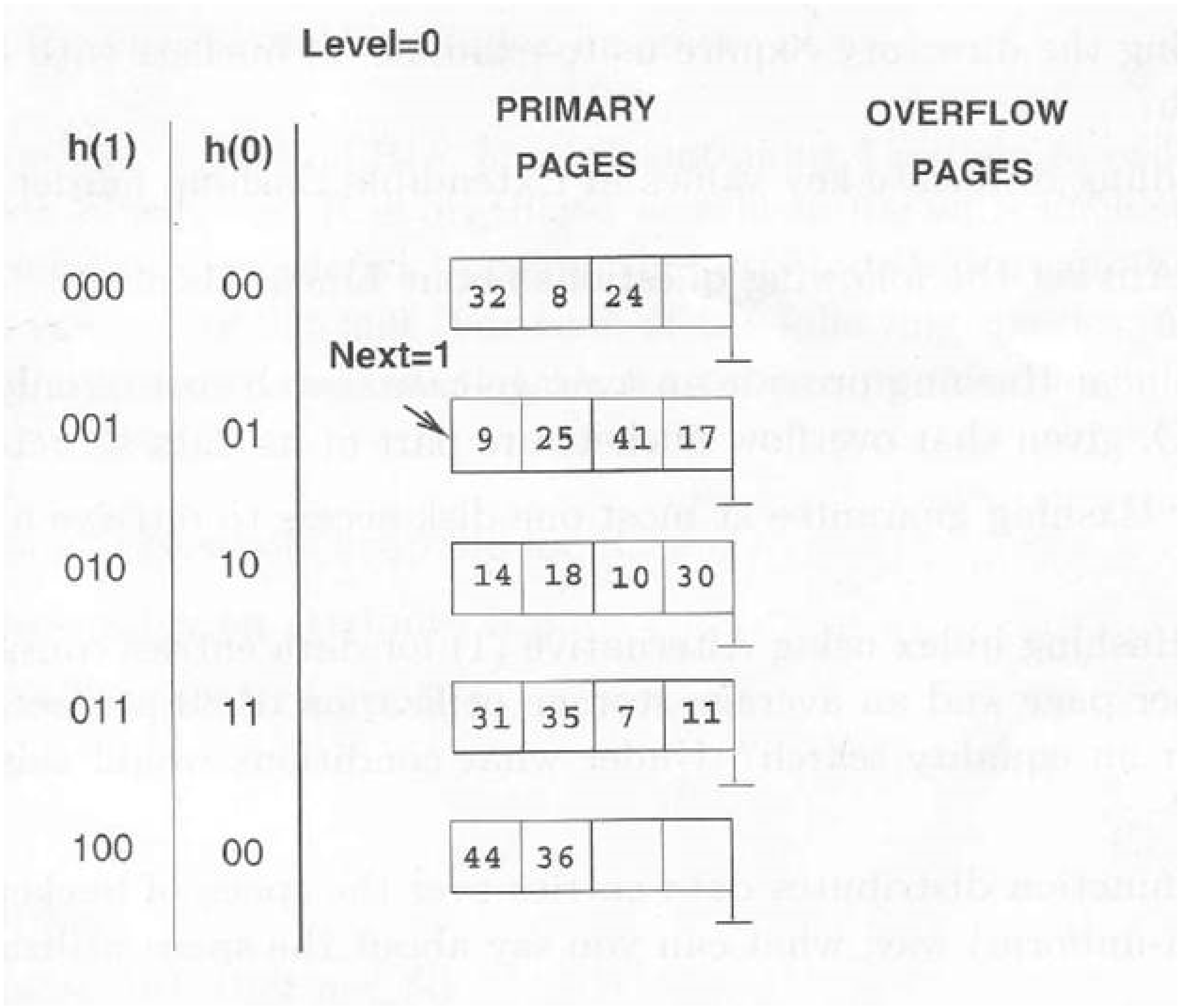
Homework 2

---

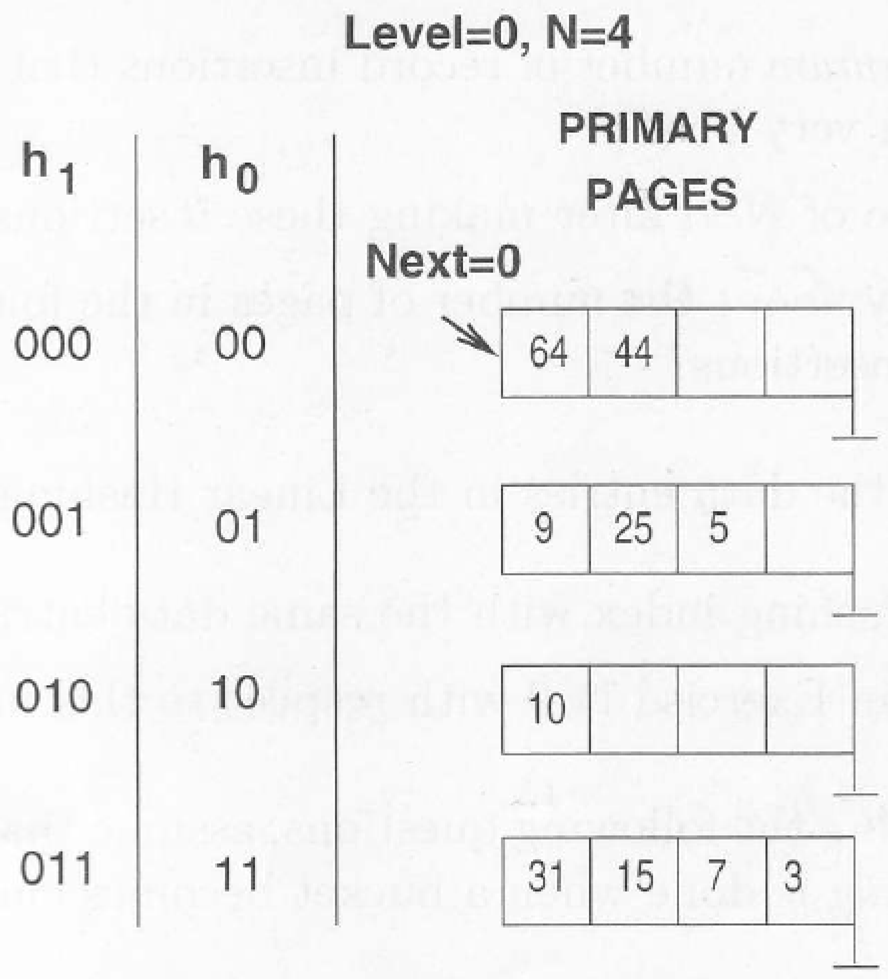
1. Consider the Linear Hast hashing index shown in the following figure. Assume that we split whenever an overflow page is created. Answer the following questions about this index:

1) Show the index after inserting an entry with hash value 12.
2) Show the original index after inserting an entry with hash value *29*.
3) Show the original index after deleting the entries with hash values 36 and 44. (Assume that the full deletion algorithm is used).
4) Find a list of entries whose insertion into the original index would lead to a bucket with two overflow pages. Use as few entries as possible to accomplish this. What is the maximum number of entries that can be inserted into this bucket before a split occurs that reduces the length of this overflow chain?



2. Consider the data entries in the Linear Hashing index shown in the following figure. Assume that a bucket split occurs whenever an overflow page is created.

1) What is the maximum number of data entries that can be inserted (given the best possible distribution of keys) before you have to split a bucket? Explain very briefly.
2) show the index after inserting a single record whose insertion causes a bucket split.
3) a. What is the minimum number of record insertions that will cause a split of all four buckets? Explain very briefly.
b. What is the value of Next after making these insertions?
c. What can you say about the number of pages in the fourth bucket shown after this series of record insertions?



Level=0, N=4
h1
h0
PRIMARY PAGES
Next=0
000
00
64
44
001
01
9
25
5
010
10
10
011
11
31
15
7
3

3. Consider the following classes of schedules: serializable, conflict-serializable, view-serializable, recoverable, avoids-cascading-aborts, and strict. For each of the following schedules, state which of the preceding classes it belongs to. If you cannot decide whether a schedule belongs in a certain class based on the listed actions, explain briefly. The actions are listed in the order they are scheduled and prefixed with the transaction name. If a commit or abort is not shown, the schedule is incomplete; assume that abort or commit must follow all the listed actions.

1). T1:R(A), T1:W(A), T1:R(B), T2:R(B), T3:W(B), T1:W(A), T2:R(B)

2). T1:R(A), T2:W(A), T1:W(A), T2:Abort, T1:Commit

3). T1:R(A), T2:R(B), T4:R(C), T3:W(A), T4:W(C), T4:Abort, T2:R(A), T2:Commit, T1:R(B), T1:Commit, T3:Commit

4). T1:R(A), T2:R(A), T3:R(B), T1:W(A), T2:W(A), T3:W(B)

5). T1:W(A), T3: R(C), T2:R(B), T1:R(B), T1:Commit, T3: W(C), T3:Commit, T2:R(A), T2:Commit

6). T1:W(A), T3:R(B),T2:R(A), T1:W(A), T3:Abort, T2:Commit, T1:Abort

7). T1:R(A), T3:W(A), T3:Commit, T1:W(A), T1:Commit, T2:R(A), T2:Commit

8). T1:W(A), T3:R(B),T2:R(A), T3:W(B), T1:W(A), T2:Abort, T1:Commit,T3:Commit

9). T1: R(A), T3:W(A), T3:Commit, T2:W(A), T2:Commit, T1:R(A), T1:W(A), T1:Commit

10). T1:W(A), T3:W(B), T2:R(A), T3:Abort, T1:W(A), T2:Commit, T1:Commit

11). T1:R(A), T2:W(A), T1:W(A), T3:R(A), T1:Commit, T2:Commit, T3:Commit

12). T1:R(A), T2:W(B), T2:W(A), T1:W(A), T2:Commit, T1:Commit

4. Consider the following sequence of actions, listed in the order they are submitted to the DBMS:

Sequence S1: T1:R(B), T2:R(C), T2:W(B), T3:R(A), T3:W(C), T1:W(A), T1:Commit, T2:Commit, T3:Commit

Sequence S2: T1: R(B), T1: W(C), T2:R(A), T3:R(A), T1:Commit, T2: R(B), T2:W(B), T2:Commit, T3: W(C), T3:W(A), T3: Commit

Assume that the timestamp of transaction Ti is i. The DBMS processes actions in the order shown. If a transaction is blocked, assume that all its actions are queued until it is resumed; the DBMS continues with the next action of an unblocked transaction. For each of the following concurrency control mechanisms, describe how the concurrency control mechanism handles the sequence.

a) Strict 2PL with timestamps used for deadlock prevention. Suppose the wait-die policy is used

b) Strict 2PL with timestamps used for deadlock prevention. Suppose the wound-wait policy is used

c) Strict 2PL with deadlock detection. (Show the waits-for graph in case of deadlocks).

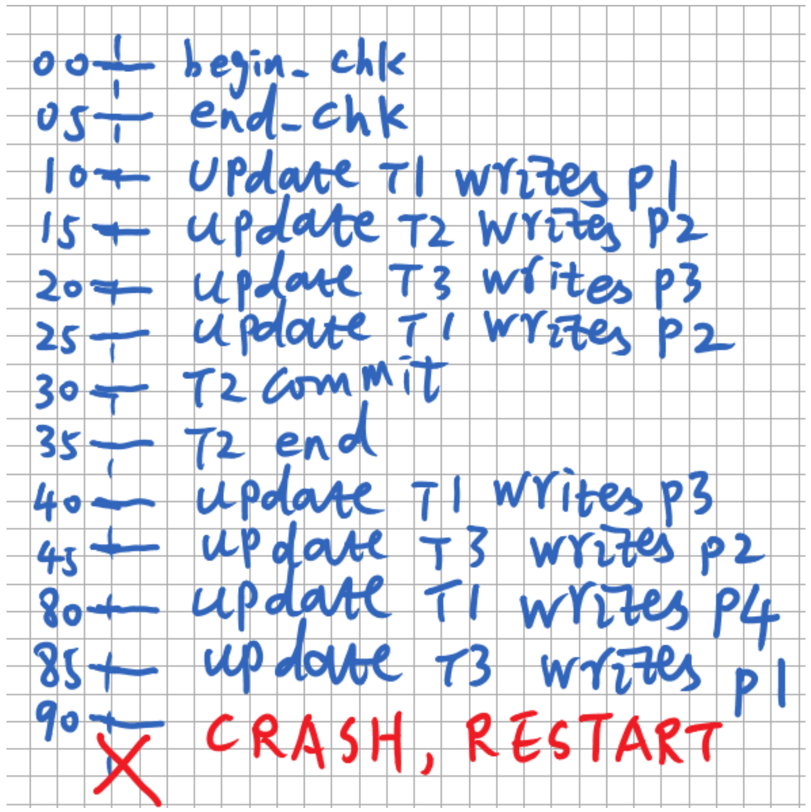
d) Conservative 2PL

e) Optimistic concurrency control

f) Timestamp concurrency control without the Thomas Write Rule

g) Timestamp concurrency control with the Thomas Write Rue

5. Consider the execution shown in following figure:



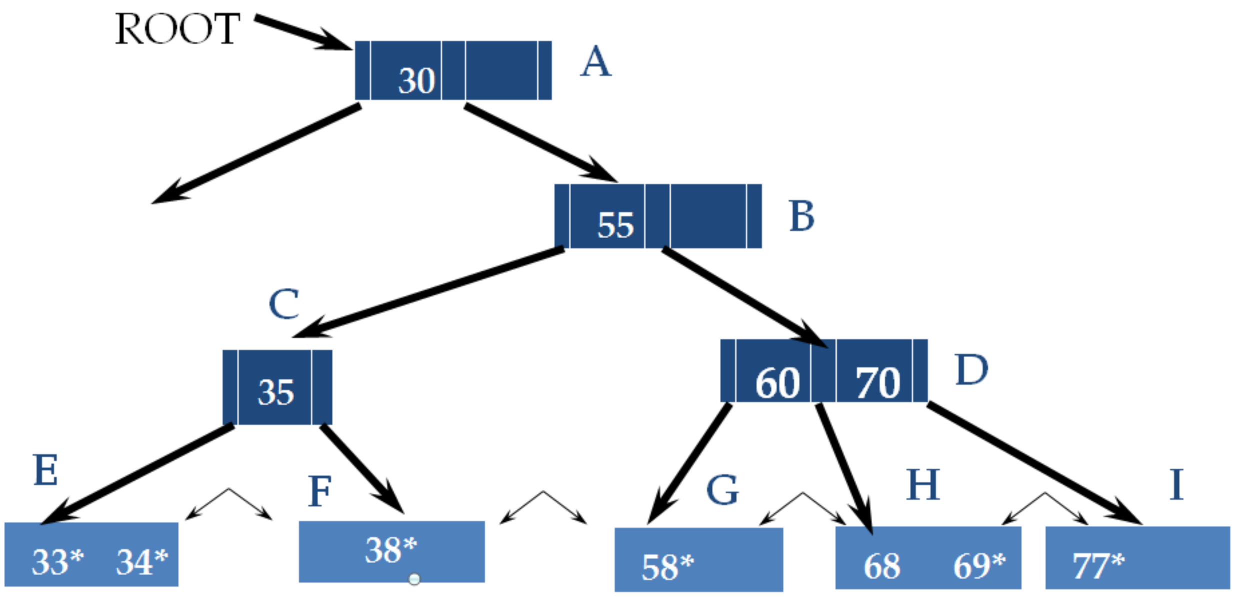
00 begin_chk
05 end_chk
10 update T1 writes p1
15 update T2 writes p2
20 update T3 writes p3
25 update T1 writes p2
30 T2 commit
35 T2 end
40 update T1 writes p3
45 update T3 writes p2
80 update T1 writes P4
85 update T3 writes p1
90 CRASH, RESTART

1) After the crash, what is done during the Analysis phase? Give the tables gotten in this phase.

2) What is done during the Redo phase? The answers should be in this format: Redo LSN xx, Redo LSN xx , …

3) Show the log records after recovery, including all prevLSN and undonextLSN values in log records. Also show the initial toUndo list and the list at each step.

6. Consider the following B+ tree with order 1, show step by step how to get and release locks for the following operations.



a) Search 38*

b) Delete 68*

c) Insert 56*

d) Insert 80*

e) Delete 34*

---

7. Consider a disk with an average seek time of 12ms, average rotational delay of 6ms, and a transfer time of 1ms for a 4K page. Assume that the cost of reading/writing a page is the sum of these values (i.e., 18ms) unless a sequence of pages is read/written. In this case, the cost is the average seek time plus the average rotational delay (to find the first page in the sequence) plus 1ms per page (to transfer data). You are given 640 buffer pages and asked to sort a file with 40,000,000 pages. Assume that you begin by creating sorted runs of 640 pages each in the first pass. Evaluate the cost of the following approaches for the subsequent merging passes:

a) Do 639-way merges.

b) Create 512 'input' buffers of 1 page each, create an 'output' buffer of 128 pages, and do 512-way merges.
c) Create 64 'input' buffers of 8 page each, create an 'output' buffer of 128 pages, and do 64-way merges.
d) Create 32 'input' buffers of 16 pages each, create an 'output' buffer of 128 pages, and do 32-way merges.
e) Create 16 'input' buffers of 32 pages each, create an 'output' buffer of 128 pages, and do 16-way merges.
f) Create 8 'input' buffers of 64 pages each, create an 'output' buffer of 128 pages, and do 8-way merges.

---

8. Suppose we have a relation Books which contains 4 attributes: bid, btitle, pid and price. All attributes are of the same length. bid is the primary key. The relation contains 10,000 pages and there are 50 buffer pages available. Consider the following query:

```
SELECT DISTINCT    bid, pid
FROM  Books
```

If you use the optimized version of the Consider the optimized version of the sorting-based projection algorithm: The initial sorting pass reads the input relation and creates sorted runs of tuples containing only attributes bid and btitle. Subsequent merging passes eliminate duplicates while merging the initial runs to obtain a single sorted result (as opposed to doing a separate pass to eliminate duplicates from a sorted result containing duplicates).

a) How many sorted runs are produced in the first pass? What is the average length of these runs? (Assume that memory is utilized well and replacement sorting is used.) What is the I/O cost of this sorting pass?

b) How many additional merge passes are required to compute the final result of the projection query? What is the I/O cost of these additional passes?

---

9. Consider a relation with this schema:

Authors(aid: integer, aname: string, age: integer, sal: integer)

Suppose that the following indices, all using Alternative (2) for data entries, exist:

- a hash index on aid
- a hash index on age
- a clustered B+ tree index on <age, sal> (the height of the tree is 4)
- an unclustered B+ tree index on sal (the height of the tree is 4)

Each author record is 60 bytes long, and you can assume that each index data entry is 15 bytes long (the index entry of the index on <age,sal> is 30 bytes long). The Author relation contains 40,000 pages and each page contains 40 tuples.

1)). Consider each of the following selection conditions and, assuming that the reduction factor (RF) for **EACH term** that matches an index is 0.1, compute the cost of the most selective access path for retrieving all Author tuples that satisfy the condition:

(a) sal > 200

(b) age = 20

(c) age > 30

(d) aid = 4000

(e) sal > 300 $\wedge$ age = 20

(f) sal > 300 $\wedge$ aname = 'Mike'

2) Suppose that , for the selection conditions (b), (e), you want to compute the average salary for qualifying tuples. Describe the least expensive evaluation method and state its cost for each of the two conditions.

---

10. Consider the join of two tables TA and TB on the condition TA.s = TB.s. The cost metric is the number of page I/Os and the cost of writing out the result should be uniformly ignored. You're given the following information

Relation TA contains 10,000 tuples and has 20 tuples per page.
Relation TB contains 3000 tuples and also has 20 tuples per page.
Attribute s of relation TA is the primary key for TA.

Both relations are stored as simple heap files.
Neither relation has any indices built on it.
32 buffer pages are available.

a). What is the cost of joining TA and TB using a page-oriented simple nested loops join? What is the minimum number of buffer pages required for this cost to remain unchanged?

b). What is the cost of joining TA and TB using a block nested loops join? What is the minimum number of buffer pages required for this cost to remain unchanged?

c). What is the cost of joining TA and TB using a sort-merge join? What is the minimum number of buffer pages required for this cost to remain unchanged?

d). What is the cost of joining TA and TB using a hash join? What is the minimum number of buffer pages required for this cost to remain unchanged?