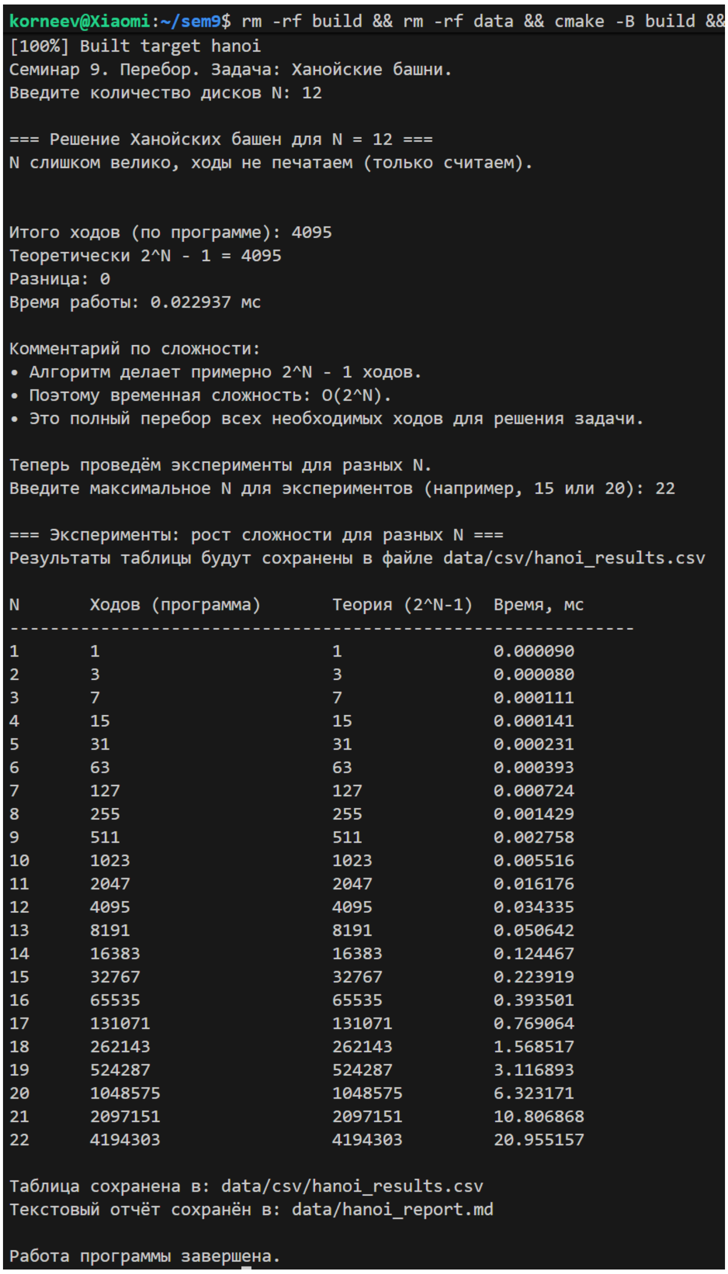
Цели работы:

- Реализовать задачу Ханойских башен методом полного перебора.
- Определить и подтвердить вычислительную сложность алгоритма.
- Исследовать поведение алгоритма на разных N и определить практический предел выполнения.
- Обосновать оптимальность алгоритма и сравнить с альтернативами.

Практическая реализация: Практическая часть работы заключалась в разработке программного решения задачи Ханойских башен методом полного перебора с последующим экспериментальным анализом вычислительной сложности. Для реализации использовался язык программирования C++**.**

Вывод терминала VS Code представлен на рисунке 1



```
korneev@Xiaomi:~/sem9$ rm -rf build && rm -rf data && cmake -B build &&
[100%] Built target hanoi
Семинар 9. Перебор. Задача: Ханойские башни.
Введите количество дисков N: 12

=== Решение Ханойских башен для N = 12 ===
N слишком велико, ходы не печатаем (только считаем).


Итого ходов (по программе): 4095
Теоретически 2^N - 1 = 4095
Разница: 0
Время работы: 0.022937 мс

Комментарий по сложности:
• Алгоритм делает примерно 2^N - 1 ходов.
• Поэтому временная сложность: O(2^N).
• Это полный перебор всех необходимых ходов для решения задачи.

Теперь проведём эксперименты для разных N.
Введите максимальное N для экспериментов (например, 15 или 20): 22

=== Эксперименты: рост сложности для разных N ===
Результаты таблицы будут сохранены в файле data/csv/hanoi_results.csv

N       Ходов (программа)       Теория (2^N-1)  Время, мс
--------------------------------------------------------------
1       1                       1               0.000090
2       3                       3               0.000080
3       7                       7               0.000111
4       15                      15              0.000141
5       31                      31              0.000231
6       63                      63              0.000393
7       127                     127             0.000724
8       255                     255             0.001429
9       511                     511             0.002758
10      1023                    1023            0.005516
11      2047                    2047            0.016176
12      4095                    4095            0.034335
13      8191                    8191            0.050642
14      16383                   16383           0.124467
15      32767                   32767           0.223919
16      65535                   65535           0.393501
17      131071                  131071          0.769064
18      262143                  262143          1.568517
19      524287                  524287          3.116893
20      1048575                 1048575         6.323171
21      2097151                 2097151         10.806868
22      4194303                 4194303         20.955157

Таблица сохранена в: data/csv/hanoi_results.csv
Текстовый отчёт сохранён в: data/hanoi_report.md

Работа программы завершена.
```

Рисунок 1 – Вывод терминала VS Code

Программа собирает данные расчетов в csv таблицу представленную на рисунке 2.

hanoi_results.csv

data > csv > hanoi_results.csv

N	Ходов (программа)	Теория (2^N-1)	Время
1	1	1	0
2	3	3	0
3	7	7	0
4	15	15	0
5	31	31	0
6	63	63	0
7	127	127	0
8	255	255	0
9	511	511	0
10	1023	1023	0.01
11	2047	2047	0.02
12	4095	4095	0.03
13	8191	8191	0.05
14	16383	16383	0.12
15	32767	32767	0.22
16	65535	65535	0.39
17	131071	131071	0.77
18	262143	262143	1.57
19	524287	524287	3.12
20	1048575	1048575	6.32
21	2097151	2097151	10.81
22	4194303	4194303	20.96

Рисунок 2 – Таблица hanoi_results.csv

По данной таблице с помощью Python-скрипта строится график роста числа ходов и времени выполнения. График представлен на рисунке 3.

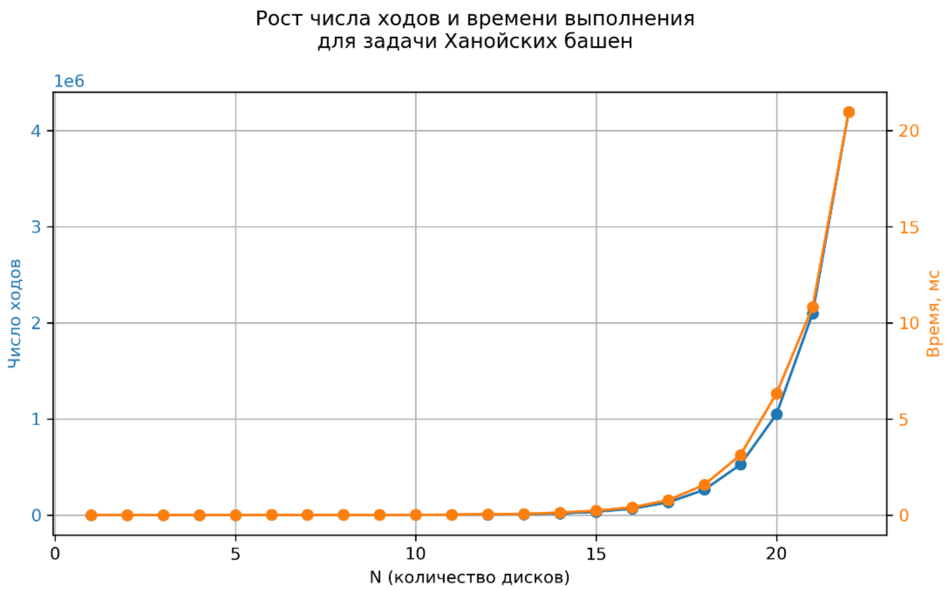


Рисунок 3 – График роста числа ходов и времени выполнения.

На рисунке 4 представлены сгенерированные графики зависимости времени от количества дисков (N) с помощью Python скрипта.

Рисунок 4 – Графики времени выполнения алгоритма.

На рисунке 5 представлена визуализация увеличения времени алгоритма при увеличении вводимого числа N

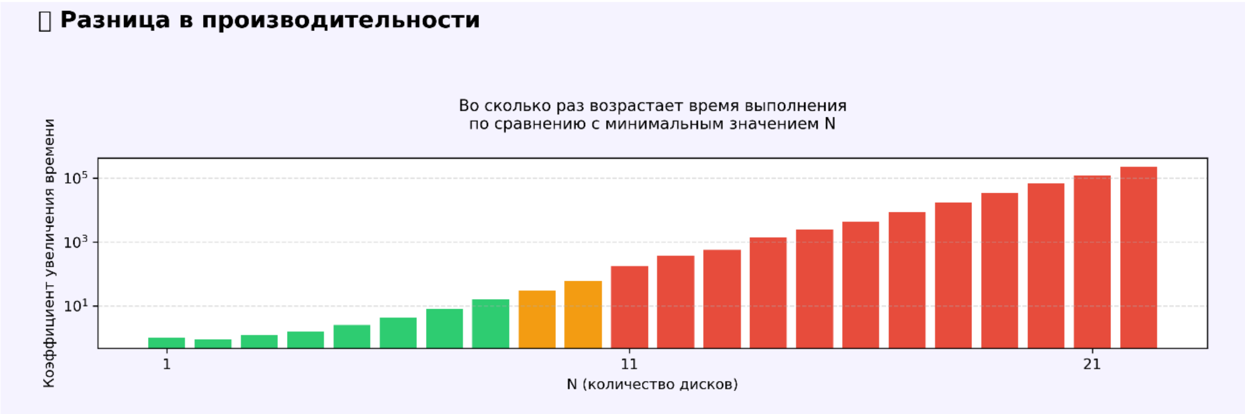


Рисунок 5 — График роста коэффициента увеличения времени выполнения алгоритма Ханойских башен при увеличении числа дисков N.

Сравнение алгоритмов и выбор лучшего

Для задачи Ханойских башен минимальное количество ходов строго равно

$$2^N - 1$$

и является **нижней теоретической границей**. Ни один алгоритм не может выполнить меньше перемещений, поэтому любой корректный метод будет иметь экспоненциальную сложность $O(2^N)$

Классический рекурсивный алгоритм уже достигает этой границы и выполняет оптимальную последовательность ходов. Улучшить асимптотику невозможно — можно сократить лишь технические накладные расходы, но не порядок роста.

Вывод: Полный перебор для Ханойских башен является одновременно и единственно возможным оптимальным алгоритмом. Его сложность совпадает с теоретическим минимумом, поэтому более быстрый метод в принципе невозможен.

Итоговый вывод: Эксперименты подтвердили, что задача Ханойских башен имеет неизбежную экспоненциальную сложность $O(2^N)$, а минимальное число ходов строго равно 2^N-1. Рост времени выполнения на практике полностью совпал с теоретическим прогнозом, что видно по построенным графикам. Полный перебор является оптимальным и единственно возможным алгоритмом, так как улучшить порядок сложности невозможно.