FUNDAMENTAL ALGORITHMS MIDTERM
SOLUTIONS

No calculators, no notes. Do all problems. Maximal score: 125.

1. (20) Let $A[1 \cdots N]$ and $B[1 \cdots N]$ be arrays of numbers that are already in increasing order. Give an efficient algorithm for creating an array $APPLE[1 \cdots 2N]$ consisting of the $2N$ entries in A and B, placed in increasing order. How long (give a short reason) does your algorithm take?
`Solution:` This is the MERGE algorithm. The time is $O(N)$.

2. (20) For the following algorithms let $T(N)$ denote the total number of times the step after the `WHILE` step is reached. For the first algorithm give an *exact* formula for $T(N)$. For the second algorithm first give $T(N)$ as a precise sum. Then find $T(N)$ is the form $T(N) = \Theta(g(N))$ for a standard $g(N)$. Reasons please!

   (a) W=1
   WHILE W < N
   do W=2*W
   END WHILE
   `Solution:` When you hit the WHILE for the t-th time $W = 2^{t-1}$. So you want the maximal t so that $2^{t-1} < N$, or $2^{t-1} \leq N-1$ or $t \leq 1 + \lg(N-1)$ so $\lfloor 1 + \lg(N-1) \rfloor$.

   (b) FOR J=1 TO N
   V=J
   WHILE $V \leq N$
   do V=2*V
   END WHILE
   END FOR
   `Solution:` The inner loop takes (ignoring floors and ceilings) $\lg(N/J)$ so you want $\sum_{J=1}^{N} \lg(N/J)$. One approach is that this is $\lg(N^N/N!)$. But by Stirling's Formula $N^N/N! \sim e^N(2\pi N)^{-1/2}$ so the lg is $\Theta(N)$.

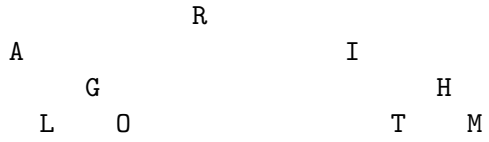
   (c) (20) Let $W[1 \cdots N]$ be an array of integers with all $1 \leq W[i] \leq K$. Give (psuedocode is fine) the algorithm `COUNTINGSORT` that ends with W in increasing order. (You may, and should, create auxilliary arrays.) Analyze the running time of `COUNTINGSORT`

when $K = N$. (Note: It is not sufficient simply to give the answer, an analysis is called for.)
`Solution:`This is basic COUNTINGSORT. When $K = N$ all three parts take time $O(N)$ for a total time of $O(N)$.

3. (25) Consider the following Binary Search Tree `TREE` with `ROOT(TREE)=R`. (The values have been deliberately excluded. Assume the values are distinct.))

vertex	A	G	H	I	L	M	O	R	T
leftchild	NIL	L	T	NIL	NIL	NIL	NIL	A	NIL
rightchild	G	O	M	H	NIL	NIL	NIL	I	NIL
parent	R	A	I	R	G	H	G	NIL	H

   (a) (5) Draw a (nice!) picture of this tree.
   `Solution:`Start at the root with R and children A,I. Continuing



   (b) (10) Which is the vertex with minimal value. Illustrate how the program `MIN` will find it.
   `Solution:`Start at root R. Go to left A. Go to left NIL. So A is the MIN.

   (c) (10) Give the vertices of the tree in increasing order of value. (Give an indication of your method, but you needn't give every detail.)
   `Solution:`IOTW(R) calls IOTW(A). Nothing on left so print A. Now call IOTW(G). Left to L, print L; print G; then print O; finish IOTW(G); finish IOTW(A); print R; Now IOTW(I). Final answer: ALGORIITHM

4. (20) Let A be an array of length 127 in which the values are distinct and in increasing order.

   (a) In the procedure `BUILD-MAX-HEAP(A)` *precisely* how many times will two elements of the array be exchanged? (Reason, please!)
   `Solution:`(from first assignment!) BUILD-MAX-HEAP(A) starts

from I = LENGTH(A)/2 DOWN to 1, every I will do Max-Heapify.
For $32 \leq I \leq 63$,there should be one exchange.
For $16 \leq I \leq 31$, there should be 2 exchanges.
For $8 \leq I \leq 17$, there should be 3 exchanges.
For $I = 4, 5, 6, 7$, there should be 4 exchanges.
For $I = 2, 3$ there should be 5 exchanges.
The root goes down to the bottom, 6 exchanges.
Total: $32 \cdot 1 + 16 \cdot 2 + 8 \cdot 3 + 4 \cdot 4 + 2 \cdot 5 + 1 \cdot 6 = 120$

(b) Now suppose the values are distinct and in decreasing order. Again, in the procedure `BUILD-MAX-HEAP(A)` *precisely* how many times will two elements of the array be exchanged? (Reason, please!)
`Solution:`Never! Each element will be placed originally in precisely its correct final spot.

5. (20) Consider an algorithm `BOHOC` for multiplication of two n digit numbers. (Don't worry about how `BOHOC` really works, we just want an analysis based on the information below.) It multiplies two n digit numbers by making five recursive calls to multiplication of two $n/2$ digit numbers plus two additions of n digit numbers. Each of the additions take time $O(n)$. Give the recursion for the time $T(n)$ for `BOHOC` and use the Master Theorem to find the asymptotics of $T(n)$. Is `BOHOC` a good algorithm to use for n large? Give brief reason for your answer.
`Solution:`$T(n) = 5T(n/2) + O(n)$. This is the low overhead case so $T(n) = \Theta(n^\alpha)$ wth $\alpha = \log_2 5$. As $\log_2 5 > 2$ this is taking more than time $O(n^2)$ which is what the standard multiplication takes so, no, this is not a good algorithm.