Sample Solution to Problem Set 1

1. (6 points) Twenty questions

A friend challenges you to find out a positive integer number n that she has on her mind. For any guess you make, she is willing to tell you which of the following three possibilities is true: n is less than, greater than, or equal to your guess. You have no clue as to how large n is and would like to determine n with the smallest number of guesses.

Describe a guessing procedure for determining n. What is the number of guesses your procedure makes, as a function of n? You must aim at keeping this function as small as possible.

Answer: Our algorithm consists of two phases. In the first phase, we find a number m such that that $m \leq n < 2m$. In the next phase, we perform binary search in the range $[m..2m-1]$.

The first phase consists of the following iterative procedure. In the ith iteration, starting from $i = 0$, we guess 2^i. Thus our guesses go as $1, 2, 4, \ldots$. If 2^i equals n, then we are done and we exit the algorithm. If 2^i is greater than n, then we stop the first phase and and set m to 2^{i-1}. Otherwise, we continue to the next iteration. When the first phase completes, we have $m \leq n < 2m$.

In the second phase, we make our guesses according to a binary search in $[m, 2m-1]$, and stop when we find n.

We now consider the number of guesses. The first phase completes in the earliest iteration i when $2^i \geq n$. That is, $2^i \leq 2n$, implying that $i \leq \lg n + 1$. Thus, the number of guesses in the first phase is $i + 1 \leq \lg n + 2$. The second phase, which is just a binary search in $[m, 2m-1]$ requires at most $\lg m$ guesses. Since $m \leq n$, $\lg m \leq \lg n$. Thus, the total number of guesses is at most $2 \lg n + 2 = O(\lg n)$.

2. (1 + 5 + 5 + 3 = 14 points) Bubblesort

Problem 2-2, page 38.

Answer:

(a) We also need to show that array $A'[1..n]$ consists of the same elements originally in $A[1..n]$.

(b) Invariant for inner loop: At the start of any iteration of the inner loop, $A[j] \leq A[k]$ for all $k \geq j$.

Initialization: For the base case, we let $j = n$, the length of the array. We need to prove that $A[n] \leq A[k]$ for all $k \geq n$; this is trivially true.

Maintenance: Assume that at the start of an iteration of the inner loop, we have $A[j] \leq A[k]$ for all $k \geq j$. In this iteration $A[j]$ is compared with $A[j-1]$ and swapped if $A[j] < A[j-1]$. If the swap does not occur, then $A[j-1]$ is at most $A[j]$. If the swap occurs, then again $A[j-1]$ is at most $A[j]$. By our assumption, $A[j]$ is at most $A[k]$ for all $k \geq j$. Thus, after

the execution of this iteration of the loop, $A[j-1] \leq A[k]$ for all $k \geq j-1$. Therefore, at the start of the next iteration, the desired claim is true.

Termination: At termination of an inner loop, $j = i$. Thus, we have: $A[i] \leq A[k]$ for all $k \geq i$.

(c) The effect of the ith outer loop is that the ith smallest element is moved to $A[i]$. In particular, we will show that the following invariant holds at the start of every iteration of the outer loop: $A[1..i-1]$ contains the $i-1$ smallest elements of A in sorted order and $A[i..n]$ contains the remaining elements of A.

Initialization: When $i = 1$, we need to show that at the start of the first iteration, $A[1..0]$ contains the 0 smallest elements of A in sorted order and $A[1..n]$ contains the remaining elements of A. This trivially true.

Maintenance: We assume first that at the start of an iteration of the outer loop, $A[1..i-1]$ contains the $i-1$ smallest elements of A in sorted order and $A[i..n]$ contains the remaining elements of A. Now consider the next iteration of the outer loop. In this iteration, we execute the inner loop. By the termination of the inner loop ($j = i$), we obtain that $A[i] \leq A[k]$ for all $k \geq i$. Since $A[1..i-1]$ contains the $i-1$ smallest elements of A in sorted order and $A[i]$ is the smallest of the remaining, we obtain that at the end of this iteration of the outer loop: $A[1..i]$ contains the i smallest elements of A in sorted order and $A[i+1..n]$ contains the remaining elements of A. This is the desired claim.

Termination: At termination of the outer loop, $i = n+1$. Thus, we have $A[1..n]$ contains the n smallest elements of A in sorted order, which establishes inequality (2.3) of text, as required.

(d) Line 3 is executed once for every iteration of the inner loop. The inner loop is executed $n-1$ times for every iteration of the outer loop. The outer loop is executed n times. Therefore, line 3 is executed $n(n-1) = n^2 - n$ times. This holds for *any* input instance.

Line 1 is executed n times, line 2 $n(n-1)$ times, and line 4 at most $n(n-1)$ times. Therefore, the worst-case running time is $\Theta(n^2)$. It is the same as that for insertion sort.

3. (1 + 3 + 6 = 10 points) Inversions and insertion sort

Parts (a) through (c), Problem 2-4, pages 39-40.

Hint for part (c): If $T(A)$ is the running time of insertion sort on input A of n numbers and $I(A)$ is the number of inversions in A, then prove that $T(A)$ is $\Theta(I(A) + n)$.

(a) The five inversions are: $(2,1), (3,1), (8,6), (8,1), (6,1)$.

(b) The array that is inverse sorted has the maximum number of inversions since every pair (i,j) (for $i < j$) is an inversion. How many pairs (i,j) such that $i < j$ are there? For $i = 1$, there are $n-1$ possible values for j. For $i = 2$, there are $n-2$ possible values for j. In general, for i, there are $n-i$ possible values for j. Therefore, the total number of inversions in an inverse sorted array equals:

$$\sum_{i=1}^{n}(n-i) = \sum_{k=0}^{n-1} k = \frac{(n-1)n}{2}.$$

(c) Consider what happens in the jth for loop of insertion sort. (See pseudocode on page 17 of text.) At the start of this for loop, all of the elements in $A[1..j-1]$ have been sorted. In the for loop, we first set *key* to $A[j]$ (line 2) and then compare it with elements $A[j-1]$, $A[j-2]$, ..., (the while-loop in lines 5 through 7) until an element smaller than or equal to *key* is found. Thus the total number of executions of the while-statement is exactly equal to 1 more than the number of elements that are in the first $(j-1)$ positions and are greater than $A[j]$. But the elements that are in the first $(j-1)$ positions and are greater than $A[j]$ exactly correspond to the number of inversions of the form $(\cdot, j)$ in A. Thus, if we add over all of the for loops, we get that the total number of while loop executions is $I(A) + (n-1)$ ($I(A)$ is the number of inversions in A). Since there are only a constant number of operations in each while loop, the running time of insertion sort is $\Theta(I(A)+n)$.

4. (10 points) Ordering functions

Arrange the following functions in order from the slowest growing function to the fastest growing function. Briefly justify your answers. (*Hint:* It may help to plot the functions and obtain an estimate of their relative growth rates. It may also help to express each function as a power of 2 and then compare.)

$$n^{1/2} \quad n^2 + \lg n \quad n(\lg n)^5 \quad (\lg \lg n)^2 \quad \lg n$$

Answer: The order from slowest growing to the fastest growing function is: $(\lg \lg n)^2$, $\lg n$, $n^{1/2}$, $n(\lg n)^5$, and $n^2 + \lg n$. While a justification based on plots and/or informal arguments was acceptable for this homework, you must be prepared to provide formal proofs for your claims such as what follows. We prove our ordering using four steps: (i) $(\lg \lg n)^2 = o(\lg n)$, (ii) $\lg n = o(n^{1/2})$, (iii) $n^{1/2} = o(n \lg^5 n)$, and (iv) $n \lg^5 n = o(n^2 + \lg n)$.

(i) We use the connection between asymptotic notation and limits. Let $m = \lg n$. We get $(\lg \lg n)^2 = \lg^2 m$ and $\lg n = m$. So we are comparing $\lg^2 m$ and m. As n tends to infinity, so does m. Therefore, we calculate the following limit.

$$\begin{aligned} \lim_{m\to\infty} \frac{\lg^2 m}{m} &= \lim_{m\to\infty} \frac{2 \lg m}{m \ln 2} \\ &= \lim_{n\to\infty} \frac{2}{m \ln^2 2} \\ &= 0. \end{aligned}$$

Note that we use L'Hopital's rule twice. We thus obtain that $\lg^2 m = o(m)$ yielding $(\lg \lg n)^2 = o(\lg n)$.

(ii) We again use the connection between asymptotic notation and limits.

$$\begin{aligned} \lim_{n\to\infty} \frac{\lg n}{n^{1/2}} &= \lim_{n\to\infty} \frac{2}{(n \ln 2)(n^{-1/2})} \\ &= \lim_{n\to\infty} \frac{2}{\ln 2\sqrt{n}} \\ &= 0, \end{aligned}$$

where we use L'Hopital's rule in the first step. Therefore, $\lg n = o(n^{1/2})$.

(iii) We again use the connection between asymptotic notation and limits.

$$\begin{aligned}\lim_{n\to\infty} \frac{n^{1/2}}{n(\lg n)^5} &= \lim_{n\to\infty} \frac{1}{n^{1/2}(\lg n)^5} \\ &= 0,\end{aligned}$$

Therefore, $n^{1/2} = o(n(\lg n)^5)$.

(iv) We again use the connection between asymptotic notation and limits.

$$\lim_{n\to\infty} \frac{n \lg^5 n}{n^2 + \lg n} = \lim_{n\to\infty} \frac{\lg^5 n}{n + \frac{\lg n}{n}}$$

Here, $\lim_{n\to\infty} \frac{\lg n}{n} = 0$, so we can focus on:

$$\begin{aligned}\lim_{n\to\infty} \frac{\lg^5 n}{n} &= \lim_{n\to\infty} \frac{5 \lg^4 n}{(\ln 2)n} \\ &= \lim_{n\to\infty} \frac{5!}{(\ln^5 2)n} \\ &= 0\end{aligned}$$

Therefore, $n(\lg n)^5 = o(n^2 + \lg n)$.