MT2505 Computing Project 2: Computing with permutations

MRQ

Semester 2, 2019–20

Overview

This is the second part of the Computing Project for MT2505. Your submission should be a single file consisting of your attempts to the first part of the Computing Project together with your attempts to this part. See the **Overview** of the first part and the `guidance` document for further information. The concepts found in this part of the project are introduced in Chapter 6 of the lecture notes.

The `groups` module

This part of the Computing Project depends upon use of the module `groups` written by Prof. J. D. Mitchell with one (extremely minor) modification by MRQ. The module is available in MMS and can be loaded using the `import` command:

```
from groups import *
```

The following commands are defined in the `groups` module:

```
G = SymmetricGroup(5) # the symmetric group of given degree

G.identity()          # returns the identity
G.order()             # returns the order of G

[x for x in G]        # produces a list of all the elements of G

x = Perm((1,2),(4,5)) # create a permutation by specifying its
y = Perm((4,3),(2,1)) # disjoint cycle structure

x.degree()            # returns the degree of the symmetric
                      # group to which the permutation x belongs
x.hit(1)              # apply the permutation x to the given
                      # point

x * x                 # multiply permutations
x ** 2                # calculate the power of a permutation
x ** -1               # invert a permutation

x == y                # test equality of two permutations
str(x)                # convert a permutation to a string

IsSymmetricGroup("banana")
              # Check if something is a symmetric group.
              # Only returns True if the input was created
              # using the SymmetricGroup command.
```

Cycle structure and the order of an element

Experiment with the commands listed above to ensure that you are happy with their behaviour.

Question 6: Write a function called `cycle_length` that takes two inputs, the first that is a permutation `x` and the second is an integer `k` between 1 and the degree of the given permutation (inclusive), and returns the length of the cycle of `x` that contains `k`.

The code for your function should therefore start with the following:

```
def cycle_length(x, k):
```

Check and Debug Your Code: Check that your function gives the correct answer for the following permutations:

(a) `Perm((1, 2, 3), (4, 5, 6), (10, 11, 12))`

(b) `Perm((1, 3), (4, 5), (6, 8, 10), (7, 9))`

(c) `Perm((1, 4, 3, 2, 12, 7, 10, 5), (9, 11))`

(d) `Perm((1, 8, 12, 9, 7, 5, 4, 11, 10, 2), (6, 13))`

(e) `Perm((1, 9, 8, 13, 10, 5), (2, 14), (3, 6, 4, 7, 11, 12))`

The order of an element: In Chapter 8 of the lecture notes, we shall meet the concept of the *order* of an element x of a group G. This is the smallest positive integer n (if it exists) such that x^n is the identity element. In particular, this definition applies to a permutation from a symmetric group of finite degree. We shall also meet a useful theorem in Chapter 8 that gives further information about the order of a permutation. You can probably solve Question 6 without that theorem, but you might consider revising your answer based on what we do in Chapter 8 when you come to solve Question 7. (Alternatively you might want to leave solving these questions until we have covered the theorem.)

Question 7: Write a function called `order_perm` to find the order of a permutation. [It might help you to start by finding the degree of the input permutation at some point in your code.]

Check and Debug Your Code: Check that your function gives the correct answer for the permutations listed above[1]. If necessary, fix your code.

Question 8: The final question concerns a permutation of very large degree. Rather than require you to download the large file storing the permutation, I have placed it on amongst my webpages so that you can access it. The following code accesses the file and unpacks its content:

```
import gzip, pickle, requests
from groups import *
open("vlp.p.gz", "wb").write(requests.get(
  "https://tinyurl.com/bigperm2020").content)
p = pickle.load(gzip.open("vlp.p.gz", "r"))
```

You can now work with the permutation `p`.

Use Python to calculate the order of this permutation `p`. Your command `order_perm` from Question 6 may fail for this permutation, or at least might fail to run within the required 20-minute window, so you may have to write new code.

[Suggestion: Experiment with small examples first. Once you have a version that works with small examples, think how you might make your code more efficient if it takes a very long time with the large example `p`.]

[1]The answers should be: (a) `3`; (b) `6`; (c) `8`; (d) `10`; (e) `6`.