Exercise 4 - Solution

Q1

a) Consider the capacity of the buffer pool is 4 and the request frame sequence is 1,2,3,4,5,4,5,4…

b) Consider the capacity of the buffer pool is 4 and the request frame sequence is 1,2,3,4,1,2,3,5,4…

Q2

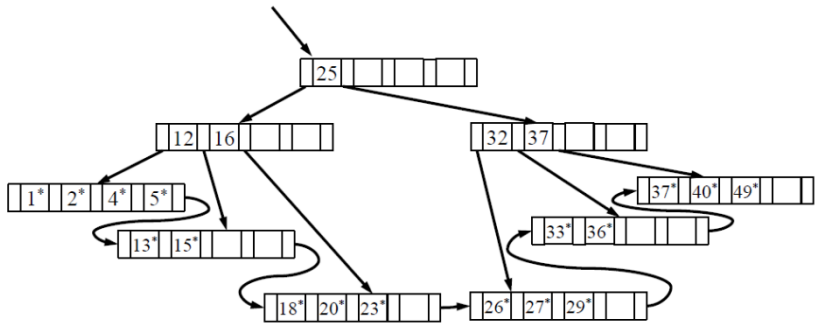
Suppose we have n frames in the buffer. The request frame sequence is

1, 2, 3, 4, ..., n, n+1, n, n+1, n, n+1, ...

We have to replace a frame every time when n or n+1 is requested.

Q3

1)



2)
According to the definition of B+-tree, the biggest B+-tree for the same set of index entries will be the tree where every node is just “half full”. Note that the answer is not unique.

Q4

The main difference between ISAM and B+ tree indexes is that ISAM is static while B+ tree is dynamic. Another difference between the two indexes is that ISAM’s leaf pages are allocated in sequence.

Q5

For equality search in a B+ tree, k nodes must be examined, where k = height of the tree. For range selection, number of nodes examined = k + m – 1, where m = number of nodes that contains elements in the range selection. For ISAM, the number of nodes examined is the same as B+ tree plus any overflow pages that exist.

Q6

(This proof is by contradiction and assumes binary locks for simplicity. A similar proof can be made for shared/exclusive locks.) Suppose we have n transactions T1, T2, ..., Tn such that they all obey the basic two-phase locking rule (i.e. no transaction has an unlock operation followed by a lock operation). Suppose that a non-(conflict)-serializable schedule S for T1, T2, ..., Tn does occur; then, according to Section 17.5.2, the precedence (serialization) graph for S must have a cycle. Hence, there must be some sequence within the schedule of the form:

S: ...; [o1(X); ...; o2(X);] ...; [o2(Y); ...; o3(Y);] ... ; [on(Z); ...; o1(Z);]…

where each pair of operations between square brackets [o,o] are conflicting (either [w, w], or [w, r], or [r,w]) in order to create an arc in the serialization graph. This implies that in transaction T1, a sequence of the following form occurs:

T1: ...; o1(X); ...; o1(Z); ...

Furthermore, T1 has to unlock item X (so T2 can lock it before applying o2(X) to follow the rules of locking) and has to lock item Z (before applying o1(Z), but this must occur after Tn has unlocked it). Hence, a sequence in T1 of the following form occurs:

T1: ...; o1(X); ...; unlock(X); ...; lock(Z); ...; o1(Z); ...

This implies that T1 does not obey the two-phase locking protocol (since lock(Z) follows unlock(X)), contradicting our assumption that all transactions in S follow the two-phase locking protocol.

Q7

1. The following schedule results in a write-read conflict: T2:R(X), T2:R(Y), T2:W(X), T1:R(X) ... T1:R(X) is a dirty read here.

2. The following schedule results in a read-write conflict: T2:R(X), T2:R(Y), T1:R(X), T1:R(Y), T1:W(X) ... Now, T2 will get an unrepeatable read on X.

3. The following schedule results in a write-write conflict: T2:R(X), T2:R(Y), T1:R(X), T1:R(Y), T1:W(X), T2:W(X) ... Now, T2 has overwritten uncommitted data.

4. Strict 2PL resolves these conflicts as follows: (a) In S2PL, T1 could not get a shared lock on X because T2 would be holding an exclusive lock on X. Thus, T1 would have to wait until T2 was finished. (b) Here T1 could not get an exclusive lock on X because T2 would already be holding a shared or exclusive lock on X. (c) Same as above.

Q8

1)

T1, T2: redo

T3: undo

2)

T2: redo

T3: undo

Q9

1)

Yes. There is no cycle in its schedule graph:

2)

There is no way to construct a schedule whose wait-for graph contains cycles.

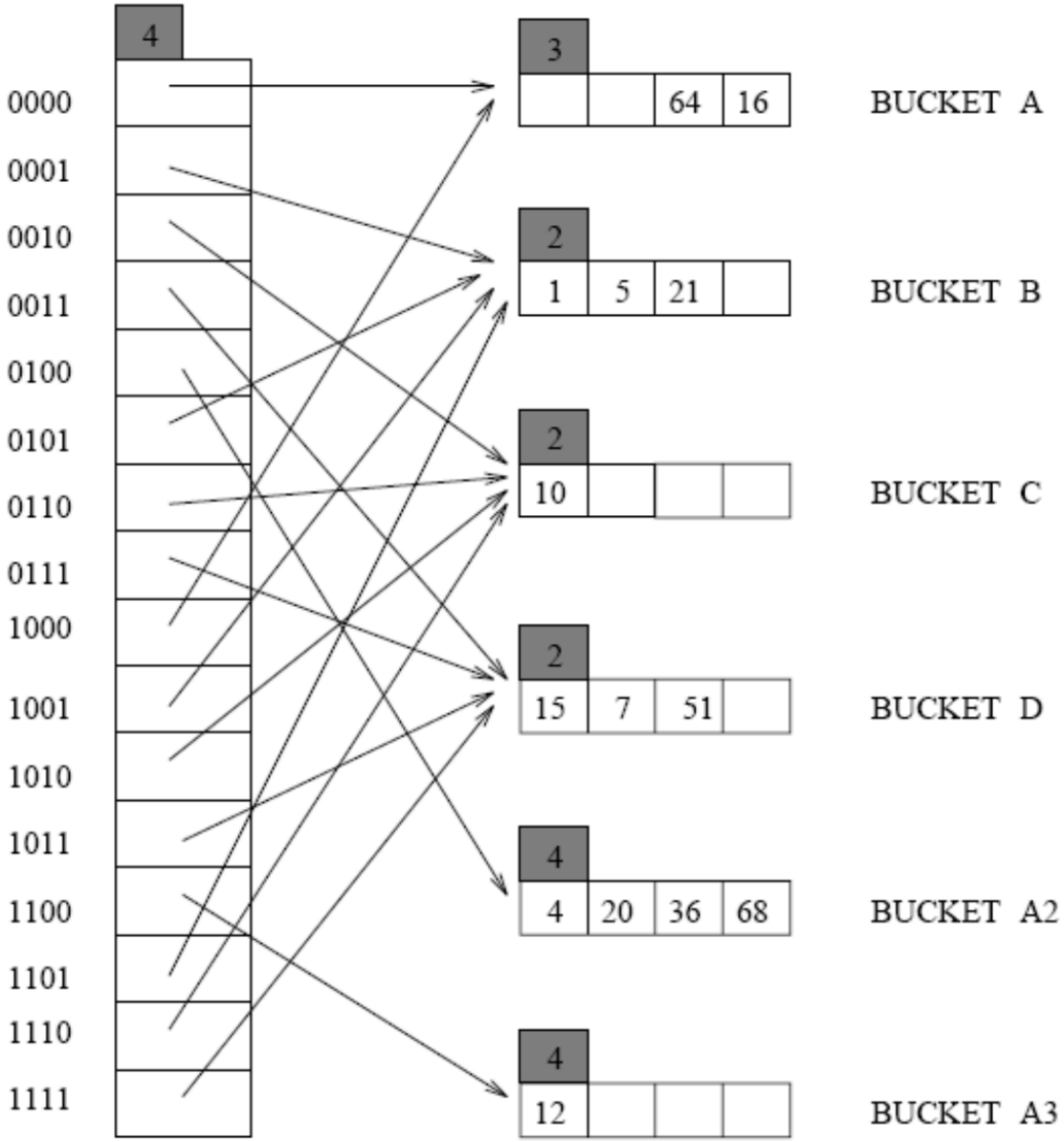
We have T1 and T3 read and write on X, we have potential to make T1 wait-for T3 or T3 wait-for T1.

We have T2 and T3 read and write on Y, we have potential to make T2 wait-for T3 or T3 wait-for T2.

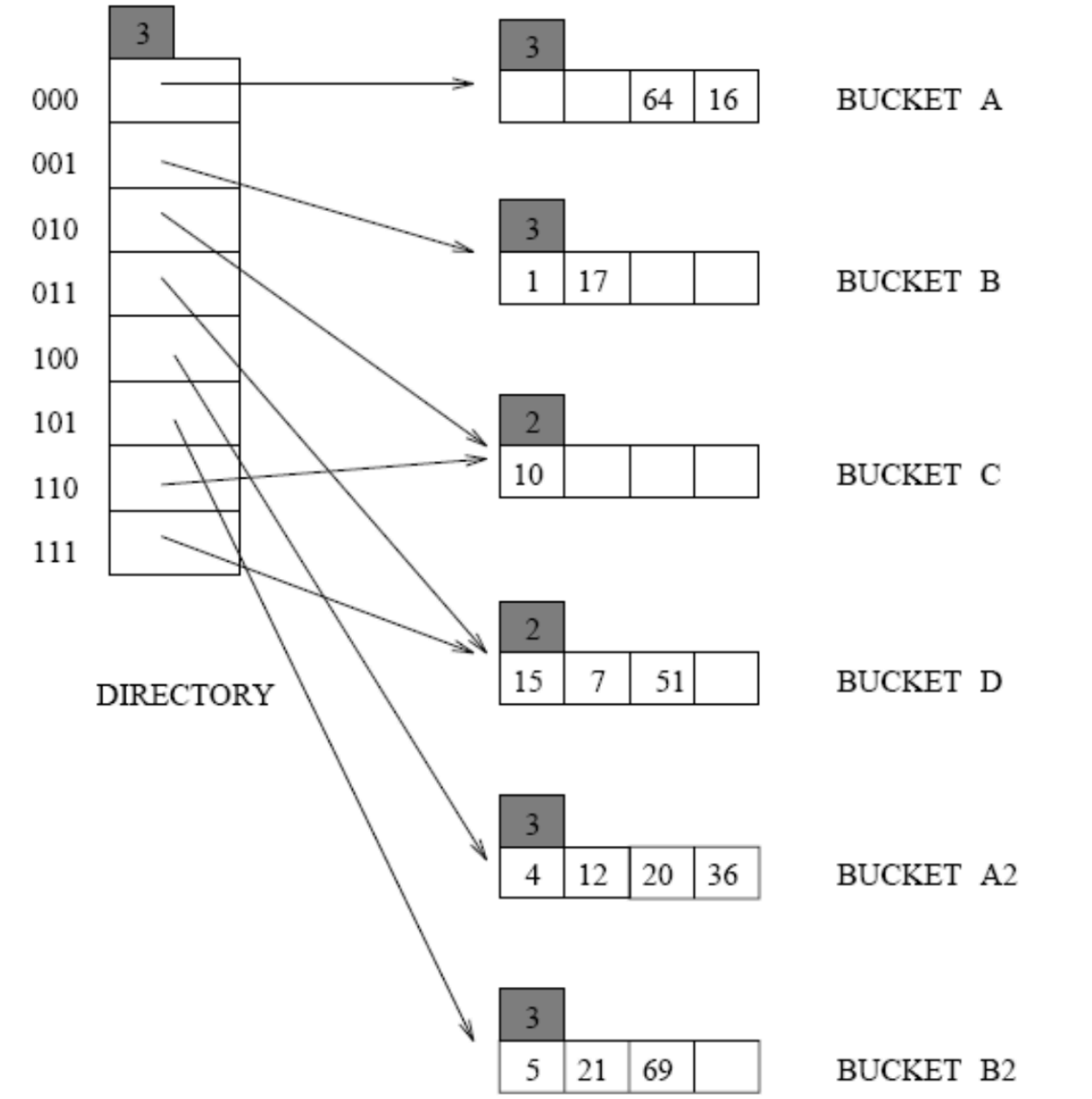
If we make T1 wait-for T3, we cannot make T3 wait-for T1 directly or through T2.

Q10

1)



2)


3
000
001
010
011
100
101
110
111
DIRECTORY
3
64
16
BUCKET A
3
1
17
BUCKET B
2
10
BUCKET C
2
15
7
51
BUCKET D
3
4
12
20
36
BUCKET A2
3
5
21
69
BUCKET B2