HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI
MATEMAATTIS-LUONNONTIETEELLINEN TIEDEKUNTA
MATEMATISK-NATURVETENSKAPLIGA FAKULTETEN
FACULTY OF SCIENCE

String Sorting in Python – Comparison of Several Algorithms

Onni Koskinen, Arturs Polis, and Lari Rasku

Comparison-based sorting is one of the most mature subfields of CS research. However, the more well-known of such algorithms have been designed with the expectation that the objects they sort can be compared in constant time. When used to sort objects that require linear-time comparison operations, such as strings, they perform a lot of wasteful work that leads to suboptimal performance. For maximum efficiency, *string sorting algorithms* are needed.

We have implemented a family of three different string sorting algorithms in Python and compared their performance against Python's native Timsort using a variety of different datasets.

TEST DATA

The empirical data gathered is intended. A 100MB and a 200MB sample of each dataset was used. The ENGLISH datasets were not used as-is, but with each word split on its own line, in order to make the algorithms sort individual words and not entire lines. The statistics file documents some stringological properties of these datasets.

TEINS, DNA and ENGLISH datasets from the Pizza&Chili Corpus, in addition to a set of URLs from Ranjan Sinha's ref1 data ref2 for his original Burstsort paper.

ref1 https://sites.google.com/site/ranjansinha/home
ref2 http://www.cs.mu.oz.au/ rsinha/resources/data/sc

ALGORITHMS

MSD RADIX SORT

MSD (most significant digit) radix sort is a divide-and-conquer algorithm that partitions the strings based on their character at a given position. The comparison position starts from 0 and increases with one at every recursion level. No position, then, is visited twice; and if the algorithm does not attempt to partition buckets of size 1 or consisting entirely of strings shorter than the recursion depth, each string is visited at most one more time than the length of its shortest distinguishing prefix. Thus, the partitioning takes at most $O(L(R) + n)$ time, where $L(R)$ is the sum of the LCP array.

However, efficient implementations require the buckets to be implemented as an array of linked lists in order to avoid the overhead of binary search tree insertions and lookups. This allows true constant time insertion to buckets, but wastes time and memory if the strings use only a fraction of the alphabet for which MSD radix sort allocates space. Likewise, if the number of strings is smaller than the size of the alphabet, standard comparison based string sorting algorithms outperform MSD radix sort.

Our implementation uses a fixed alphabet size of 256 and falls back to ternary quicksort when the size of the bucket drops below it.

QUICKSORT ALGORITHMS

Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text Quicksort text

BURST SORT

Burst sort text Burst sort text Burst sort text Burst sort text Burst sort text Burst sort text Burst sort text Burst sort text Burst sort text Burst sort text Burst sort text Burst sort text Burst sort text

REFERENCES

[1] U. Lauther and T. Lukovszki. Space efficient algorithms for the Burrows-Wheeler backtransformation. In *Proc. 13th Annual European Symposium on Algorithms*, volume 3669 of *LNCS*, pages 293–304. Springer, 2005.

[2] J. Seward. Space-time tradeoffs in the inverse B-W transform. In *Proc. IEEE Data Compression Conference*, pages 439–448. IEEE, 2001.

[3] U. Lauther and T. Lukovszki. Space efficient algorithms for the Burrows-Wheeler backtransformation. In *Proc. 13th Annual European Symposium on Algorithms*, volume 3669 of *LNCS*, pages 293–304. Springer, 2005.

[4] J. Seward. Space-time tradeoffs in the inverse B-W transform. In *Proc. IEEE Data Compression Conference*, pages 439–448. IEEE, 2001.

[5] U. Lauther and T. Lukovszki. Space efficient algorithms for the Burrows-Wheeler backtransformation. In *Proc. 13th Annual European Symposium on Algorithms*, volume 3669 of *LNCS*, pages 293–304. Springer, 2005.

[6] J. Seward. Space-time tradeoffs in the inverse B-W transform. In *Proc. IEEE Data Compression Conference*, pages 439–448. IEEE, 2001.