Bubble Sort

```
for i = n downto 2 do
    for j = 1 to i-1 do
        if A[j] > A[j+1]
            then A[j] ↔ A[j+1]
        end if
    end for
end for
```

```
for i = n downto 2 do
    for j = 1 to i-1 do
        if A[j] > A[j+1]
            then A[j] ↔ A[j+1]
        end if
    end for
end for
```

Bubble Sort (Modified)

```
i ← n-1
while i > 0 do
    t ← 1
    for j = 1 to i do
        if A[j] > A[j+1] then
            A[j] ↔ A[j+1]
            t ← j
        end if
    end for
    i ← t-1
end while
```

Cocktail-Shaker Sort

No code.

Insertion Sort with Sentinel

```
A[0] ← −∞
for i = 2 to n do
    t ← A[i]
    j ← i-1
    while t < A[j] do
        A[j+1] ← A[j]
        j ← j-1
    end while
    A[j+1] ← t
end for
```

```
A[0] ← −∞
for i = 2 to n do
    t ← A[i]
    j ← i-1
    while t < A[j] do
        A[j+1] ← A[j]
        j ← j-1
    end while
    A[j+1] ← t
end for
```

Insertion Sort (without sentinel)

```
for i = 2 to n do
    t ← A[i]
    j ← i-1
    while j>0 and A[j]>t do
        A[j+1] ← A[j]
        j ← j-1
    end while
    A[j+1] ← t
end for
```

```
for i = 2 to n do
    t ← A[i]
    j ← i-1
    while j>0 and A[j]>t do
        A[j+1] ← A[j]
        j ← j-1
    end while
    A[j+1] ← t
end for
```

Selection Sort

```
for i = n downto 2 do
    k ← 1
    for j = 2 to i do
        if A[j] > A[k] then k ← j
    end for
    A[k] ↔ A[i]
end for
```

```
for i = n downto 2 do
    k ← 1
    for j = 2 to i do
        if A[j] > A[k] then k ← j
    end for
    A[k] ↔ A[i]
end for
```

Merge Sort

```
procedure MergeSort(A,p,r)
    if p<r then
        q ← ⌊(p+r)/2⌋
        MergeSort(A,p,q)
        MergeSort(A,q+1,r)
        Merge(A,(p,q),(q+1,r))
    end if
end procedure

procedure Merge(A,(p,q),(q+1,r))
    copy (A,p,r) into (B,p,r)
    i ← p
    j ← q+1
    k ← p
    while i ≤ q AND j ≤ r do
        if B[i] ≤ B[j] then
            A[k] ← B[i]
            i ← i+1
        else
            A[k] ← B[j]
            j ← j+1
        end if
        k ← k+1
    end while
    if i>q
        then copy (B,j,r) into (A,k,r)
        else copy (B,i,q) into (A,k,r)
    end if
end procedure
```

Heap Sort

```
proc heapsort(A: list,  n: list size)
    {Create heap}
    for r = ⌊n/2⌋ downto 1 do
        sift(r,n)
    end for
    {Finish Sort}
    for m = n downto 2 do
        A[1] ↔ A[m]
        sift(1,m-1)
    end for
end proc

proc sift(p: root,  m: size of list)
    c ← 2*p
    while c ≤ m do
        if c < m then
            if A[c+1] > A[c] then c ← c+1 end if
        end if
        if A[c] > A[p] then
            A[p] ↔ A[c]
            p ← c
            c ← 2*p
        else
            exit while loop
        end if
    end while
end proc
```

Quicksort

```
procedure quicksort(A,p,r)
    if p<r then
        q ← partition(A,p,r)
        quicksort(A,p,q-1)
        quicksort(A,q+1,r)
    end if
end procedure

function partition(A,p,r)
    X ← A[r]
    i ← p-1
    for j = p to r-1 do
        if A[j] ≤ X then
            i ← i+1
            A[i] ↔ A[j]
        end if
    end for
    A[i+1] ↔ A[r]
    return(i+1)
end function
```