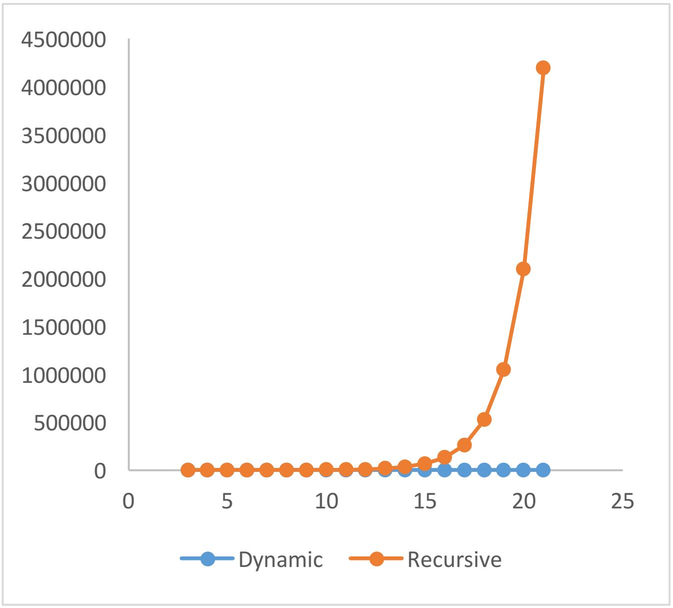
Number of Method Calls per Hanoi solve() Implementation

k (# of rings)	Dynamic	Recursive
3	12	15
4	15	31
5	18	63
6	21	127
7	24	255
8	27	511
9	30	1023
10	33	2047
11	36	4095
12	39	8191
13	42	16383
14	45	32767
15	48	65535
16	51	131071
17	54	262143
18	57	524287
19	60	1048575
20	63	2097151
21	66	4194303



Pattern Between K and the Number of the Method Calls

Assuming k, the number of rings, is at least 3:

Recursive Implementation

Exponential increase of the number of method calls by factor of 2 (plus an additional 1) for every additional ring k OR **f(k) = 2^(k+1)-1**.

This pattern is due to the fact that for every movement of a ring to its destination tower, the method has to call itself to determine the sub-movements that are needed to accomplish that movement. These sub-movements may have sub-movements of their own, resulting in an exponential increase in method calls to execute the movement.

Dynamic Implementation

Linear increase of 3 method calls for every additional ring k OR **f(k) = 3*k + 3**.

This pattern is due to the fact that the implementation saves each movement of a ring from a starting tower to a destination tower, storing them in a hash map so that fetching the movements requires O(1) complexity. Therefore, only a linear increase in method calls is needed to calculate the movements of an additional ring or, more simply, only the additional number of movements that only relate to the additional ring need to be calculated.