Iterations of Superpermutations

To what extent is there a method to finding the number of iterations and the contents of minimal superpermutations?

749071

An Extended Essay in Mathematics

Session: May 2022

Word Count: 3370

Table of Contents

Introduction

Research Question: To what extent is there a method to finding the number of iterations and the contents of minimal superpermutations?

In this extended essay, different methods of creating and shortening superpermutations will be explored. The goal would to be able to prove and demonstrate that there is a viable method in determining the number of iterations of minimal superpermutation for any given value of n>7 (subject to change) and additionally the contents of supra mentioned strings. Superpermutations are from the area of combinatorial mathematics and are an extension of normal permutations and combinations. A superpermutation about symbol n is a string that contains all permutations of n. Things get more interesting when overlapping is introduced and strings can be shortened. More overlap results in more efficiency and thus, shorter strings. This extended essay will focus on the shortest string for each value of n (minimal permutations). As n grows larger, the length of superpermutations increases exponentially due to the inherent nature of normal permutations and their property of increasing in factorials. Few real-world applications of superpermutations exist and it's seen as a trivial area of mathematics.

All permutations of n = 3:

123, 132, 213, 231, 321, 312

Superpermutation of n = 3:

123132213231321312

Minimal base superpermutation of n=3:

123121321

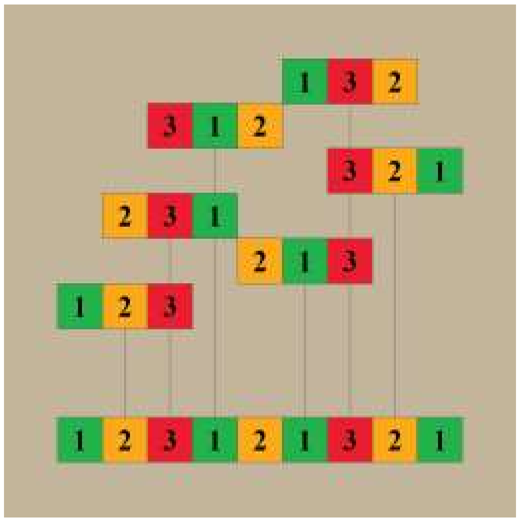


Fig 1. Visualization the minimal base superpermutation of n=3. Created with Adobe Illustrator.

Pre-examination of the Problem

In order to solve this problem. different methods of constructing superpermutations must and will be considered. However, as different methodologies of construction will result in varying patterns in the essence of the created superpermutation. Only one method will be considered in the context of the research question. But even before that, certain terms must be defined. Frames and weight are terms that can describe efficiency when creating superpermutations. Frames are continuous blocks of n characters. E.g., 7 frames of 3 in a string of length 9.

123 -> 231 -> 312 -> 121 -> 213 -> 132 -> 321 = 123121321

Weight is how many characters are wasted when connecting 2 separate permutations in the process of creating a superpermutation. E.g., combining 1234 and 2341 would have a weight of 1 whereas 1234 and 3412 or 3421 would have a weight of 2.

1234 -> 2341 = 12341

1234 -> 3412 = 1234$\underline{12}$

1234 -> 3421 = 1234$\underline{21}$

By evaluating the weight of combining certain permutations, one can find subsequent permutations that may be better or worse. This becomes more apparent and clearer after the first method that was investigated. Throughout this essay, certain superpermutations will be referred to as base or identity. These will simply be superpermutations that begin with the smallest permutation of the given value of n. For example, 121 would be the base superpermutation of n=2 as opposed to 212.

This extended essay will go over more than one method of constructing superpermutations. However, it will ultimately focus on superpermutations constructed by the recursive formula when attempting to answer the research question. This is due to the large amount of randomness and lack of patterns that exist in superpermutations created by Aaron Williams method or brute force. Both factors would make it far harder to answer my research question. Nevertheless, all methods will be briefly mentioned as the latter two aforementioned methods can find shorter and more efficient minimal superpermutations for values of $n \geq 6$, the computational power required to produce them are also exponentially higher.

Approaching the Problem

As the superpermutations for $n = 1$ and 2 were both extremely short and self-explanatory, it would only be logical to begin focusing on this problem with $n = 3$. When examining the base minimal superpermutation of $n = 3$, 1123121321. It is observed that there are 4 weights of 1 and weight of 2. In other words, there is only a single wasted frame, '121'. When attempting to reconstruct the superpermutation from just the base permutations of $n = 3$, a simple method practiced was to add single digits unique to the previous 2 digits unless by doing so would repeat a permutation of $n = 3$. In that situation, increase the digit by 1 and if the digit in question was a 3, wrap it around and count it as 1. Repeat the above steps until all permutations of $n = 3$ have been reached. For example, when beginning with the permutation '123' you would add '1' then '2'. However, after that you would need to change something as it would repeat a previous permutation if the rule was followed for another digit '123123'. Thus, the 6th digit would instead be '1'. Continuing with the original steps, the last three digits of the superpermutation would be '3', '2' and '1'. It ends here as all 6 permutations of $n = 3$ can be found within the superpermutation.

123-1-2-1-3-2-1

213-2-1-2-3-2-1

312-3-1-3-2-1-3

123	132
213	231
312	321

Permutations of n=3

All minimal superpermutations of n=3

123121321	213212312	312313213
132131231	231232132	321323123

The aforementioned method also works with all other minimal superpermutations of n=3. However, the main difference stems from the different permutation that each superpermutation begins with. As there are only 6 different permutations of $n = 3$, only 6 different iterations of the minimal superpermutations of $n = 3$ exist.

When adapting this method for larger values of n, you would simply increase the number of previous digits needed to look back for. More precisely, n-1 digits. For example, when looking for the next digit in constructing a superpermutation for $n = 6$:

123456 (2, 3, 4, 5, 6)

It would be obvious that the following digit should be a 1, as all other digits between 1-6 are present within the last 5 $(n - 1)$ digits.

This method would work in theory for creating any superpermutation but in practice would be unviable to use for values of $n > 3$ as the need to constantly be checking whether or not the previous string of n digits would be a repeated permutation would cost far too much time. Additionally, permutations create with this method would only be considered minimal superpermutations prior to n=6 as with the introduction of the far more complex method developed by Aaron Williams. See section ‘Aaron William’s Method’.

The Second Approach

A far more reliable method for constructing superpermutations would be what was come across when experimenting with permutations of $n = 4$. Rather than focus on adding single digits onto the end of a string of integers, permutations were grouped and categorized based on their respective weights to other permutations. The first permutation was "1234" and simply connected subsequential permutations with as much overlap as possible, starting with n-1 digits of overlap. For example, "1234" would be followed by "2341" and continued by "3412".

123412 (12 digits of permutations compressed into 6 digits!)

After being unable to add further permutations with $n - 1$ overlapping digits, $n - 2$ and subsequently $n - 3$. However, it would reset back and attempt $n - 1$ after each step. Causes for not finding a suitable permutation is most likely caused by the required permutation already being used. For example, the continuation of the string of permutations above would lead to "4123".

1234 -> 2341 -> 3412 -> 4123 (combined to be 1234123)

The most optimal permutation to follow would be "1234". However, it has already been used and so we must choose between "2314" and "2341". As the latter has also already been used, "2314" is the only choice. The continuation of this method would eventually result in the construction of a full superpermutation. My raw work is shown on the next page.

There are 24 permutations of n = 4.

1234	1243	1324	1342	1423	1432
2134	2143	2314	2341	2413	2431
3124	3142	3214	3241	3412	3421
4123	4132	4213	4231	4312	4321

Small arrows signify weight-1

1234 -> 2341 -> 3412 -> 4123 -> ~~1234~~

↓ weight-2

2314 -> 3142 -> 1423 -> 4231 -> ~~2314~~

~~2341 -> 3412 -> 4123~~

↓ weight-2

3124 -> 1243 -> 2431 -> 4312

↓ weight-3

2134 -> 1342 -> 3421 -> 4213

↓ weight-2

1324 -> 3241 -> 2413-> 4132

↓ weight-2

3214 -> 2143 -> 1432 -> 4321

Final superpermutation: 123412314231243121342132413214321

1234123 | 14231 | 24312 | 134213 | 24132 | 14321

Length: 33

Weights: 18 of 1, 4 of 2, 1 of 3

It was around this time in my extended essay process that I decided to program a script in Python 3 that allowed me to quickly check whether a text document of countless superpermutation of any length and any value of n was indeed a valid superpermutation of the given value of n. The script works by firstly splitting the lines of the text document into separate values and it will proceed to ask for a value of n and then generate all permutations of previously mentioned n. Lastly, it will check whether each permutation can be found within the potential superpermutations and either add it to a list of valid superpermutations or invalid ones. Thus, the program will be able to confirm valid superpermutations consistently; however, it will not exclusively seek minimal superpermutations. After going through all the potential superpermutations, it will return all the valid ones in the python shell. Nevertheless, the raw program is attached in the references as both proof and for confirmation that it is original work. Additionally, all infra mentioned superpermutations will be confirmed with this

Immediately after confirming that the superpermutation constructed was valid, I wondered if similar results could be achieved when starting with a different number. The successful construction of a superpermutation with 33 digits. As an extension to the question, I asked to what extent does it matter for which permutation leads? I asked these questions because my initial decision to start with the permutation “1234” was simply because it made the most logical sense, being both the smallest permutation and easiest to work with.

Continuation of the Second Approach

After further investigation and testing. The starting permutation was chosen to be “3214”. Once again, chosen somewhat by chance. The results shared many similarities.

Final superpermutation: 321432134213241323142312431234123

3214321 | 34213 | 24132 | 314231 | 24312 | 34123

Length: 33

A similar process was gone through just like the first approach. However, it was far easier as only the premade groups had to be matched together and change a few digits to make sure there were no redundant or recurring numbers. Thus, the starting permutation does not matter only to a certain extent. The extent being that the first permutation must be the first of a string of consecutive permutations with the weight of 1. E.g., groups of 4 permutations in my worked example above. If the starting permutation is not the leading permutation of a string of consecutive permutations with the weight of 1, there would need to be wasted characters and frames somewhere within the superpermutation to accommodate for the missed permutation(s). Making the end result a superpermutation with a length of more than 33 digits and thus not a minimal superpermutation. In conclusion, there are 6 different variants of minimal superpermutations for n=4, each starting with the leading permutation of a string of consecutive permutations all with the weight of 1. I would like to note that both superpermutations mentioned above have been double-checked by my program to confirm that they are indeed valid superpermutations for $n = 4$. Additionally, the other 4 minimal superpermutations of $n = 4$ that were not mentioned in this section above have also been double-checked.

Regarding the first part of my research question, my next question would be regarding how many iterations are possible for any superpermutation created with the recursive formula. Not quite the minimal iterations the research question is ultimately searching for, but this is a good steppingstone. So far, it can be proved from this method that there should be at least 6 unique variations of length 33 as there are 6 unique strings of 4 permutations with connecting weights of 1.

The Recursive Formula

A more formal name for this method would be the recursive formula and its method is well displayed in the diagram below. [1]

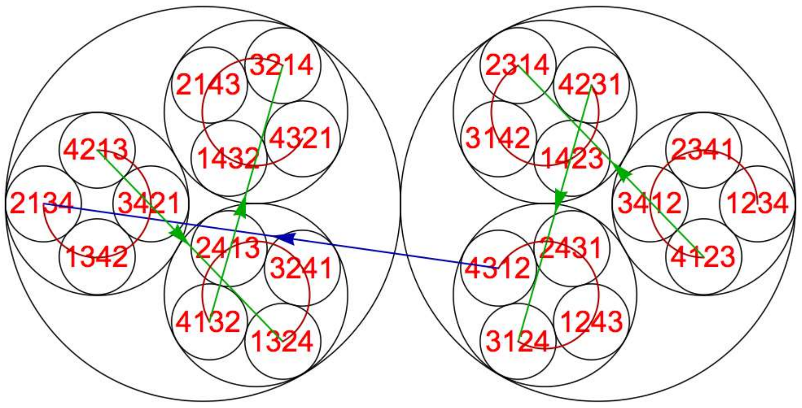


Fig 2. Egan, Greg. Superpermutations as Paths through a graph. [1]

Red is for paths with weights of 1, green for 2 and blue for 3.

[1] Egan, Greg. *Superpermutations*, 20 Oct. 2018, www.gregegan.net/SCIENCE/Superpermutations/Superpermutations.html.

Shorter superpermutations are created by reducing the amount of wasted characters. When using the recursive formula, with each increment of n, previous paths have their weight increase by 1. From this, we can see that, though the recursive formula is a reliable, albeit rather slow method of creating superpermutations, it loses effectiveness in creating the minimal superpermutations the larger the n gets. E.g., a superpermutation of 10 would have weights as high as 9, which is extremely ineffective when creating superpermutations with the goal of short lengths. See table 3 for additional details. The recursive formula has been proven to begin losing effectiveness at $n \geq 3$, with a one-off method beating out the original 783 digits by one. See table 2. However, through other more consistent methods, the difference in superpermutation length only increases as n gets larger.

$$\sum_{k=1}^{n} k!$$

$$k^2(k-1)!, for\ 1 \leq k \geq n-1$$

An alternate method of constructing minimum superpermutations would be the method developed by Aaron Williams. As was mentioned before, this next method is indeed able to construct minimal superpermutations smaller than ones created with the recursive formula after $n = 6$. However, the methodology is completely different and due to difficulties in comparing minimal superpermutations with two completely separate methods of construction, Aaron William’s method will not be considered in the context of the scope of the research question.

Alternate Methods

Aaron William's Method

Aaron Williams is a postdoctoral researcher at McGill University with a PhD in computer science and he was able to come up with a method of constructing superpermutations with lower lengths than ones created by the recursive formula.[2] His method is based on Hamiltonian paths, which is a path between an array of nodes that only visits each vertex exactly once.

Williams utilized this by creating an array of nodes with permutations and finding many cycles of routes with only weights 1 and 2. He would later combine different cycles until he had a string of integers with only one of each permutation. This method of creating superpermutations of shorter lengths relative to ones formed with the recursive formula gains effectiveness as n becomes increasingly larger. This is because, superpermutations created with the recursive formula will always have weights in relation to the value of n as there is an exponential relationship between the number of permutations sharing a weight in n in relation to $n + 1$.

For example, a superpermutation of $n = 5$ would only have 5 total weights that are higher than 2. Whereas one of $n = 8$ would have 719 weights above 2. See graph 3 for more details.

Currently, a script written by Greg Egan which is adapted from Aaron William's work is only able to construct superpermutations of shorter lengths than ones created from the recursive formula starting from $n = 3$ and onwards.

[2] Egan, Greg. *Superpermutations*, 20 Oct. 2018, www.gregegan.net/SCIENCE/Superpermutations/Superpermutations.html.

Thus, for the simplicity of focusing on the research question of searching for variations of minimal superpermutations, only those of $n \geq 5$ will be considered. As anything larger will either nearly impossible to compare due to the different methods of constructions or they will no longer be a minimal superpermutation if made with the recursive formula.

Ben Chaffin's Method

In March of 2014, Chaffin was able to discover 8 unique minimal superpermutations of $n = 5$ through brute force and searching for strings of all 120 permutations of $n = 5$ while wasting the least number of digits.[3] The found minimal superpermutations are the same length as ones created by the recursive formula but all of them begin with the identity permutation (12345).

Final Approach

After observing the variants of minimal superpermutations of $n = 3$ and 4, there are 6 unique variants in both cases. However, the limiting factors are different for the two values of n. For minimal superpermutations of $n = 3$, the maximum number of variants is equivalent unique permutations of 3. Whereas the limiting factor for superpermutations is the same as the number of permutation strings that are held together exclusively by weight one edges. Additionally, there are only two variants of minimal superpermutations of $n = 2$ and a single variant for $n = 1$. See data table 1. For my final approach, investigation was done on minimal superpermutations of $n = 5$, which has the length of 153 digits.

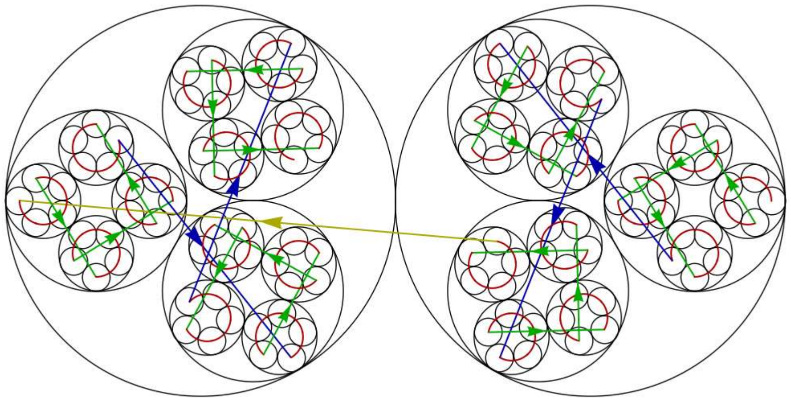
[3] Johnston, Nathaniel. "All Minimal Superpermutations on Five Symbols Have Been Found." *Nathaniel Johnston*, 22 Aug. 2014, www.njohnston.ca/2014/08/all-minimal-superpermutations-on-five-symbols-have-been-found/.

Fig 2. Egan, Greg.

Superpermutations as Paths

through a graph. [4]

Red	Weight-1
Green	Weight-2
Blue	Weight-3
Yellow	Weight-4



The minimal base superpermutation for $n = 5$ is listed below:

123451234152341253412354123145231425314235142315423124531243512431524312543121

345213425134215342135421324513241532413524132541321453214352143251432154321

This is the same superpermutation but divided at weights 2, 3 and 4:

123451234 152341 253412 354123

1452314 253142 351423 154231

2453124 351243 152431 254312

13452134 251342 153421 354213

2451324 153241 352413 254132

1453214 352143 251432 154321

Vertical spacing is weight-2

Single horizontal spacing is weight-3

Double horizontal spacing is weight-4

See Data table 3 for more information on weights.

[4] Egan, Greg. *Superpermutations*, 20 Oct. 2018, www.gregegan.net/SCIENCE/Superpermutations/Superpermutations.html.

A parallel between the formatted superpermutation and Figure 3 can also be drawn to increase understanding between both mediums. There are many similarities between this and the minimal superpermutations in the second approach. The minimal superpermutations for $n = 4$ all had 18 edges of weight-1, 4 of weight-2 and 1 of weight-3. Those numbers have been bumped up a weight and now $n = 5$ has 96 edges of weight-1. This shared property of having 18 edges of a weight allow there to be 6 unique variants of the minimal superpermutation of $n = 5$.

Conclusion

In conclusion, the limiting factors change between different values of n. However, after getting past the anomalies of $n = 1$ and 2 a recognizable pattern has been able to develop. $n = 3$, 4 and 5 all have six unique variants of minimal superpermutations constructed in the style of the recursive formula. Disregarding the fact that $n = 3$ is capped at 6 variations because of the fact that there are only 6 permutations of 3. $n = 4$ and 5 both are capped by the strings of low-weight permutations, as the minimal superpermutation of $n = 4$ has 18 permutations of weight-1 edges and likewise, $n = 5$ has 18 permutations of weight-2 edges. As there will always be a group of weighted edges with 18 permutations in 6 groups of 4, all minimal superpermutations created with the recursive formula and with a value of $n \leq 5$ will have 6 unique variants. Evidence for the aforementioned claims and conclusions can be found in the second and final approaches of this extended essay. Through the displaying of superpermutations in the segmented method that is unique to this extended essay, the pattern can be easily recognized within the predetermined ranges of n. Most notably with in the examples used in this extended essay of $n = 4$ and $n = 5$.

Data tables

Normally, superpermutations with doubled digits (e.g., 1221) would still be considered valid.

Length is based on the recursive formula so shorter versions will exist at $n \geq 6$.

Table 1.

n	String(s) generated from the recursive formula	Variants	Length[5]
0		0	0
1	1	1	1
2	121 or 212	2	3
3	123121321	6	9
4	123412314231243121342132413214321	6	33
5	123451234152341253412354123145231425314235142 315423124531243512431524312543121345213425134 215342135421324513241532413524132541321453214 352143251432154321	8 Source[6]	153
6	123456123451623451263451236451234651234156234 152634152364152346152341652341256341253641253 461253416253412653412356412354612354162354126 354123654123145623145263145236145231645231465 231425631425361425316425314625314265314235614 235164235146235142635142365142315642315462315	?	873

	426315423615423165423124563124536124531624531 264531246531243561243516243512643512463512436 512431562431526431524631524361524316524312564 312546312543612543162543126543121345621345261 345216345213645213465213425613425163425136425 134625134265134215634215364215346215342615342 165342135642135462135426135421635421365421324 561324516324513624513264513246513241563241536 241532641532461532416532413562413526413524613 524163524136524132564132546132541632541362541 326541321456321453621453261453216453214653214 356214352614352164352146352143652143256143251 643251463251436251432651432156432154632154362 154326154321654321		
7		?	5913
8		?	46 233
9		?	409 113
10		?	4 037 913
11		?	43 954 713
12		?	522 956 313

13		?	6 749 977 113
14			93 928 268 313
15			1 401 602 636 313

[5]Only based off the recursive formula

[6]http://www.njohnston.ca/superperm5.txt

Table 2.

n	Shortest found superpermutations	Variants	Length
6	12345612345162345126345123645132645136245136425136452136451234651234156234152634152364152346152341652341256341253641253461253416253412653412356412354612354162354126354123654132654312645316243516243156243165243162543162453164253146253142653142563142536142531645231465231456231452631452361452316453216453126435126431526431256432156423154623154263154236154231654231564213564215362415362145362154362153462135462134562134652134625134621536421563421653421635421634521634251634215643251643256143256413256431265432165432615342613542613452613425613 4	1	872

	265134261532465132465312463512463152463125463 215463251463254163254613254631245632145632415 632451632456132456312465321465324165324615326 415326145326154326514362514365214356214352614 352164352146352143651243615243612543612453612 435612436514235614235164235146235142635142365 143265413625413652413562413526413524613524163 52413654213654123		
7	Here[7]	"Few dozen" (Egan)	5906
8	Here[8]	?	46 205

[7]http://www.gregegan.net/SCIENCE/Superpermutations/7_5906_nsk666646664466646666_2SYMM_FS.txt

[8]http://www.gregegan.net/SCIENCE/Superpermutations/8_46205.txt

Weight = how many new characters are needed to add a new permutation. E.g., combining 1234 and 2341 would have a weight of 1 whereas 1234 and 3412 or 3421 would have a weight of 2.

Number of weights for values of n, 1-10. Higher amounts of lower weight are more efficient for creating minimal superpermutations. Empty box is equivalent to zero.

Table 3.

	1	2	3	4	5	6	7	8	9
1									
2	1								
3	4	1							
4	18	4	1						
5	96	18	4	1					
6	600	96	18	4	1				
7	4320	600	96	18	4	1			
8	35280	4320	600	96	18	4	1		
9	322560	35280	4320	600	96	18	4	1	
10	3265920	322560	35280	4320	600	96	18	4	1

a^2 * (a-1)! to calculate weight for n-a edges

Works Consulted

Berry, Nick. *De Bruijn Sequences*, 2 Oct. 2013, datagenetics.com/blog/october22013/index.html.

Egan, Greg. *Superpermutations*, 20 Oct. 2018, www.gregegan.net/SCIENCE/Superpermutations/Superpermutations.html.

Griggs, Mary Beth. "An Anonymous 4chan Post Could Help Solve a 25-Year-Old Math Mystery." *The Verge*, The Verge, 24 Oct. 2018, www.theverge.com/2018/10/24/18019464/4chan-anon-anime-haruhi-math-mystery.

Grime, James, director. *Superpermutations - Numberphile*. *YouTube*, YouTube, 29 Jan. 2018, www.youtube.com/watch?v=wJGE4aEWc28.

Honner, Patrick. "Unscrambling the Hidden Secrets of Superpermutations." *Quanta Magazine*, 16 Jan. 2019, www.quantamagazine.org/unscrambling-the-hidden-secrets-of-superpermutations-20190116/.

Johnston, Nathaniel. "All Minimal Superpermutations on Five Symbols Have Been Found." *Nathaniel Johnston*, 22 Aug. 2014, www.njohnston.ca/2014/08/all-minimal-superpermutations-on-five-symbols-have-been-found/.

Johnston, Nathaniel. "The Minimal Superpermutation Problem." *Nathaniel Johnston*, 22 Aug. 2014, www.njohnston.ca/2013/04/the-minimal-superpermutation-problem/.

Klarreich, Erica. "Computer Scientists Break Traveling Salesperson Record." *Quanta Magazine*, 8 Oct. 2020, www.quantamagazine.org/computer-scientists-break-traveling-salesperson-record-20201008/.

Parker, Matt. “Superpermutations: The Math Problem Solved by 4chan.” *YouTube*, YouTube, 28 Jan. 2019, www.youtube.com/watch?v=OZzIvl1tbPo&t=3s.

Sarkar, Sohail. “Superpermutations.” *Medium*, Betamat, 30 Oct. 2020, medium.com/betamat-en/superpermutations-f7378eebcaf4.

Python Program

I created this to check whether a given string of numbers is a superpermutation for any given value of n.

```
import itertools


n = [int(input("n = "))]

nList = [j+1 for i in n for j in range(i)]

rawList = list(itertools.permutations(nList))

#print(rawList)

#n = []

#superPermutationList = []

yes = []

no = []


dataFile = (open("data.txt", "r"))

data = dataFile.read()
```

```
superPermutationList = data.split()


permutationList = []

for i in rawList:

    a = [str(b) for b in i]

    c = "".join(a)

    d = int(c)

    #print(d)

    permutationList.append(d)


e = len(permutationList)

for superPermutation in superPermutationList:

    f = 0

    for permutation in permutationList:

        if str(permutation) in str(superPermutation):

            f += 1

            if f == e:

                print("Found a superpermutation, " + str(superPermutation) + "!")

                yes.append(superPermutation)

        else:

            break

    no.append(superPermutation)
```

```
for element in no:
    if element in yes:
        no.remove(element)
#print("Yes: " + str(yes))
#print("No: " + str(no))
```