**2 A Closed Formula for the Symmetric Group *S\_n***

Before presenting the formula for generating all permutations of the symmetric group *S\_n*, we will describe what is a **permutation**, then show some initial cases of generating permutations, and then finally present the formula.

**Definition 2.1.** *Permutations are functions over a finite product of a totally ordered set to itself, which rearrange the terms of the product set. That is, if A is a set, and n a natural number, then τ = [j\_1j\_2 ... j\_n] denotes a* ***permutation, τ*** *: A x A x A ... A --> A x A ... x A (n times), which rearranges the input (a\_1, a\_2, ..., a\_n) with respect to τ, [j\_1j\_2 ... j\_n], where j\_1, j\_2, ... , j\_n are in {1, 2, ... , n} and, j\_i <> j\_k if and only if, i <> k, i, k in {1, 2, ... , n} so that 1st location of the n-tuple (a\_1, a\_2, ... , a\_n) takes the value a\_{j\_1} and the j\_1st location takes the value a\_1, the 2nd the value of a\_{j\_2} and the j\_2nd location the value a\_2, and so on.*

**Definition 2.2.** *Transpositions are functions as permutations, only that a* ***transposition σ****, is a [ij], where the i-th location of the tuple (a\_1, a\_2, ..., a\_n) gets the a\_j and the j-th location the a\_i. That is an interchange of terms i, j.*

From the above two definitions, one can say that a permutation is composed of at least one transposition. The set of permutations is a finite group, called *symmetric group S\_n of order n!*, with composition law the trivial function composition, and special ("unit") element the permutation τ = [123...n].

Now, let's try a method of generating all permutations on initial cases. Consider for the beginning we get [1234].



From Figure 2.1., one can see that the following formula can be derived which generates the group *S\_n*. Let first denote the equalities (1):

A\_1 = [12]

A\_2 = [23]

A\_j = [j(j+1)]

.

.

.

A\_(n-1) = [(n-1)n]

**Theorem 2.1.** *The formula:*

*(A\_{j\_1} A\_{j\_2} ... A\_{j\_{k\_1}})^N\_1 (A\_{i\_1} A\_{i\_2} ... A\_{i\_{k\_2}})^N\_2 ... (A\_{s\_1} A\_{s\_2} ... A\_{s\_{k\_m}})^N\_m, (2),*

*where j\_1, j\_2, ...,j\_{k\_1}, i\_1, i\_2, ..., i\_{k\_2},..., s\_1, s\_2, ..., s\_{k\_m} are in {1, 2, ... (n-1)}, and 0 <= k\_j <= n - 1, 1 <= j <= m, and N\_j, 1 <= j <= m, can be any natural number, generates all possible permutations of the group S\_n.*

*Proof:* It will be proved by applying induction on the number of transpositions a permutation can be broken down. Initially assume that a permutation τ is composed from a single transposition [ij] (j = i + m, m > 0). Then the effect of this transposition is equivalent to applying the composition of transpositions, [(j - 1)j] ... [(i + 1)(i + 2)] [i(i + 1)] followed by the composition of transpositions [i(i + 1)] ... [(j - 2)(j - 1)]. (Recall that transpositions are function, and composition of transpositions are function composition applied to an n-tuple.) By the equalities (1), we have (A\_i A\_(i + 1) ... A\_(j - 2) A\_(j - 1) A\_j ... A\_(i + 1) A\_i), which is a straightforward application of (2) with N\_m = 1, with m = 1.

Assume now that any permutation τ composed from k transpositions is described by (2) that is, τ = (A\_{j\_1} A\_{j\_2} ... A\_{j\_{k\_1}})^N\_1 (A\_{i\_1} A\_{i\_2} ... A\_{i\_{k\_2}})^N\_2 ... (A\_{s\_1} A\_{s\_2} ... A\_{s\_{k\_m}})^N\_m. Of course, **Σ**N\_i \* k\_i = k, where 1 <= i <= m. Subsequently, consider the application of one extra transposition σ = [ij] on τ. Then στ = [ij]τ. The transposition σ, as we saw in the initial case of the induction, breaks down to a form as (2). Therefore, στ has the form of (2). QED

**Remark 2.1.** We can further see that **Σ**N\_i, 1 <= i <= m, is bounded below. To see that, one can write (2) as,

(A^(0,1)\_1 A^(0,1)\_2 ... A^(0,1)\_{(n - 1)})^N\_1 (A^(0,1)\_1 A^(0,1)\_2 ... A^(0,1)\_{(n - 1)})^N\_2 ... (A^(0,1)\_1 A^(0,1)\_2 ... A^(0,1)\_{(n - 1)})^N\_m,

where A^(0,1) means that either the transpositions as in (1) appears, or not. Therefore, we have at most M = 2^{(n - 1) \* **Σ**N\_i} (1 <= i <= m) permutations generated by the formula (2). Therefore M >= n! => (n - 1) \* **Σ**N\_i >= **Σ**logj, (1 <= j <= n) => **Σ**N\_i = o(**Σ**logj/(n - 1)) (1 <= j <= n) (3). That is, it suffices to apply (2) at least [**Σ**logj/(n - 1)] + 1 on **Σ**N\_i, to generate *S\_n*. ([x] gives the integer part of the number x.)

A piece of algorithmic code that generates all permutations applying (2) that will be used to the series of quantum algorithms in the coming sections, is as below:

S\_n = {}

M = [**Σ**logj/(n - 1)] + 1

**for** N\_1, N\_2, ..., N\_M = 1 **to** **Σ**N\_i ( 1<= i <= M) = 1 **to** M **do**

**for** b\_{(1,1)}, b\_{(1, 2)}, ..., b\_{(1,(n - 1))} = 0, 1 **do**

**for** b\_{(2,1)}, b\_{(2, 2)}, ..., b\_{(2,(n - 1))} = 0, 1 **do**

.

.

.

**for** b\_{((n - 1),1)}, b\_{((n - 1), 2)}, ..., b\_{(M,(n - 1)))} = 0, 1 **do**

S\_n= S\_n **U Π**\_m (A^{b\_{(m,1)}}\_1 A^{b\_{(m, 2)}}\_2 ... A\_^{b\_{((m,(n - 1))}}\_{(n - 1)})^N\_m (1 <= m <= M)

**end**

**end**

.

.

.

**end**

**end**

**end**