Shellsort

Computational complexity

Best Case	Average Case	Worst Case
$O(n \cdot log(n))$	$O(n^2)$	$O(n^2)$ (worst known steplength sequence) $O(n \cdot log(n)^2)$ (best known steplength sequence)

WORST CASE: The conditions under which the worst case occurs depend on steplengths that are used. For Shell's originally proposed steplength sequence based on 2^n (eg. steplengths = [8,4,2,1]), the worst case performance may appear when the binary representation of the number of elements contains many consecutive zeroes (eg. $1024_{10} \rightarrow 10000000000_2$). This steplength-sequence leaves out uneven numbers until it gets to 1 and therefore is not the optimal choice.

BEST CASE: This occurs when the array is already sorted. Because the number of elements in the steplengths-array is $log(n)$ (again for Shell's original proposal) and for each steplength we need to confirm that the order of elements in the array is correct (in $O(n)$ as a sorted array is the best case for Insertionsort), the resulting complexity is $O(n \cdot log(n))$.

Pseudocode

`array`: array with numbers that are to be sorted

`steplengths`: array with steplengths that are to be used. In our definition, steplengths are arranged in descending order (eg. steplengths = [8,4,2,1]). A practical steplengths-array must further fulfill the conditions $steplengths[1] < array.length$ and $steplengths[steplengths.length] == 1$.

```
procedure shellsort(array, steplengths)
-----------------------------------------
begin
  for each steplength in steplengths do
  begin
    for i := 1 to steplength do
    begin
          sort(array, i, steplength)
    end
  end
end

procedure sort(array, i, steplength)
--------------------------------------
perform Insertionsort using the following positions in the array:
 i, i + steplength, i + steplength*2, ...
```