

Avancerade algoritmer  
Uppgift A

Peter Boström  
*[pbos@kth.se](mailto:pbos@kth.se)*

2010-10-01

## Assignment

In the lectures you've seen how to sort  $n$  word-sized integers on a unit-cost RAM model in  $O(n \log \log n)$  time. In this homework you will study special cases where it's possible to find easier algorithms or better time bounds.

- 1 Give a short description of the unit-cost RAM model and explain how the word-size  $w$  gives an upper bound on the number of integers  $n$  in the sorting problem above. (5p)**

The unit-cost RAM model gives a simplified computing model where all operations, arithmetic as well as loading/storing are performed in constant time.

A RAM model using a word size  $w$ , uses word-sized address pointers as well. Each element to be sorted (or even stored) requires a unique address. A pointer using a word size of  $w$  bits, cannot be used to address more than  $2^w$  elements. This introduces a limit to the number of elements that can be addressed, and therefore sorted, to  $2^w$ .

- 2 If the maximum element  $m$  is  $O(n)$  you may sort in linear time using a simple algorithm. Describe how. (5p)**

Counting sort. (Note that all integers are positive or zero.)

Reserve space for  $m + 1$  elements, initialize all to zero. This is assumed to be done in  $O(m)$  and therefore  $O(n)$  time. This array, *counts*, will be used to count the number of occurrences of each element. Increase *counts*[ $i$ ] for each element  $i$  in the list. All elements are now represented by the *counts* array, and the sum of each element in the *counts* array will be equal to the length of the original array, that is  $n$ . Now start writing back the counts of the array. Iterate through each possible value  $i$  which goes from 0 to  $m$ , and insert it into the array *counts*[ $i$ ] times. This is an  $O(n) + O(m)$  operation, that is  $O(n)$ .

- 3 Give an algorithm that sorts in linear time when the maximum element  $m$  is  $O(n^k)$ , where  $k$  is a positive constant. (5p)**

Radix sort. Each number will be looked upon  $k$  times and sorted into  $n$  different lists depending on their  $i$ th digit in lexical order.

- 4 Describe an algorithm that sorts in linear time if there are many repeated elements. How many distinct elements can your algorithm handle and how fast is it? (5p)**
- 5 The algorithm discussed in class uses large amounts of memory. How much? By doing the radix sorting phase in more than two steps you may reduce the memory requirements while increasing the running time. Explain how to do this and give a formula describing the time-space tradeoff. (5p)**