$T^R$. Step 2, in essence, computes all but the leftmost son of a node $X$. Each of these sons is obtained with one table look-up. The total number of sons cannot exceed $n$, since there are only that many nodes in the entire tree. Thus the growth of step 2 is $0(n)$. Step 3 merely sets up the sublist of sons for each node in the adjacency list of $T^R$. Hence step 3 also takes $0(n)$ time. In step 4, we find the first son of each node (except the root). The first son of a node in $T^R$ is the next node of the node under consideration in the adjacency list of $T$. Hence the growth of step 4 is also linear. We determine in step 5 whether a node in $T^R$ has a son other than the leftmost son found in step 4. This step is executed exactly $(n - 1)$ times. Thus the growth of the entire algorithm is $0(n)$.

It can easily be seen that the storage requirement of the algorithm is also linear in the number of nodes. In step 3 of the algorithm, the entry $X\ S(1)\ S(2) \ldots S(K)\ 0$ is added to the adjacency list of $T^R$. However, each $S(I)$ has to be accompanied by a link where the link should point to the beginning of the sublist where $S(I)$'s sons are stored. But this pointer to the sons of $S(I)$ will not be available when the sons of $X$ are computed. The following modification to the algorithm will get around this problem; the growth of the algorithm will still be linear. In step 0 of the algorithm, as a preprocessing step, change the values of the nodes to $1, 2, \ldots, N$ in the adjacency list of $T$. Store these values in a one dimensional array such that if a value $v$ is changed to the integer $J$, then $S(J)$ contains $v$. This array will be used to obtain the original values of the nodes at the end of the algorithm. Let $R$ be a $n \times 2$ array. The array entry $R(J, 1)$ can then be used to store the location of the link for the list of sons of node $J$ in the adjacency list of $T^R$. The entry $R(J, 2)$ can be used to store the actual value of the link to the sons of $J$ when it becomes available. Storing this information in $R$ takes only a constant amount of time for each $J$. When we exit the algorithm in step 4, we can complete filling in the links in the adjacency list of $T^R$ using $R$ again. This can be done in linear time for all links.

## 5. Acknowledgements

We wish to express our appreciation to Professor Carter Bays of the University of South Carolina for his many helpful suggestions in improving the readability of this paper.

## References

DE BRUIJN, N. G., KNUTH, D. E. and RICE, S. O. (1972). The average height of planted plane trees, *Graph theory and computing*, edited by R. C. Read, Academic Press, New York and London, pp. 15-22.

DEBRUIJN, N. G. and MORSELT, B. J. M. (1976). A note on plane trees, *Jour. of Comb. Theory*, Vol. 2, pp. 27-34.

HARARY, F., PRINS, G. and TUTTE, W. T. (1964). The number of plane trees, Neder Akad. Wetensch. Proc. Ser A67 (*Indag. Math.*, Vol. 26, pp. 319-329).

HIBBARD, T. N. (1962). Some combinatorial properties of certain trees with applications to searching and sorting, *JACM*, Vol. 9 No. 1, pp. 13-18.

HOROWITZ, E. and SAHNI, S. (1976). *Fundamentals of Data Structures*, Computer Science Press, Inc., 4566 Poe Avenue, Woodland Hills, Calif.

KLARNER, D. A. (1970). Correspondence between plane trees and binary sequences, *Jour. of Comb. Theory*, Vol. 9, pp. 401-411.

KNUTH, D. E. (1968). *The Art of Computer Programming*, Vol. 1: Fundamental Algorithms, Addison-Wesley, Reading, MA.

RENYI, A. and SZEKERES, G. (1967). On the height of trees, *Austral. J. Math.* Vol. 7, pp. 497-507.

# Book reviews

*Machine and Assembly Language Programming of the PDP-11* by Arthur Gill, 1979; 191 pages. (*Prentice-Hall*, £10·75)

This book is an expensive, slim volume which gives a very sound introduction to the PDP-11 machine organisation and programming. The author recommends that the reader be equipped with the manufacturer's processor, assembler and peripherals handbooks in order to undertake the exercises set in the text and also expects the reader to have encountered the basic concepts of algorithms, flowchart and stored program computer.

He commences with a strange first chapter which is a general review of number systems and various conversion algorithms—the reviewer would, however, admit its usefulness. It might have been better as an appendix referred to as necessary. The second chapter introduces the basics of PDP-11 machine organisation showing how the memory can be addressed by byte and by word (two bytes)—the even addresses. The general machine registers and peripheral registers are also introduced. The peripherals handled are limited to the console keyboard, console printer and line clock. The third chapter contrasts numbers and characters introducing integer, floating point and string representations. It is the reviewer's experience that students always have difficulty distinguishing coded representations from numeric value representations and the early emphasis on this in this text is to be applauded. The fourth chapter completes the groundwork for the rest of the text, dealing in full with the differing instruction forms and lengths and the various forms of addressing. The chapter includes several example coding sequences.

Having completed the machine code treatment, Chapter 5 introduces assembly language programming. Chapter 6 deals with stacks and subroutines introducing the system stack for the control of subroutine linkage and exit. The remaining chapters are: arithmetic operations, traps and interrupts, Assembler and Linkage Editor, Advance (Macro) Assembly techniques.

This is a well developed text with well developed examples. The author emphasises good programming style and dedicates an appendix to this. The book is to be recommended for anyone wishing to learn something of the PDP-11 class of computer.

A. H. WISE (Leicester)

*Assembly Language Fundamentals* by Rina Yarmish and J. Yarmish, 1978; 768 pages. (*Addison-Wesley*, £13·50)

This book sets out to teach Assembler for IBM/360 or /370 machines, under OS/VS and DOS/VS. The material is introduced gradually and is well explained, with many examples and copious exercises.

My only criticism is that it is rather too much of a good thing. The authors start from square one, assuming that the reader may not even know what a computer is like. Perhaps they would have served most of their readers by assuming at least some background in computing. Another economy could have been made by making the book more of a bridge to the manufacturer's manuals. As it is, the book cannot hope to cover everything, and the reader is in the end referred to the manuals.

If the aim of the book had been thus curtailed, it would have been just about as useful, though greatly reduced in bulk and price. But perhaps it is in the publishers' interest to produce expensive tomes?

A. COLIN DAY (London)