Computational Complexity of Isomorphism

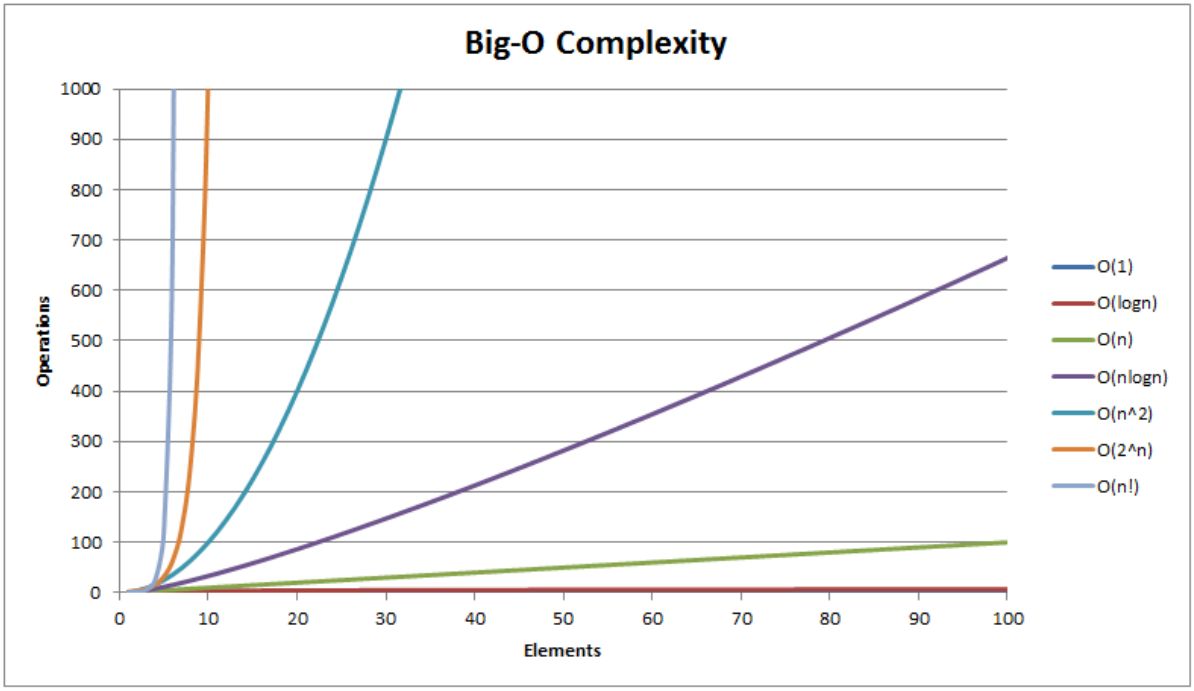
Time/space

The time and space complexity of permutation is 0(n). The reason for this is because there are N possible permutations in a length of N. In the code snippet below this is further explained.

The code shown below demonstrates an adjacency matrix. There are 9 different numbers (space complexity) that can be arranged in 9 possible ways. When the code goes through the array A, it iterates 9 times (time complexity) to find the number. When the number is found, it is taken out of the array and put into the function matrix_print(A) which will print out the numbers in a three-by-three format. This process is repeated until all the numbers are out of array A and in function matrix_print(A). This ties into the reason at the start, there are 9 possible permutations in a length of 9.

This explains why permutation has a space and time complexity of 0(n)

```
In [36]: # Here is a 3x3 version of A.
         A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]], dtype=np.uint8)
         matrix_print(A)

         1 2 3
         4 5 6
         7 8 9

In [37]: # Permute columns using P.
         matrix_print(A @ P)

         2 1 3
         5 4 6
         8 7 9

In [38]: # Permute rows using P.
         matrix_print(P @ A)

         4 5 6
         1 2 3
         7 8 9

In [39]: # P and P.T are equal again.
         np.all(P.T == P)
```



Sources

https://learnersbucket.com/examples/algorithms/program-to-print-all-the-permutation-of-string/

https://adrianmejia.com/most-popular-algorithms-time-complexity-every-programmer-should-know-free-online-tutorial-course/

images

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.hackerearth.com%2Fpractice%2Fnotes%2Fbig-o-cheatsheet-series-data-structures-and-algorithms-with-thier-complexities-1%2F&psig=AOvVaw3VQmeFXhnrkEJC13KdaOOf&ust=1652113402671000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCJC646Co0PcCFQAAAAAdAAAAABAD