Shellsort

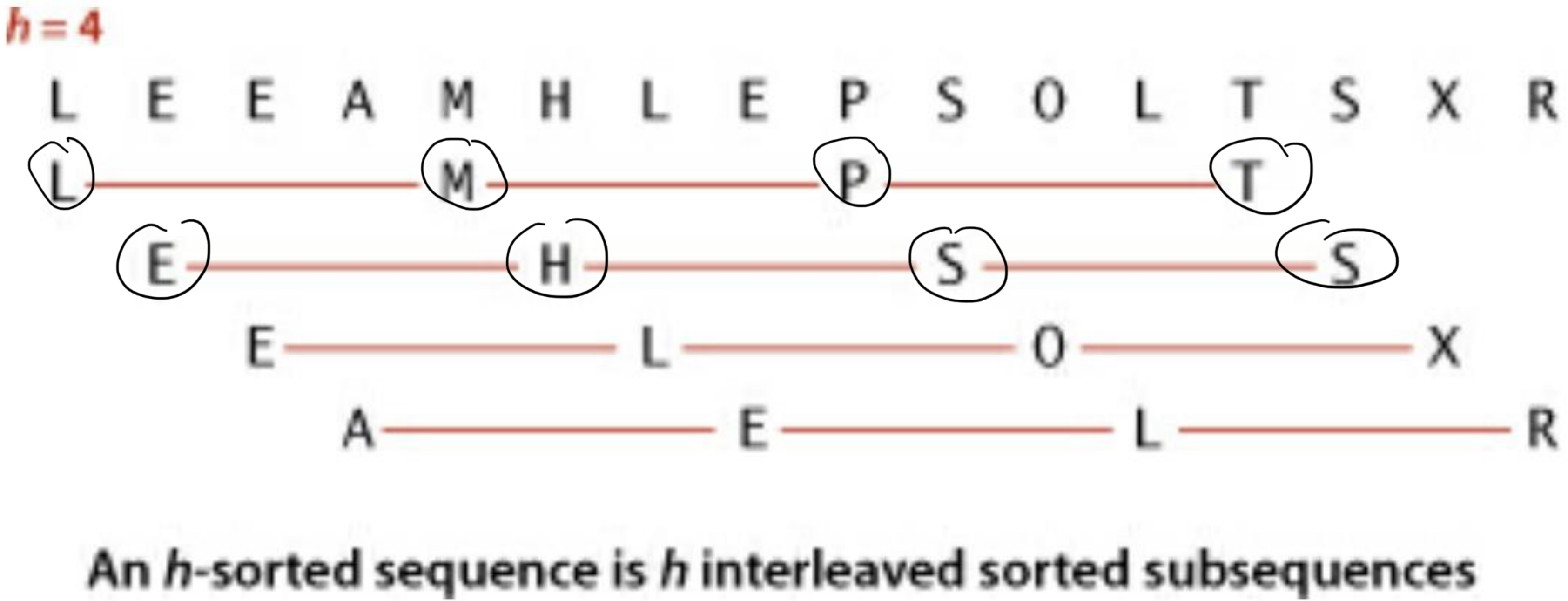
Intuition

Insertion Sort is slow for large values of N partly because elements can only move through the array **one position at a time**. So...why not alter it so that elements can move through the array more quickly (e.g. **13, 4, etc. positions at a time**)?

Shellsort: Overview

- Use insertion sort on every h^{th} value, creating sorted subsequences.
- Decrement the value of *h* according to an **increment sequence,** until $h = 1$ and you are just performing insertion sort on the array.
- When *h* is larger, the subsequence you are sorting is smaller.
- As *h* gets smaller, the subsequence you are sorting gets bigger, but it is also partially sorted.

Shellsort: h–sorted sequence



Picture from [1]

h = 4

3 4 1 0 9 8 2

4-sorted

h = 1

3 4 1 0 9 8 2

1 3 4 0

0 1 3 4 9 8

0 1 3 4 8 9 2

0 1 2 3 4 8 9

Shellsort: Example

input	S	H	E	L	L	S	O	R	T	E	X	A	M	P	L	E
13-sort	P	H	E	L	L	S	O	R	T	E	X	A	M	S	L	E
4-sort	L	E	E	A	M	H	L	E	P	S	O	L	T	S	X	R
1-sort	A	E	E	E	H	L	L	L	M	O	P	R	S	S	T	X

Shellsort trace (array contents after each pass)

Picture from [1]

input	S	H	E	L	L	S	O	R	T	E	X	A	M	P	L	E
13-sort	P	H	E	L	L	S	O	R	T	E	X	A	M	S	L	E
	P	H	E	L	L	S	O	R	T	E	X	A	M	S	L	E
	P	H	E	L	L	S	O	R	T	E	X	A	M	S	L	E
4-sort	L	H	E	L	P	S	O	R	T	E	X	A	M	S	L	E
	L	H	E	L	P	S	O	R	T	E	X	A	M	S	L	E
	L	H	E	L	P	S	O	R	T	E	X	A	M	S	L	E
	L	H	E	L	P	S	O	R	T	E	X	A	M	S	L	E
	L	H	E	L	P	S	O	R	T	E	X	A	M	S	L	E
	L	E	E	L	P	H	O	R	T	S	X	A	M	S	L	E
	L	E	E	L	P	H	O	R	T	S	X	A	M	S	L	E
	L	E	E	A	P	H	O	L	T	S	X	R	M	S	L	E
	L	E	E	A	M	H	O	L	P	S	X	R	T	S	L	E
	L	E	E	A	M	H	O	L	P	S	X	R	T	S	L	E
	L	E	E	A	M	H	L	L	P	S	O	R	T	S	X	E
	L	E	E	A	M	H	L	E	P	S	O	L	T	S	X	R
1-sort	E	L	E	A	M	H	L	E	P	S	O	L	T	S	X	R
	E	E	L	A	M	H	L	E	P	S	O	L	T	S	X	R
	A	E	E	L	M	H	L	E	P	S	O	L	T	S	X	R
	A	E	E	L	M	H	L	E	P	S	O	L	T	S	X	R
	A	E	E	H	L	M	L	E	P	S	O	L	T	S	X	R
	A	E	E	H	L	L	M	E	P	S	O	L	T	S	X	R
	A	E	E	E	H	L	L	M	P	S	O	L	T	S	X	R
	A	E	E	E	H	L	L	M	P	S	O	L	T	S	X	R
	A	E	E	E	H	L	L	M	P	S	O	L	T	S	X	R
	A	E	E	E	H	L	L	M	O	P	S	L	T	S	X	R
	A	E	E	E	H	L	L	L	M	O	P	S	T	S	X	R
	A	E	E	E	H	L	L	L	M	O	P	S	T	S	X	R
	A	E	E	E	H	L	L	L	M	O	P	S	S	T	X	R
	A	E	E	E	H	L	L	L	M	O	P	S	S	T	X	R
	A	E	E	E	H	L	L	L	M	O	P	R	S	S	T	X
result	A	E	E	E	H	L	L	L	M	O	P	R	S	S	T	X

Detailed trace of shellsort (insertions)

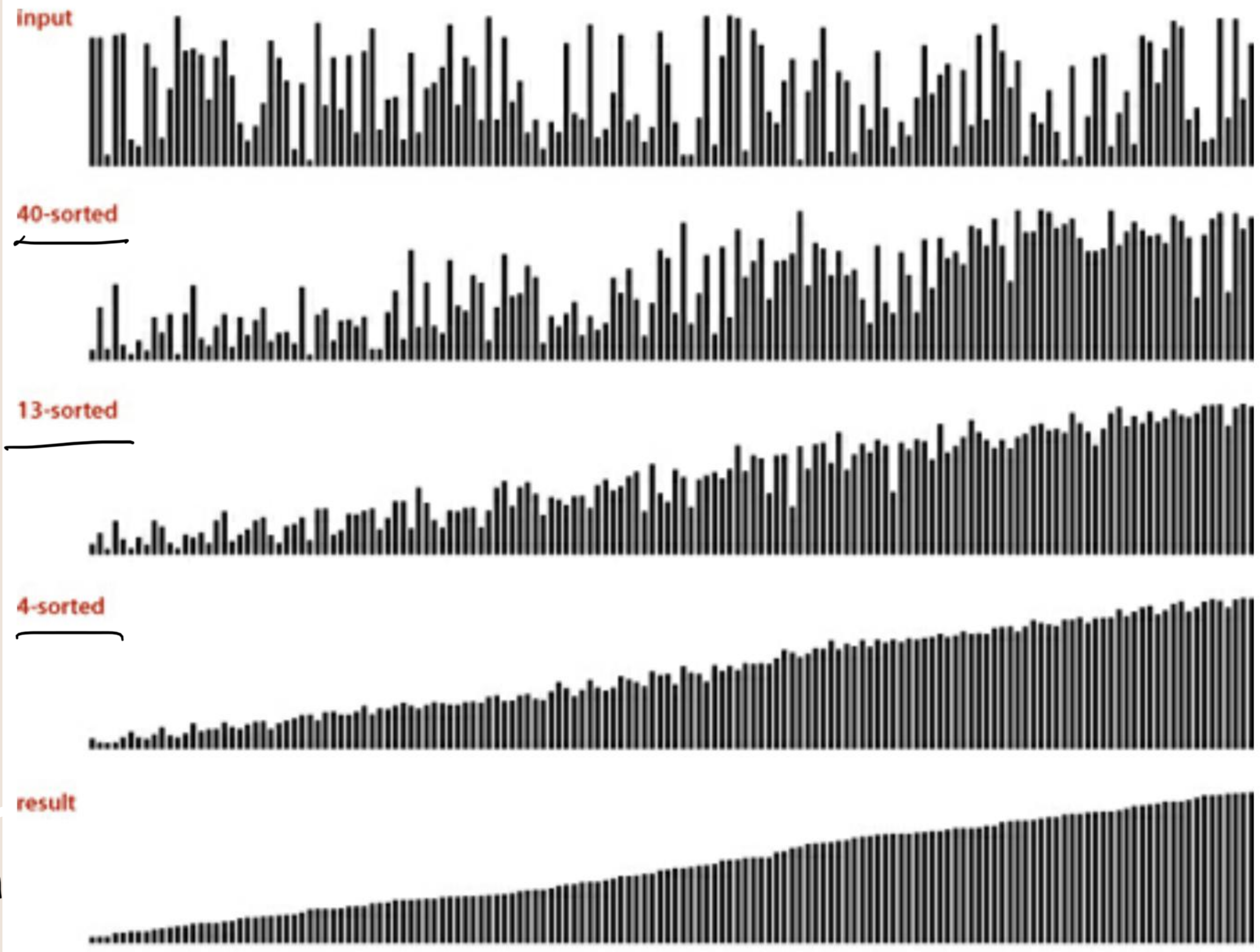
Picture from [1]

$3(13) + 1 =$

$3(4) + 1 =$

$3(1) + 1 =$

$h = 1$



Visual trace of shellsort

Picture from [1]

Analysis

- The increment sequence does matter, as it affects HOW sorted the subsequences are.
- Performance depends on:
    - Number of increments
    - Arithmetical interactions among increments (e.g. size of common divisors)
- This makes analysis of the performance difficult!
- So far, no one has found a **provably best** increment sequence.
- Our implementation: worst-case number of compares is $\mathbf{O(N^{3/2})}$, which is still better than quadratic.
- Our *h*-sequence is determined by $h_k = 3h_{k-1}+1$ and $h_0=0$, and the first *h* is chosen to be the first one where $h >= N/3$.

Shellsort Summary

- Small alteration to insertion sort beats quadratic time!
- Generally acceptable running time for moderately large arrays
- Requires small amount of code
- Requires no extra memory

References

[1] *Algorithms, Fourth Edition;* Robert Sedgewick and Kevin Wayne (and associated slides)