LEMPLE–ZIV FACTORIZATION

Lempel-Ziv factorization $LZ(T)$ of string $T$ is a greedy partition of $T$ into longest previous factors (LPFs). LPF at position $i$ is the longest factor $T[i..i+\ell]$ that also occurs at some position $j < i$.

Example:

$$T = \text{ABABBABBAB}$$

$$LPF[2] = AB \ (j = 0)$$

$$LPF[5] = ABBAB \ (j = 2)$$

$$LZ(T) = \text{ABBABABBAB}$$

EXPERIMENTAL COMPARISON

We implemented and compared all LZ factorization algorithms depicted on the right. The algorithms were executed on varying size prefixes of two test files: a large data set containing English text (left) and a database of Wikipedia articles containing many versions of the same articles (right). All algorithms were allowed to use 3.5GiB of internal memory. The results depend on the amount of repetitions in the input test. LZscan dominates all other algorithms for highly repetitive input but performs poorly when the data is less repetitive, such as the English text file. The fastest algorithm for such data is determined by the ratio of input size to available RAM.

DETAILED RUNTIME BREAKDOWN

Below we present a detailed runtime breakdown of LPF-eSAISlcp and LPF-SAscan executed on English text. The graphs reveal that most of the time is spent during the computation of supporting data structures (SA and LCP). The LCP array construction is significantly accelerated with the use of our new algorithm (LCPscan) which makes SA construction the slowest phase of the factorization. The main challenge in efficient and scalable LZ factorization is therefore developing new methods for suffix sorting, possibly using parallel or distributed computation.

REFERENCES


