D0012E Laboration 1

2024-11-27

Laboranter:
Arvid Persson[1]
Emma Stiger[2]
Oscar Lundborg[3]

[1] arvper-3@student.ltu.se
[2] emasit-2@student.ltu.se
[3] osclun-3@student.ltu.se

1. Specification

Given an empty stack A containing n elements, and an auxiliary stack B, initially empty, put the elements of A in sorted (ascending) order on one stack.

Allowed stack operations are **push**, **pop** and **is-empty**.

The choice was made to store the elements in B. The stability of the sorting was not considered, and so cases where two elements are equal are not considered in this analysis.

2. Overview

- Until A is empty, do the following:
  - Pop an element x from A.
  - Until B is empty, or $x < y$, where y is the top value of B, do the following[4]:
    - Pop y from B and push it onto A.
  - Push x onto B.

3. Observations

- During each iteration of over A, the stack shrinks if and only if $y > x$ (as no value is pushed back onto A in this case). Recall that y was the value of x in the previous iteration. This means that A shrinks at its fastest in the case that each value on the stack is smaller than the previous, i.e. a stack sorted in descending order.
  - In the best case, all elements are moved once— sorting is performed in linear time.
  - The opposite is also true; A shrinks at its slowest in the case that the stack was already sorted in ascending order. As such, in the worst case, each element is processed once for every subsequent element— quadratic time.

4. Complexity

- Each of the n elements, it will have to go through all $n - 1$ other elements. Thus, the number of comparions is

$$n(n-1) = n^2 - n.$$

- B will be popped from once for each comparison. Additionally, A will be popped from a total of n times before A shrinks (and this repeats). Thus, the number of pops is

$$n(n-1) + \sum_{k=1}^{n} k = n^2 - n + \frac{n^2 + n}{2} = \frac{3n^2 - n}{2}.$$

  Since the number of elements present (including elements temporarily stored outside of either stack) is constant, we have the same number of pushes.

This confirms the quadratic time complexity.

[4] As **peek** operations are not permitted, this includes popping y from B and possibly then pushing it back onto B in the case that $x < y$.

5. Experimentation

The decision was made for experiments to be run on data sets of sizes 10^k. As $k = 4$ turned out to be the largest value that would finish in any reasonable amount of time, a stack of 50000 was also tried. For each size, five stacks were generated and sorted:

- One sorted in ascending order (the worst case).
- One stack sorted in descending order (the best case).
- One sorted in ascending order, but with a random selection of elements out of place.
- One sorted in descending order, but with a random selection of elements out of place.
- One entirely randomized.

Below are the results from a single run of the tests. Time is measured in seconds. For more accurate results, several runs given the same conditions should have been averaged.

Size	Sorted (asc.)	Semi-sorted (asc.)	Sorted (desc.)	Semi-sorted (desc.)	Random
1	$4.53 \cdot 10^{-6}$	$3.34 \cdot 10^{-6}$	$2.15 \cdot 10^{-6}$	$1.91 \cdot 10^{-6}$	$1.43 \cdot 10^{-6}$
10	$9.11 \cdot 10^{-5}$	$8.77 \cdot 10^{-5}$	$1.22 \cdot 10^{-5}$	$1.36 \cdot 10^{-5}$	$4.91 \cdot 10^{-5}$
100	$8.92 \cdot 10^{-3}$	$8.56 \cdot 10^{-3}$	$1.19 \cdot 10^{-4}$	$1.27 \cdot 10^{-4}$	$3.94 \cdot 10^{-3}$
1000	0.722	0.713	$9.96 \cdot 10^{-4}$	$1.06 \cdot 10^{-3}$	0.353
10000	74.9	73.6	$1.09 \cdot 10^{-2}$	$1.19 \cdot 10^{-2}$	39.1
50000	1875.8	1837.15	0.102321	0.172322	880.92

As expected, the fully sorted cases are the extremes with the semi-sorted not far behind. The best cases is several orders of magnitude faster than the worst case for larger n. The fully random version maintains a steady average of the two. Furthermore, each order of magnitude increases the running time by about two orders of magnitude.