**TITLE**

SUBTITLE

Text

**SHELL SORT ALGORITHM**

Brief History

Shel Sort Algorithm was invented by Donald Shell in 1959. He acquired his Ph.D. in Mathematics from the University of Cincinnati in 1959, after publishing the shell sort algorithm in the Communications of the ACM in July the same year.

Gap Sequence

* In Shell Sort, the sorting process is done in intervals.
* The interval between the elements in each pass is reduced based on the **gap sequence** used.
* Choosing the optimal gap sequence can help in improving the efficiency of the process.

Time and Space Complexity of Shell Sort

|  |  |
| --- | --- |
| **Time Complexity (Best)** | O(n log n) |
| **Time Complexity (Average)** | Depends on gap sequence used |
| **Time Complexity (Worst)** | O(n(4/3)) |
| **Space Complexity** | O(1) |

Shell Sort

Shell sort is a generalized version of the insertion sort algorithm. It first sorts elements that are far apart from each other and successively reduces the interval between the elements to be sorted. It is essentially several stages of Insertion Sort designed with the purpose of speeding up to the said algorithm itself.

Pros

* Shell Sort runs faster than an insertion sort
* It improves upon bubble sort and insertion sort by moving out of order elements more than one position at a time.
* Shell Sort is efficient for medium size list of elements.

Cons

* Shell sort algorithm is not stable method, as its performance depends on the choice of increment sequence

**Variations**

Shell Sequence

void shellSort(int array[], int n) {

// Rearrange elements at each n/2, n/4, n/8, ... intervals

for (int interval = n / 2; interval > 0; interval /= 2) {

for (int i = interval; i < n; i += 1) {

int temp = array[i];

int j;

for (j = i; j >= interval && array[j - interval] > temp; j -= interval) {

array[j] = array[j - interval];

}

array[j] = temp;

}

}

}

* Uses a gap sequence of (N / 2^p), where p is the number of passes of sorting.
* Worst case scenario occurs for N equal to a power of two when elements greater and smaller than the median occupy odd and even positions respectively.

Knuth’s Interval

void shellSortv2(int array[], int n) {

int interval, i, j, temp;

for(interval = 1; interval < n; interval = 3 \* interval + 1);

// 1, 4, 13, 40, 121, 364, 1093, ...

for (; interval > 0; interval = interval / 3) {

for (i = interval; i < n; i++) {

temp = array[i];

for (j = i; j >= interval && array[j - interval] > temp; j -= interval) {

array[j] = array[j - interval];

}

array[j] = temp;

}

}

}

* Uses a gap sequence of 1, 4, 13, 40, 121, ..., 3h(sub(p-1)) + 1, where h(sub(p-1)) is the previous element of the sequence.
* Recommended for N < 1000. (Knuth, 1998)