8th International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010.

Radix Sort for Linked Lists: Implementations

Tibor Ásványi

Faculty of Informatics, Eötvös Loránd Univ.
e-mail: asvanyi@inf.elte.hu

Abstract

We sort a linked list. The items' key is unsigned integer with k digits using the number system base m.

Radix sort uses m temporary lists indexed by the possible values of digits. It sorts the original list in k passes. First, it distributes each item of the list to the temporary list indexed by the least significant digit of the item's key. At the end of the pass, the items are collected from the temporary lists... Last, they are redistributed according to the most significant digit of the key, and collected again. In each pass the items are kept in order.

Therefore, when the first pass is done, the list is ordered according to the least significant digit, when the second pass is done, it is ordered according to the two least significant digits, and so on, when the last pass is done, it is ordered according to the whole key. It is easy to see that the time complexity of radix sort is $\Theta(k*(n+m))$, in practice $\Theta(n)$, where n is the length of the list.

In this paper we develop and compare three different implementations of this algorithm. First we use simple lists (with head and tail pointers). Next we introduce (dummy) sentinel head for the lists and find that we can reduce the code complexity. At last we use a relatively unknown programming technique (invented by the author and Nick Parlante in parallel): indirect pointers[1] [2]. Using this technique, we can reduce the run-time significantly and the code complexity dramatically.

Keywords: linked list, radix sort, indirect pointer, code complexity

MSC:

Tibor Ásványi
Budapest, Pázmány Péter sétány 1/c, H-1117

[1] `http://aszt.inf.elte.hu/~asvanyi/pp/p2p.pdf`
[2] `http://cslibrary.stanford.edu/`