SIAM J. COMPUT.
Vol. 13, No. 1, February 1984

PUSHDOWN PERMUTER CHARACTERIZATION THEOREM*

PRAKASH V. RAMANAN†

Abstract. There is a well-known class of algorithms for permuting symbols which has been formally characterized by a device called a pushdown permuter. Two theorems attempting to characterize the type of permutations that can be achieved by a pushdown permuter have appeared in the literature. Counterexamples to these theorems have also appeared in the literature. This paper presents a correct statement of this theorem.

Key words. algorithm, permutation, stack, pushdown permuter

1. Introduction. There is a well-known algorithm which reads infix arithmetic expressions from left to right, one character at a time up to an end-marker, and, using a pushdown stack, produces from left to right, one character at a time, the suffix form of the expressions. This algorithm belongs to a well-known class of algorithms, characterized by their use of a single pushdown stack and a finite number of random access memory cells, to shuffle the characters about between the input string and the output string. Reingold [4] has developed a formal model for this class of algorithms, which he calls a pushdown permuter, and has attempted to characterize the type of permutations that can be achieved by a pushdown permuter. Carlson [1] has presented a counterexample to Reingold's characterization theorem. Shyamasundar [5] has also attempted to characterize this type of permutations. Ramanan [3] has presented a counterexample to Shyamasundar's characterization. In this paper we present a correct theorem characterizing the type of permutations that can be achieved by a pushdown permuter.

2. Pushdown permuter. Reingold defines a *pushdown permuter* (*p.d.p.*) to be a variant of a one-way, deterministic, finite state pushdown transducer whose finite input, output, and stack alphabets coincide. Informally, a p.d.p. can perform only the following kinds of steps: (a) It can read the input string one character at a time from left to right until it reaches an end-marker. (b) The character read from the input may be put directly into the output, which is also produced one character at a time from left to right, or it may be put on the top of a pushdown stack. (c) At any time, the only element accessible on the stack is the top element which can, if desired, be popped off the stack allowing access to the second element in the stack. The element popped off the stack can be thrown away, or it can be put into the output string. Once a symbol has been put in the output string it is forever after inaccessible and immutable.

A p.d.p. is nothing more than a control for a "switchyard" arrangement between the input string, the stack, and the output string. When the function of a p.d.p. is limited to just the permutation of the symbols from the input string, the capability of a p.d.p. to throw away symbols is not needed.

3. Pushdown permuter characterization. To simplify the notation we consider $12\cdots n$ to be the input string and $p_1p_2\cdots p_n$ to be the output string. Knuth [2, § 2.2.1, ex. 5] has shown that if the p.d.p. can have access to nothing except the top

* Received by the editors January 22, 1982, and in revised form February 10, 1983. This research was supported in part by the Office of Naval Research under grant N00014-79-C-0775.

† Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

stack symbol, then $p_1p_2 \cdots p_n$ can be produced if and only if there is no subsequence[1] $p_ip_jp_k$ of $p_1p_2 \cdots p_n$ for which $p_i > p_k > p_j$.

Reingold's pushdown permuter characterization theorem attempts to generalize the result stated in the previous paragraph to the case in which some fixed number of symbols can be stored in a random access memory, in addition to the stack. His characterization is as follows:

"A p.d.p. with M memory cells can permute the input string $12 \cdots n$ to $p_1p_2 \cdots p_n$ if and only if there is no subsequence $xy_1 \cdots y_{M+1}z_1 \cdots z_{M+1}$ of $p_1p_2 \cdots p_n$ such that for all i and j, $x > z_i > y_j$."

Carlson has presented a counterexample to this characterization. We present the following characterization:

THEOREM. *A p.d.p. with M memory cells can permute the input string $12 \cdots n$ to $p_1p_2 \cdots p_n$ if and only if there is no subsequence $p_rp_{i_1}p_{i_2} \cdots p_{i_{M+1}}$ of $p_1p_2 \cdots p_n$ satisfying the following conditions:*

i) *p_r is the largest symbol preceding p_{i_1} in $p_1p_2 \cdots p_n$. Moreover $p_r > p_{i_t}$ for all t, $1 \leqq t \leqq M+1$.*

ii) *Let $w = \max(p_{i_1}, p_{i_2}, \cdots, p_{i_{M+1}})$. For any prefix $p_1p_2 \cdots p_m$, $r \leqq m \leqq i_{M+1}$, of $p_1p_2 \cdots p_n$, let $f_m = \max(p_1, p_2, \cdots, p_m)$. Then*

$$|\{p_j \mid j > m,\ w < p_j \leqq f_m\}| \geqq |\{i_t \mid i_t \leqq m\}|.$$

Proof. First we shall prove the *if* part of the theorem. For a permutation $p_1p_2 \cdots p_n$ of $12 \cdots n$ consider the p.d.p. which behaves as follows: At each input symbol e the p.d.p. places e into a vacant memory cell, if one exists; otherwise it examines e and the symbols in the M memory cells, and, of those $M+1$ symbols, it puts the one which appears right-most in $p_1p_2 \cdots p_n$ onto the stack. If at any point $p_1p_2 \cdots p_k$ has been put into the output, and p_{k+1} is in one of the memory cells, is at the top of the stack, or is the next symbol in the input string, then p_{k+1} is put into the output and if a memory cell is vacated, it is filled with the top symbol in the pushdown stack, which is popped up. If there are no more input symbols, then, since we kept the left-most occurring symbols in the memory cells, we put them into the output in the appropriate order, filling the vacated cells with symbols taken from the top of the pushdown stack. This process continues until all of the symbols have been put into the output, or, perhaps, until the p.d.p. gets stuck with all memory cells filled and the symbol which must be put next into the output inaccessible on the stack. Clearly, if the p.d.p. does not get stuck, it will produce $p_1p_2 \cdots p_n$. We must verify that if the p.d.p. gets stuck, then $p_1p_2 \cdots p_n$ has a subsequence of the form mentioned in the theorem.

Suppose at some point the p.d.p. gets stuck with all memory cells filled and the symbol u which must be put into the output next, inaccessible below the top of the stack. Consider the p.d.p. at the time the symbol u is put into the stack for the last time, never to come out again, and call the contents of the M memory cells at that time $y_1, y_2, \cdots, y_M$. Since the y_i stay in the memory cells while u is put into the stack, they must precede u in $p_1p_2 \cdots p_n$, and since the p.d.p. does not get stuck until trying to put u into the output, the y_i are all in the output at the time the p.d.p. does get stuck. Let x be the largest symbol in the output preceding all of the y_i at the time the p.d.p. gets stuck. Clearly, $x > y_i$ and $x > u$ because at the time u goes into the stack for the last time, the p.d.p. is "waiting" for some symbol still in the input, while u and the y_i have already been read.

[1] We define $p_{i_1}p_{i_2} \cdots p_{i_k}$ to be a subsequence of $p_1p_2 \cdots p_n$ provided that $1 \leqq i_1 < \cdots < i_k \leqq n$.

So far our proof is identical to that of Reingold's. At the time the symbol u is put into the stack for the last time, never to come out again, the largest symbol the p.d.p. has read from the input is $\max(y_1, y_2, \cdots, y_M, u)$. Consider the p.d.p. at some later time t' before it gets stuck. Our assumption concerning the last time the symbol u is put into the stack implies that the number of symbols read from the input by the p.d.p. after u is put into the stack and until time t' be at least as large as the number of symbols put into the output by the p.d.p. during the same time period; otherwise, at some time during this period, the p.d.p. will pop up u from the stack and place it into a vacant memory cell. This argument holds true at any time, after u is put into the stack for the last time. Hence, we obtain the following condition: Let $p_1p_2\cdots p_m$ be any prefix of $p_1p_2\cdots p_n$ such that $x\in\{p_1, p_2, \cdots, p_m\}$, and $u\notin\{p_1, p_2, \cdots, p_m\}$. Let $w=\max(y_1, y_2, \cdots, y_M, u)$ and $f_m=\max(p_1, p_2, \cdots, p_m)$. Then

$$f_m - w \geqq |\{p_j \mid 1\leqq j\leqq m, p_j > w\}| + |\{y_i \mid y_i = p_j \text{ for some } j\leqq m\}|.$$

Moreover, it is clear that if we take the prefix $p_1p_2\cdots p_k$ up to, but not including u, strict inequality should hold; otherwise, u will be accessible at the top of the stack. If we let $p_r = x$, $\{p_{i_1}, p_{i_2}, \cdots, p_{i_M}\} = \{y_1, y_2, \cdots, y_M\}$, and $p_{i_{M+1}} = u$, for some $i_1 < i_2 < \cdots < i_{M+1}$, we have proved the existence of a subsequence of the form mentioned in the theorem.

Now we shall prove the *only if* part of the theorem. Let $p_1p_2\cdots p_n$ contain a subsequence of the form mentioned in the theorem. Since $p_r > p_{i_t}$ for all t, $1\leqq t\leqq M+1$, all of the $M+1$ p_{i_t} must be read and stored before we read p_r and put it into the output. Since there are $M+1$ p_{i_t}, they cannot all be stored within the M memory cells. So at least one of the p_{i_t} must be on the stack at the time p_r is put into the output. Let p_{i_q} be the bottom-most p_{i_t} on the stack at this time. Condition ii) of the theorem for $r\leqq m < i_q$ guarantees that p_{i_q} can never be popped off the stack, and placed in a memory cell. When $m = i_q$, it guarantees that p_{i_q} is not at the top of the stack. Hence the p.d.p. gets stuck when it tries to put p_{i_q} into the output.

Acknowledgments. The author is grateful to Edward Reingold and M. S. Paterson for their valuable suggestions during the preparation of this paper.

REFERENCES

[1] C. R. CARLSON, *A counterexample to Reingold's pushdown permuter characterization theorem*, this Journal, 8 (1979), pp. 199–201.

[2] D. E. KNUTH, *The Art of Computer Programming, Vol.* 1, *Fundamental Algorithms*, Addison-Wesley, Reading, MA, 1968.

[3] P. V. RAMANAN, *A counterexample to Shyamasundar's characterization of pushdown permuters*, Theoret. Comp. Sci., 23 (1983), pp. 103–105.

[4] E. M. REINGOLD, *Infix to prefix translation: The insufficiency of a pushdown stack*, this Journal, 1 (1972), pp. 350–353.

[5] R. L. SHYAMASUNDAR, *On a characterization of pushdown permuters*, Theoret. Comp. Sci., 17 (1982), pp. 333–341.