MA U34605 Quiz 01 w/e 9/10/20 answers

Answer any 3 questions. Submit them using the `submit-work` program as pdfs, either handwritten and scanned, or typeset. They should be submitted before 1pm on Tuesday 13 October. All questions carry 20 marks.

(1) Show that $an + b$ is $O(n)$.

Answer. (Assuming a, b are nonnegative) $an + b \leq (a + b)n$ true for all n.

(2) Binary search works on a sorted array — but 'sorted' was deliberately left undefined. What should the definition be, and what are the implications?

Answer. Sorted could mean 'strictly increasing' or 'nondecreasing.' In both instances binary search works, in that it returns -1 or the index of a matching array entry, but in the second instance all but one of the matching entries will be hidden, which may be undesirable.

(3) Estimate the runtime of the 'inefficient' sorting method (and prove that it always halts). Hint: if $a[0..n-1]$ is an array, an *inversion* for that data is a pair i, j where $0 \leq i < j \leq n-1$ and $a[i] > a[j]$.

Answer. At any time the maximum possible number of inversions is $n(n-1)/2$. Suppose that *ooo* does swap two entries, k and $k+1$, say.

Answer. Before the swap, items $0 \ldots k$ were in nondecreasing order, so all the inversions i, j occurring had $k \leq i < j < n$. The only inversions which are altered would be

$$\begin{aligned} a[k] > a[k+1] &\qquad \text{lost} \\ a[k] > a[j], \qquad k+1 < j &\qquad \text{some lost} \\ a[k+1] > a[j], \qquad k+1 < j &\qquad \textit{preserved} \end{aligned}$$

At least one inversion is lost each time an adjacent pair is swapped, so there are at most $n(n-1)/2$ calls to `ooo()`. Each takes at most n steps so the overall cost is $O(n^3)$.

(4) Give a proportionate (e.g., $\sim n^4$, which is wrong) lower bound for the performance of the inefficient sorting method.

Answer. We consider sorting the array with the maximum possible number of inversions, such as $5, 4, 3, 2, 1$.

```
EG: sorting 5 starting in decreasing order
  5  4  3  2  1
  4  5  3  2  1
  4  3  5  2  1
  3  4  5  2  1
  3  4  2  5  1
  3  2  4  5  1
  2  3  4  5  1
  2  3  4  1  5
  2  3  1  4  5
  2  1  3  4  5
  1  2  3  4  5
```

The pattern is clear. Given $n, n-1, \ldots, 1$ to start with, the final 1 remains in place until all of $2 \ldots n$ are put in sorted order.

Next, the inverted pair $n, 1$ is found, the n in the $n-1$-st position, so it is found after $n-1$ steps, and swapped. This leaves n in its correct place, and an inverted pair $n-1, 1$, which is found in the $n-2$nd position after $n-2$ steps, and swapped. This continues until the inverted pair $2, 1$ is found in the first position. Adding the cost of finding these inverted pairs, we get $n(n-1)/2$.

Therefore if $s(n)$ is the cost of sorting n numbers, initially in reverse order, we get

$$s(n) = s(n-1) + n(n-1)/2$$
$$s(n) = \sum_{r=1}^{n} r(r-1)/2 =$$
$$(n+1)n(n-1)/6 : \qquad \Omega(n^3). \quad \blacksquare$$

(5) The analysis of mergesort was simplified, replacing $an + bS$ by $(a+b)n$. Of course in the $O()$ notation it makes no difference, but this question is: calculate a sharper estimate.

Answer. The total cost of the r-th pass is bounded by

$$an + bS$$

where $S = \lceil n/2^r$. Let $K = \lceil \log_2(n+1) \rceil$, an upper bound on the number of passes. At the K-th pass the array may be copied, but there are no further merge steps. Glossing over that, the overall cost is

$$\sum_{r=1}^{K-1} (an + \lceil \frac{n}{2^r}) \quad \leq \quad an(K-1) + b \sum_{r=1}^{K-1} (\frac{n}{2^r} + 1) \quad \leq \quad anK + bn + K - 1$$

so a sharper estimate, since $K \leq 1 + \log_2(n+1)$, is

$$an(1 + \log_2(n+1)) + bn + \log_2(n+1)$$

which is $O(n \log n)$.

(6) We know that one can reconstruct a binary tree given its inorder and postorder rankings. Is it possible to reconstruct the tree from preorder and postorder rankings? Give reasons, of course.

Answer. No.