Minimal Mergesort

Tadao TAKAOKA

Department of Computer Science
University of Canterbury
Christchurch, New Zealand

December, 1996

Abstract

We present a new adaptive sorting algorithm, called minimal merge sort, which merges the ascending runs in the input list from shorter to longer, that is, merging the shortest two lists each time. We show that this algorithm is optimal with respect to the new measure of presortedness, called entropy.

Keywords: adaptivesort, minimal mergesort, ascending runs, entropy

1 Introduction

Adaptive sorting is to sort the list of n numbers into increasing order as efficiently as possible by utilizing the structure of the list which reflects some presortedness. See Estivill-Castro, and Wood [1] for a general survey on adaptive sorting. There are many measures of presortedness. The simplest one is the number of ascending runs in the list. Let the given list $X = (a_1, a_2, \cdots, a_n)$ be divided into k ascending runs X_i $(i = 1, \cdots, k)$, that is, $X = (X_1, X_2, \cdots, X_k)$ where $X_i = (a_1^{(i)}, \cdots, a_{n_i}^{(i)})$ and $a_1^{(i)}$ is the $|X_1| + \cdots + |X_{i-1}| + 1$-th element in X. We denote the length of list X by $|X|$. Note that $a_1^{(i)} \leq \cdots \leq a_{n_i}^{(i)}$ for each X_i and $a_{n_i}^{(i)} > a_1^{(i+1)}$ if X_i is not the last list. The sort algorithm called natural merge sort [2] sorts X by merging adjacent two lists for each phase halving the number of ascending runs after each phase so that sorting is completed in $O(n \log k)$ time. Mannila [3] proved that this method is optimal under the measure of the number of ascending runs.

In this paper we generalize the measure $RUNS(X)$ of the number of ascending runs into that of the entropy of ascending runs in X, denotes by $H_{RUNS}(X)$ and sometimes denoted by $H(X)$ for simplicity. Then we invent a sorting algorithm, called minimal merge sort, that sorts X by merging two minimal length runs successively until we have the sorted list. We show that the time for this method is $O(nH(X))$ and is optimal under the measure of $H(X)$. Logarithm is taken with base 2 throughout the paper.

2 Entropy of ascending runs

Let $n_i = |X_i|$ and $p_i = n_i/n$. Note that $\sum p_i = 1$. We define the entropy of ascending runs in X, $H_{RUNS}(X)$ or $H(X)$ for simplicity, by

$$H_{RUNS}(X) = -\sum_{i=1}^{k} p_i \log p_i.$$

Since p_i $(i = 1, \cdots, k)$ can be regarded as a probability measure, we have

$$0 \le H(X) \le \log k$$

and the maximum is obtained when $|X_i| = n/k$ $(i = 1, \cdots, k)$.

LEMMA 2.1 *Any sorting algorithm takes at least $\Omega(nH(X))$ time when the entropy of ascending runs in X is $H(X)$ and $|X_i| \ge 2$ for $i = 1, \cdots, k$.*

Proof. Sorting X into $X' = (a'_1, \cdots, a'_n)$ where $a'_1 \le \cdots \le a'_n$ means that X' is a permutation of X. Let $X_i = (a_1^{(i)}, \cdots, a_{n_i}^{(i)})$. Let $a_1^{(i)}$ $(i = 1, \cdots, k)$ occupy the first k positions in X'. Then there are $\binom{n-k}{n_1-1}$ possibilities of X_1 being scattered in X'. Since we have a constraint of $a_{n_1}^{(1)} > a_1^{(2)}$, we put $a_1^{(2)}$ among the first k positions. Then we have $\binom{n-k-n_1+1}{n_2-1}$ possibilities of X_2 being scattered in X'. Repeating this calculation yields the number of possibilities N as

$$\begin{aligned} N &= \frac{(n-k)!}{(n_1-1)!\,(n-k-n_1+1)!} \cdot \frac{(n-k-n_1+1)!}{(n_2-1)!\,(n-k-n_1-n_2+2)!} \\ &\qquad \cdots \frac{(n_{k-1}-1)!}{(n_k-1)!\,0!} \\ &= \frac{n\,!}{n_1! \cdots n_k!} \cdot \frac{n_1 \cdots n_k}{n(n-1)\cdots(n-k+1)}. \end{aligned}$$

Since the number of possible permutations is not fewer than this, we have the lower bound T on the computing time based on the binary decision tree model approximated by

$$T = \log N \ge n \log n - \sum_{i=1}^{k} n_i \log n_i + \sum_{i=1}^{k} (\log n_i - \log(n-i+1))$$

where we use Stirling's formula. T is evaluated by

$$T \ge \sum_{i=1}^{k} n_i \log \frac{n}{n_i} - k \log n + 2(k-1) + \log(n-2k+2),$$

since $\sum \log n_i$ is minimum when $n_1 = \cdots = n_{k-1} = 2$ and $n_k = n - 2(k-1)$. Now noting that the first term is minimum with the same condition, we have

$$\begin{aligned} 2T - nH &\ge \sum n_i \log \frac{n}{n_i} - 2k \log n + 4(k-1) + 2\log(n-2k+2) \\ &\ge 2(k-1)\log\frac{n}{2} + (n-2k+2)\log\frac{n}{n-2k+2} \\ &\qquad -2k\log n + 4(k-1) + 2\log(n-2k+2) \\ &= (n-2k)\log\frac{n}{n-2k+2} + 2(k-1) \\ &\ge 0, \end{aligned}$$

since $1 \leq k \leq n/2$. Thus we have $T \geq nH(X)/2 = \Omega(nH(X))$. ■

3 Minimal mergesort

All list are maintained in linked list structures in this section. Let $X = (X_1, \cdots, X_k)$ is the given input list such that each X_i is sorted in ascending order. Rearrange X into $X' = (X_{i_1}, \cdots, X_{i_k})$ in such a way that $|X_{i_j}| \leq |X_{i_{j+1}}|$ $(j = 1, \cdots, k-1)$, that is, $(X_1, \cdots, X_k)$ is sorted with $|X_i|$ as key. We call this "meta-sort." Since each $|X_{i_j}|$ is an integer we can obtain X' in $O(n)$ time by radix sort. Now we sort X' by merging two shortest lists repeatedly. Formally we have the following. Let M and L are lists of lists, whereas W_i $(i = 1, 2)$ and W are ordinary lists. By the operation $M \Leftarrow L$, the leftmost list in L is moved to the rightmost part of M. By the operation $W_i \Leftarrow M$ $(i = 1, 2)$ the leftmost list of M is moved to W_i. By the operation $M \Leftarrow W$, W is moved to the rightmost part of M. First (L) is the first list in L.

ALGORITHM 3.1 (Minimal mergesort)

```
1  Meta-sort X into X' by length of X_i;
2  Let L = X';
3  M := ∅;
4  M ⇐ L;
5  if L ≠ ∅ then M ⇐ L;
6  for i := 1 to k − 1 do begin
7     W_1 ⇐ M;
8     W_2 ⇐ M;
9     W := merge (W_1, W_2);
10    while L ≠ ∅ and |W| > |first(L)| do M ⇐ L;
11    M ⇐ W
12 end
   {W is the sorted list}.
```

THEOREM 3.1 *The algorithm minimal mergesort sorts $X = (X_1, \cdots, X_k)$ where each X_i is an ascending sequence in $O(nH(X))$ time where $H(X)$ is the entropy of X.*

Proof. Consider the moment when W_1 and W_2 are merged at line 9. Note that M and L are meta-sorted throughout the computation. From Lemma 3.1 we have $|W_2| \leq \frac{2}{3}|W|$ if W_2 is a merged list. Since $|W_1| \leq |W_2|$, W_1 and W_2 will go to a wider lists at least 3/2 as large if they are not original X_i's. Therefore each element in X_i will go to wider lists at most $\lceil \log_{3/2} n/|X_i| \rceil + 1$ times. Since at each merge $|W_1| + |W_2| - 1$ comparisons are performed, we can charge 1 comparison on each element in X_i if X_i is in one of the merged lists. Thus the total number of comparisons can be bounded by

$$\sum_{i=1}^{k} n_i \log \frac{n}{n_i} = nH(X).$$

■

LEMMA 3.1 *If W_2 is a merged list at line 9, it holds that $|W_2| \leq \frac{2}{3}|W|$.*

Proof. Suppose to the contrary that $|W_2| > 2|W_1|$. Then for the previously merged lists V_1 and V_2, that is, $W_2 = \text{merge}\,(V_1, V_2)$, we have $|V_1| > |W_1|$ or $|V_2| > |W_1|$. Thus V_2 or V_1 must have been merged with W_1 or a shorter list, a contradiction. ■

EXAMPLE. Let $|X_1| = 2$, $|X_i| = 2^{i-1}$ $(i = 2, \cdots, k-1)$ and $n = 2^k$. Then minimal mergesort sorts X in $O(n)$ time, since $H(X) = const$, whereas natural mergesort takes $O(n \log\log n)$ time to sort X.

4 Concluding remarks

When we scan X, we can identify ascending runs and descending runs alternatingly. By reversing descending runs, we can satisfy the condition of $|X_i| \geq 2$ in Lemma 2.1. This modified version of Algorithm 3.1 with this prescanning is thus optimal with respect to the entropy measure.

We showed that the entropy measure covers the measure of $RUNS$. It remains to be seen whether the entropy measure can cover other measures of presortedness.

References

[1] V. ESTIVILL-CASTRO AND D. WOOD, A survey of adaptive sorting algorithms, *ACM Computing Surveys* **24** (1992), 441–476.

[2] D. E. KNUTH, "The Art of Computer Programming, Vol.3, Sorting and Searching," Addison-Wesley, Reading, Mass., 1974.

[3] H. MANNILA, Measures of presortedness and optimal sorting algorithms, *IEEE Trans. Comput. C-34* (1985), 318–325.