Permutations

Permutations

We can work with permutations in Sage:

```
p = Permutation([2,3,1,5,4])
p
```

```
[2, 3, 1, 5, 4]
```

There are several ways to represent permutations. The above presentation is one-line notation. Permutations can also be represented by their corresponding permutation matrix:

```
p.to_matrix()
```

```
[0 0 1 0 0]
[1 0 0 0 0]
[0 1 0 0 0]
[0 0 0 0 1]
[0 0 0 1 0]
```

Or in cycle notation:

```
p.cycle_string()
```

```
'(1,2,3)(4,5)'
```

Sign and inversions

We can count the number of inversions of a permutation:

```
p
```

```
[2, 3, 1, 5, 4]
```

```
p.inversions()
```

```
[[0, 2], [1, 2], [3, 4]]
```

The number of inversions tell us whether the permutation is even or odd:

```
p.is_even()
```

```
False
```

Reduced words

Every permutation can be written as a product of simple transposition which interchange two adjacent letters i and i+1. The reduced word tells you which simple transpositions occur in the product.

```
p.reduced_word()
```

```
[1, 2, 4]
```

```
p1 = Permutation([2,1,3,4,5])
p2 = Permutation([1,3,2,4,5])
p3 = Permutation([1,2,3,5,4])
p3*p2*p1
```

```
[2, 3, 1, 5, 4]
```

Matrix representation

We can compose permutations. Unfortunately, in sage the left permutation is applied first (as we already saw when we looked at the reduced word).

```
p
```

```
[2, 3, 1, 5, 4]
```

```
q = Permutation([2,5,4,3,1])
q*p
```

```
[3, 4, 5, 1, 2]
```

The matrix of the composition of two permutations corresponds to the matrix product of the matrices for each permutation (again up to changing the order):

```
(q*p).to_matrix()
```

```
[0 0 0 1 0]
[0 0 0 0 1]
[1 0 0 0 0]
[0 1 0 0 0]
[0 0 1 0 0]
```

```
p.to_matrix()*q.to_matrix()
```

```
[0 0 0 1 0]
[0 0 0 0 1]
[1 0 0 0 0]
[0 1 0 0 0]
[0 0 1 0 0]
```

```
(q*p).to_matrix() == p.to_matrix()*q.to_matrix()
```

```
True
```