

ONE-PASS ALGORITHM

Nicole Schweikardt

Institute for Computer Science

Johann Wolfgang Goethe-Universität Frankfurt am Main

Robert-Mayer-Str. 11–15, D-60325 Frankfurt am Main, Germany

schweika@informatik.uni-frankfurt.de

SYNONYMS

1-pass algorithm; streaming algorithm; data stream algorithm

DEFINITION

A one-pass algorithm receives as input a list of data items x_1, x_2, x_3, \dots . It can read these data items only once, from left to right, i.e., in increasing order of the indices $i = 1, 2, 3, \dots$. Critical parameters of a one-pass algorithm are (1) the size of the memory used by the algorithm, and (2) the processing time per data item x_i . Typically, a one-pass algorithm is designed for answering one particular query against the input data. To this end, the algorithm stores and maintains a suitable data structure which, for each i , is updated when reading data item x_i . The two parameters *processing time per data item* and *memory size* are usually measured as functions depending on the size N of the input (different measures of the *input size* are considered in the literature, among them, e.g., the number of data items occurring in the input, as well as the total number of bits needed for storing the entire input). The ultimate goal when designing a one-pass algorithm is to keep the processing time per data item and the memory size *sublinear*, preferably polylogarithmic, in N . In particular, one typically aims at algorithms whose memory size is far smaller than the size of the input.

MAIN TEXT

The design and study of one-pass algorithms has a long tradition in many areas of computer science. For example, they are used in the area of *data stream processing*, where streams of huge amounts of data have to be monitored *on-the-fly* without prior storing the entire data. But also, e.g., a deterministic finite automaton on words can be viewed as a (very simple) example of a one-pass algorithm whose memory size and processing time per data item is constant, i.e., does not depend on the input size. For most computational problems, however, the amount of memory necessary for solving the problem grows with increasing input size. *Lower bounds* on the memory size needed for solving a problem by a one-pass algorithm are usually obtained by applying methods from *communication complexity* (see, e.g., [1, 2] for typical examples of according lower bounds).

For many concrete problems it is even known that the memory needed for solving the problem by a deterministic one-pass algorithm is at least linear in the size N of the input. For some of these problems, however, *randomized* one-pass algorithms can still compute good *approximate* answers while using memory of size sublinear in N (cf., [1, 2, 3]). Typically, such algorithms are based on *sampling*, i.e., only a “representative” portion of the data is taken into account, and *random projections*, i.e., only a rough “sketch” of the data is stored in memory (see [3] for a comprehensive survey of according algorithmic techniques).

In the context of database systems these techniques are relevant, for example, for maintaining information needed for cost based query optimization, e.g., estimates for the number of distinct values of an attribute, or the self-join size of a database relation. Efficient one-pass algorithms for incrementally updating these estimates can be found in [1].

In some application areas, rather than just a single pass, a small number P of sequential passes over the data may be available; the resulting algorithms are called *multi-pass algorithms* (see e.g. [2] for an analysis of the trade-off between the memory size and the number of passes necessary for solving particular problems).

CROSS REFERENCE

REFERENCES

- [1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58:137–147, 1999.
- [2] M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. In *External memory algorithms*, volume 50, pages 107–118. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1999.
- [3] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.