華中科技大學

研究生课程报告

姓　　名	赖家晨
学　　号	M202474131
系、年级	计算机科技与技术系 2024 级
类　　别	课程报告
报告科目	人工智能

2025 年 1 月 3 日

1　算法步骤

在梵塔问题中，我们可以用递归的思想去解决。具体来说，如图 1.1所示，给定梵塔问题有三个柱子，1 号柱上放置 n 个从小到大的圆盘，我们需要将 1 号柱上的所有圆盘移动到 3 号柱上，要求小圆盘必须要在大圆盘上且在移动过程中也要保证该要求。

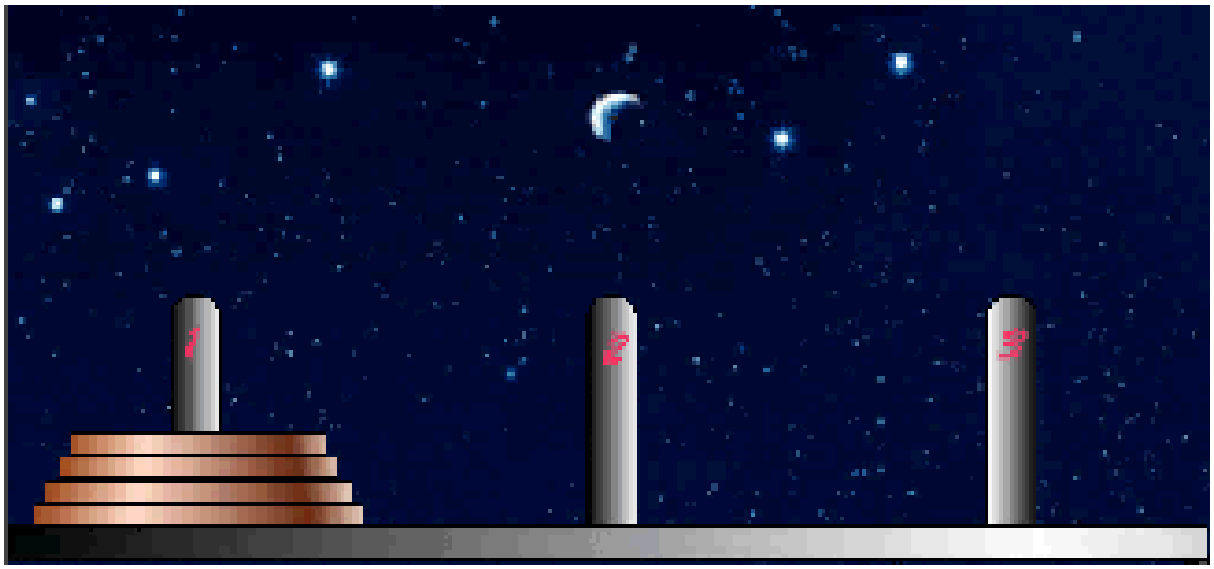


图 1.1 梵塔

在梵塔问题中，我们必须要先将 1 号柱上的前 n-1 个圆盘先放置在 2 号柱上，再将 1 号柱上的第 n 个圆盘移动到 3 号柱上。整个梵塔问题就是上述过程不断重复，直到 1 号柱上第 1 个圆盘移动到 3 号柱上。于是，n 个圆盘的梵塔问题就分解成①将 n-1 个圆盘从起始柱移动到中间柱上；②将第 n 个圆盘从起始柱移动到目标柱上；③将 n-1 个圆盘从中间柱移动到目标柱。因此，n 个圆盘的梵塔问题的圆盘移动次数等于两倍的 n-1 个圆盘的梵塔问题的圆盘移动次数加上 1（第 n 个圆盘的移动次数）。设 n 阶梵塔问题移动圆盘的次数为 $h(n)$, 有

$$h(n) = 2h(n-1) + 1 \tag{1.1}$$

考虑 1 阶梵塔问题移动圆盘次数 $h(1)$ 为 1，根据递推公式式 1.1有，n 阶梵塔问题移动圆盘的次数：

$$h(n) = 2^n - 1 \tag{1.2}$$

2 代码实现

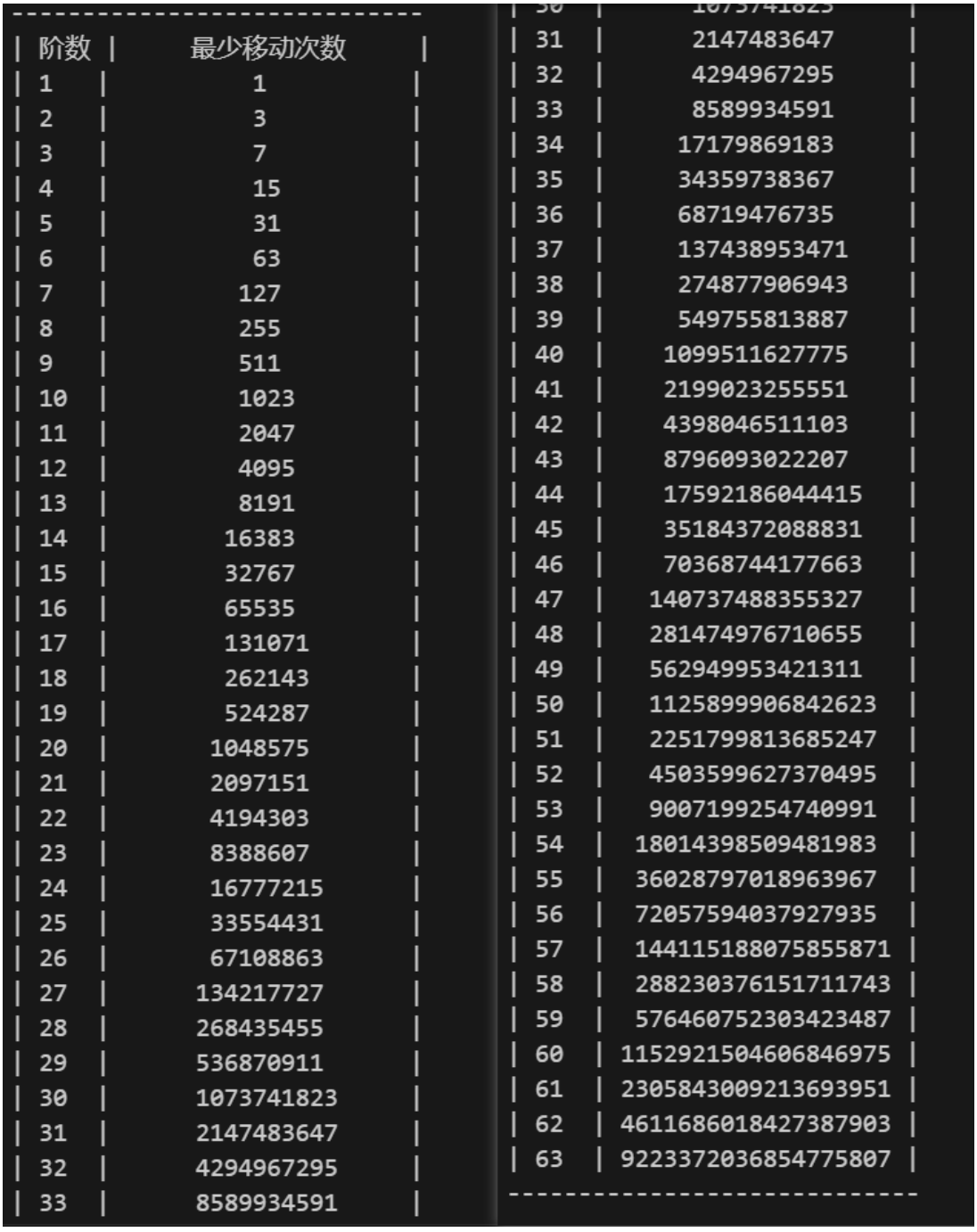
依照式 1.2得出的公式，可以很轻松的实现相关代码。代码实现如下：

```cpp
#include <iostream>
#include <cstdint>
#include <sstream>

using namespace std;

int main() {
	cout << "----------------------------" << endl;
	cout << "| 阶数 |      最少移动次数      |" << endl;
	for(int i = 1; i < 64; ++i) {
		printf("| %-2d  | ", i);
		string str = to_string(((uint64_t)1 << i) - 1);
		int size = (20 - str.size()) / 2;
		int mod = (20 - str.size()) % 2;
		string str1(size, ' '), str2(size + mod, ' ');
		cout << str1 << str << str2 << "|" << endl;
	}
	cout << "----------------------------" << endl;
}
```

输出结果如图 2.1所示



阶数	最少移动次数
1	1
2	3
3	7
4	15
5	31
6	63
7	127
8	255
9	511
10	1023
11	2047
12	4095
13	8191
14	16383
15	32767
16	65535
17	131071
18	262143
19	524287
20	1048575
21	2097151
22	4194303
23	8388607
24	16777215
25	33554431
26	67108863
27	134217727
28	268435455
29	536870911
30	1073741823
31	2147483647
32	4294967295
33	8589934591
34	17179869183
35	34359738367
36	68719476735
37	137438953471
38	274877906943
39	549755813887
40	1099511627775
41	2199023255551
42	4398046511103
43	8796093022207
44	17592186044415
45	35184372088831
46	70368744177663
47	140737488355327
48	281474976710655
49	562949953421311
50	1125899906842623
51	2251799813685247
52	4503599627370495
53	9007199254740991
54	18014398509481983
55	36028797018963967
56	72057594037927935
57	144115188075855871
58	288230376151711743
59	576460752303423487
60	1152921504606846975
61	2305843009213693951
62	4611686018427387903
63	9223372036854775807

图 **2.1** 输出结果