E-402-STFO PROBLEMS FOR MODULE 2

CREATED BY HENNING ULFARSSON

This is the first module concernded with permutations. You get a perfect score for this module by getting 60 points or more.

1. Inversions and descents (20 points)

Let $\pi = a_1a_2 \dots a_n$ be a permutation. An inversion in π is a pair (i, j), with $1 \le i < j \le n$, such that $a_i > a_j$. A descent is a number i such that $\pi_i > \pi_{i+1}$. Let $\mathrm{inv}(\pi)$ and $\mathrm{des}(\pi)$ denote the number of inversion and descents in π, respectively. For instance, if $\pi = 24153$ then $\mathrm{inv}(\pi) = 4$ and $\mathrm{des}(\pi) = 2$. You can use the built in function `Permutations(n)` to generate the permutations of $\{1, \dots, n\}$.

Problem 1 (5 points)**.** Implement a function `m2p1(p)` which counts the number of inversions for a given permutation `p`.

- Input: `m2p1(Permutation([2,4,1,5,3]))`
- Output: `4`

Problem 2 (5 points)**.** Implement a function `m2p2(p)` which counts the number of descents for a given permutation `p`.

- Input: `m2p1(Permutation([2,4,1,5,3]))`
- Output: `2`

Problem 3 (10 points)**.** Implement a function `m2p3(n)` which counts the number of permutations of length `n` that have the same number of inversions and descents.

- Input: `m2p3(4)`
- Output: `5`

Note: Your function will be tested with very large inputs.

2. Avoidance (40 points)

Problem 4 (10 points)**.** A classical pattern in a permutation is a subsequence of letters, possibly with gaps, in a particular order. For example, the letters 523 in the permutation 15243 form the classical pattern 312. Write a function `m2p4(p,cl)` that takes as input a permutation `p` and a classical pattern `cl` and returns `True` if the permutation contains the pattern, but `False` otherwise.

- Input: `m2p4(Permutation([1,5,2,4,3]),Permutation([3,1,2]))`
- Output: `True`

Problem 5 (10 points)**.** The bubble-sort operator (B) moves the elements of a permutation to the right, until they reach a larger element. For example, when we apply B to the permutation 532614 the 5 starts moving past the 3 and the 2. Then it hits the 6 and stops. The 6 then moves past 1 and 4 until it reaches the end. So $B(532614) = 325146$. Write a function `m2p5(p,cl)` that takes as input a permutation `p` and applies the bubble-sort operator to it.

- Input: `m2p5(Permutation([5,3,2,6,1,4]))`
- Output: `Permutation([3,2,5,1,4,6])`

Date: Updated September 3, 2013.

Problem 6 (10 points)**.** Let's give ourselves the fact that permutations that bubble-sort to the identity (in one pass) are exactly the permutations that avoid two mystery classical patterns. Write a function `m2p6()` that outputs these patterns.

Problem 7 (10 points)**.** Look at what the function `to_standard()` does to a string like `[1,3,7,5]` and implement a faster version of your own called `m2p7(p)`

- Input: `m2p7([1,3,7,5])`
- Output: `Permutation([1,2,4,3])`

Note: To access the function `to_standard()` you first have to execute the command `from sage.combinat.permutation import to_standard`.

3. Learning classical patterns (30 points)

Problem 8 (30 points)**.** Many interesting infinite sets of permutations are defined by the avoidance of classical patterns. To name just two,

- The permutations sortable by one pass through a stack are the avoiders of 231 (Knuth)
- The permutations that correspond to smooth Schubert varieties are the avoiders of 2413 and 1324 (Lakshmibai and Sandhya)

Other infinite sets are *not* defined by the avoidance of classical patterns (we will look at what kind of patterns we need for them later).

Given some infinite set of permutations $\mathcal{S}$ we can look at a finite piece $\mathcal{S}_{\text{fin}}$ and try to guess what classical patterns $\mathcal{S}$ avoids. Write a function `m2p8(L)` that outputs the classical patterns if possible – otherwise `False`.

- Input: `m2p8([Permutation([1]), Permutation([1, 2]), Permutation([2, 1]), Permutation([1, 2, 3]), Permutation([1, 3, 2]), Permutation([2, 1, 3]), Permutation([2, 3, 1]), Permutation([3, 1, 2]), Permutation([1, 2, 3, 4]), Permutation([1, 2, 4, 3]), Permutation([1, 3, 4, 2]), Permutation([1, 4, 2, 3]), Permutation([2, 1, 3, 4]), Permutation([2, 1, 4, 3]), Permutation([2, 3, 1, 4]), Permutation([2, 3, 4, 1]), Permutation([2, 4, 1, 3]), Permutation([3, 1, 2, 4]), Permutation([3, 1, 4, 2]), Permutation([3, 4, 1, 2]), Permutation([4, 1, 2, 3]), Permutation([1, 2, 3, 4, 5]), Permutation([1, 2, 3, 5, 4]), Permutation([1, 2, 4, 5, 3]), Permutation([1, 2, 5, 3, 4]), Permutation([1, 3, 4, 5, 2]), Permutation([1, 4, 5, 2, 3]), Permutation([1, 5, 2, 3, 4]), Permutation([2, 1, 3, 4, 5]), Permutation([2, 1, 3, 5, 4]), Permutation([2, 1, 4, 5, 3]), Permutation([2, 1, 5, 3, 4]), Permutation([2, 3, 1, 4, 5]), Permutation([2, 3, 1, 5, 4]), Permutation([2, 3, 4, 1, 5]), Permutation([2, 3, 4, 5, 1]), Permutation([2, 3, 5, 1, 4]), Permutation([2, 4, 1, 5, 3]), Permutation([2, 4, 5, 1, 3]), Permutation([2, 5, 1, 3, 4]), Permutation([3, 1, 2, 4, 5]), Permutation([3, 1, 2, 5, 4]), Permutation([3, 1, 4, 5, 2]), Permutation([3, 1, 5, 2, 4]), Permutation([3, 4, 1, 2, 5]), Permutation([3, 4, 1, 5, 2]), Permutation([3, 4, 5, 1, 2]), Permutation([3, 5, 1, 2, 4]), Permutation([4, 1, 2, 3, 5]), Permutation([4, 1, 2, 5, 3]), Permutation([4, 1, 5, 2, 3]), Permutation([4, 5, 1, 2, 3]), Permutation([5, 1, 2, 3, 4])])`
- Output: `[Permutation([3,2,1]), Permutation([1,3,2,4])]`

School of Computer Science, Reykjavik University, Menntavegi 1, 101 ReykjavÃŋk, Iceland

E-mail address: `henningu@ru.is`