Patience sorting

From Wikipedia, the free encyclopedia

In computer science, **patience sorting** is a sorting algorithm inspired by, and named after, the card game patience. A variant of the algorithm efficiently computes the length of a longest increasing subsequence in a given array.

Patience sorting

Class	Sorting algorithm
Data structure	Array
Worst-case performance	$O(n \log n)$
Best-case performance	$O(n)$; occurs when the input is pre-sorted[1]

Contents

Overview

The algorithm's name derives from a simplified variant of the patience card game. This game begins with a shuffled deck of cards. These cards are dealt one by one into a sequence of piles on the table, according to the following rules.[2]

1. Initially, there are no piles. The first card dealt forms a new pile consisting of the single card.
2. Each subsequent card is placed on the leftmost existing pile whose top card has a value greater than or equal the new card's value, or to the right of all of the existing piles, thus forming a new pile.
3. When there are no more cards remaining to deal, the game ends.

This card game is turned into a two-phase sorting algorithm, as follows. Given an array of n elements from some totally ordered domain, consider this array as a collection of cards and simulate the patience sorting game. When the game is over, recover the sorted sequence by repeatedly picking off the minimum visible card; in order words, perform an k-way merge of the p piles, each of which is internally sorted.

Analysis

The first phase of patience sort, the card game simulation, can be implemented to take $O(n \log n)$ comparisons in the worst case for an n-element input array: there will be at most n piles, and by construction, the top cards of the piles form an increasing sequence from left to right, so the desired pile can be found by binary search.[1] The second phase, the merging of piles, can be done in $O(n \log n)$ time as well using a priority queue.[1]

When the input data contain natural "runs", i.e., non-decreasing subarrays, then performance can be strictly better. In fact, when the input array is already sorted, all values form a single pile and both phases run in $O(n)$ time. The average-case complexity is still $O(n \log n)$: any uniformly random sequence of values will produce an expected number of $O(\sqrt{n})$ piles,[3] which take $O(n \log \sqrt{n}) = O(n \log n)$ time to produce and merge.[1]

An evaluation of the practical performance of patience sort is given by Chandramouli and Goldstein, who show that a naïve version is about ten to twenty times slower than a state-of-the-art quicksort on their benchmark problem. They attribute this to the relatively small amount of research put into patience sort, and develop several optimizations that bring its performance to within a factor two of that of quicksort.[1]

If values of cards are in the range $1, \ldots, n$, there is an efficient implementation with $O(n \log \log n)$ worst-case running time for putting the cards into piles, relying on a Van Emde Boas tree.[3]

Relations to other problems

Patience sorting is closely related to a card game called Floyd's game. This game is very similar to the game sketched earlier:[2]

1. The first card dealt forms a new pile consisting of the single card.
2. Each subsequent card is placed on *some* existing pile whose top card has a value no greater than the new card's value, or to the right of all of the existing piles, thus forming a new pile.
3. When there are no more cards remaining to deal, the game ends.

The object of the game is to finish with as few piles as possible. The difference with the patience sorting algorithm is that there is no requirement to place a new card on the *leftmost* pile where it is allowed. Patience sorting constitutes a greedy strategy for playing this game.

Aldous and Diaconis suggest defining 9 or fewer piles as a winning outcome for $n = 52$, which happens with approximately 5% probability.[4]

Algorithm for finding a longest increasing subsequence

First, execute the sorting algorithm as described above. The number of piles is the length of a longest subsequence. Whenever a card is placed on top of a pile, put a back-pointer to the top card in the previous pile (that, by assumption, has a lower value than the new card has). In the end, follow the back-pointers from the top card in the last pile to recover a decreasing subsequence of the longest length; its reverse is an answer to the longest increasing subsequence algorithm.

S. Bespamyatnikh and M. Segal[3] give a description of an efficient implementation of the algorithm, incurring no additional asymptotic cost over the sorting one (as the back-pointers storage, creation and traversal require linear time and space). They further show how to report *all* the longest increasing subsequences from the same resulting data structures.

History

Patience sorting was named by C. L. Mallows, who attributed its invention to A.S.C. Ross in the early 1960s.[1] According to Aldous and Diaconis,[4] patience sorting was first recognized as an algorithm to compute the longest increasing subsequence length by Hammersley,[5] and by A.S.C. Ross and independently Robert W. Floyd as a sorting algorithm. Initial analysis was done by Mallows.[6] Floyd's game was developed by Floyd in correspondence with Donald Knuth.[2]

Use

The patience sorting algorithm can be applied to process control. Within a series of measurements, the existence of a long increasing subsequence can be used as a trend marker. A 2002 article in SQL Server magazine includes a SQL implementation, in this context, of the patience sorting algorithm for the length of the longest increasing subsequence.[7]

References

1. Chandramouli, Badrish; Goldstein, Jonathan (2014). *Patience is a Virtue: Revisiting Merge and Sort on Modern Processors* (PDF). SIGMOD/PODS.
2. Burstein, Alexander; Lankham, Isaiah (2006). "Combinatorics of patience sorting piles" (PDF). *Séminaire Lotharingien de Combinatoire*. **54A**.
3. Bespamyatnikh, Sergei; Segal, Michael (2000). "Enumerating Longest Increasing Subsequences and Patience Sorting". *Information Processing Letters*. **76**: 7–11. doi:10.1016/s0020-0190(00)00124-1.
4. Aldous, David; Diaconis, Persi. "Longest increasing subsequences: from patience sorting to the Baik-Deift-Johansson theorem". *Bulletin (new series) of the AMS*. **36** (4): 413–432. doi:10.1090/s0273-0979-99-00796-x.
5. Hammersley, John (1972). *A few seedlings of research*. Proc. Sixth Berkeley Symp. Math. Statist. and Probability. **1**. University of California Press. pp. 345–394.
6. Mallows, C. L. (1973). "Patience sorting". *Bull. Inst. Math. Appl.* **9**: 216–224.
7. Kass, Steve (April 30, 2002). "Statistical Process Control". *SQL Server Pro*. Retrieved 23 April 2014.