Comb sort - Summary

By: Jackie Xu

- **History**
  - Originated from Bubble sort (1950s)
  - Developed by Włodzimierz Dobosiewicz in 1980
  - Re-discovered and optimized by Stephen Lacey and Richard Box in 1991
- **General Information**
  - Method of sorting: Exchanging
  - Based off of: Bubble sort (behaves like Bubble near the end)
- **Time complexity**
  - Efficiency
    - Best case: O(n)
    - Worst case: $O(n^2)$
    - Average case: O(n log n)
  - Memory
    - O(1) - No extra memory needed during sorting process
- **Pros**
  - Easy to write - Uses simple structures (Decisions, while loops, arrays/lists)
  - Memory efficient - Does not need to reserve more memory during the sort
  - On average more efficient than Bubble, Selection and Insertion sorts
    - It improves Bubble sort by the eliminating "turtles" in the beginning

○ **Cons**

- Unstable sort - It cannot sort lists/arrays with complex objects (multiple attributes) (Eg. Cards)

○ **Key terms / variables**

- "turtles" - Small values at the end of a list that tends to slow down a sorting method
- Shrink factor - A number that is used to determine the gap value in a Comb sort (ideally 1.3)
- gap size - The number of offsets from the current counter, crucial in a Comb sort
- stable sort - A sort that will guarantee that all the attributes in a list will be sorted
  (Eg. Suits and numbers in a list of cards will be properly sorted)
- unstable sort - A sort that will NOT guarantee that all the attributes in a list will be sorted
  (Eg. Suits in a deck of cards may not be properly sorted, only the numbers)

○ **When to and when not to use**

- USE when sorting a list of primitive variables (Eg. floats, integers, chars...)
- DON'T use when sorting a list of objects with multiple attributes (Eg. Cards)

○ **Pseudocode**

```
// Comb sort
shrink = 1.3     // shrink factor (ideally 1.3)
gap = input.size
while (gap != 1 or swapped == true)     // terminates when list is sorted
        gap = int (gap / shrink)
        if (gap < 1)
                gap = 1     // The sorts behaves like bubble sort after this

        swapped = false    // Initializes condition variable
        i = 0
        while (i + gap < input.size)     // loop that sets boundaries
                if (input [i] > input [i + gap])    // swapping algorithm
                        swap (input [i], input [i + gap])
                        swapped = true
                i++
```