
# Design and Analysis of Algorithms

## ShellSort:Advanced Sorting Algorithms

**Astana IT University** | **Course:** Design and Analysis of Algorithms

Topic

Shell Sort (Student A)

Student

Ildar Savzikhanov

Instructor

Sayakulova Zarina

Semester 3

# The Shell Sort Algorithm: Core Principles

## Improved Insertion Sort

ShellSort is anoptimizationof the standard Insertion Sort, allowing the exchange of items that are far apart.

## Gapped Comparisons

Itsortselements usingasequence of decreasing gaps (or intervals)between compared elements, progressively reducing the distance.

## Final Pass

Theprocess concludes with a standard InsertionSort(gap = 1), ensuring the array is fully sorted.

## Algorithmic Complexity

The efficiency of Shell Sort heavily depends on the chosen gap sequence. While the worst-case complexity can be high, the practical performance is generally excellent.

- **Average Time Complexity:** O(n $\log^2$ n)
- **Worst-Case (varies):** O($n^2$)
- **Space Complexity:** O(1) 4 an in-place sorting algorithm

# Gap Sequences and Implementation Details

ThecorechallengeinShellSortisselecting an optimalgapsequencetominimizethenumber ofcomparisons andswaps.

| | | |
|---|---|---|
| Shell | n/2, n/4, ..., 1 | The original, simplest approach. Basic but inefficient for some inputs. |
| Knuth | 1, 4, 13, 40, ... (3h + 1) | Significantly faster in practice and widely adopted as a standard improvement. |
| Sedgewick | 1, 5, 19, 41, 109, ... | Derived from mathematical analysis, offering the best empirical results. |

### ShellSor t.java

Core algorithm implementation with support for dynamic gap sequences.

### Per formanceTracker.java

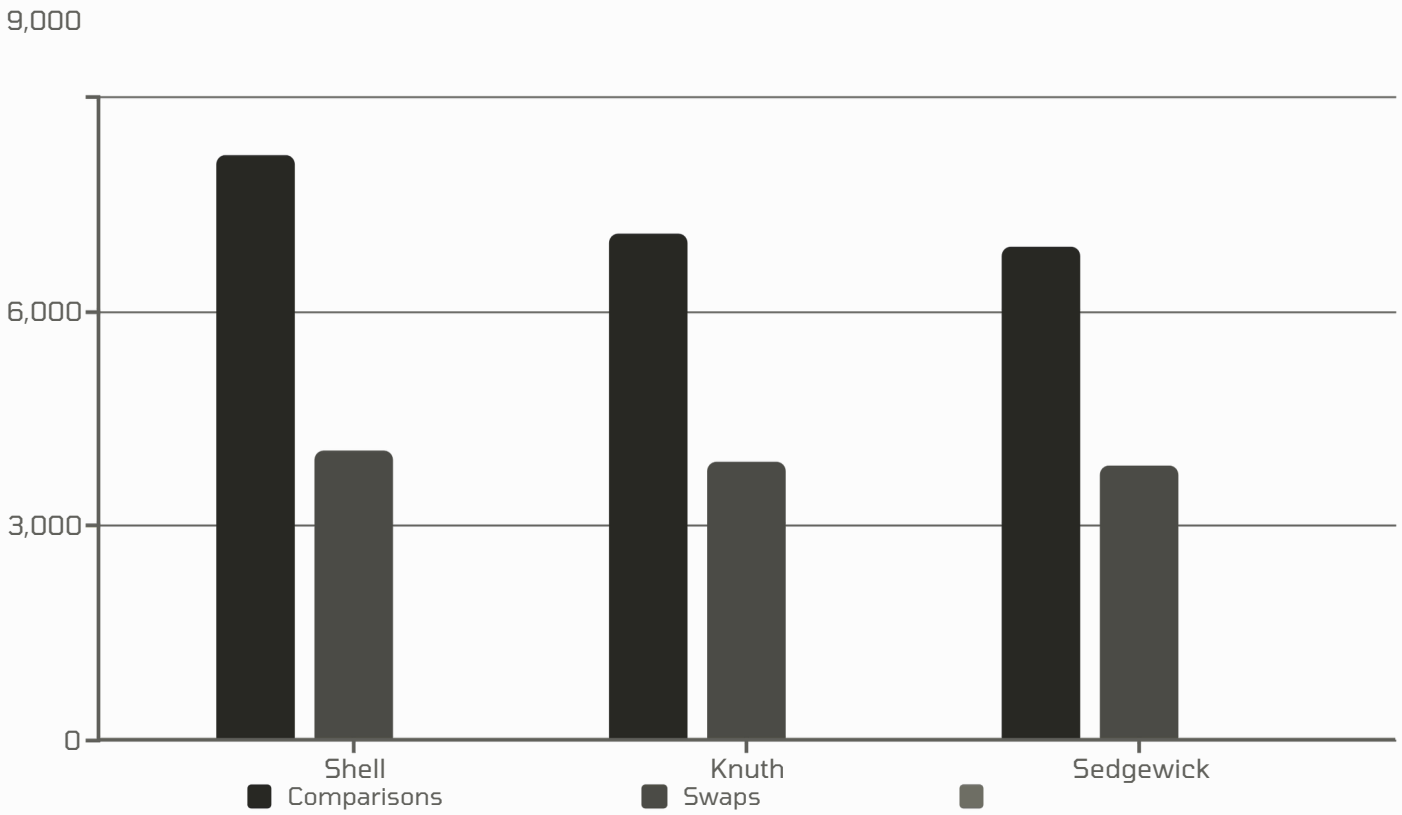
Module for precisely capturing execution metrics (comparisons, swaps, time).

### BenchmarkRunner.java

Command-line interface (CLI) driver for running tests and generating CSV output.

# Benchmarking Results: Comparing Gap Sequences

We conducted experimental analysis on a data set of $n=1000$ elements to measure the efficiency of the three gap sequences.




## Key Observation

The **Sedgewick sequence** consistently yielded the lowest count for comparisons and swaps, resulting in the fastest execution time for the tested array size.

# Optimization Strategy and Conclusion

### Unified Metrics

Implementedanacc(k) method to ensure consistent and unified tracking of performance metrics across all test runs.

### Default Knuth Gaps

Setthe proven efficientKnuth sequence as the default choice for general sorting applications within the implemented class.

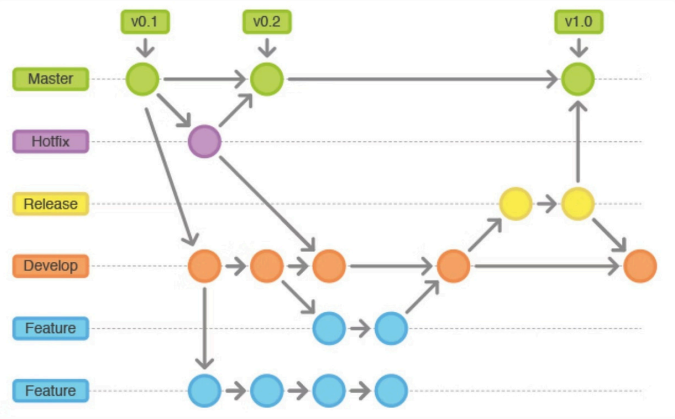
### Refactor for Clarity

Thecodebasewasrefactored to improve readability, modularity, and maintainability, adhering to best practices.

## Development Workflow

- Feature branches were used for developing and testing each gap sequence independently.
- All changes were rigorously tested before being merged into the main branch, culminating in the v1.0 release.

**Source Code Repository: github.com/1B0-d/Assigment2_daa**




## Final Conclusion

Shell Sort is a simple yet powerful sorting algorithm, particularly efficient for medium-sized arrays. Utilizing the mathematically derived **Sedgewick sequence** provides the optimal performance profile.