CompSci 260P Project #1 — Shellsort

Shellsort, the famous, classic sorting algorithm, is really a family of different sorting algorithms, depending on the gap sequence that it uses. For this project, you need to implement five (5) different versions of the Shellsort algorithm. Specifically, you need to implement Shellsort with the following five (5) different gap sequences:

1. The original Shell sequence, $n/2^k$, for k=1,2,...,log n.
2. The sequence, 2^k-1, for k=log n, ..., 7, 3, 1.
3. The Pratt sequence, 2^p3^q, ordered from the largest such number less than n down to 1.
4. The A036562 sequence, in reverse order, starting from the largest value less than n, down to 1.
5. A sequence different than any of these, which you invented.

Please test out multiple possibilities for your invented sequence, #5, with the goal of making it the best of the five.

Following up on this, you need to do an experimental time analysis by implementing each of the five above versions of Shellsort, as n grows, with multiple runs for random permutations for each problem size. That is, you need to perform an empirical comparative analysis by running these five implementations on a variety of test cases to experimentally determine the relative efficiencies of these algorithms in the worst and average cases as a function of n.

For each implementation, you need to perform runtime experiments on random permutations, with multiple runs for each problem size, for increasing problem sizes. Specifically, you need to do a set of experiments for each of the following distributions:

- **Uniformly distributed permutations**, where all permutations are equally likely.
- **Almost-sorted permutations**. These are generated by starting with a sorted array/vector of n numbers, say, the numbers 1,2,3,...,n, in this order. Then, independently choose 2log n pairs, (i,j), where i and j are uniformly-chosen random integers in the range from 0 to n-1, and swap the numbers at positions i and j in the array/vector.

You must plot the results on a log-log scale (with uniformly distrubted permuations on one plot and almost-sorted permutations on another) to empirically determine the algorithm's asymptotic running time for each type of distribution. You then need to rank all your sorting implementations from asymptotically slowest to fastest based on your experimental results.

Deliverables (to be submitted via EEE dropbox) include the following:

- A ZIP file that includes the following:
  - Source code (C, C++, Java, or Python) of your program that implements and experimentally tests the running times of the five versions of the Shellsort algorithm
- A Report (in PDF) that includes the following:
  - Comparative experimental analysis and conclusions drawn from the experiments ranking the quality of the gap sequences you tested. This portion of the report should include a pseudo-code summary of your algorithm, design choices implemented in the source code (such as data structures and representations), and visual plots showing comparitive empirical performance of the different versions of the Shellsort algorithm. The plots should be on a log-log scale and you should use the slope of a best line fit to determine the exponent of the polynomial for a best estimate of the running time in terms of n for each version.