Sorting Algorithm.

1. Descendants of insertion sort

1A. /* insertion sort */

(1 ≤ j ≤ #seq) k = j-1; (while k ≥ 0 and seq(k+1) < seq(k))

〈seq(k), seq(k+1)〉〈seq(k+1), seq(k)〉; k = k-1; end while;; exit;

1B. /* centered insertion sort */

/*/ initialize */ tseq = nl; median = 0; dptr = 0; uptr = 0;
tseq(0) = seq(1);

(2 ≤ j ≤ #seq) if seq(j) < tseq(median)

then /* insert to the left of the median */

k = dptr-1; tseq(k) = seq(j); (while tseq(k) ≥ tseq(k+1))

〈tseq(k), tseq(k+1)〉〈tseq(k+1), tseq(k)〉; k = k+1;

end while; dptr = dptr+1;

else /* insert to the right */

k = uptr+1; tseq(k) = seq(j); (while tseq(k) ≤ tseq(k-1))

〈tseq(k-1), tseq(k)〉〈tseq(k), tseq(k-1)〉; k = k-1;

end while; uptr = uptr+1; end else;

median = (uptr + dptr)/2; end ∀;

/*/ copy tseq back into seq */

j = 1; (dptr < m ≤ uptr) seq(j) = tseq(m); j = j+1; end ∀m;

exit;

1C. /* sliding insertion sort */

k = log2_#seq; int = 2**k-1;
Ll: (l ≤ ∀m ≤ int) /* do for each subsequence /*
(Q ≤ ∀j ≤ (#seq-m)/int) tseq(j+1) = seq(,j*int+m);
/* copy it into tseq /*
call insertsort (tseq);
(0 ≤ ∀j ≤ (#seq-m)/int) seq(j*int+m) = tseq(j+1);
/* copy sorted tseq into seq /*
end ∀m;
M = M/2; if M gt 0 go to Ll; else exit;;

LD. /* calculated insertion sort /*
(l ≤ ∀j ≤ #seq) call elinpcts(seq(j), pockts); /* PCKTS is
sequence of pockets and ELINPCTS puts SEQ(J) into appropriate
pocket - PCKTS(K) /*
n=1; (l ≤ ∀k ≤ #pockts) call insertsort(pockts(k));
/* pockets are sorted /*
(l ≤ ∀m ≤ #pockts(k)) seq(n) = pockts(k,m); n = n+1;
end ∀m; /* sorted pockets are written into seq /*
end ∀j; exit;

10. /* tree insertion sort /*
e1t = nl; r = nl; l = nl;
tree = nl; ntop = newat; elt(ntop) = seq(1);
(2 ≤ ∀j ≤ #seq) x = seq(j); top = ntop;
Ll: if x gt elt(top) then if r(top) = then r(top) = newat;
e1t(r(top)) = x;
else top = r(top); go to L1;

else if l(top) = ∅ then l(top) = newat;
elt(l(top)) = x;
else top = l(top); go to L1;
end else;

end Vj; seq = nl; traverse ntop; exit;
define traverse top; external seq;
if top eq ∅ then return; else traverse l(top); seq(#seq+1)=elt(top);
traverse r(top); return; end traverse;