1. The implementation is in Q1.py file. You can test the algorithm with any input to the main, just pay attention to the notation of the input.
   The code in commented completely. I've used 2 python functions, combinations, for selecting all possible tuples of the reverses, because there is no difference between (2,5) and (5,2) here. And product, for combining these possible tuples together and includes every possible sequence of them.
   The whole logic is just like the hint, m starts from 0 to the length of the permutation and the first time that a sequence of reversals with length m results in the identity, the brute force ends.
2. The source code is in Q1.py. The code in commented completely. Here's a sample output:

permutation	d(permutation)	lower bound	execution time
[1]	0	0	0
[1, 2]	0	0	0
[2, 1]	1	1	0
[2, 1, 3]	1	1	0
[2, 3, 1]	2	2	0
[3, 2, 1]	1	1	0
[3, 1, 2]	2	2	0
[1, 3, 2]	1	1	0
[3, 2, 1, 4]	1	1	0
[2, 4, 3, 1]	2	2	0
[2, 4, 1, 3]	3	3	0

[1, 3, 4, 2]	2	2	0
[2, 3, 1, 4]	2	2	0
[4, 1, 5, 3, 2]	3	3	0
[1, 5, 2, 4, 3]	2	2	0.00100017
[3, 4, 2, 5, 1]	3	3	0
[2, 5, 4, 3, 1]	2	2	0
[1, 5, 2, 3, 4]	2	2	0
[5, 4, 1, 3, 2, 6]	3	2	0.00109076
[4, 6, 3, 1, 5, 2]	4	4	0.0567541
[5, 6, 4, 3, 1, 2]	3	2	0.000976562
[4, 2, 6, 3, 1, 5]	4	4	0.147376
[6, 4, 1, 3, 5, 2]	4	4	0.0366974

[5, 7, 1, 6, 2, 3, 4]	3	3	0.0144253
[6, 5, 7, 4, 3, 1, 2]	4	3	0.0606337
[2, 4, 5, 7, 3, 6, 1]	5	4	2.80486
[3, 5, 4, 2, 1, 7, 6]	4	3	0.0675828
[7, 5, 2, 1, 6, 3, 4]	3	3	0.00516438
[1, 7, 5, 2, 8, 3, 4, 6]	4	4	0.924185
[7, 2, 6, 8, 3, 1, 4, 5]	5	4	5.89242
[5, 4, 6, 8, 1, 2, 3, 7]	4	3	0.246729
[8, 7, 3, 6, 1, 5, 2, 4]	5	4	7.82728
[7, 6, 5, 3, 4, 1, 2, 8]	3	2	0.0144172

I calculate the lower boundaries with the algorithm in the lecture, using the number of breakpoints.

You can see the brute force algorithm execution time increases exponentially.

3. The code is in Q3.py. I calculate the upper bound of the d(permutation) this way:
    1. Find the breakpoints and strips with respect to them.
    2. Find the minimum number within the descending strips. And if there isn't any descending, just reverse the first ascending that is longer than 1 length and goto 1.
    3. Reverse between the index of minimum_descending-1 and minimum descending then goto 1.

We continue this until there isn't any breakpoint. This is the result:

```
Lower bound = 3
Higher bound = 4
The exact d(pi)= 3
```

Let's find the bounds here manually to check the bounds.

- Lower bound:
  Extend: 0, 4, 3, 6, 5, 1, 8, 7, 2, 9
  Find breakpoints: 0| 4, 3| 6, 5| 1| 8, 7| 2| 9
  So [6/2] = 3.

- Upper bound:

  Extend: 0, 4, 3, 6, 5, 1, 8, 7, 2, 9

  Find breakpoints: 0| 4, 3| 6, 5| 1| 8, 7| 2| 9

  Minimum between descending strips = 1

  After first reverse: 0, 1|5, 6| 3, 4| 8,7| 2| 9

  Minimum between descending strips = 2

  After second reverse: 0, 1, 2|7, 8| 4, 3| 6,5| 9

  Minimum between descending strips = 3

  After third reverse: 0, 1, 2, 3, 4|8, 7, 6, 5| 9

  Minimum between descending strips = 5

  After fourth reverse: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

  So the upper bound is 4 as we can find a solution that can map the permutation to the identity with 4 reversals.