Shell Sort (invention of computer scientist Donald Shell)

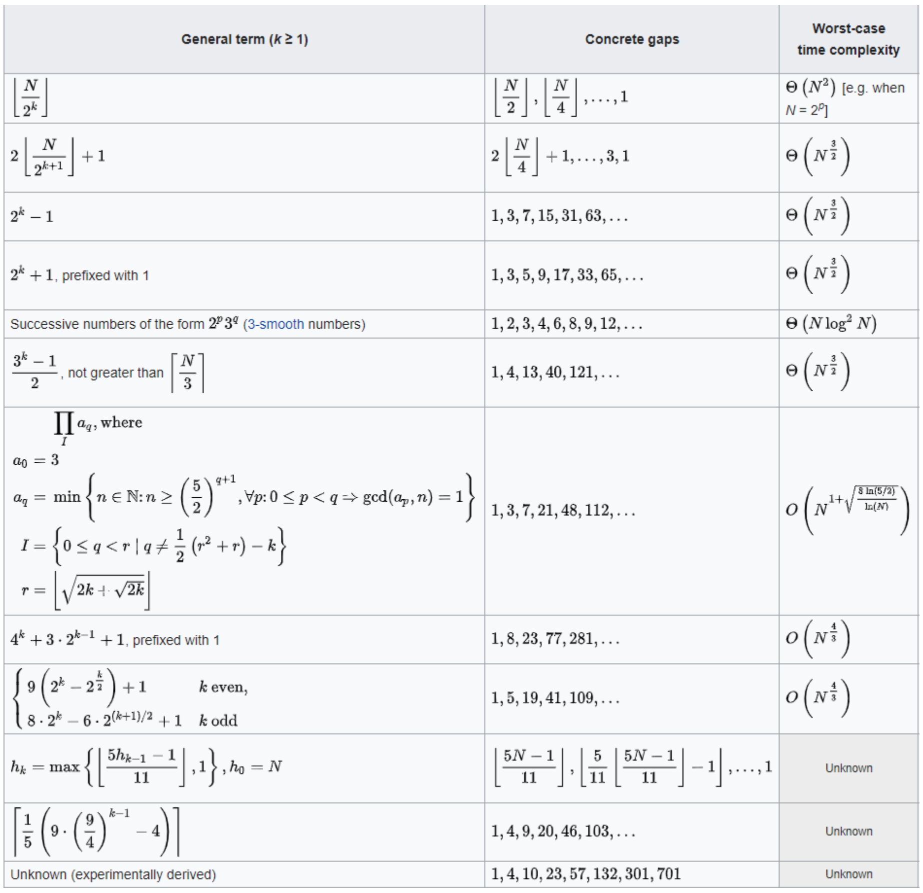
- it's a variation of Insert Sort
- insertion sort choose which element to insert using gap of 1
- Shell Sort start out using a larger gap value
- As the algorithm runs the gap is reduced
- Goal is to reduce the amount of shifting required
- The last gap value is always 1
- Gap value of 1 is equivalent to insertion sort
- It's unstable algorithm

Algorithm traverses the array with certain gap value and after it's done its first traversal with the initial gap value, it decreases the gap and does it again until the gap value is 1

- It is a preliminary work to presort the array so the part of insertion sort (gap is 1) will go faster (less shifting require)

There are several ways to pick gap number with different time complexity

As we are reading from Wikipedia:



General term ($k \geq 1$)	Concrete gaps	Worst-case time complexity
$\left\lfloor \frac{N}{2^k} \right\rfloor$	$\left\lfloor \frac{N}{2} \right\rfloor, \left\lfloor \frac{N}{4} \right\rfloor, \ldots, 1$	$\Theta\left(N^2\right)$ [e.g. when $N = 2^p$]
$2\left\lfloor \frac{N}{2^{k+1}} \right\rfloor + 1$	$2\left\lfloor \frac{N}{4} \right\rfloor + 1, \ldots, 3, 1$	$\Theta\left(N^{\frac{3}{2}}\right)$
$2^k - 1$	$1, 3, 7, 15, 31, 63, \ldots$	$\Theta\left(N^{\frac{3}{2}}\right)$
$2^k + 1$, prefixed with 1	$1, 3, 5, 9, 17, 33, 65, \ldots$	$\Theta\left(N^{\frac{3}{2}}\right)$
Successive numbers of the form $2^p 3^q$ (3-smooth numbers)	$1, 2, 3, 4, 6, 8, 9, 12, \ldots$	$\Theta\left(N \log^2 N\right)$
$\frac{3^k - 1}{2}$, not greater than $\left\lceil \frac{N}{3} \right\rceil$	$1, 4, 13, 40, 121, \ldots$	$\Theta\left(N^{\frac{3}{2}}\right)$
$\prod_I a_q$, where $a_0 = 3$, $a_q = \min\left\{n \in \mathbb{N} \colon n \geq \left(\frac{5}{2}\right)^{q+1}, \forall p \colon 0 \leq p < q \Rightarrow \gcd(a_p, n) = 1\right\}$, $I = \left\{0 \leq q < r \mid q \neq \frac{1}{2}\left(r^2 + r\right) - k\right\}$, $r = \left\lfloor \sqrt{2k + \sqrt{2k}} \right\rfloor$	$1, 3, 7, 21, 48, 112, \ldots$	$O\left(N^{1+\sqrt{\frac{8 \ln(5/2)}{\ln(N)}}}\right)$
$4^k + 3 \cdot 2^{k-1} + 1$, prefixed with 1	$1, 8, 23, 77, 281, \ldots$	$O\left(N^{\frac{4}{3}}\right)$
$\begin{cases} 9\left(2^k - 2^{\frac{k}{2}}\right) + 1 & k \text{ even}, \\ 8 \cdot 2^k - 6 \cdot 2^{(k+1)/2} + 1 & k \text{ odd} \end{cases}$	$1, 5, 19, 41, 109, \ldots$	$O\left(N^{\frac{4}{3}}\right)$
$h_k = \max\left\{\left\lfloor \frac{5h_{k-1} - 1}{11} \right\rfloor, 1\right\}, h_0 = N$	$\left\lfloor \frac{5N - 1}{11} \right\rfloor, \left\lfloor \frac{5}{11}\left\lfloor \frac{5N - 1}{11} \right\rfloor - 1 \right\rfloor, \ldots, 1$	Unknown
$\left\lceil \frac{1}{5}\left(9 \cdot \left(\frac{9}{4}\right)^{k-1} - 4\right) \right\rceil$	$1, 4, 9, 20, 46, 103, \ldots$	Unknown
Unknown (experimentally derived)	$1, 4, 10, 23, 57, 132, 301, 701$	Unknown

One common gap value is the Knuth sequence:

- gap is calculated using $(3^k – 1) / 2$
- we set k based on the length of the array
- we want the gap to be as close as possible to the length of the array we want to sort without being greater than the length

k	Gap(interval)
1	1
2	4
3	13
4	40
5	121

If the array have 20 elements the closest k value that is less then array length following the pattern would be 3

- k = 3 gives interval of 13
- k = 4 gives interval of 40 so that’s over the 20 elements array length

Gap value based on array’s length:

- gap will be initialized to array.length / 2
- on each iteration, we’ll divide the gap value by 2 to get next gap value

🟩 - comparing

🟨- shifting

_ -inserting position

nextElement – next value we want to insert

shft – shift

ins – insert

EOA – end of array

Dn – do nothing

i – starts from the gap, then increment as usually till the end of array

we compare elements of [i] and [i – gap]

Step	20	35	-15	7	55	1	-22	Gap	Elem.	Description	
0	20	35	-15	7	55	1	-22	3	7	7 < 20	shft
1	20	35	-15	20	55	1	-22	3	7	EOA	ins
2	7	35	-15	20	55	1	-22	3	55	55 > 35	Dn
3	7	35	-15	20	55	1	-22	3	1	1 > -15	Dn
4	7	35	-15	20	55	1	-22	3	-22	-22 < 20	shft
5	7	35	-15	20	55	1	20	3	-22	-22 < 7	shft
6	7	35	-15	7	55	1	20	3	-22	EOA	ins
7	-22	35	-15	7	55	1	20	3/2 = 1	35	35 > -22	dn
8	-22	35	-15	7	55	1	20	1	-15	-15 < 35	shft
9	-22	35	35	7	55	1	20	1	-15	-15 > -22	ins
10	-22	-15	35	7	55	1	20	1	7	7 < 35	Shft
11	-22	-15	35	35	55	1	20	1	7	7 > -15	ins
12	-22	-15	7	35	55	1	20	1	55	55 > 35	ins
13	-22	-15	7	35	55	1	20	1	1	1 < 55	shft
14	-22	-15	7	35	55	55	20	1	1	1 < 35	shft
15	-22	-15	7	35	35	55	20	1	1	1 < 35	Shft
16	-22	-15	7	7	35	55	20	1	1	1 > -15	ins
17	-22	-15	1	7	35	55	20	1	20	20 < 55	Shft
18	-22	-15	1	7	35	55	55	1	20	20 < 35	Shft
19	-22	-15	1	7	35	35	55	1	20	20 > 7	ins
20	-22	-15	1	7	20	35	55	SORTED			

This algorithm can be implemented into bubble sort ! (same idea, using gap, instead of shifting we swap elements)