2/9/06

4. Homework

Due **2/16/06** before class

1. **Mergesort decision tree**
   Draw the decision tree for mergesort of three elements. Assume that the first split occurs after the first element (so, the first recursive call will be for only one element, and the second recursive call will be for two elements). Remember that the comparisons only take place in the merge routine. Annotate the tree with comments as to which algorithm part a comparison belongs to.

2. **Radix sort with most significant digit first**
   Try to sort the numbers

   $$535, 229, 218, 457, 256, 436, 215, 456, 227, 555, 544$$

   using radix sort but starting with the **most** significant digit (so, from left to right, not from right to left).

   Why would a program that implements this strategy be much more complicated than the radix sort that starts with the least significant digit? (Hint: What kind of variables or data structures would you have to maintain?)

3. **Radix Sort vs. Insertion Sort**

   Given n numbers between 0 and $n!$. Which sorting algorithm is faster: Insertion Sort or Radix Sort?

4. **Randomized code snippets**
   Analyze the **expected** runtimes of the following code snippets. Clearly define your random variables. *Hint: Define a separate random variable for each iteration of the loop. And remember, random variables are functions.*

   a)
   `RandomBit()` takes $O(1)$ time and returns 0 or 1 each with probability 1/2.

```
for(i=1; i<=n; i++){
  if(RandomBit()==1){
    for(j=1; j<=n; j=j+i){ //we had this loop in class before...
      print(``hello'');
    }
  }
}
```

   b)
   `RandomInteger(i)` takes $O(1)$ time and returns an integer between 1 and i, each with probability $1/i$.

```
for(i=1; i<=n; i++){
  if(RandomInteger(i)==i){
    for(j=1; j<=n; j++){
      print(``hello'');
    }
  }
}
```

FLIP OVER TO BACK PAGE ⟹

5. **Sorting Algorithms**

   Consider the following sorting algorithms: Insertion Sort, Heapsort, Mergesort, Quicksort, Counting Sort, Radix Sort.

   For each of these algorithms specify: Best-case runtime together with a best-case input, worst-case runtime together with a worst-case input, expected runtime (if it applies), is this algorithm stable, does this algorithm sort in place, is it a comparison sort or not.

   Justify all your answers briefly.

Related questions from previous PhD Exams

Just for your information. You **do not** need to solve them for homework credit.

P1 (a) A comparison sort is an algorithm that sorts based only on comparisons between pairs of input elements. Which of the following sorting algorithms are comparison sorts: bucket sort, counting sort, heapsort, insertion sort, merge sort, quicksort, radix sort?

(b) A binary decision tree is a binary tree in which the internal nodes are boolean expressions and the leaves are the outcomes. The execution of the decision tree traces a path from the root to a leaf. At each internal node, its expression is evaluated, and the left branch is taken if the expression is true; otherwise the right branch is taken.

i. Describe the decision tree model of comparison sorts.

ii. What are the expressions in the internal nodes?

iii. What are the outcomes at the leaves?

iv. Explain why there are at least $n!$ leaves.

(c) Show an example of the decision tree model for $n = 3$.

(d) The height of the decision tree model represents the worst-case number of comparisons. Show that the height of the decision tree model must be at least $\lg(n!)$.

(e) Show that $\lg(n!)$ is $\Theta(n \lg n)$. Hint: Use the fact that $\lg(mn) = \lg(m) + \lg(n)$.