Partial Super Permutation Algorithm*

Arogya Dahal

July 2021

1 Keywords

- *array*: The array containing the super permutation
- n: The number we are finding partial super permutation for
- *Position*: The position of n in the last n number of term in array. So, if we are finding the super permutation for 5, and our current matrix is (1), the ***position*** will be 2.

$$\begin{pmatrix} 1, & 2, & 3, & 4, & 5 \\ 1, & 2, & 3, & 4, & 1 \\ 5, & 2, & 3, & 4, & 1 \\ 2, & 5, & 3, & 4, & 1 \end{pmatrix} \tag{1}$$

- *Shift*: The interchanging of position of n with what is on it's right. In (1), the position of 5(on the second last row) is shifted with 2 to get the last row.
- *Ledge Jump*: When n jumps from its current position to right. In (1), 5 ledge jumps on the first row.

*I am writing this after quite a long time, so there might be some errors. I would like to apologize.

2 Algorithm

Here is the explanation of the algorithm in not so good English. The first explanation will be in words, then the algorithm will be shown and then a demo. I think super permutation for 5 will be a good fit as it is short and contains most of the concepts. Again, I don't know if this algorithm has already been found, and if it has been found, I will be more than happy to remove it ☺.

2.1 In Words

Let A be that will be containing the superpermutation for n^th number. For super permutation of n, we start with a sorted array starting from 1 and ending with n. Initially,

$$A = \begin{pmatrix} 1, & 2, & 3, & 4, & \cdots, & n-1, & n \end{pmatrix} \tag{2}$$

We shift the n by 1 units and check if the number so formed is in the array. So here n with shift with 1 and check if the array $\begin{pmatrix} n, & 2, & 3, & 4, & \cdots, & n-1, & 1 \end{pmatrix}$ belongs in A, if it belongs in A we shift n by another 1 unit and check if that resulting array is in A or not. Here, the second matrix doesn't belong in the matrix original A matrix so the resulting matrix will look something like this.

$$A = \begin{pmatrix} 1, & 2, & 3, & 4, & \cdots, & n-1, & n \\ 1, & 2, & 3, & 4, & \cdots, & n-1, & 1 \\ n, & 2, & 3, & 4, & \cdots, & n-1, & 1 \end{pmatrix} \tag{3}$$

The array that is added to A is added after the position of n and the number is with which a unique array was formed is interchanged. Here, 1 and n's interchanging created a new array, so we added the array to the matrix. What about the second array on the second?! Here, we are changing the position of n and 1 keeping all the digits the same, since 1 is after the ledge jump, the replacement should be after the jump. So, n jumps to the next row keeping 1 on first column of second row safe.

Now the same pattern holds true for the entirety of the execution, n shifts by 1 unit to the right, we check if it occurs in the array, if it doesn't we add the resulting array to our A;however, if the resulting array is in A, we shift again to the right, we do this until a new array is found. If no array is found the execution is stopped. What if a new array is found after multiple shifting? Well in that case the new array is formed by interchanging the position of n and the digit and reversing everything in between n and the position of the number and adding it to A. So let's say in (3), no new array is found until n replaces with 4, in that case the array that is added to A will be:

$$\begin{pmatrix} 4, & 3, & 2, & n, & \cdots, & n-1, & 1 \end{pmatrix} \tag{4}$$

Thus the final A matrix will look something like this:

$$A = \begin{pmatrix} 1, & 2, & 3, & 4, & \cdots, & n-1, & n \\ 1, & 2, & 3, & 4, & \cdots, & n-1, & 1 \\ \vdots, & \vdots, & \vdots, & \vdots, & \vdots, & \vdots & \vdots \\ n, & 2, & 3, & 4, & \cdots, & n-1, & 1 \\ 4, & 3, & 2, & n, & \cdots, & n-1, & 1 \end{pmatrix}$$

This same rule holds during the ledge jump as well.

2.2 In Algorithm

In term of algorithm it can be shown as follows:

Algorithm: Partial Super Permutation

```
A = (1, 2, 3, 4, ⋯, n − 1, n);
C = () ;
while All possible shift isn't in A do
    C = A[::-5];
    Shift the position of n by 1 step to the right ;
    if C is not in A then
        Reverse every number between the original position and the
          unique position and add it to the original array;
    end
end
```

2.3 C++ Code

The code can be found here arogyad.

2.4 Demo

Lets do the demo for $n = 5$. We begin with:

$$\begin{pmatrix} 1, & 2, & 3, & 4, & 5 \end{pmatrix}$$

We shift from the right hand side, since the interchanging of 1 and 5 results in a new array; we can add this new array to our older one. So the new formed array can be given as:

$$\begin{pmatrix} 1, & 2, & 3, & 4, & 5 \\ 1, & 2, & 3, & 4, & 1 \\ 5, & 2, & 3, & 4, & 1 \end{pmatrix}$$

We, then, interchange the position of 2 and 5. Since the resulting array doesn't appear in the original array before, it belongs to the original array.

$$\begin{pmatrix} 1, & 2, & 3, & 4, & 5 \\ 1, & 2, & 3, & 4, & 1 \\ 5, & 2, & 3, & 4, & 1 \\ 2, & 5, & 3, & 4, & 1 \end{pmatrix}$$

Repeating the same steps, we can form the next array as well.

$$\begin{pmatrix} 1, & 2, & 3, & 4, & 5 \\ 1, & 2, & 3, & 4, & 1 \\ 5, & 2, & 3, & 4, & 1 \\ 2, & 5, & 3, & 4, & 1 \\ 2, & 3, & 5, & 4, & 1 \end{pmatrix}$$

On to the next step, when we interchange 5 and 4, the resulting array already exists i.e $\begin{pmatrix} 2, & 3, & 4, & 5, & 1 \end{pmatrix}$ already exists in the array (Five digits starting from first row second colum forms $\begin{pmatrix} 2, & 3, & 4, & 5, & 1 \end{pmatrix}$). So, we interchange 5 and 1, which doesn't exist in the array, so the next 5 digits are $\begin{pmatrix} 2, & 3, & 1, & 4, & 5 \end{pmatrix}$ So the main array becomes:

$$\begin{pmatrix} 1, & 2, & 3, & 4, & 5 \\ 1, & 2, & 3, & 4, & 1 \\ 5, & 2, & 3, & 4, & 1 \\ 2, & 5, & 3, & 4, & 1 \\ 2, & 3, & 5, & 4, & 1 \\ 2, & 3, & 1, & 4, & 5 \end{pmatrix}$$

Following this algorithm, we can create the complete super permutation for $n \leq 6$ and partial super permutation for $n \geq 7$.