7 Algorithms (fms27)

Consider alternative algorithms for sorting an array of n items.

(*a*) The *BST-sort* algorithm looks at each element of the array in turn, starting at position 0, and inserts it into a BST (pass 1). Having processed all elements, it repeatedly extracts the minimum from the BST, refilling the array from position 0 onwards (pass 2).

(*i*) Derive, with justification, the computational complexity of each of the two passes of BST-sort. [2 marks]

(*ii*) Describe a way of asymptotically speeding up pass 2 without changing the data structure, yielding *enhanced BST-sort*, and give the new computational complexity of pass 2 and of the overall algorithm. [2 marks]

(*iii*) Compare enhanced BST-sort against heapsort, mergesort and quicksort with respect to asymptotic worst-case time and space complexity, saying when (if ever) it would be preferable to any of them. [3 marks]

(*b*) The *enhanced 2-3-4-sort* algorithm is obtained by replacing the BST with a 2-3-4 tree in enhanced BST-sort.

(*i*) Perform pass 1 of enhanced 2-3-4 sort on the array $\{6,9,3,1,4,3,6,7,5,0,2\}$, redrawing the tree at each insertion. [*Hint:* Remember to split 4-nodes on the way down when inserting, and to put $\leq$ keys in the left child and $>$ in the right.] [5 marks]

(*ii*) How much space will enhanced 2-3-4-sort require to sort an array of n items, if each item is m bits long? Give exact upper and lower bounds in terms of n and m rather than an asymptotic estimate. [3 marks]

(*iii*) Repeat question (*a*)(*i*) for enhanced 2-3-4-sort. [2 marks]

(*iv*) Repeat question (*a*)(*iii*) for enhanced 2-3-4-sort. [3 marks]