Problem #1
Tell whether each of the following is true or false. In each case, give a brief explanation where possible; for false entries, this should take the form of a counter-example. Assume that comparisons always take constant time.

a. If all items in a heap are distinct, the second-smallest item in a heap has no children (assuming the heap is ordered with the largest item at the top).

b. Suppose that an ordinary binary tree (not a search tree) obeys the heap property - that the value of any node is at least as large as that of any of its children. Then inserting a new item into this tree while maintaining the heap property requires time $O(lg n)$, where $n$ is the number of nodes in the tree.

c. Insertion sort always requires $(n^2)$ time to sort $n$ elements.

d. Quicksort always requires $(nlg n)$ time to sort $n$ elements.

e. Suppose that $RB$ is a red-black tree and that $M$ comparisons are required to discover that a key, $x$, is not present in $RB$. Then if $y$ also is not present in $RB$, it will require at least $M/2$ comparisons to discover this.

Problem #2
Assume that $h(k, n)$ is a hash function that takes a key $k$ and integer $n$ and produces a number in the range $[0...n-1]$. Assume that $K_1, K_2, ...$ is an infinite sequence of keys, and that $h$ distributes these keys "evenly." More precisely, assume that for any $M > 0$, the number of keys $k \in \{K_1, ... K_M\}$ such that $h(k, n) = p$ is $M/n \pm 1$, for each $p \in [0...n - 1]$. Answer the following, giving brief explanations for each answer.

a. Suppose that you are to input a value $M$ and then insert $K_1, ..., K_M$ into a hash table. How can you arrange that each insertion runs in worst-case time $O(1)$?

b. Suppose that $M$ is not explicitly input, and that instead you read keys $K_1, K_2, ...$, inserting them into a hash table as you go, until you encounter some marker indicating the end of the data. Assume that you try to manage the hash table so as to minimize the time required to do all the insertions. What is the (asymptotic) worst-case time for inserting $K_m$ (that is, the $m^{th}$ item you insert) as a function of $m$? Why?

c. Under the same assumptions as (b), what is the total worst-case asymptotic time for doing all $M$ insertions? Why?

Problem #3
A threaded tree is essentially a tree whose nodes are additionally linked together into a list such that following the list gives an inorder traversal of the tree. Consider the following representation.

```cpp
class TTree {
    public:
```
TTree(KeyType k, TTree* L, TTree* R) :
    key(k), left(L), right(R), next(NULL)
{/}

KeyType key;
TTree* left;
TTree* right;
TTree* next;
};

Fill in the following function body to make the comment correct. You may define any additional functions you need.

/* Set the 'next' fields in the tree with root T, so as to link them */
/* into an inorder list. Return a pointer to the first node in */
/* this inorder list (or NULL if there is none.) */
TTree* linkTree(TTree* T)
{ // FILL IN HERE

Problem #4
When the program below is executed, it prints exactly the following three lines:

The answer is
42
What is the question?

Define the classes Action, IntAction, and StrAction so that the program behaves like this.

class Action // FILL IN
{/}
class IntAction // FILL IN
{/}
class StrAction // FILL IN
{/}
typedef List<Action*> AList;

main()
{
    AList* L = new AList(
        new StrAction("The answer is"),
        new AList(
            new IntAction(42),
            new AList("What is the question?", NULL)));
    while (L != NULL) {
L->head->exec(cout);
cout << \'n\';
L = L->tail;
}
}

Problem #5
When *Sefer Yezirah* says "Go on and obtain numbers which the mouth cannot express and the ear cannot hear," to what numbers does it refer?

Problem #6
You are given a set of ordered pairs of real numbers, \((a_i, b_i)\), satisfying \(0 \leq a_i < b_i \leq 1\) for \(0 < i \leq N\). Consider each pair as representing an interval on the real line:

\((a_i, b_i)\) represents \([a_i...b_i] = \{ x | a_i \leq x < b_i \}\)

As quickly as possible, I would like to find and print the endpoints of all intervals between 0 and 1 of numbers that are *not* contained in any of these intervals. For example, if the inputs are

\((0.1, 0.25), (0.0, 0.2), (0.8, 0.9), (0.4, 0.6), (0.35, 0.65)\)

the output should be

\((0.25, 0.35), (0.65, 0.8), (0.9, 1.0)\)

(For the purposes of this problem, we'll ignore the distinction between closed intervals - those that include their endpoints - and open intervals - those that do not.) Present (in pseudo-code) the fastest algorithm you can think of to do this. Keep things at a reasonably high level. You may assume that you have available to you implementations of any of the data structures and algorithms we've discussed, without going into details of how they are implemented. For example, you can say "store the \(a_i\) in hash table \(H\)..." or "heapify the array of \(b_i\)'s" without going into further detail.

---

Posted by HKN (Electrical Engineering and Computer Science Honor Society) University of California at Berkeley
If you have any questions about these online exams please contact examfile@hkn.eecs.berkeley.edu.