Longest Increasing Subsequence of Random Permutation

Author: Xinyu Hu, Ryan Jiau

Introduction

The objective of this project is to build a model that takes samples from random permutations of length n, calculate the average longest increasing sequence of the sample, and estimate the expected length of the longest increasing subsequence for random permutation of length n as a function of n.

Analysis strategy

We decided to compute the longest increasing subsequence using the deterministic dynamic programming algorithm for our sample random permutations.

We used the python built-in function **np.random.permutation()** to get random permutations of length n. At first, we assumed that for large n, the probability of getting the same random permutation each time we sample is small, so we decided to just append the new permutations into a list of samples without checking for duplicate random permutations.

However, we discovered that for small n and big m, we will get a lot of duplicated samples. These duplicated samples will not affect the average LIS, but it will run the algorithm on the same input excessively. Therefore we added the code to check for **duplicate permutations**. After we get the samples, we can run them through our algorithm and get the LIS.

Algorithm description

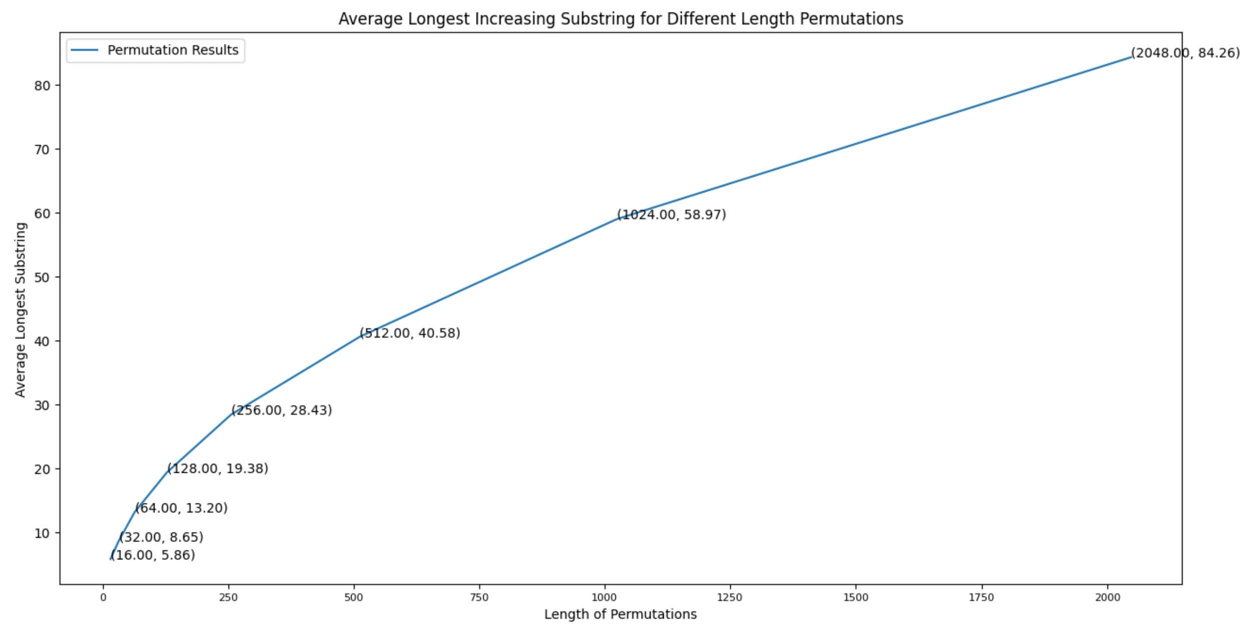
```
Input(p)        p is the random permutation
  1. Declare the list of LIS, max_count[], of length n and initialize all index of
     max_count[] to 1
  2. Compute the optimized LIS values by checking whether there are longer LIS that
     end at every index.
     For i from index 1 to length(p):
         For j from index 0 to i:
             If p[i]>p[j]:
                 max_count[i] = max(max_count[i], max_count[j]+1)
  3. Return max(max_count)
```

The complexity of the algorithm is O(n^2).

Experiment process

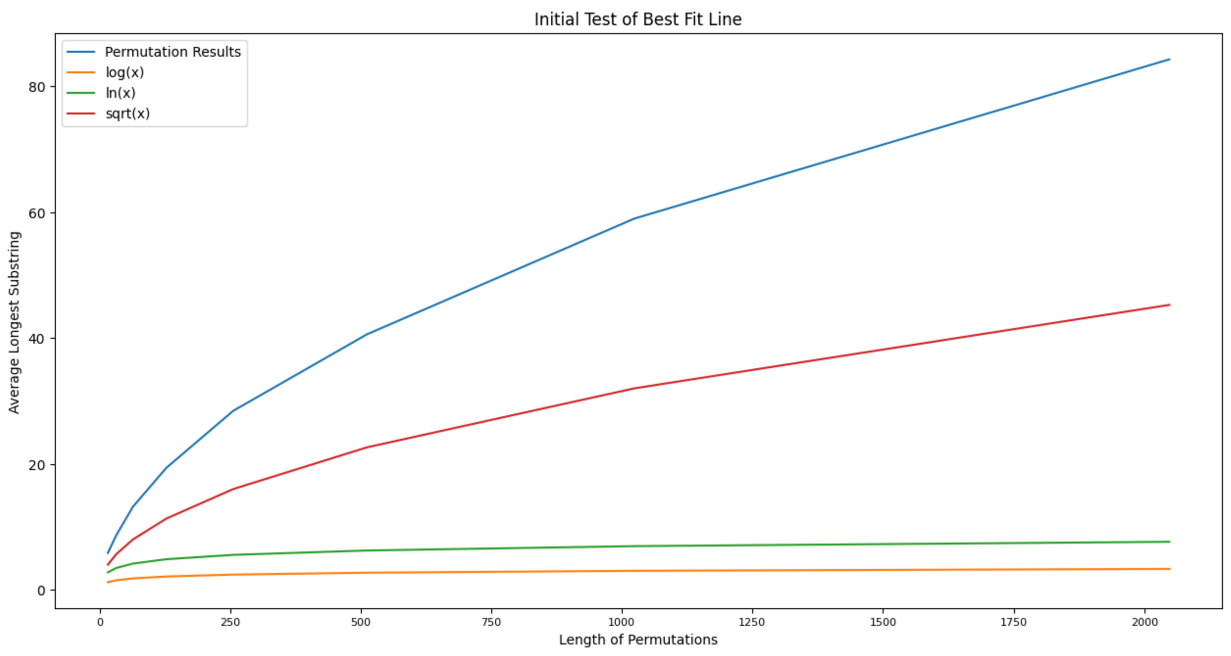
We counted the permutation for various values of n. For our experiment, we decided to use n = 16, 32, 64, 128, 256, 512, 1024, 2048. For each n, we tested m different permutations. In our experiment we set m = 100. For each n, we calculated the LIS for each of the m permutations. Then we took the average LIS for that n. Once we calculated the average LIS for multiple n, we graphed the average LIS against n.

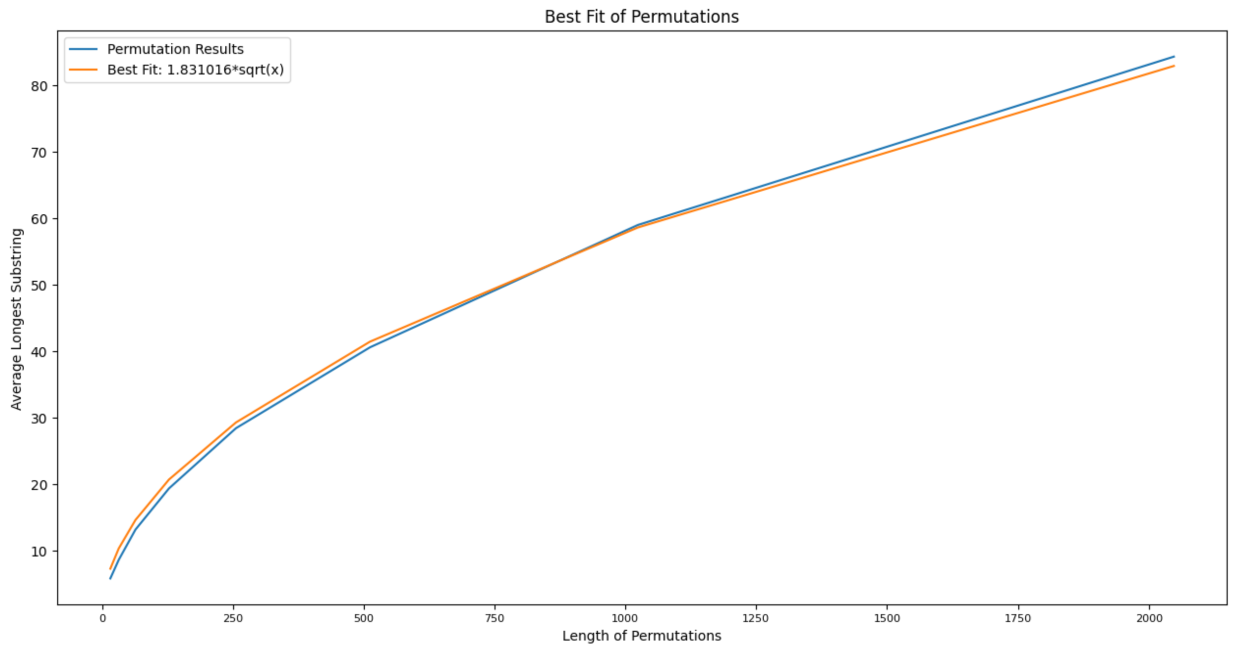
Conclusion



Pattern:

First thought after we see the graph, it looks like some sort of a log(n), ln(n), or $\sqrt{n}$ graph. But we soon realized it couldn’t be log or ln because the graph is increasing much faster than log and ln.Take n=2048 as example, log(n) ~=3.3, ln(n)~=7.6. But looking at **sqrt(n)~=44, and the fact that f(2048)=84.96**, we decided that the best estimate will be **c***$\sqrt{n}$ **for some constant c**. We then used the function **curve_fit** from the python package **scipy** and got c=1.831016.





As we can see, f(x) and y=1.831016*$\sqrt{n}$ are close to each other, and even cross each other around 750~1250.

Accuracy and Precision:

We used Chebyshev's inequality to calculate accuracy and precision. Below is a table for the calculated values for each n we used in our experiment.

The form of Chebyshev's inequality we used is Pr(|x - Ex| ≥ tσ(x)) ≤ $\frac{1}{t^2}$. The standard deviation was calculated using all m=100 LIS. We then found tσ(x) to be |x - Ex|, where x is the average LIS and Ex is the LIS using our best fit curve, y=1.831016*$\sqrt{n}$. From there we can calculate t and plug it into $\frac{1}{t^2}$.

n	t	σ(x)	$\frac{1}{t^2}$
16	1.44208933471225	1.01523744548114	0.4808566983042391
32	1.6446290535080528	1.0384039812533599	0.36971245696924615
64	1.1686977548763025	1.2390938363794468	0.7321424370451576
128	0.6912650612141328	1.93207900794178	2.0927183692815468
256	0.39238181662120775	2.2076771466494787	6.495046324962106
512	0.36983900596229097	2.3014268740426354	7.310962792966095
1024	0.15887595044096903	2.3760377849595646	39.61719076710863
2048	0.48410192719505185	2.8871712065199002	4.267036687448566

From our calculations it shows that as n gets larger, the farther away from the expected LIS the n=100 LIS are. It also shows that for larger n, the average LIS, from our experiment, is closer to the LIS from the best fit curve with n=1024 being the closest.