We start from a network with small N_0 (for example 10), everything is fully connected (we could also start with an ER).

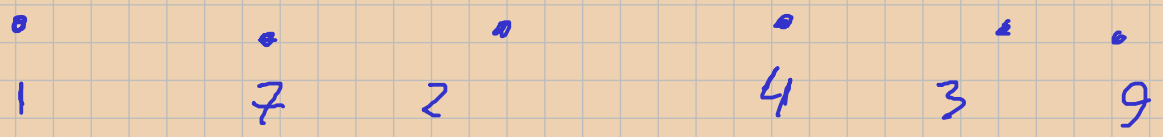
We define how many nodes we want in the end, NN.

At every step, we need to know the total link number L at that time, and the degree of the node candidates to link.

When introducing each node, we want to make m connections, for example m=5.

(we can calculate the k of the candidates at the start of the process to get m links or each time for every m connection, for small m shouldn't make a big difference)

$p_i = \frac{k_i}{2L}$ this is the probability to link the new node to the i node.



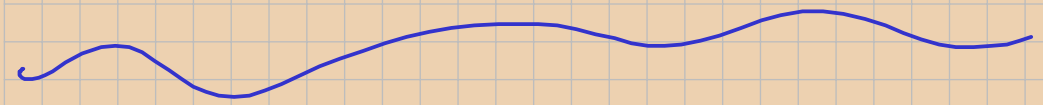
those are 6 candidate nodes, with their degree under them

We can make an array with 1 time the 1st, 7 times the 2nd, 2 times the 3rd, 4 times the 4th, 3 times the 5th, 9 times the 6th.
We then extract a random number from that list, and that is the node that we link it with!

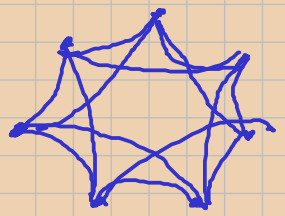
After extracting for example the 4th node, we remove every istance of 4 from that array, and we extract a 2nd node to connect from that list, we repeat this m times to form m link.

Of course we won't have 6 nodes, but all the nodes present in the network at each step when we want to add a new node with m connections.

I think the efficiency of this code is in how we compute the value of k for each candidate node at each step (a step is when we add a new node, and make m connections)



For watts-strogatz:
We start from a ring regular network, we can construct it in multiple ways



K=4
N=7

We have to be careful, we want k even