Code Jam 2010 - World Finals

Analysis: Letter Stamper

The problem idea originated from the report of one of Robert Tarjan's talks, where a quadratic solution for four letters was claimed. We found the easier three-letter version was already very interesting, and presented it to you as the first problem for this Code Jam final.

We note that the general case where the alphabet size is not restricted to a small constant such as 3 can be solved by dynamic programming in O(n^3). It had appeared in several programming contests before.

The current state is characterized by a pair `(S, K)`, where `S` is the suffix of the input string you still need to print (so the next letter you need to print is the head of `S`), and `K` is the stack. For each situation, we need to decide if the next step is a push, a pop, or a print.

We state some rules that are satisfied by an optimal sequence. These will make the solution quite clear. There can be other optimal sequences, but you can always reduce them to one that satisfies these conditions.

Rule 1. If the first letter of `S` is the same as the top of `K`, then the next step is a print.

Rule 2. You never need to push the letter that's already on top of the stack. Therefore no letters appears on the stack twice in a row.

Rule 3. Immediately after you push a letter `x`, the next step is to print `x`.

The rules above are trivial and we omit the reasoning. But it might worth spending a little time convincing yourself formally.

Rule 4. There should never be three consecutive letters in the stack of the form `xyx`.

Let us justify this rule. Suppose an optimal solution has XYX on the stack. Let's modify it. At some point the top of the stack is XY and we were going to push another X. Instead, we will pop the Y. Then we continue as before, until we were going to pop the second X that now doesn't exist. Instead, we will push Y, arriving at the same state as before. This way we have shortened the stack while achieving a solution of the same length. By induction you can keep simplifying the solution until all 4 rules are satisfied.

Rule 4, together with Rule 2, implies that the stack always contains a cycle of 3 letters, e.g. `ABCABCABC...`. The state of the stack then is completely defined by the first two letters and the stack height. There are only 6 patterns, and the height of the stack is never bigger than `n`. Hence the number of possible states `(S, K)` is `O(n^2)`. This gives a quadratic dynamic programming solution.