

CSE3358 Problem Set 2

01/21/05

Due 01/28/05

Please review the Homework Honor Policy on the course webpage
<http://enr.smu.edu/~saad/courses/cse3358/homeworkpolicy.html>

Problem 1: Practice with asymptotic notations

(a) (3 points) Explain clearly why the statement “The running time of algorithm A is at least $O(n^3)$ ” does not make sense.

(b) (6 points) Let $f(n)$ and $g(n)$ be asymptotically nonnegative functions. Let $h(n) = \max(f(n), g(n))$. Using the precise mathematical definition of Θ -notation, show that $h(n) = \Theta(f(n) + g(n))$. *Hint:* Remember this means you have to show that $h(n) = O(f(n) + g(n))$ and $h(n) = \Omega(f(n) + g(n))$.

Can we say the same thing about $h'(n) = \min(f(n), g(n))$, i.e. is $h'(n) = \Theta(f(n) + g(n))$? Explain.

(c) (3 points) Show that for any real constants a and b , where $b > 0$, $(n + a)^b = \Theta(n^b)$.

(d) (4 points) Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$?

(e) (4 points) Find two non-negative functions $f(n)$ and $g(n)$ such that neither $f(n) = O(g(n))$ nor $g(n) = O(f(n))$.

Problem 2: Recursive Insertion sort

We have seen in class how Merge sort sorts an array by recursively sorting smaller subarrays. A pseudocode similar to the one we saw in class is shown below for convenience.

```
MERGE-SORT( $A, p, r$ )
  if  $p < r$ 
    then  $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$ 
         MERGE-SORT( $A, p, q$ )
         MERGE-SORT( $A, q + 1, r$ )
         MERGE( $A, p, q, r$ )
```

where $\text{MERGE}(A, p, q, r)$ merges the two sorted subarrays $A[p..q]$ and $A[q + 1..r]$.

Insertion sort can also be expressed as a recursive procedure: In order to sort $A[1..n]$, we recursively sort $A[1..n - 1]$ and insert $A[n]$ into the sorted array $A[1..n - 1]$.

(a) (10 points) Write the pseudocode of Insertion sort as a recursive procedure.

(b) (10 points) Let $T(n)$ be the running time of Insertion sort on an array of size n . By identifying the running time of each line in the code of part (a) (either as a Θ notation or as a function of T on smaller arguments), obtain a recurrence equation for $T(n)$.

(c) (10 points) Solve for $T(n)$ by expanding it in a tree-like structure as we did in class.

Problem 3: Insertion sort and merge sort

Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n , insertion sort runs in $8n^2$ steps, while merge sort runs in $64n \log n$ steps.

(a) (5 points) For which values of n does insertion sort beat merge sort?

Although merge sort runs in $\Theta(n \log n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort make it faster for small n . Therefore, it makes sense to use insertion sort within merge sort when subproblems become sufficiently small. Consider a modification of merge sort in which subarrays of size k or less (for some k) are not divided further, but sorted explicitly with Insertion sort.

```
MERGE-SORT( $A, p, r$ )
  if  $p < r - k + 1$ 
    then  $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$ 
      MERGE-SORT( $A, p, q$ )
      MERGE-SORT( $A, q + 1, r$ )
      MERGE( $A, p, q, r$ )
    else INSERTION-SORT( $A[p..r]$ )
```

(b) (5 points) Show that the total time spent on all calls to Insertion sort is in the worst-case $\Theta(nk)$.

(c) (5 points) Show that the total time spent on merging is in the worst-case $\Theta(n \log(n/k))$.

(d) (5 points) Therefore, this modified merge sort runs in $\Theta(nk) + \Theta(n \log(n/k))$ (sorting + merging). The total running time is therefore $\Theta(nk + n \log(n/k))$. What is the largest asymptotic (Θ -notation) value of k as a function of n for which the modified algorithm has the same asymptotic running time as standard merge sort? How should k be chosen in practice?

(10 points) [This is intended to (1) practice the implementation of recursive functions and (2) really appreciate the difference in efficiency between Insertion sort and Merge sort on large size inputs]

Implement in a programming language of your choice this modified version of Merge sort. Therefore, you have to have Insertion sort implemented as well (but hopefully you can use your implementation of Problem Set 1). Compare the running times of Merge sort and Insertion sort on large values of n and report your findings.