FUNDAMENTAL ALGORITHMS FINAL EXAM SOLUTIONS

`Part I:` Do any eight of these problems. I.e., you must **omit** one problem.

1. (20) In Kruskal's MST algorithm there are functions $SIZE[v]$ and $\pi[v]$ (parent). Let the graph have V vertices.
   (a) (15) Show that (at *any* time in the running of the algorithm) if $\pi[v] = u$ then $SIZE[u] \geq 2 \cdot SIZE[v]$.
   `Solution:`At the moment that it was decided to make $\pi[v] = u$ we had $SIZE[u] \geq SIZE[v]$. The new value of $SIZE[u]$ is then $SIZE[u] + SIZE[v] \geq 2 \cdot SIZE[v]$. *After* the value of $SIZE[v]$ never changes since v is no longer a root whereas the value of $SIZE[u]$ might go up but certainly never goes down so the inequality $SIZE[u] \geq 2 \cdot SIZE[v]$ with remain valid.
   (b) (5) Consider the loop
   WHILE $x \neq \pi[x]$
   $\quad x \leftarrow \pi[x]$ (*Call this LoopStep*)
   Give an upper bound, in terms of V, on the number of times LoopStep is applied in this WHILE loop. Reason please!
   `Solution:`The initial $SIZE[x] \geq 1$. After Loopstep is applied t times we have $SIZE[x] \geq 2^t$ from the previous part. But $V \geq SIZE[x]$ as V is the total number of vertices. Thus $2^t \leq V$ and $t \leq \lg V$.

2. (20) Use the Master Theorem to give, in Thetaland, the asymptotics of these recursions:
   (a) (5) $T(n) = 6T(n/2) + n\sqrt{n}$
   `Solution:`$\log_2 6 > 1.5$ so this is Low Overhead, $T(n) = \Theta(n^c)$ with $c = \log_2 6$.
   (b) (5) $T(n) = 4T(n/2) + n^5$
   `Solution:`$\log_2 4 = 2 < 5$ so this is High Overhead, $T(n) = \Theta(n^5)$.
   (c) (5) $T(n) = 4T(n/2) + 7n^2 + 2n + 1$
   `Solution:`$\log_2 4 = 2$ so this is the just right overhead regime and $T(n) = \Theta(n^2 \ln n)$.
   (d) (5) $T(n) = 3T(n/2) + 10n$.
   `Solution:`$\log_2 3 > 1$ so this is low overhead and $T(n) = n^c$ with $c = \log_2 3$. (This happens in Karatsuba's Algorithm.)

3. (20) Let T be a BST. Suppose, in addition to the usual parent, leftchild, rightchild, ROOT, we have a function $desc[v]$. This gives, for each node v in the tree, the total number of descendents of v, including v itself. Now suppose we apply $INSERT[z]$ which adds new vertex z to T.

   (a) (15) *Update* the values $desc$ in an efficient manner. (`Warning:` It is not sufficient to set $desc[z] = 1$. Some other vertices v will have their value $desc[v]$ changed.
   `Solution:`We must increment by one z and all of the ancestors of z One way:
   WHILE $z \neq NIL$
   $desc[z]++$
   $z \leftarrow \pi[z]$.

   (b) (5) Suppose also we have a parameter SAM which is the sum of all of the values $desc[v]$. Update SAM in an efficient manner.
   `Solution:`Simply increment SAM everytime $desc$ is incremented. E.g.:
   WHILE $z \neq NIL$
   $desc[z]++$
   $SAM++$
   $z \leftarrow \pi[z]$.

4. (20) Let G be a DAG [Directed Acyclic Graph]. Suppose $w \in Adj[v]$. Let $d[x], f[x]$ denote, as usual, the discovery and finishing times for vertex x when running $DFS[G]$.

   (a) (10) Suppose $d[v] < d[w]$. Give *with logical argument* the order in which $d[v], f[v], d[w], f[w]$ will appear.
   `Solution:`As $w \in Adj[v]$, w will be discovered before $DFS - VISIT[v]$ is completed. Then $DFS - VISIT[w]$ will be called, this is inside $DFS = VISIT[v]$, so $DFS - VISIT[w]$ will finish before $DFS = VISIT[v]$, so the order is $d[v] < d[w] < f[w] < f[v]$.

   (b) (10) Suppose $d[w] < d[v]$. Give *with logical argument* the order in which $d[v], f[v], d[w], f[w]$ will appear.
   `Solution:`As G is a DAG during the running of $DFS - VISIT[w]$ the vertex v will not be discovered. (If it were there would be a path from w to v and that would make a cycle.) Thus the order is $d[w] < f[w] < d[v] < f[v]$.

5. (20) Here is a segment of `COUNTINGSORT`. We assume $A[1\cdots N]$ has all $0 \leq A[I] \leq K$ and that $B[1\cdots N]$ and $C[0\cdots K]$ are initially all zeroes. We assume $K \geq 7$ to avoid trivialities below.
FOR $I = 1$ TO N; $C[A[I]] = ++$; ENDFOR (*end part one*)
FOR $I = 1$ TO K; $C[I] = C[I] + C[I-1]$; ENDFOR (*end part two*)
FOR $I = N$ DOWN TO 1
    $VALUE = A[I]$
    $PLACE = C[VALUE]$
    $B[PLACE] = VALUE$
    $C[VALUE] = C[VALUE] - 1$
ENDFOR (*end part three*)

   (a) (5) Describe $C[7]$ at the end of part one.
   `Solution:` $C[7]$ is the number of J for which $A[J] = 7$

   (b) (5) Describe $C[7]$ at the end of part two.
   `Solution:` $C[7]$ is the number of J for which $A[J] \leq 7$

   (c) (10) Describe $C[7]$ at the end of part three.
   `Solution:` In part 3 $C[7]$ will be decremented for each time 7 appears so at the end $C[7]$ is the number of J for which $A[J] \leq 6$.

   The descriptions should be in terms of the values $A[J]$. They could be something like: $C[7]$ is the sum of all values $A[J]$ – but of course that would be the wrong answer. The descriptions must be in clear concise *words*.

6. (20) List the parenthesizations of $ABCD$.
`Solution:` $(A(BC))D; ((AB)C)D; A(B(CD)); A((BC)D); (AB)(CD)$ Let $P(n)$ denote the number of parenthesizations of $A_1 \cdots A_n$. Give a recursive formula for $P(n)$ and an argument for why it holds.
`Solution:` The formula is

$$P(n) = \sum_{i=1}^{n-1} P(i)P(n-i)$$

Any parenthesization has a lead break – for some $1 \leq i \leq n-1$ the products $A_1 \cdots A_i$ and $A_{i+1} \cdots A_n$ are parenthesized and then multiplied. There are $P(i)$ choices for the left parenthesization and $P(n-i)$ choices for the right parenthesization and, as these are independent choices, they are multiplied. Each i gives a term and the total number is the sum over possible i.

7. (20) Let $A[1\cdots N], B[1\cdots N], C[1\ldots N]$ each be arrays in increasing order. Assume, for convenience, that all values are distinct. Give a procedure `TRIDENT` which creates an array $D[1\cdots(3N)]$ which has the $3N$ values in increasing order. Pseudocode allowed. *Please* include comments describing what is happening in your procedure.
`Solution:`First increment the arrays by setting $A[N+1] = B[N+1] = C[N+1] = \infty$. Create three markers am, bm, cm, all initially 1. Now for $j = 1$ to $3n$ check which is the smallest of $A[am], B[bm], C[cm]$. If it is $A[am]$ set $D[j] = A[am]$ and move am one to the right - $am{+}{+}$. If it is $B[bm]$ set $D[j] = B[bm]$ and move bm one to the right - $bm{+}{+}$. If it is $C[cm]$ set $D[j] = C[cm]$ and move cm one to the right - $cm{+}{+}$.

8. (20) Let $A[1\cdots 127]$ be an array in no particular order. Apply the following procedure:
   FOR I = 63 DOWN TO 8 (*Warning: Check endvalue!*)
   MAX-HEAPIFY(A,i)
   Let $OLDA[1\cdots 127]$ and $NEWA[1\cdots 127]$ below denote the original and final values of A respectively.
   (a) (10) OMITTED
   `Solution:`OMITTED
   (b) (5) Give the relationship between $NEWA[5]$ and the values in $OLDA$.
   `Solution:`$NEWA[5] = OLDA[5]$, it hasn't changed!
   (c) (5) Make a very small change in the procedure above to create a well studied procedure. What is the name of that procedure?
   `Solution:BUILD-HEAP` is the procedure when the first step is FOR I=63 DOWN TO 1.

9. (20) Assume modulo m multiplication of two numbers can be done in one nanosecond. Let s be a positive integer, at most 10^{100}, given in binary form. Show how 3^s modulo m can be computed in less than a microsecond.
`Solution:`Say

$$s = 2^{s_r} + \ldots + 2^{s_1} \text{ with } s_r > \cdots > 0$$

As $10^{100} < 2^{400}$ (leaving lots of room!) we have $s_r < 400$. Set $X_0 = 3$ and recursively calculate $X_t = X_{t-1}^2 \bmod m$ for $1 \le t \le s_r$. Thats at most 400 nanoseconds. Now we want the product of the X_{s_i}. Multiply

X_{s_1} times X_{s_2}, then that result times X_{s_3}, then that result times X_{s_4} (all mod m), so in $r-1$ multiplications – again at most 400 nanosecond, we get the 3^s. Total at most 800 nanoseconds, less than a microsecond.

`Part II:` Do any *three* of these problems. That is, you must **omit** one problem. Whichever you choose you **must** give substantial comments describing what the algorithm is doing and what the various functions represent. Pseudocode is fine. You must give a (brief!) explanation for how long the program takes and *why* it takes that long.
`Solution:` These programs are all given in the text.

1. (20) Describe Dijkstra's Algorithm. The input is a directed graph G given by Adjacency List Representation, a source vertex s, and a weight function $w[x, y] \geq 0$ defined for all edges (x, y) of the graph. The output will be $d[v]$, the length of the shortest path from s to v, and a parent function $\pi[v]$.

2. (20) Let G be a graph with N vertices and E edges. Let s be a vertex of G. Give the algorithm $BFS[G, s]$ for Breadth First Search.

3. (20) Give an efficient sorting algorithm (your choice!). The input is an array $A[1 \cdots n]$ of values and the output should be the same numbers in a sorted array $B[1 \cdots n]$. (You may make no assumption about the nature of the data.)

4. (20) Give the LCS (longest common subsequence) algorithm. The input will be $A[1 \cdots N], B[1 \cdots N]$ sequences with all values either zero or one. The output will be the length of the Longest Common Subsequence. (You are *not* being asked to output the LCS itself!) In this algorithm there will be an array $c[i, j]$. What will the value $c[i, j]$ represent?