CS2223
HW#1

DUE: Tuesday, November 2 (**vote!!!**)

1. Consider the following algorithm to compute the MAXIMIM and the MINIMUM elements of an array $A[1..n]$, where $n \geq 2$. We assume that all n elements of A are distinct.

if $A[1]<A[2]$ **then** $\{Big \leftarrow A[2]$
$\quad Little \leftarrow A[1]\}$
else $\{Big \leftarrow A[1]$
$\quad Little \leftarrow A[2]\}$
for $i \leftarrow 3$ **to** n **do**
$\quad$ **if** $Big < A[i]$ **then** $Big \leftarrow A[i]$
$\quad$ **else if** $Little > A[i]$ **then** $Little \leftarrow A[i]$

We want to count the total number of executions of the comparisons

$$\{ A[1]<A[2],\ Big < A[i],\ Little > A[i] \}$$

a (2 points) How many comparisons are executed in the worst-case by the above algorithm?
b (4 points) How many worst-case instances of A are there?
c (2 points) How many comparisons are executed in the best-case by the above algorithm?
d (2 points) How many best-case instances of A are there?

2. (15 points) One algorithm to estimate the size, n, of a set of labelled objects, is to select members of the set randomly, **with replacement**, until any object is selected a second time. If k distinct objects are drawn before the first duplicate, then we estimate n to be $2k^2/\pi$. Test this by writing and executing a program for the following algorithm:

Pick a fixed $n \geq 1$.
$S \leftarrow \varnothing$
$a \leftarrow$ random integer in the range $[1,...,n]$
repeat
$\quad S \leftarrow S \cup \{a\}$
$\quad a \leftarrow$ random integer in the range $[1,...,n]$
until $a \in S$
return $2|S|^2/\pi$

a What operations are performed on *S*?
b Describe a reasonable data structure to implement *S*.
c What is the worst-case time to perform each of the operations of part ***a*** for your data structure of part ***b***? You should use Θ-notation.
d Time your program and try to determine the rate of growth of its execution time as a function of *n*. That is, for each of several values of *n*, execute and time your program. Try to express your program's execution time as a function of *n*.

Describe your implementation (the machine and compiler you are using). Submit a listing of your program, along with evidence that it executes correctly. The evidence should include your selected values of *n* and your estimates of *n*. Are your program's estimates of *n* reasonable?

CS2223
HW#1 SOLUTIONS

1. ***a*** $1+2(n-2)=2n-3$

b In a worst-case instance, the comparison $Big < A[i]$ is always violated. This happens when the MAXIMUM element is among the first two elements of *A*. There are $(n-1)!$ permutations of *A* in which the largest element is the first element of *A*, and $(n-1)!$ permutations of *A* in which the largest element is the second element of *A*. Hence, there are $2(n-1)!$ worst-case instances of *A*.

c n-1

d In a best-case instance, the comparison $Big < A[i]$ is always satisfied. This happens if *A*[1] and *A*[2] contain the two smallest elements of *A*, and then *A*[3..*n*] contain the *n*-2 largest elements in increasing order. There are exactly 2 best-case instances, one is when *A* is sorted in increasing order, and the other is the same instance with the first two elements swapped.

2. ***a*** MAKENULL(*S*), INSERT(*a*,*S*), MEMBER?(*a*,*S*)

b,***c*** One possibility is to store the elements of *S* in the first *k* positions of array int *S*[1..*n*] in the order in which they arrive (unsorted)

MAKENULL(*S*) - $\Theta(1)$
INSERT(*a*,*S*) - $\Theta(1)$
MEMBER?(*a*,*S*) - $\Theta(k)$ where $|S|=k$

If the array *S* were sorted, then the analysis would be

MAKENULL(*S*) - $\Theta(1)$
INSERT(*a*,*S*) - $\Theta(k)$ where $|S|=k$
MEMBER?(*a*,*S*) - $\Theta(\lg n)$