Introduction to Stack Sortable Permutations

Stuart Paton

December 19, 2012

1 Introduction

Stack sortable permutations were introduced by Donald Knuth in the 1960's with a problem involving the movement of railways cars across a railroad switching network. [1] [2]
A formal description of the stack sorting problem is as follows:
Consider an n-sized permutation $\sigma = \alpha_1\alpha_2...\alpha_{n-1}\alpha_n$. This is known as the 'input'. To start with we push α_1 on to the stack. Secondly, we compare it with the element α_2. If $\alpha_1 < \alpha_2$ then we push α_2 onto the stack, otherwise we pop α_1 from the stack and add it to the output and push α_2 on to the stack.

We continue this process of taking the leftmost element of our permutation and comparing it with the top element on the stack and repeating our comparison until the input is empty, the stack is empty and the output is full.

2 Stack Sortable Permutations

Definition A:
The identity permutation a permutation σ such that the image is in lexicographic ordering. This is $\sigma = \alpha_{1'}\alpha_{2'}...\alpha_{n-1'}\alpha_{n'}$ such that $\alpha_{1'} < \alpha_{2'} < ... < \alpha_{n-1'} < \alpha_{n'}$.

Definition B:
We say that a permutation σ is *single pass stack sortable* if the image $s(\sigma)$ is the identity permutation.

Theorem A: Consider the permutation $\sigma = \rho_1\rho_2...\rho_{n-1}\rho_n$.
Let $n = max(\rho_1, \rho_2, ..., \rho_{n-1}, \rho_n)$
Let α and β be the terms such that $\sigma = \alpha n\beta$.
Then:
$s(\sigma) = s(\alpha)s(\beta)n$
Proof: Every element before n will enter and leave the stack, and hence α will be sorted before n enters as it is larger. In the same fashion, after n enters the

stack, every element will enter and leave the stack and hence β will be sorted. Finally n will leave the stack. Hence our theorem is proven. $\square$

3 Single Pass Stack Sortable Permutations

Now let's look at where a given permutation is single pass stack sortable.//
Theorem B:
A permutation is single pass stack sortable if and only if the permutation avoids a 231-pattern.
Proof:
If a permutation σ constains a 231-pattern then, by definition, $s(\alpha)$ will contain an element larger than an element in $s(\beta)$, hence the image is not an identity permutation.
Conversely if the permutation σ does not contain a 231-pattern then consider the following:
For any two elements a and b such that a preceeds b, if $a > b$ then $\nexists c$ such that c is between a and b and $c > a$ (avoiding 231). Thus, a will enter the stack and not leave until b has left the stack hence b now preceeds a in $s(\sigma)$.
If $a < b$ then a will enter and leave the stack before b hence a will preceed b in $s(\sigma)$.
Hence $s(\sigma)$ is the identity pattern so σ is stack sortable. $\square$

Knuth proved that the number of permutations which are single pass stack sortable is the Catalan number C_n. Here is my proof of this:
Theorem C:
The number of single pass stack sortable permutations is the Catalan number C_n.
Proof:
We know from Theorem B that every permutation which avoids the pattern 231 is stack sortable.
Let's define $f(n)$ to be the number of single pass stack sortable permutations and $f(0) = 1$. Consider the permutation $\sigma_m = \alpha_1\alpha_2...\alpha_{m-1}\alpha_m$ and let $n = max(\alpha_1, \alpha_2, ..., \alpha_{m-1}, \alpha_m)$ such that $\sigma_m = \alpha n \beta$. Now from Theorem A we know that every element on the left of n must be smaller than every element on the right of n. So, from Theorem A we also see that the number of sortable permutations must be the number of sortable sub-permutations on the left of n multiplied by the number of sortable sub-permutations on the right of n. Formally this is: $|s(\sigma)| = |s(\alpha)| * |s(\beta)|$.
Summing all the possible permutations we get:
$f(n) = \sum_{i=0}^{n} f(i-1)f(n-i)$
This is analogous to our recursive definition of C_n:
$C_0 = 1$ and $C_n = \sum_{i=0}^{n} C_{i-1}C_{n-i}$
$\square$

References

[1] Donald E. Knuth, *The art of computer programming*, vol. 1, Addison-Wesley, Reading, Massachusetts, 1968.

[2] ———, *The art of computer programming*, vol. 3, Addison-Wesley, Reading, Massachusetts, 1973.