used for the Students file? Explain briefly.

- b) (5 pts) How many disk accesses is needed to read the name of the student who has student id 20122015 if a sorted file (according to ids) and a sparse index (built on the student id) is used? Explain briefly.

- c) (5 pts) How many disk accesses is needed to find the name of the department of the student with id 20457789 if both files are heap files and using a 2-level index? Explain briefly.

- d) (10 pts) The students file is sorted based on the id (without an index) and the department file is also sorted based on the department id (without an index). Recall that department names are unique. At the best case, what is the number of disk accesses required to find the number of students who are studying 'Computer Science'? Explain briefly.

3. Hash Based Indexing (30 pts)

The least significant (rightmost) bits of the key values is used as the hash key. Each bucket can store up to 4 entries. Consider that you insert the following keys in order.

key	binary key
5	101
64	1000000
9	1001
25	11001
31	11111
15	1111
10	1010
7	111
3	11
44	101100

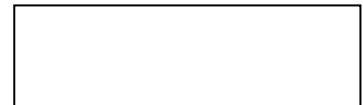
Linear hashing

- a) (5 pts) Does linear hashing always split the bucket that overflows? Explain briefly.
- b) (5 pts) Display the final structure with the decimal key values and the next pointer.
- c) (5 pts) What is the largest key (in decimal) less than 25 whose insertion cause a split? Explain briefly.

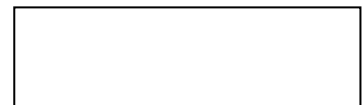
Extendible Hashing

d) (5 pts) Display the final structure with the decimal key values. Indicate the global and local depths. Explain briefly.

e) (5 pts) Give an example key value whose insertion cause a split but no change in the directory? Explain briefly.



f) (5 pts) What is the largest key (in decimal) less than 44 whose insertion cause the directory to double in size? Explain briefly.



4. Tree-Based Indexing (20 pts)

In the following questions, while explaining your solutions please recall that level 0 refers to root level.

B-Tree Indexing

Consider that you are going to create a B tree index for a file of 3000 records with unique keys. The order of buckets is set to 9.

- a) (5 pts) Consider that you are going to create a B tree index for the given set of records. The order of buckets is set to 9. At the worst case, what is be the maximum number of disk page access to locate a record with a given key? Explain briefly.

- b) (5pts) What is the minimum number of levels in the B tree to accommodate such number of records? Explain briefly.

B+ Tree Indexing

Consider that you are going to define a B+ tree index for the same file of 3000 records with unique keys. The order of the data buckets is 9 but the order of internal nodes is 200.

- c) (5 pts) At the worst case, what is be the maximum number of disk page access to locate a record with a given key? Explain briefly.

- d) (5 pts) What is the minimum number of levels in the B+ tree to index such number of records? Explain briefly.