First lets come up with a simple dp $O(n^2)$ solution.
Lets define $dp_{i,j}$ as number of permutations of length i that $parts(p) = j$.
We can see that $dp_{i,j} = dp_{i-1,j-1} + dp_{i-1,j} \times (i-1)$ because:

- Consider we want to add number i to the permutation of numbers from 1 to $i-1$.
- One case is to add i to end of permutation, in this case we have $p_i = i$, so i must be in s.
- Other case is to add i in some where else like j $(1 \leq j \leq i-1)$ and set p_i to p_j. In this case minimum size of s does not change and we have $i-1$ ways to choose j in this case.

Now the problem becomes calculating $dp_{i,j}$ in an efficient way.
We can extract from recursive definition of $dp_{i,j}$ that:

- $dp_{i,j} = \sum_{S \subseteq \{1,2,\ldots,i-1\}}^{|S|=i-j} (\prod_{j \in S} j)$

Consider this polynomial $P_n(x) = (1+x) \times (1+2x) \times \ldots \times (1+(n-1)x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1}$.
Lemma: $dp_{n,i} = a_{n-i}$.
Proof: to prove this we show how each subset (S) in non-recursive definition of $dp_{n,i}$ is mapped to a choice of each parenthesis in $P_n(x)$.
For each j $(1 \leq j \leq i-1)$ that $j \in S$ we choose jx in jth parenthesis in $P_n(x)$ in others we choose 1 in that parenthesis.
So the product will be ax^b that $a = \prod_{j \in S} j$ and $b = i - j$.

Now the problem becomes calculating $P_n(x)$ in a fast way.
We use divide and conquer here.

- Lets define $calc(l,r) = (1+lx) \times (1+(l+1)x) \times \ldots \times (1+(r-1)x)$.
- $calc(l,l+1) = (1+lx)$.
- $calc(l,r) = calc(l,mid) \times calc(mid,r)$ that $mid = \frac{l+r}{2}$.

If we use FFT for multiplication of two polynomials the time complexity of algorithm is:
$T(n) = 2 \times T(n/2) + O(n \times log(n))$ that is $O(n \times log(n)^2)$.