

Bubble Sort

Pseudocode implementation

The algorithm can be expressed as (0-based array):

```
procedure bubbleSort( A : list of sortable items )
  n = length(A)
  repeat
    swapped = false
    for i = 1 to n-1 inclusive do
      /* if this pair is out of order */
      if A[i-1] > A[i] then
        /* swap them and remember something changed */
        swap( A[i-1], A[i] )
        swapped = true
      end if
    end for
  until not swapped
end procedure
```

Optimizing bubble sort

The bubble sort algorithm can be easily optimized by observing that the n-th pass finds the n-th largest element and puts it into its final place. So, the inner loop can avoid looking at the last n-1 items when running for the n-th time:

```
procedure bubbleSort( A : list of sortable items )
  n = length(A)
  repeat
    swapped = false
    for i = 1 to n-1 inclusive do
      if A[i-1] > A[i] then
        swap(A[i-1], A[i])
        swapped = true
      end if
    end for
    n = n - 1
  until not swapped
end procedure
```

More generally, it can happen that more than one element is placed in their final position on a single pass. In particular, after every pass, all elements after the last swap are sorted, and do not need to be checked again. This allows us to skip over a lot of the elements, resulting in about a worst case 50% improvement in comparison count (though no improvement in swap counts), and adds very little complexity because the new code subsumes the "swapped" variable: To accomplish this in [pseudocode](#) we write the following:

```
procedure bubbleSort( A : list of sortable items )
  n = length(A)
  repeat
    newn = 0
    for i = 1 to n-1 inclusive do
      if A[i-1] > A[i] then
        swap(A[i-1], A[i])
        newn = i
      end if
    end for
    n = newn
  until n = 0
end procedure
```

Alternate modifications, such as the [cocktail shaker sort](#) attempt to improve on the bubble sort performance while keeping the same idea of repeatedly comparing and swapping adjacent items.