





WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Featured content](#)
[Current events](#)
[Random article](#)
[Donate to Wikipedia](#)
[Wikipedia store](#)

Interaction
[Help](#)
[About Wikipedia](#)
[Community portal](#)
[Recent changes](#)
[Contact page](#)

Tools
[What links here](#)
[Related changes](#)
[Upload file](#)
[Special pages](#)
[Permanent link](#)
[Page information](#)
[Wikidata item](#)
[Cite this page](#)


Print/export
[Create a book](#)
[Download as PDF](#)
[Printable version](#)

Languages 
[Deutsch](#)
 [Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)



Samplesort

From Wikipedia, the free encyclopedia



This article **needs attention from an expert in Computer science**. Please add a *reason* or a *talk* parameter to this template to explain the issue with the article. [WikiProject Computer science](#) (or its [Portal](#)) may be able to help recruit an expert. *(April 2009)*

Samplesort is a [sorting algorithm](#) that is a [divide and conquer algorithm](#) often used in parallel processing systems.^[1] Conventional divide and conquer sorting algorithms partitions the array into sub-intervals or [buckets](#). The buckets are then sorted individually and then concatenated together. However, if the array is non-uniformly distributed, the performance of these sorting algorithms can be significantly throttled. Samplesort addresses this issue by selecting a sample of size *s* from the *n*-element sequence, and determining the range of the buckets by sorting the sample and choosing *m* - 1 elements from the result. These elements (called splitters) then divide the sample into *m* equal-sized buckets.^[2] Samplesort is described in the 1970 paper, "Samplesort: A Sampling Approach to Minimal Storage Tree Sorting", by W D Frazer and A C McKellar. In recent years, the algorithm has been adapted to implement [randomized sorting](#) on [parallel computers](#).

Contents [\[hide\]](#)

- 1 Algorithm
 - 1.1 Complexity
- 2 Sampling the Data
 - 2.1 Oversampling
- 3 Selecting the Splitters
- 4 Uses in Parallel Systems
- 5 See also
- 6 References
- 7 External links

Algorithm [\[edit\]](#)

Samplesort can be broken down into 3 parts

- Find splitters, values that break up the data into buckets, by sampling the data.
- Use the sorted splitters to define buckets and place data in appropriate buckets.
- Sort each of the buckets

Complexity [\[edit\]](#)

Find the splitters.

$$O(n/P + \log(P))$$

Send to buckets.

$$\begin{aligned} O(P) & \text{ for reading all node} \\ O(\log(P)) & \text{ for broadcasting} \\ O(n/P * \log(P)) & \text{ for binary search for all keys} \\ O(n/P) & \text{ to send keys to bucket} \end{aligned}$$

Sort Buckets

$$O(c/P) \text{ where } c \text{ is complexity of underlying sequential sorting method}^{[1]}$$

Sampling the Data [\[edit\]](#)

The data may be sampled through different methods. Some methods include:

1. Pick evenly spaced samples.
2. Pick randomly selected samples.

Oversampling [\[edit\]](#)

The oversampling ratio determines how many data elements to pull as samples. The goal is to get a good representation of the distribution of the data. If the data values are widely distributed, in that there are not many duplicate values, then a small sampling ratio is needed. In other cases where there are many duplicates in the distribution, a larger oversampling ratio will be necessary.

Selecting the Splitters [\[edit\]](#)

The ideal is to pick splitters that separate the data into j buckets of size n/j , where n is the number of elements to be sorted. This is to achieve an even distribution among the buckets, this way no one bucket takes longer than others to be sorted. This can be accomplished by selecting splitters in the sample by stepping through the sorted sample using a/j . Where sample size is a , and bucket size is j such that the values are a/j , $2a/j$, ... $(j - 1)a/j$.



Uses in Parallel Systems [\[edit\]](#)

Samplesort is often used in parallel systems. This is done by splitting the sorting for each processor, where the number buckets is equal to the number of processor. Sample sort is efficient in parallel systems because each processor receives approximately the same bucket size. Since the buckets are sorted concurrently, the processors will complete the sorting at approximately the same time, thus not having a processor wait for others.

See also [\[edit\]](#)




- Flashsort
- Quicksort

References [\[edit\]](#)

1. ^{a b} "Samplesort using the Standard Template Adaptive Parallel Library"  (PDF). Texas A&M University.
2. ^a "Bucket and Samplesort" .

External links [\[edit\]](#)

Frazer and McKellar's samplesort and derivatives:

- Frazer and McKellar's original paper 
- <http://www.springerlink.com/content/p70564506802n575/> 
- <http://www.springerlink.com/content/l211p1q526j84174/> 

Adapted for use on parallel computers:

- <http://citeseer.ist.psu.edu/91922.html> 
- <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.214> 

Categories: Sorting algorithms

This page was last modified on 6 August 2015, at 22:56.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

