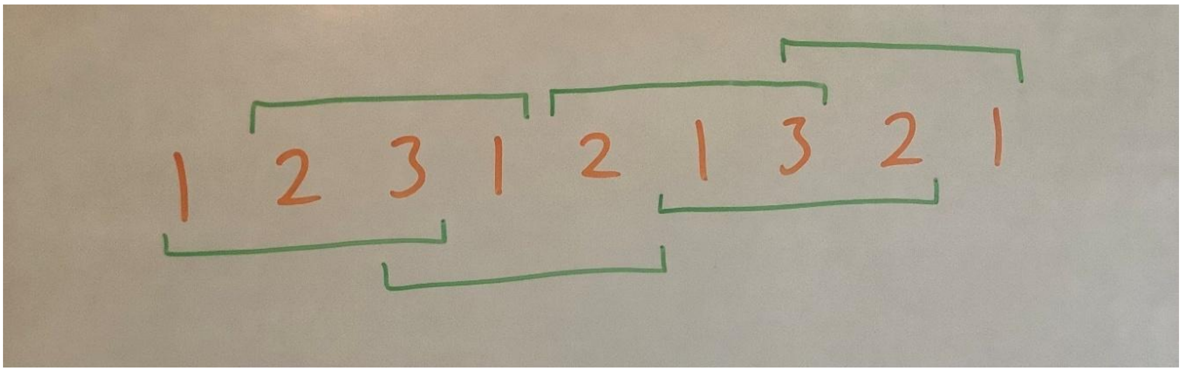
**Superpoems: rhyme time**

There are many different forms of poetry; most are at least partially defined by their rhyme scheme. Taking similar letters to denote lines that rhyme and different letters for lines that don't, ABAB CDCD EFEF GG is the romantic sonnet, while AABBA is the jaunty limerick. These 14 and 5 line poems contain both rhyme schemes possible in a 2 line poem – AA and AB. But both fall shy of every 3 line rhyme scheme: the sonnet has no AAB, the limerick has no ABA and neither contain AAA. Can we create a poem containing every rhyme scheme of a given length?

Obviously yes – we could just list all rhyme schemes of length n (n-rhymes) and write a poem using one after another. But this is very inefficient. The number of n-rhymes grows very quickly so the question becomes, what are the shortest such poems?

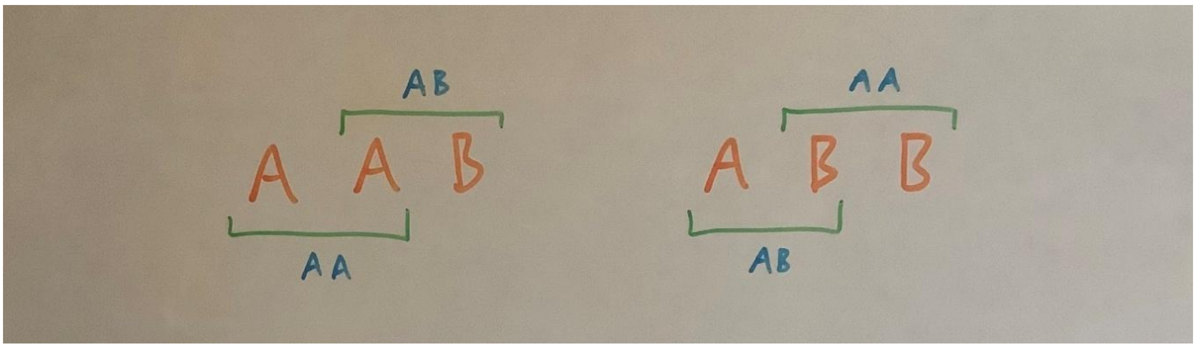
Superpermutations

This is akin to the **superpermutation**: a string that contains each permutation of n symbols. For example, 123121321 is a superpermutation as it contains all permutations of 3 symbols, namely 123,132, 213, 231, 312, 321 (check for yourself!) and this cannot be achieved this in less than the 9 symbols used. Thus, our poems will be termed Superpoems.




*The overlaps of permutations of 3 symbols in a superpermutation*

Superpermutations achieve their minimal lengths through overlaps so maybe Superpoems can do the same. To make a poem including AA and AB, overlap the middle A to create AAB. Actually, ABB works too as BB signifies the same thing as AA: 2 consecutive lines that rhyme with each other. It is easy to see that we cannot contain AA and AB in less than 3 symbols as 2 symbols is only enough to express AA or AB but not both.



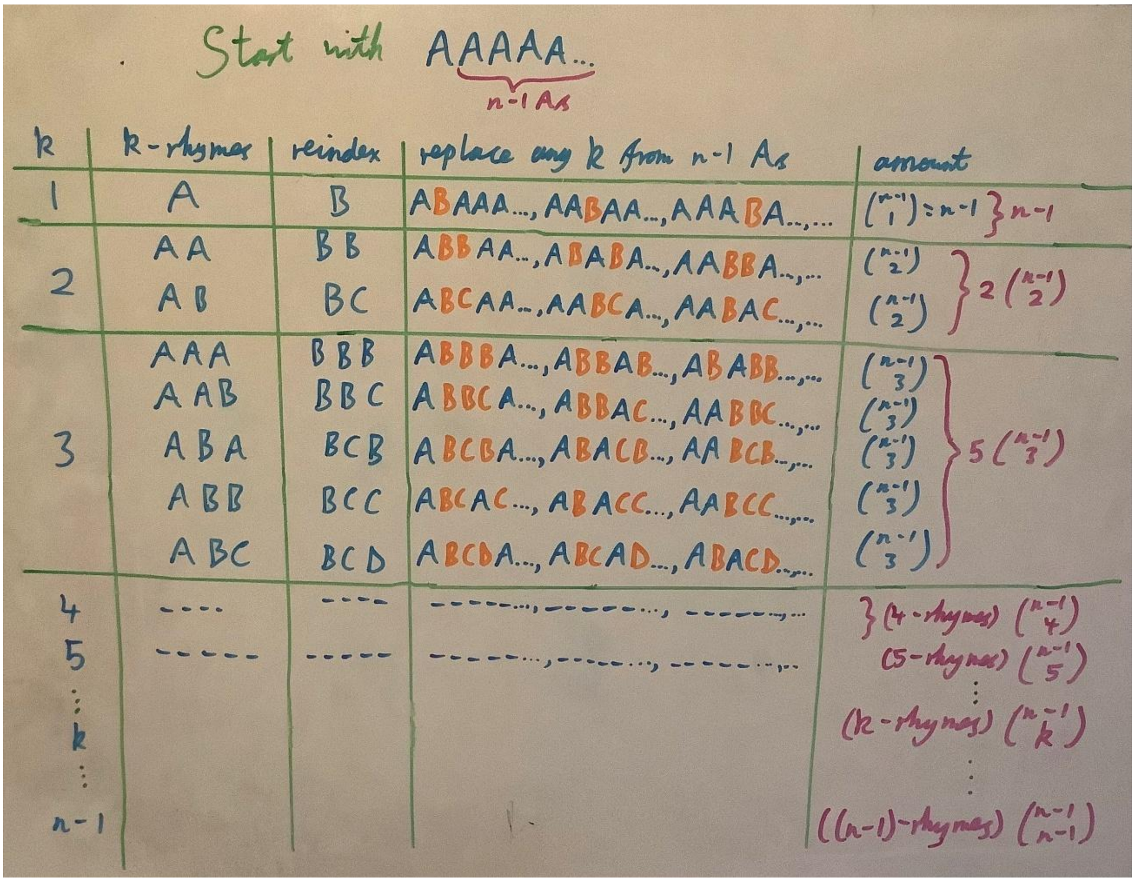

*The overlaps of the 2-rhymes within the 2-Superpoems*

Likewise, there is a hard lower limit on n-Superpoems for n>2, but to find it we will need to know how many n-rhymes there are in general.

## How many can be found?

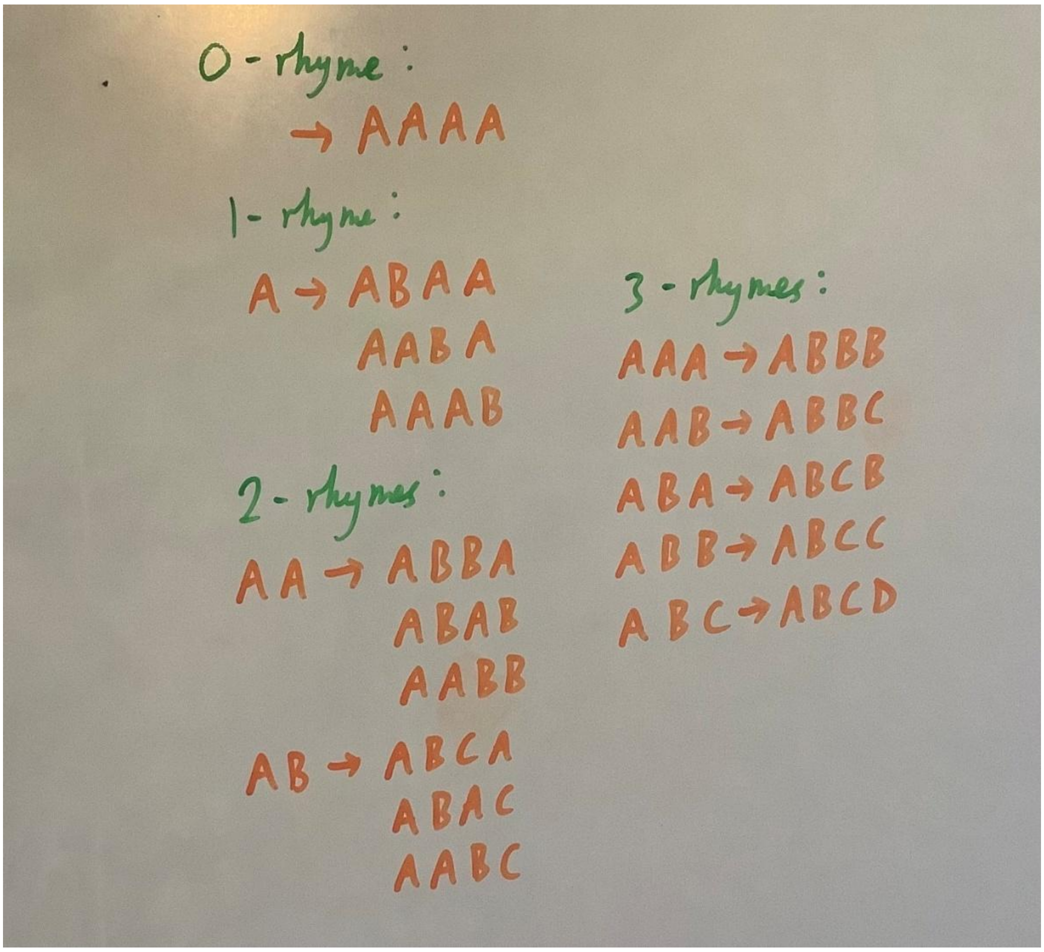
We have seen that there are 2 2-rhymes: AA and AB. Going back to 1-rhymes, it is clear there is only A. Adding some As, Bs and a C to the 2-rhymes, we can find that there are 5 3-rhymes: AAA, AAB, ABA, ABB and ABC. Finding more manually gets tiring, so how can we find the general case of n-rhymes? In other words, how many ways are there to fill n places to give a valid rhyme scheme?

First, fill all n places with As. Rhyme schemes always begin with A as convention so we now have n-1 places to do whatever we want with. Beginning with our singular 1-rhyme, we are going to replace one of the filler As. We will need to reindex our 1-rhyme from A to a B, but that's okay. There are n-1 places we can replace. Now move to our 2-rhymes (AA and AB). After reindexing, we want to choose 2 of the n-1 filler As to replace. How many ways can we do that? Thankfully, we don't need to worry: there's a function for it! The choose function (with notation $\binom{n}{k}$) is defined as the number of ways to pick k objects out of n. So there are $\binom{n-1}{2}$ ways to replace our filler As for any of our 2-rhymes. Similar, there are $\binom{n-1}{3}$ ways to replace our filler As with any of our 3-rhymes.




*A table showing why there is $\binom{n-1}{k}$ of each k-rhyme in the construction of n-rhymes*

So on and so on, until we eventually get to (n-1)-rhymes. Since we are replacing all n-1 filler As, there is only one way to do this – which is consistent with $\binom{n-1}{n-1}$ as it equals 1 for any n (not a coincidence!). Summing these up will get us every rhyme scheme apart from the string of just As – but if we say there is one 0-rhyme of " ", it is included.




*Constructing all 4-rhymes from all k-rhymes with k<4*

Thus, ↓ (this symbol means 'sum for k=0,1,2,3...up to k=n-1')

$$n - rhymes = \sum_{k=0}^{n-1} \binom{n-1}{k} (k - rhymes), with\ 0 - rhymes = 1$$

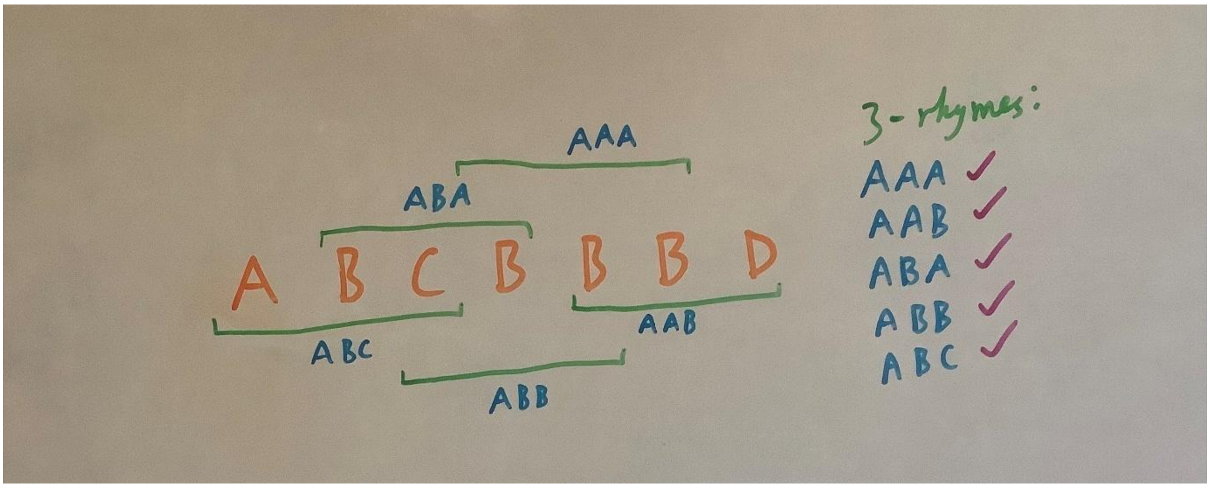
In the case of n=4 as in the picture above, 4-rhymes=(1)(1)+(3)(1)+(3)(2)+(1)(5)=15.

This gives us the sequence 1, 1, 2, 5, 15, 52, 203, 877, 4140....These are normally called **Bell numbers** and crop up in all sorts of situations, from card shuffling to classic Japanese literature to juggling notation! The bad news is that they grow faster than exponential - but this makes our quest for the shortest Superpoem all the more crucial. Now we know the number of n-rhymes, we can find the hard lower limit on the length of possible Superpoems.

<u>The lower bound</u>

If the shortest possible Superpoem exists, it needs to overlap every rhyme scheme as much as possible. This means that every new character needs to introduce a new rhyme scheme to the poem. Imagine we have a list of all n-rhymes and we want to form such a Superpoem. We need n characters for the first n-rhyme. Then, for ever rhyme after the first, only 1 extra character is needed to be added as every new character introduces a new rhyme scheme. Since we have (n-rhymes)-1 left, our lower limit will be (n-rhymes)+n-1. But can such Superpoems exist?

For n=2, our limit is 2+2-1=3. As we have seen, AAB/ABB fulfil this nicely. For n=3, our limit is 5+3-1=7. One such Superpoem is ABCBBBD but it turns out there are many distinct 3-Superpoems. Try to find another!

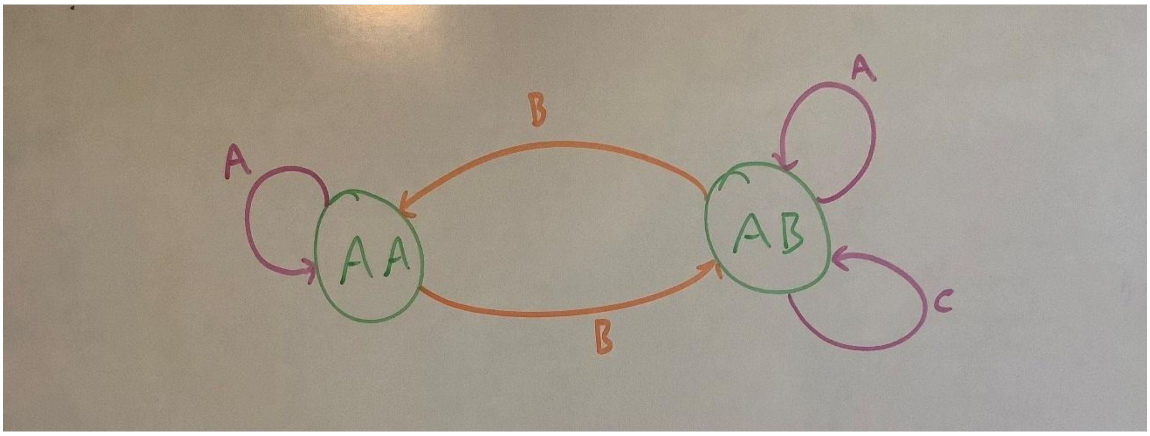



*The overlaps of all 3-rhymes within a 3-Superpoem*

Our lower limits for n-Superpoems is as follows: 0, 1, 3, 7, 18, 56, 208, 883, 4147....Much better than the strategy of listing rhyme schemes which would have led to a length of n-rhymes multiplied by n: 0, 1, 4, 15, 60, 260, 1218, 6139, 33120...! Now we know what to aim for, we can begin our quest to find some...

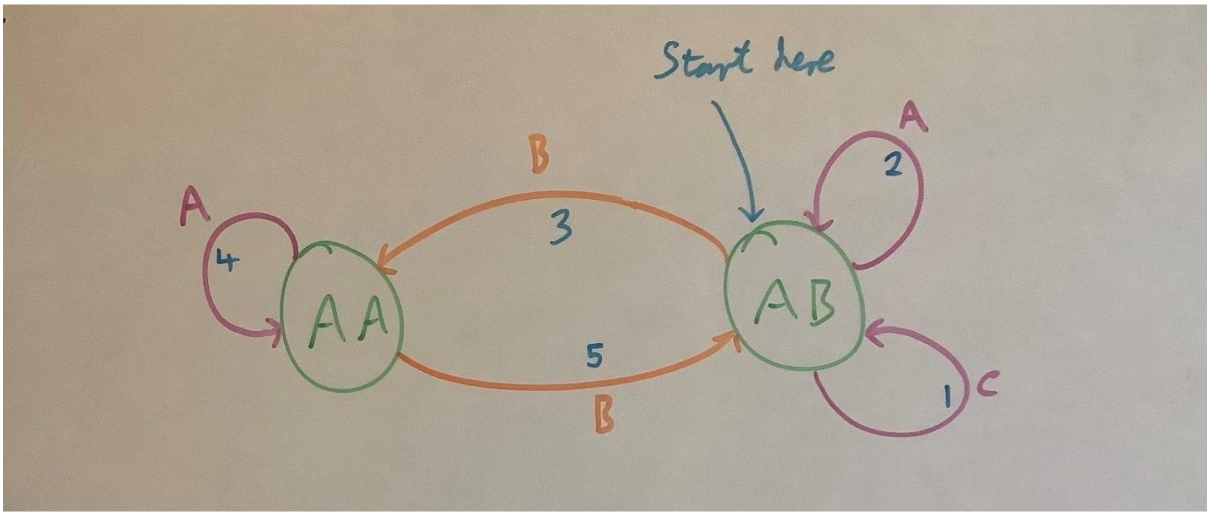
<u>Arrows whirl around</u>

If you tried finding a 3-Superpoem, you will have found that the choices in the characters you write down are crucial. Some choices force you to repeat a rhyme while others allow a Superpoem to be formed. To map out these choices, we can put the overlapped characters into **nodes**. The overlapped characters will always be of length n-1 so we will need (n-1)-rhymes nodes. Now we can draw **arrows** between the nodes to find what rhymes occur with each possible choice. Since these represent every combination of an (n-1)-rhyme plus another character, there will be n-rhymes arrows.



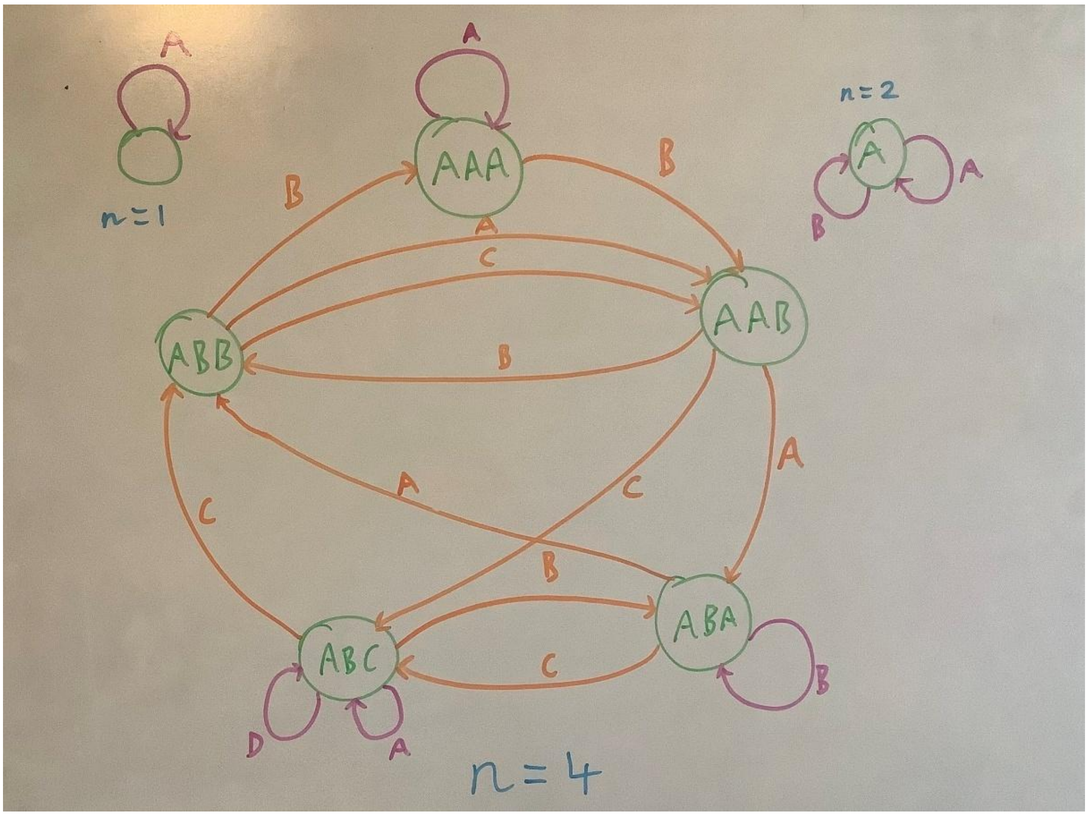

*The graph for n=3, showing which rhymes are obtained through each character added*

This is called a **directed graph** and has some interesting features. There are arrows that point to the node they came from, like ABC because it starts and ends with the overlapped AB (BC is the same as AB as both mean 2 consecutive lines that don't rhyme). There are arrows that point to different nodes like ABB because it begins with AB and ends with AA. Most importantly, we can construct a path that travels along each edge exactly once. Since each edge corresponds to a new added letter, and each letter corresponds to a new added rhyme, this path (called an **Eulerian path**) will correspond to a Superpoem!




*The Eulerian path (numbered) corresponding to the ABCBBBD Superpoem*

We can take this graph approach down to n=2 (or even n=1) and up to n=4. After that, the number of nodes and arrows would become unwieldy – remember there are (n-1)-rhymes nodes and n-rhymes arrows! Try to find a 4-Superpoem using the graph! Cross out each edge once you've used it. Just try not to get stuck at a node without a way out.



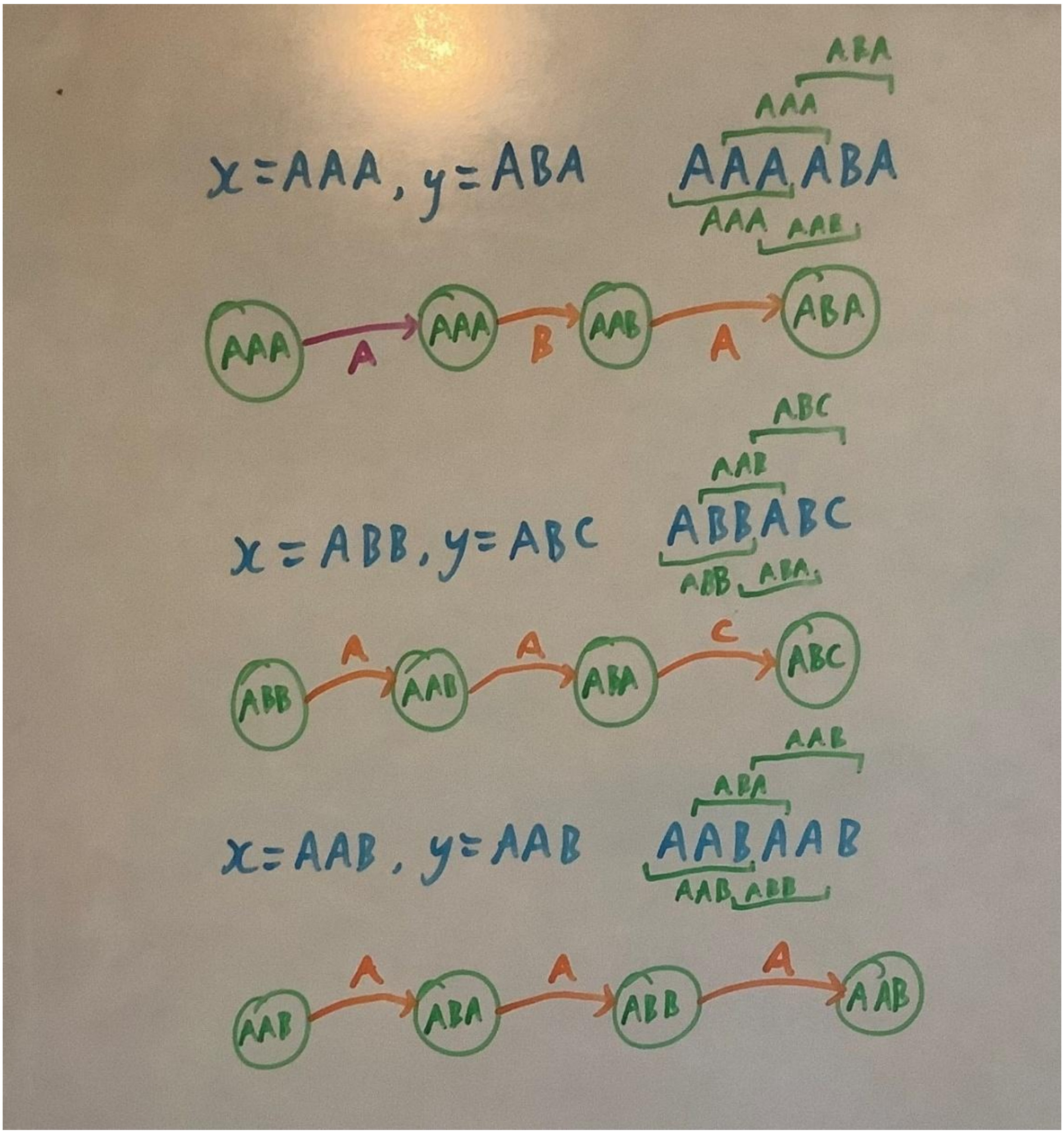

*The graphs for n=1,2 and 4. There are 2 arrows from ABB to AAB, can you see why?*

Though the graphs for n>4 may not be feasible to draw, we know they exist and can deduce certain properties they must exhibit. But which properties do we need for an Eulerian path to exist?

<u>All nodes included?</u>

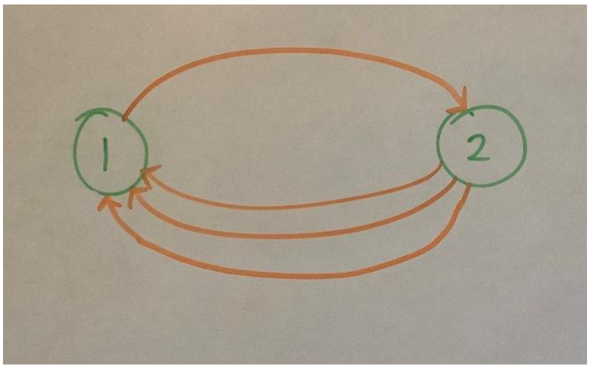
Obviously, we could not have an Eulerian path if any node has no arrows connected to it. So we require all nodes to have a path to or from any other node. Will this be true in our graphs?

Yes. Remember our nodes are all the (n-1)-rhymes so we want to get from any (n-1) rhyme to any other (n-1)-rhyme. Let's call our starting (n-1)-rhyme x and our ending (n-1)-rhyme y. Begin with x, then write y after it. It is that simple. In terms of the graph, we begin at the x node. Then we take the arrow corresponding to the first character of y. Then we take the arrow corresponding to the second character of y. Repeat until we have written y, meaning we must have reached the y node.




*Examples of going from a 3-rhyme to another 3-rhyme with the corresponding arrows and nodes.*

We have arrived at y purely by following arrows on our graph and thus there is a path from any (n-1)-rhyme x to any other (n-1)-rhyme y! Note that this may not be the most efficient path from x to y but it doesn't have to be, and it is guaranteed to always work! But this is not quite enough. Consider the following graph:

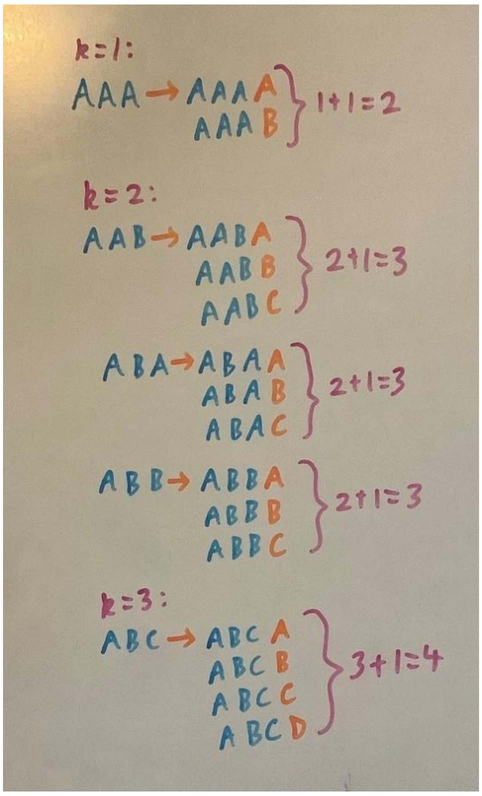



*A graph that has all connected nodes. But no Eulerian path.*

All nodes are connected but there is no Eulerian path; node 2 has too many outgoing arrows and not enough incoming arrows. Thus, we require that the number of incoming and outgoing arrows should be equal for all nodes. Check for yourself this is true for our graphs with n=1,2,3,4. Why is this true and is it true for all n?

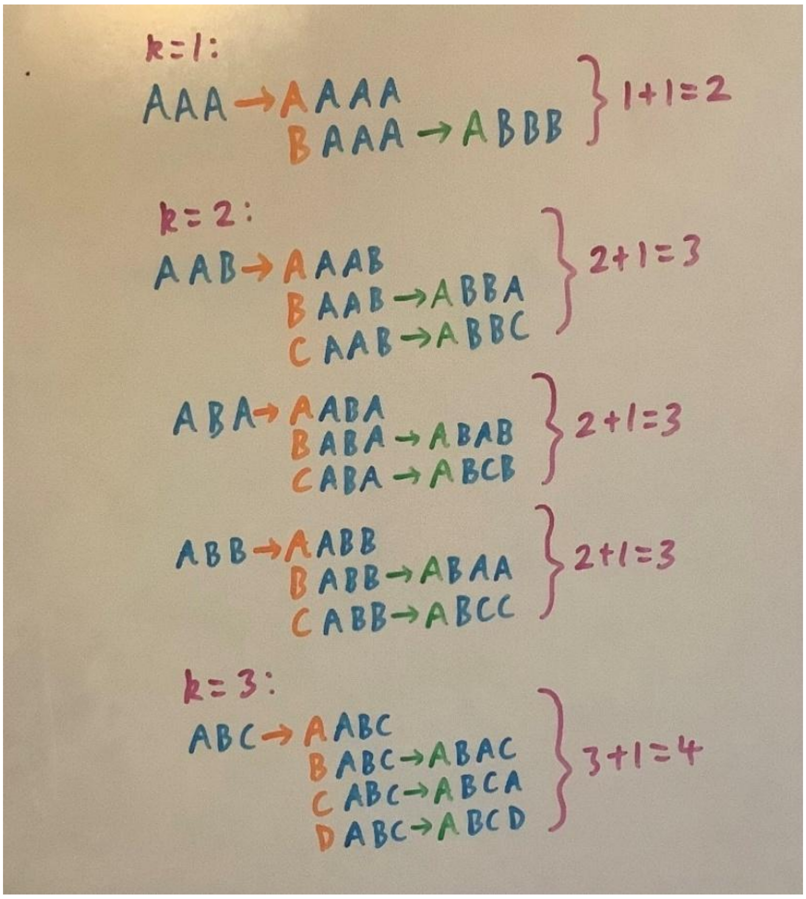
<u>The justifications</u>

Consider how many outgoing arrows there are for any node. This is the same as asking how many different characters we can put after the rhyme to get a valid rhyme. Let k be the number of different characters used in the rhyme. K=1 for any string of As, k=2 for any string of just As and Bs and so on. Then, we can put k+1 different characters on the end, ranging from A,B,C...to the last letter used (alphabetically) and the next one after that. Anything else would be equivalent to one of those. Thus, we have k+1 outgoing arrows.




*Illustrating k+1 outgoing arrows for n=4*

Now consider how many incoming arrows there are. This is the same as asking how many characters can precede a rhyme to get a valid rhyme. Using the same logic as before, we can cycle through A,B,C…to the last alphabetical letter used, plus the one after that. Thus – after reindexing most of them – we get k+1 incoming arrows!




*Illustrating k+1 incoming arrows for n=4*

Now we know all nodes are connected and the number of incoming arrows = the number of outgoing arrows for all nodes. It turns out these 2 conditions are sufficient to conclude that an Eulerian path always exists. Thus, Superpoems always exist to contain every n-rhyme!

<u>Everything concluded</u>

So a Superpoem always exists. But how can we find them with their rapidly increasing length? How else could this be concluded but with a couple of Superpoems? Here's 2: one about Superpoems themselves and the other a compilation of the subheadings you've been reading!

**<u>Superpoem</u>**

Made to enthral,
This Superpoem,
Though small,
All at one time,
Contains every rhyme,
Of a length that's prime.
Can you find them all?

**<u>Subheadings</u>**

Superpermutations
How many can be found?
The lower bound
Arrows whirl around
All nodes included?
The justifications
Everything concluded