Graduate Examination

Department of Computer Science
The University of Arizona
Fall 2003

Oct 24, 2003

Instructions

This examination consists of ten problems. The questions are in three areas:

1. Theory: CSc 545, 573, and 520;
2. Systems: CSc 552, 553, and 576; and
3. Applications: CSc 560, 525, 522, and 533.

You are to answer any two questions from each area (i.e., a total of six questions). If more than two questions are attempted in any area, the two highest scores will be used.

You have two hours to complete the examination. Books or notes are not permitted during the exam.

The exam sheets you have been given contain the last 4 digits of your CSID number as identification. Please write your answers directly on this exam. Use the flip side of the page if you need additional space. Keep your answers brief and to the point.

Results will be posted using these four-digit numbers.

Some problems will ask you to write an algorithm or a program. Unless directed otherwise, use an informal pseudo-code. That is, use a language with syntax and semantics similar to that of C or Pascal. You may invent informal constructs to help you present your solutions, provided that the meaning of such constructs is clear and they do not make the problem trivial. For example, when describing code that iterates over the elements of a set, you might find it helpful to use a construct such as

\[
\text{for } c \in S \text{ do } Stmt
\]

where $S$ is a set.

SOLUTIONS
1 Theory 1 (CSc 520: Principles of Programming Languages)

Consider the following program:

```pascal
VAR i : INTEGER;
VAR a : ARRAY 2 OF INTEGER;

PROCEDURE P ( x : INTEGER );
BEGIN
  i := i + 1;
  x := x + 1;
END;

BEGIN
  i := 1;
  a [1] := 1;
  a [2] := 2;
  P ( a [ i ] );
  WRITE a [1], a [2];
END
```

1. What does the program print if \( x \) is a value parameter?
2. What does the program print if \( x \) is a reference parameter?
3. What does the program print if \( x \) is a name parameter?

In each case, motivate your answer!

Solution

1. Pass by value?
   - The call \( P ( a [i] ) \) does not alter \( a [1] \) since the value is copied over as part of the call.
   - \( \Rightarrow \) The program prints 1, 2.

2. Pass by reference?
   - The call \( P ( a [i] ) \) increments \( a [2] \) since its address is passed over as part of the call.
   - \( \Rightarrow \) The program prints 2, 2.

3. Pass by name:
   - \( x := x + 1 \) becomes \( a [i] := a [i] + 1 \)
   - Since \( i \) is incremented before \( x \), we get \( a [2] := a [2] + 1 \).
   - \( \Rightarrow \) The program prints 1, 3.
2 Theory 2 (CSc 545: Design and Analysis of Algorithms)

(List with positional access) Suppose we want to support a list-like data structure with the following operations:

- **List()**
  Create and return an empty list.

- **Insert** ($x$, $i$, $L$)
  Insert object $x$ before position $i$ in list $L$. (After this operation, $x$ is at position $i$, and the elements of $L$ that were at positions $i, \ldots, n$ are now at positions $i + 1, \ldots, n + 1$.)

- **Delete** ($i$, $L$)
  Delete and return the object at position $i$ in list $L$. (After this operation, the elements that were at positions $i + 1, \ldots, n$ are now at positions $i, \ldots, n - 1$.)

Show how to use a balanced binary search tree to implement this data structure so that Insert and Delete on a list of $n$ elements take $O(\log n)$ time.

(Remarks: Keep in mind that in the worst case, a single Insert or Delete can change the positions of $\Omega(n)$ elements if it occurs near the front of the list.

You may add new fields to tree nodes in your solution, but you must describe how they are maintained.

Your answer does not have to contain detailed pseudocode. Describe the ideas in your solution; this is often best done using pictures and prose.)

Solution
The solution uses an augmented search tree, as describe for example in the textbook *Introduction to Algorithms* by Cormen, Leiserson, Rivest, and Stein. We repeat the basic ideas below.

We store the objects in a balanced search tree $T$, at which the objects are stored at the leaves of the tree. A very slight modification yields a version at which objects are stored at each node of the tree. Each node $v$ of $T$ maintains a counter $n_v$ which counts the number of objects in the subtree rooted at $v$. To find the $i$’th object (for the operation Delete($i$, $L$)) we start traversing $T$ from the root. We branch to the left subtree if the number $n_v$ of objects in this subtree is $\leq i$. Otherwise we branch to the right, at which we recurse to seek the $(i - n_v)$’s object. We continue until the search reaches a leaf, which is the $i$’th object. After the deletion, we traverse the tree bottom up from the location of the removed object, balance the tree, and update the fields $n_v$ of the nodes along the way.

Using a similar method, for performing Insert($x$, $i$, $L$), we locate the appropriate location in $T$ to insert $x$, balance the tree (using rotation in Red-Black or AVL trees), and “correct” the values of the fields $n_v$ in a bottom-up fashion, for all nodes whose subtrees were changed (there are only $O(\log n)$ ) such nodes.
3 Theory 3 (CSc 573: Theory of Computation)

Given: A set of rectangles $S = \{r_1 \ldots r_n\}$. Each rectangle $r_i$ has height 1, and width $w_i$ larger than 1. Also given another rectangle $D$ of size $w \times h$. The problem is to find a subset $S' \subseteq S$ (if it exists) so that one can relocate the rectangles of $S'$, so that in their new locations, no two rectangles of $S'$ intersect (except possibly at their boundaries), and they fully cover $D$.

The rectangle $D$ with a cover of $D$

![Diagram of rectangles]

Prove that under the assumption that $P \neq NP$, there is no polynomial-time algorithm for finding $S'$. (Hint: use reduction from subset-sum).

Solution

The problem is NP-hard, by a reduction from subset-sum. Assume that the height of $D$ is also 1. Let $Q = \{w_1 \ldots w_n\}$ denote the widths of the rectangles of $S$. Then a subset $S'$ exists if and only there is a subset $Q' \subseteq Q$ whose sum equals $w$. 
4 Systems 1 (CSc 552: Advanced Operating Systems)

What are the five basic abstractions in Mach? Briefly explain what they are and do. How do these abstractions in Mach compare to the basic abstractions in Unix of process and file, in particular describing how a Unix process is emulated using Mach abstractions?

Solution
The 5 basic Mach abstractions are:

1. task— an execution enviroment in which threads run. Tasks include a virtual address space consisting of an ordered collection of mappings to memory objects. A task is the basic unit of resource allocation in Mach.

2. thread—the basic unit of CPU utilization in Mach. A program counter operating within a task. All threads in a task share access to the task’s resources.

3. port—the communication channel in Mach. A queue of messages protected by the kernel. They provide “object references” giving Mach an “object-oriented” design interface. The fundamental operations on ports are "send" and "receive”.

4. message—a typed collection of data objects for communication between threads. They can contain data, pointers or capabilities for ports. Messages can be sent "copy-on-write.”

5. memory object—a collection of data that is managed by a server that can be mapped into the address space of a task.

The key Unix abstraction of a process can be emulated by a task with a single thread running though it. Files in Mach are mapped into a processes virtual address space, thus allowing file systems to exist in user space managed by external servers.
5 Systems 2 (CSc 553: Principles of Compilation)

Consider the following basic block: here, x, y, z, u, and v represent distinct memory locations, while r1...r6 represent registers. Assume that load operations have a latency of one cycle.

(1) r1 := x
(2) r3 := y
(3) r2 := r1+r3
(4) r4 := z
(5) r2 := r2+r4
(6) r5 := u
(7) r5 := r2+1
(8) r6 := v
(9) r6 := r5+r6
(10) v := r6+1

(a) How many load stalls will be incurred if this instruction sequence is executed as given?

(b) Draw a graph expressing instruction scheduling constraints in the instruction sequence shown above.

(c) Use your scheduling graph to reorder the instructions above such that the resulting sequence is legal (i.e., computes the same result as the original sequence) but which does not incur any load stalls.

Solution

(a) 3 stalls.

(b) There are many possible alternatives, two are shown here:

(1) r1 := x
(2) r3 := y
(4) r4 := z
(6) r5 := u
(8) r6 := v
(3) r2 := r1+r3
(5) r2 := r2+r4
(7) r5 := r2+1
(9) r6 := r5+r6
(10) v := r6+1

(1) r1 := x
(2) r3 := y
(4) r4 := z
(6) r5 := u
(8) r6 := v
(3) r2 := r1+r3
(5) r2 := r2+r4
(7) r5 := r2+1
(9) r6 := r5+r6
(10) v := r6+1
6 Systems 3 (CSc 576: Computer Architecture)

Provide brief answers to the following questions:

1. Identify the three main factors that determine the average memory access time. Give examples of techniques aimed at impacting these factors to result in an overall reduction in average memory access time.

2. Why is a two-bit branch predictor more effective than a one-bit predictor in the handling of loops?

3. How does Scoreboarding differ from Tomasulo’s algorithm in its handling of WAW and WAR hazards?

Solution

1. The average memory access time is a function of: (a) cache hit time; (b) cache miss rate; and (c) cache miss penalty. Cache hit time can be reduced by using virtually-indexed and physically-tagged cache. Victim caches are used to reduce cache miss rate. By giving priority to reads over writes we can reduce cache (read) miss penalty.

2. Once the branch predictor is warmed up, the one-bit prediction scheme will mispredict the loop branch during the first and last loop iteration. On the other hand, a two-bit prediction scheme will only mispredict the loop branch during the last loop iteration.

3. Through register renaming, Tomasulo’s algorithm effectively eliminates these hazards. Each instruction is assigned a unique destination register to which it writes its result by the register renaming mechanism thereby eliminating these hazards. In case of scoreboarding the execution of an instruction is delayed till the hazards are resolved.
Consider a set of processes, $R_0$ to $P_{n-1}$. Each process is connected by a one-way communication channel to its next highest numbered neighbor (with wrap-around so $P_{n-1}$ is connected to $R_0$). Each process, $P_i$, can send on channel $i$, and can receive on channel $i-1$ ($P_0$ receives on channel $n-1$).

A message consists of the destination process number and some data. A process that receives a message not destined for itself, sends the message on its out-bound channel. Message sends and receives are synchronous.

Processes $P_1$ through $P_{n-1}$ execute a loop that does a receive then a send.

(a) [4 points] Process $P_0$ sends $K$ messages, then receives the $K$ messages. Deadlock may occur as a result. What is the lowest value of $K$ for which deadlock will happen? Explain.

(b) [6 points] Let $P_0$ send $K$ messages, then receive messages until it has received the $K$ messages that it originated. Process $P_1$ also sends $K$ messages, then receives messages until it has received the $K$ messages that it originated. Both $P_0$ and $P_1$ will forward any received messages that are not addressed to themselves. Processes $P_2$ through $P_{n-1}$ execute as before. What is the lowest value of $K$ for which deadlock will happen? Explain.

Solution

(a) Deadlock will occur at $K = n$ messages. Since the communication channels are synchronous, a process blocks on a send until the receiving process executes its receive. $P_0$ is only sending and does not receive until all of its messages have been sent. Thus, $P_{n-1}$ will block on its send until $P_0$ does its first receive. $P_{n-2}$ will then block on its send, etc. When each process is blocked on a send, deadlock occurs. For $K = n - 1$ messages, $P_0$ will be doing its first receive when $P_{n-1}$ is doing its first send; no deadlock.

(b) A first, wrong, attempt is to say $K = n - 2$: $P_0$ will block on its first send until $P_1$ has completed all of its sends. $P_1$ will be able to send up to $n - 2$ messages successfully, and begin its receives. At this point, $P_0$ will be able to complete its first send, with $P_1$ forwarding that message (and blocking on that send, since $P_2$ is blocked trying to send to $P_3$, etc). However, $P_0$ will then attempt another send and block since all the processes are blocked on sends and not able to receive. This then points to the correct solution: The communication channels can hold at most $n - 1$ messages in transit; $P_{n-1}$ will be blocked on a send to $P_0$. $P_0$ and $P_1$ have to begin receiving messages by the time the communication channels fill. $P_0$’s first received message will be a message from $P_1$ that $P_0$ will need to forward and $P_1$ will have to be ready to receive. Thus, $P_1$ can send out no more than $\lceil n/2 \rceil$ messages. $P_1$ then is ready to begin receiving. $P_0$ can now send no more than $\lfloor n/2 \rfloor$ messages, which $P_1$ will begin forwarding. When $P_0$ has finished sending its messages, it begins receiving and at this point the system is full of messages (each process other than $P_0$ is blocked on a send trying to forward its current message).
8 Applications 2 (CSc 525: Principles of Computer Networking)

Describe and contrast the following kind of packet-switched networks:

1. Virtual-circuit switching
2. Datagram switching or routing

What are advantages and disadvantages of each of these methods of packet switching for different types of networks and services?

Solution

1. A virtual-circuit network is connection-oriented, meaning that a connection must be established at every switch in the path before any data packets can be sent. At every switch, a virtual-circuit identifier is negotiated at connection establishment time which is unique between the two switches and this happens for every pair of switches in the end-to-end path. The data packets are then sent with virtual circuit identifiers, which are re-written at every switch, instead of globally unique addresses. This can save space in the packet header at the expense of maintaining state information for every connection in each switch. Resources can also be reserved at connection-establishment time for subsequent packets, thus providing an advantage for connections passing many data packets. State in the switches must be reclaimed after data transmission over either by a connection tear-down phase or by garbage collecting soft-state in the switches after a period of inactivity. Routing of a path is done once per-connection, rather than the once per packet of datagram and source routed networks.

2. Datagram switching or routing routes each packet individually based on the destination (and sometimes also the source) address in the packet which must be globally unique. No connection state need be maintained by the router, so the network is connection-less. However, dynamic routing protocols are typically used to communicate the best path to a destination from each datagram switch or router and these routing messages as well as the typically large header sizes of each packet carrying complete addresses count as overhead in datagram networks. No connection tear down phase is necessary and if there is damage to a switch, other paths can be found in the network and communication can proceed without intervention by the sending and receiving hosts.
9 Applications 3 (CSc 533: Computer Graphics)

1. Explain why it is common to use homogenous coordinates.

2. Given the unit cube, whose vertices are

\{(0,0,0), (0,0,1), (0,1,1), (0,1,0), (1,0,0), (1,0,1), (1,1,1), (1,1,0)\}

find a transformation matrix that rotates the cube by \(\theta\) degrees around the line that contains the vertices \((0,0,1)\) and \((1,0,1)\). You can express your answer as a multiplication of several matrices. Describe by words the way you attempt to solve the problem, even if you do not succeed to solve it completely.

3. You are given a matrix describing a picture in a resolution of 300 x 300 pixels. Each entry in the matrix describes the intensity RGB values of one of the pixels. You need to create the same picture with lower resolution, of only 100 x 100 pixels. The proposed algorithm for doing it is divide the large picture into sets of pixels, each set is of 3 x 3 pixels, and to generate the smaller picture by taking the upper left pixels from each set. In other words, if the matrices describing the larger picture is \(M[m_1, m_2]\), then the RGB value of the \((i,j)\) pixel of the output picture is \(M[3i, 3j]\). (note that the upper left pixel of the large pixel is stored in \(M[0,0]\).

List possible sources of aliasing in the new picture, and proposed better algorithm for generating the smaller picture.

Solution

1. The use of homogenous coordinates enables representing many operations such as translations and perspective projections as linear transformations, and hence can be computed as multiplications of matrices.

2. Let \(e\) denote the edge of the cube containing the vertices \((0,0,1)\) and \((1,0,1)\). Note that \(e\) is contained in the plane \(y = 0\), and is parallel to the \(x\)-axis.

We first translate the cube “downward” by one unit, so that \(e\) coincides with the \(x\)-axis. Next we rotate the cube around the \(x\)-axis, by \(\theta\) degrees. Finally, we translate the cube upward by a unit, so \(e\) is in its original location. So the matrix solution is \(T^{-1} \cdot R \cdot T\), where

\[
T = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & -1 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
R = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \theta & -\sin \theta & 0 \\
0 & \sin \theta & \cos \theta & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

3. If the RGB values of the pixels of the smaller picture are merely the RGB values of some pixels of the larger picture, there is the possibility that this subset of pixels would form a shape that does not exist in the larger picture. This phenomenon is known as moire pattern. This phenomenon can be avoided in many cases by using the following simple idea: The RGB values of each pixel in the smaller picture are the average of the corresponding RGB values of the corresponding pixels of the larger picture.
A relational database stores its schemas as tables. Consider a graphical tool to create and modify Entity-Relationship schemas. It makes sense for that tool to store the ER schemas in a relational database. For example, there could be a table

\[
\text{Attribute}(\text{AttributeName}, \ldots)
\]

which would store information about an attribute defined in the ER schema.

Read through all the following questions before starting to answer them.

(a) [6 points] Provide the schema (relation name, attributes, primary key, relevant foreign key) for three of the relations that our ER tool would need.

(b) [6 points] Using this schema, express the following query in relational algebra and SQL.

"List the relationship types that have a participating entity type named ‘Employee’.

(c) [3 points] Express the following query in SQL.

“List the names of the non-binary relationship types.”

(d) (extra credit) Express the query in part (c) in relational algebra.

Solution

(a) This will vary. Here are some sample relations. The first two, or something like them, are needed for the queries.

\[
\begin{align*}
\text{EntityType} & (\text{EntityName}) \\
\text{Primary Key} & \text{EntityName} \\

\text{ParticipatingEntityType} & (\text{RelationshipName}, \text{EntityName}) \\
\text{Primary Key} & \text{RelationshipName} \\
\text{Foreign Key} & \text{EntityName References EntityType} \\

\text{RelationshipType} & (\text{RelationshipName}) \\
\text{Primary Key} & \text{RelationshipName} \\

\text{EntityPrimaryKey} & (\text{EntityName}, \text{AttributeID}) \\
\text{Primary Key} & \text{EntityName, AttributeID} \\
\text{Foreign Key} & \text{AttributeID References Attribute} \\

\text{Attribute} & (\text{AttributeID}, \text{AttributeName}, \text{EntityName}, \text{AttributeType}, \text{MultivaluedFlag}) \\
\text{Primary Key} & \text{AttributeID}
\end{align*}
\]

(b) Relational Algebra:

\[
\pi_{\text{RelationshipName}} (\text{ParticipatingEntityType} \bowtie \sigma_{\text{EntityName} = \text{'Employee'}} (\text{EntityType}))
\]

SQL:

```
SELECT RelationshipName
FROM ParticipatingEntityType AS P, EntityType AS E
WHERE E.EntityName = 'Employee' AND P.EntityName = E.EntityName
```

(c) SQL:

```
SELECT DISTINCT RelationshipName
FROM ParticipatingEntityType AS P, EntityType AS E
WHERE P.EntityName = E.EntityName
GROUP BY RelationshipName
HAVING COUNT(*) > 2
```

(d) Relational Algebra:

\[
\pi_{\text{RelationshipName}} (\sigma_{\text{Count} > 2} (\text{RelationshipName} \bowtie \text{Count}(\text{ParticipatingEntityType} \bowtie \text{EntityType})))
\]