Constant-Working-Space Algorithms: How Fast Can We Solve Problems without Using Any Extra Array?

Tetsuo Asano

School of Information Science, Jaist,
1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan

In this talk I will present a new direction of algorithms which do not use any extra working array. More formally, we want to design efficient algorithms which require no extra array of size depending on input size n but use only constant working storage cells (variables), each having $O(\log n)$ bits. As an example, consider a problem of finding the median among n given numbers. A linear-time algorithm for the problem is well known. An ordinary implementation of the algorithm requires another array of the same size. It is not very hard to implement the algorithm without using any additional array, in other words, to design an in-place algorithm. Unfortunately, it is *not* a constant working space algorithm in our model since it requires some space, say $O(\log n)$ space, for maintaining recursive calls. A good news is that an efficient algorithm is known which finds the median in $O(n^{1+\epsilon})$ time using $O(1/\epsilon)$ working space for any small positive constant ϵ. The algorithm finds the median without altering any element of an input array. In other words, the input array is considered as a read-only array.

A main interest in this talk is how to design algorithms using only constant working space in addition to input arrays. There are two different situations depending on whether input arrays are read-only or not. If input data are stored in a read-only array and only constant working space is allowed in an algorithm, it is called a *constant working space algorithm with a read-only array.* If we can read any array element and write any information of $\log n$ bits into any array element in constant time, it is called a *constant working space algorithm with a read-write array.* The latter one is usually called as an in-place algorithm.

In this talk I will introduce several constant working space algorithms with read-write input arrays or with read-only input arrays. Such problems have been investigated in the community of complexity theory under the name of *log-space* computation. The *log-space* implies the working space of $O(\log n)$ bits for an input size n. There is no difference but their names. In the complexity theory a main concern is whether a problem belongs to *log-space*, that is, whether it is solvable in polynomial time using only small working space of $O(\log n)$ bits in total. My concern is not only polynomial-time solvability but also computational performance of the algorithm.

S.-H. Hong, H. Nagamochi, and T. Fukunaga (Eds.): ISAAC 2008, LNCS 5369, p. 1, 2008.