SETL Newsletter Number 10

December 9, 1970
Kurt Maly

Sorting Algorithm.

1. Descendants of insertion sort

1A. /*/ insertion sort /*/

```
(1 ≤ ∀j ≤ #seq) k = j-1; (while k gt 0 $ and seq(k+1) lt 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) lt tseq(median)
     then /*/ insert to the left of the median /*/
          k = dptr-1; tseq(k) = seq(j); (while tseq(k) gt 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) lt 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;
```

```
L1:  (1 ≤ ∀m ≤ int) /*/ do for each subsequence /*/
     (0≤∀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 L1; else exit;;

1D.  /*/ calculated insertion sort /*/

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

1E.  /*/ tree insertion sort /*/
elt = nl; r = nl; l = nl;
tree = nl; ntop = newat; elt(ntop) = seq(1);
(2 ≤ ∀j <#seq) x = seq(j); top = ntop;
L1:  if x gt elt(top) then if r(top) = ∧ then r(top) = newat;
elt(r(top)) = x;
```

```
                                    else top = r(top); go to L1;;
            else if l(top) = Λ then l(top) = newat;
            elt(l(top)) = x;
                                    else top = l(top); go to L1;;
                                    end else;
end ∀j; seq = nl; traverse ntop; exit;
define  traverse top; external seq;
if top eq Λ then return; else traverse  l(top); seq(#seq+1)=elt(top);
traverse  r(top); return; end traverse;
```