CS245 Database System Principles - Winter 2001

- PLEASE write your serial number on the top of your first page.
- This assignment is due in class on Thursday, March 8th.
- State all assumptions.
- Email questions to **cs245ta-win01@lists.stanford.edu**.

Assignment 7

Problem 1 (20 points)

Below are two transactions, described in terms of their effect on two database elements *A* and *B*, which we may assume are integers.

T_1: `READ(A,t); t:=t+2; WRITE(A,t); READ(B,t); t:=t*3; WRITE(B,t);`
T_2: `READ(B,s); s:=s*2; WRITE(B,s); READ(A,s); s:=s+3; WRITE(A,s);`

We assume that, whatever consistency constraints there are on the database, these transactions preserve them in isolation. Note that $A = B$ is *not* the consistency constraint.

a) It turns out that both serial orders have the same effect on the database; that is, (T_1,T_2) and (T_2,T_1) are equivalent. Demonstrate this fact by showing the effect of the two transactions on an arbitrary initial database state.
b) Give one example each of a serializable schedule and a nonserializable schedule of the 12 actions above.
c) How many serializable schedules of the 12 actions are there?

Problem 2 (20 points)

For each of the following schedules:

a) $S_a = r_1(A); w_1(B); r_2(B); w_2(C); r_3(C); w_3(A);$
b) $S_b = r_1(A); r_2(A); r_1(B); r_2(B); r_3(A); r_4(B); w_1(A); w_2(B);$

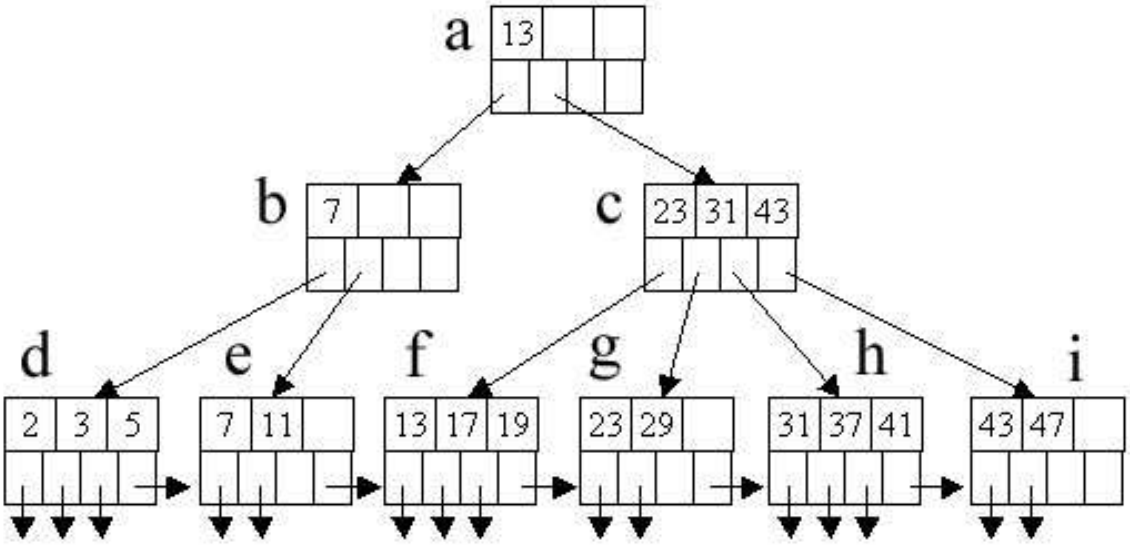
Answer the following questions:

i) What is the precedence graph for the schedule?
ii) Is the schedule conflict-serializable? If so, what are all the equivalent serial schedules?

Problem 3 (20 points)

Suppose we perform the following actions independently on the B-tree given in the figure(i.e 35 is inserted directly in the given tree and not after deleting 23). If we use the tree protocol, when can we release a write-lock on each of the nodes accessed? Note that we would like to unlock a node as early as possible to maximize concurrency. Use the notation L(node), UL(node), R(node), W(node) to indicate locking, unlocking, reading and writing a node respectively.

a) `Delete 23`
b) `Insert 35`



Problem 4 (20 points)

In the following sequences of events, we use $R_i(X)$ to mean "transaction T_i starts, and its read set is the list of the database elements X." Also, V_i means "T_i attempts to validate," and $W_i(X)$ means that "T_i finishes, and its write set was X.". State what happens when each sequence is processed by a validation-based scheduler. In particular, for each V_j action, indicate if the validation is successful or not.

a) $R_1(A,B)$; $R_2(B,C)$; $R_3(C)$; V_1; V_2; V_3; $W_1(A)$; $W_2(B)$; $W_3(C)$;
b) $R_1(A,B)$; $R_2(B,C)$; $R_3(B)$; V_1; V_2; V_3; $W_1(C)$; $W_2(B)$; $W_3(A)$;

Problem 5 (20 points)

In a lock table, the system keeps a "group mode" that records the "strongest" lock type of the transactions that have currently locked an object. In particular, say object *O* is locked in modes M_1, ... M_j and let the group mode of *O* be *GM(O)*. Then, for any possible lock mode N,

- **i)** *GM(O)* and *N* are not compatible if and only if there is an M_i element of $\{M_1, ..., M_j\}$ such that *N* and M_i are not compatible; and
- **ii)** *GM(O)* and *N* are compatible if and only if for all M_i element of $\{M_1, ..., M_j\}$, *N* and M_i are compatible.

When a new lock request arrives, for lock mode *N*, the system can simply check if *N* is compatible with *GM(O)*, instead of checking *N* against all locks currently held on object *O*.

Consider the multiple granularity locking mechanism (Section 9.6). In each of sub-problems (a) through (e) below, the modes of currently held locks on an object *O* are given. (For instance, in case (a), object *O* is locked in mode *IX* by two transaction and in mode *IS* by two transactions.) For each case, give the group mode, if there is one. (Be careful, some of the cases below are impossible! In those cases, just say there is no group mode and explain why it is so)

- **a)** IX, IS, IX, IS
- **b)** SIX, IS
- **c)** S, IS
- **d)** S, IS, IX, SIX
- **e)** IX, S, IS