WIKIPEDIA

Merge-insertion sort

In computer science, **merge-insertion sort** or the **Ford–Johnson algorithm** is a comparison sorting algorithm published in 1959 by L. R. Ford Jr. and Selmer M. Johnson.[1][2][3][4] It uses fewer comparisons in the worst case than the best previously known algorithms, binary insertion sort and merge sort,[1] and for 20 years it was the sorting algorithm with the fewest known comparisons.[5] Although not of practical significance, it remains of theoretical interest in connection with the problem of sorting with a minimum number of comparisons.[3] The same algorithm may have also been independently discovered by Stanisław Trybuła and Czen Ping.[4]

Contents

Algorithm

Merge-insertion sort performs the following steps, on an input X of n elements:[6]

1. Group the elements of X into $\lfloor n/2 \rfloor$ pairs of elements, arbitrarily, leaving one element unpaired if there is an odd number of elements.
2. Perform $\lfloor n/2 \rfloor$ comparisons, one per pair, to determine the larger of the two elements in each pair.
3. Recursively sort the $\lfloor n/2 \rfloor$ larger elements from each pair, creating a sorted sequence S of $\lfloor n/2 \rfloor$ of the input elements, in ascending order.
4. Insert at the start of S the element that was paired with the first and smallest element of S.
5. Insert the remaining $\lceil n/2 \rceil - 1$ elements of $X \setminus S$ into S, one at a time, with a specially chosen insertion ordering described below. Use binary search in subsequences of S (as described below) to determine the position at which each element should be inserted.

The algorithm is designed to take advantage of the fact that the binary searches used to insert elements into S are most efficient (from the point of view of worst case analysis) when the length of the subsequence that is searched is one less than a power of two. This is because, for those lengths, all outcomes of the search use the same number of comparisons as each other.[1] To choose an insertion ordering that produces these lengths, consider the sorted sequence S after step 4 of the outline above (before inserting the remaining elements), and let x_i denote the ith element of this sorted sequence. Thus,

$$S = (x_1, x_2, x_3, \dots),$$

where each element x_i with $i \geq 3$ is paired with an element $y_i < x_i$ that has not yet been inserted. (There are no elements y_1 or y_2 because x_1 and x_2 were paired with each other.) If n is odd, the remaining unpaired element should also be numbered as y_i with i larger than the indexes of the paired elements. Then, the final step of the outline above can be expanded into the following steps:[1][2][3][4]

- Partition the uninserted elements y_i into groups with contiguous indexes. There are two elements y_3 and y_4 in the first group, and the sums of sizes of every two adjacent groups form a sequence of powers of two. Thus, the sizes of groups are: 2, 2, 6, 10, 22, 42, ...
- Order the uninserted elements by their groups (smaller indexes to larger indexes), but within each group order them from larger indexes to smaller indexes. Thus, the ordering becomes

$$y_4, y_3, y_6, y_5, y_{12}, y_{11}, y_{10}, y_9, y_8, y_7, y_{22}, y_{21} \cdots$$

- Use this ordering to insert the elements y_i into S. For each element y_i, use a binary search from the start of S up to but not including x_i to determine where to insert y_i.

Analysis

Let $C(n)$ denote the number of comparisons that merge-insertion sort makes, in the worst case, when sorting n elements. This number of comparisons can be broken down as the sum of three terms:

- $\lfloor n/2 \rfloor$ comparisons among the pairs of items,
- $C(\lfloor n/2 \rfloor)$ comparisons for the recursive call, and
- some number of comparisons for the binary insertions used to insert the remaining elements.

In the third term, the worst-case number of comparisons for the elements in the first group is two, because each is inserted into a subsequence of S of length at most three. First, y_4 is inserted into the three-element subsequence (x_1, x_2, x_3). Then, y_3 is inserted into some permutation of the three-element subsequence (x_1, x_2, y_4), or in some cases into the two-element subsequence (x_1, x_2). Similarly, the elements y_6 and y_5 of the second group are each inserted into a subsequence of length at most seven, using three comparisons. More generally, the worst-case number of comparisons for the elements in the i th group is $i + 1$, because each is inserted into a subsequence of length at most $2^{i+1} - 1$.[1][2][3][4] By summing the number of comparisons used for all the elements and solving the resulting recurrence relation, this analysis can be used to compute the values of $C(n)$, giving the formula[7]

$$C(n) = \sum_{i=1}^{n} \left\lceil \log_2 \frac{3i}{4} \right\rceil \approx n \log_2 n - 1.415n$$

or, in closed form,[8]

$$C(n) = n \left\lceil \log_2 \frac{3n}{4} \right\rceil - \left\lfloor \frac{2^{\lfloor \log_2 6n \rfloor}}{3} \right\rfloor + \left\lfloor \frac{\log_2 6n}{2} \right\rfloor.$$

For $n = 1, 2, \ldots$ the numbers of comparisons are[1]

0, 1, 3, 5, 7, 10, 13, 16, 19, 22, 26, 30, 34, ... (sequence A001768 in the OEIS)

Relation to other comparison sorts

The algorithm is called merge-insertion sort because the initial comparisons that it performs before its recursive call (pairing up arbitrary items and comparing each pair) are the same as the initial comparisons of merge sort, while the comparisons that it performs after the recursive call (using binary search to insert elements one by one into a sorted list) follow the same principle as insertion sort. In this sense, it is a hybrid algorithm that combines both merge sort and insertion sort.[9]

For small inputs (up to $n = 11$) its numbers of comparisons equal the lower bound on comparison sorting of $\lceil \log_2 n! \rceil \approx n \log_2 n - 1.443n$. However, for larger inputs the number of comparisons made by the merge-insertion algorithm is bigger than this lower bound. Merge-insertion sort also performs fewer comparisons than the sorting numbers, which count the comparisons made by binary insertion sort or merge sort in the worst case. The sorting numbers fluctuate between $n \log_2 n - 0.915n$ and $n \log_2 n - n$, with the same leading term but a worse constant factor in the lower-order linear term.[1]

Merge-insertion sort is the sorting algorithm with the minimum possible comparisons for n items whenever $n \leq 15$ or $20 \leq n \leq 22$, and it has the fewest comparisons known for $n \leq 46$.[10][11] For 20 years, merge-insertion sort was the sorting algorithm with the fewest comparisons known for all input lengths. However, in 1979 Glenn Manacher published another sorting algorithm that used even fewer comparisons, for large enough inputs.[3][5] It remains unknown exactly how many comparisons are needed for sorting, for all n, but Manacher's algorithm and later record-breaking sorting algorithms have all used modifications of the merge-insertion sort ideas.[3]

References

1. Ford, Lester R. Jr.; Johnson, Selmer M. (1959), "A tournament problem", *American Mathematical Monthly*, **66**: 387–389, doi:10.2307/2308750 (https://doi.org/10.2307%2F2308750), MR 0103159 (https://www.ams.org/mathscinet-getitem?mr=0103159)
2. Williamson, Stanley Gill (2002), "2.31 Merge insertion (Ford–Johnson)" (https://books.google.com/books?id=YMIoy5JwdHMC&pg=PA66), *Combinatorics for Computer Science*, Dover books on mathematics, Courier Corporation, pp. 66–68, ISBN 9780486420769
3. Mahmoud, Hosam M. (2011), "12.3.1 The Ford–Johnson algorithm" (https://books.google.com/books?id=kM5v2YqMVuoC&pg=PA286), *Sorting: A Distribution Theory*, Wiley Series in Discrete Mathematics and Optimization, **54**, John Wiley & Sons, pp. 286–288, ISBN 9781118031131
4. Knuth, Donald E. (1998), "Merge insertion", *The Art of Computer Programming, Vol. 3: Sorting and Searching* (2nd ed.), pp. 184–186
5. Manacher, Glenn K. (July 1979), "The Ford-Johnson Sorting Algorithm Is Not Optimal", *Journal of the ACM*, **26** (3): 441–456, doi:10.1145/322139.322145 (https://doi.org/10.1145%2F322139.322145)
6. The original description by Ford & Johnson (1959) sorted the elements in descending order. The steps listed here reverse the output, following the description in Knuth (1998). The resulting algorithm makes the same comparisons but produces ascending order instead.
7. Knuth (1998) credits the summation formula to the 1960 Ph.D. thesis of A. Hadian. The approximation formula was already given by Ford & Johnson (1959).
8. Guy, Richard K.; Nowakowski, Richard J. (December 1995), "*Monthly* Unsolved Problems, 1969-1995", *American Mathematical Monthly*, **102** (10): 921–926, doi:10.2307/2975272 (https://doi.org/10.2307%2F2975272)

9. Knuth (1998), p. 184: "Since it involves some aspects of merging and some aspects of insertion, we call it *merge insertion*."
10. Peczarski, Marcin (2004), "New results in minimum-comparison sorting", *Algorithmica*, **40** (2): 133–145, doi:10.1007/s00453-004-1100-7 (https://doi.org/10.1007%2Fs00453-004-1100-7), MR 2072769 (https://www.ams.org/mathscinet-getitem?mr=2072769)
11. Peczarski, Marcin (2007), "The Ford-Johnson algorithm still unbeaten for less than 47 elements", *Information Processing Letters*, **101** (3): 126–128, doi:10.1016/j.ipl.2006.09.001 (https://doi.org/10.1016%2Fj.ipl.2006.09.001), MR 2287331 (https://www.ams.org/mathscinet-getitem?mr=2287331)