

Problem 1: Generating Permutations (20 pts.)**Background**

Here is one of the standard algorithms for a successor operation on permutations of $[n]$: given as input a permutation $A = (a_1, a_2, \dots, a_n)$ of $[n]$, the algorithm will compute the “next” permutation. Here is an informal description:

- Find the largest position $i < n$ such that $a_i < a_{i+1}$.
- Find the least element $a_j > a_i$ in positions $i < j \leq n$.
- Swap a_i and a_j .
- Sort the tail-end a_{i+1}, \dots, a_n of the permutation.

This works as long as the position i in the first step exists. Otherwise the whole permutation is already descending, i.e., $A = (n, n-1, \dots, 2, 1)$, in which case we wrap around and return the identity permutation. Below are the first few iterations of the algorithm on the identity permutation for $n = 4$.

```

1 2 3 4
1 2 4 3
1 3 2 4
1 3 4 2
1 4 2 3
1 4 3 2
2 1 3 4
2 1 4 3
2 3 1 4
2 3 4 1
2 4 1 3

```

You might want to implement this algorithm so you are absolutely clear how it works.

Task

- Show that this algorithm is correct: the orbit of any permutation on $[n]$ under this operation consists of all permutations of $[n]$.
- Carefully explain the running time of this operation. Any clever ideas?

Problem 2: Necklaces (30 pts.)**Background**

A necklace is a circular string of colored beads. Two necklaces are considered to be the same if we can rotate one to obtain the other, but reflections are not allowed here (as opposed to bracelets where reflections are allowed). There are two parameters: the length n of the necklace and the number k of colors, correspondingly we speak of (k, n) necklaces.

Task

- A. Compute the number of binary necklaces of length 8.
- B. How many of these necklaces have exactly 4 black and 4 white beads?
- C. Compute the pattern inventory for these necklaces.
- D. Find a general description for the number of (k, n) necklaces.

Comment

Try to make the formula in part D as nice as possible (there will not be a really simple closed form).

Problem 3: Boolean Circuits (50 pts.)**Background**

We have seen how to count Boolean functions modulo the equivalence relation induced by inverting some inputs. Several other modifications produce similar classifications of Boolean functions on k inputs. The space of configurations here is

$$X = \mathbf{2}^k \rightarrow \mathbf{2}$$

Task

1. Count the number of Boolean functions when inputs and the output can be inverted.
2. Count when inputs can be rotated. So if the rotation is by one place we have

$$f(x_1, x_2, \dots, x_k) = g(x_2, x_3, \dots, x_k, x_1)$$

3. Count when inputs can be arbitrarily permuted.

$$f(x_1, x_2, \dots, x_k) = g(x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(k)})$$

Comment

Always explain clearly what exactly the group is that acts on the space of Boolean functions.

Try to come up with a concise, elegant answer for the count but don't necessarily hope for a simple closed-form solution.