ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 4

«Задача о покрытии множеств»

Выполнил работу

Феоктистов Виктор

Академическая группа C3100

Принято

Ментор Вершинин Владислав Константинович

Санкт-Петербург

2024

1. Введение

Цель: реализовать задачу о покрытии множеств, комбинаторным способом, не превышая сложность O(2^N).

Задачи:

1. Реализовать теоретический алгоритм решения задачи по функциям;
2. Разработать теоретический алгоритм функций;
3. Изучить необходимые типы данных и алгоритмы для реализации в коде;
4. Реализовать алгоритм на языке c++
5. Разработать тесты;
6. Выполнить отладку ошибок;
7. Теоретическая подготовка

Для более понятной работы программы и более удобного взаимодействия с функциями, генерация масок должна быть реализована внутри класса. Для проверки правильности выполнения программы, нужны лишь уникальные значения внутри контейнера set. Ошибки выводятся с помощью библиотеки stdexcept, а для нахождения минимальной метрики количества множеств для покрытия, нужно большое число для сравнения в библиотеке limits. Подсчет времени реализован с помощью библиотеки chrono.

1. Реализация

В первую очередь был использован класс комбинаторного перебора масок ProductIterator(приложение A). Функция next дает следующую комбинацию маски, current\_combination генерирует текущую комбинацию индексов, increment\_indicies изменяет индексы. После класса были реализованы вспомогательные функции (приложение A). invertToVec инвертирует множество векторов в единый объект set без дубликатов. Функция isCover проверят покрыло множество векторов множество юниверса, с помощью предыдущей функции. Само задание реализовано в функции cover(приложение A). По текущей комбинации маски в функции next получаем набор нулей и единиц, если на определенной позиции в маске стоит единица, то во временное множество векторов добавляется вектор, чья позиция во входном векторе соответствует позиции единицы в маске.

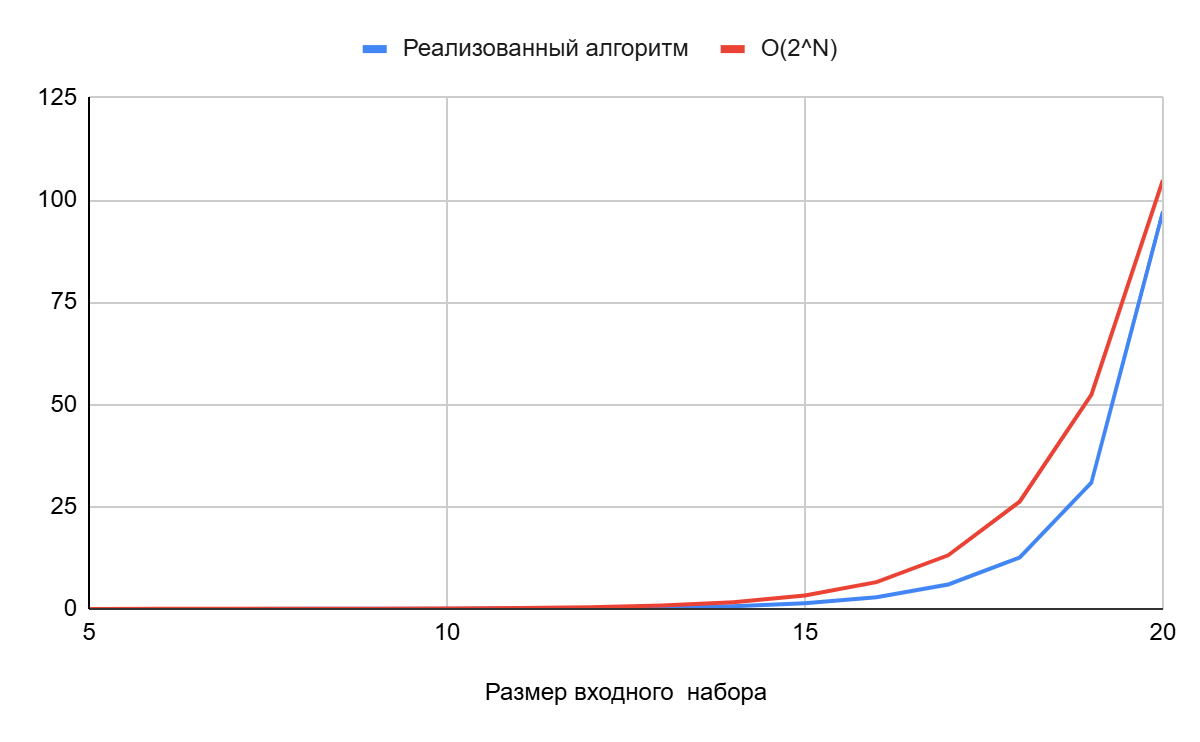
1. Экспериментальная часть

Теоретическая асимптотика: O(2^N), память: 6(n\*4) + 21 байт = n\*24 + 21 байт (подсчет приведен в приложении). Теоретически заданная сложность задачи составляет O(2^N) и более. Для тестирования алгоритма была собрана статистика, приведенная в таблице № 1.

Таблица №1 - Подсчёт сложности реализованного алгоритма

| Размер входного набора | Время, с. | O(2^N) |
| --- | --- | --- |
| 5 | 0,0006 | 0,0032 |
| 6 | 0,001 | 0,0064 |
| 7 | 0,003 | 0,0128 |
| 8 | 0,01 | 0,0256 |
| 9 | 0,02 | 0,0512 |
| 10 | 0,04 | 0,1024 |
| 11 | 0,07 | 0,2048 |
| 12 | 0,16 | 0,4096 |
| 13 | 0,31 | 0,8192 |
| 14 | 0,66 | 1,6384 |
| 15 | 1,40 | 3,2768 |
| 16 | 2,85 | 6,5536 |
| 17 | 5,96 | 13,1072 |
| 18 | 12,54 | 26,2144 |
| 19 | 30,85 | 52,4288 |
| 20 | 97,19 | 104,8576 |

График представляющий визуально удобный формат данных из таблицы №1 представлен на изображении №1.



Изображение №1 - График работы алгоритма

График реализованного алгоритма близок к графику нотации O(2^N). Следовательно, теоретический подсчет асимптотики был верный.

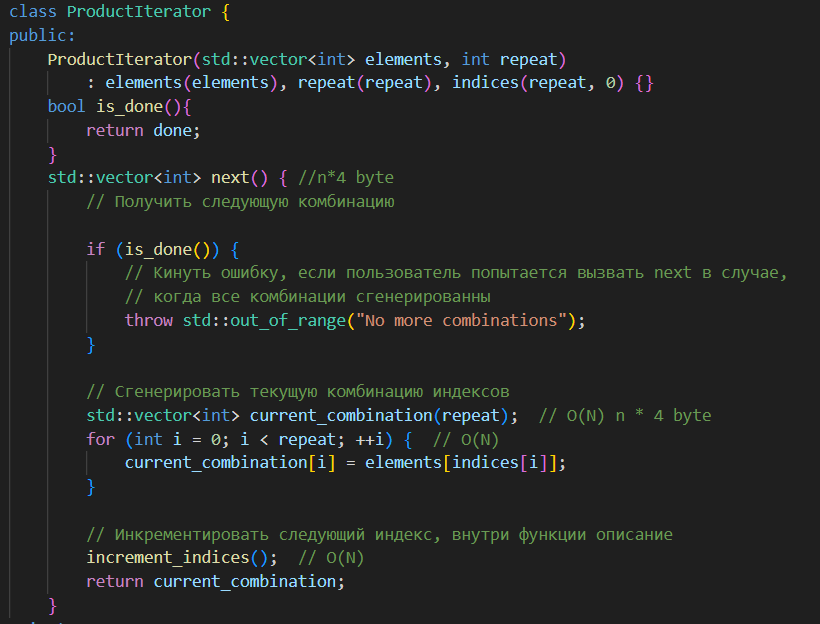
1. Заключение

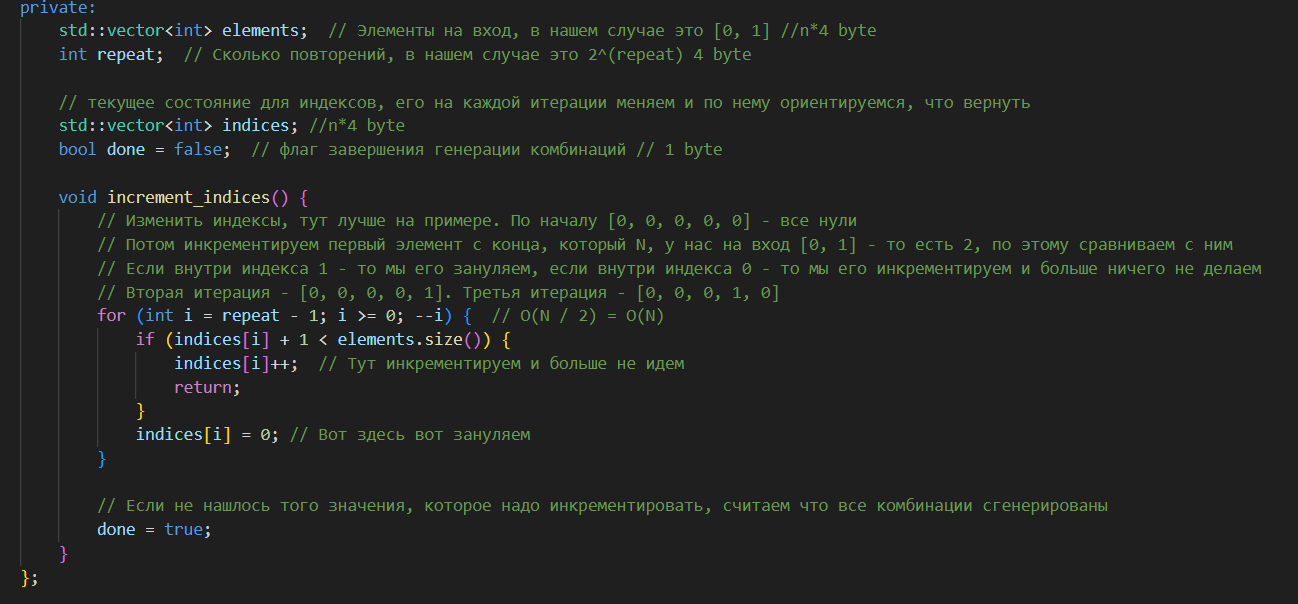
В ходе выполнения работы мною был реализован алгоритм о покрытии множеств. Цель работы была достигнута путём реализации класса, комбинаторно перебирающего индексы и перебора множеств на основе этих индексов. Полученные результаты также совпадают с теоретическими оценками сложности алгоритма.

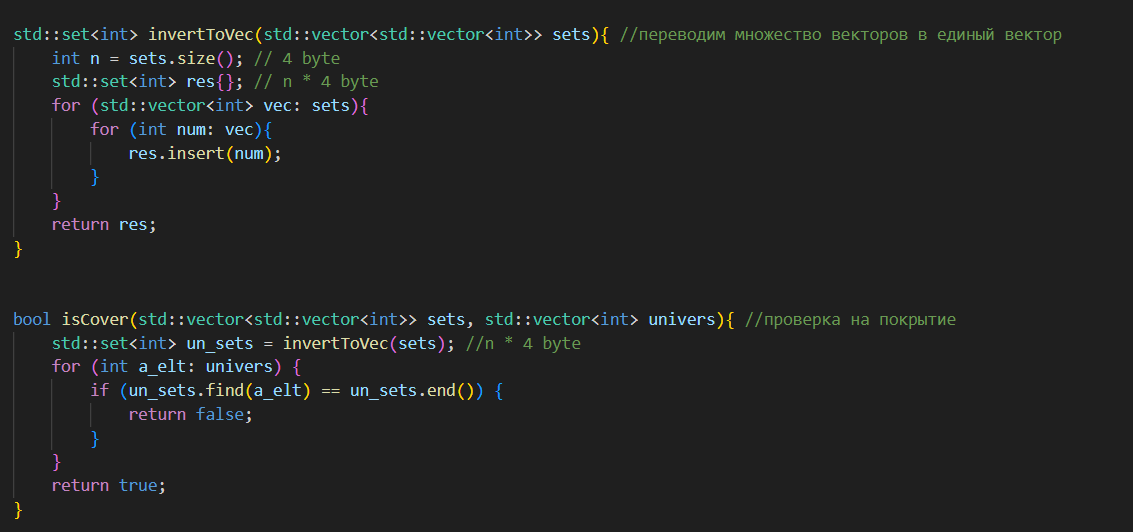
В качестве дальнейших исследований можно предложить оптимизацию алгоритма с точки зрения уменьшения затрат использования памяти, а также рассмотреть версии алгоритма, не перебирающего теоретически невозможные комбинации.

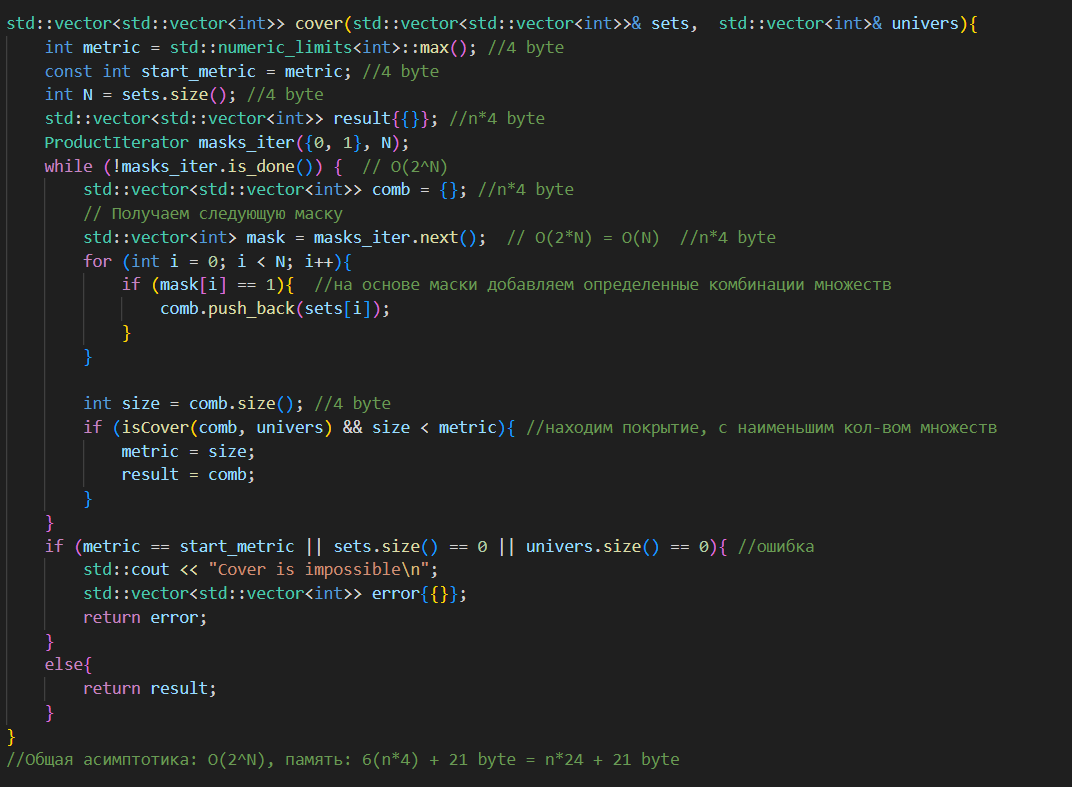
Приложение А

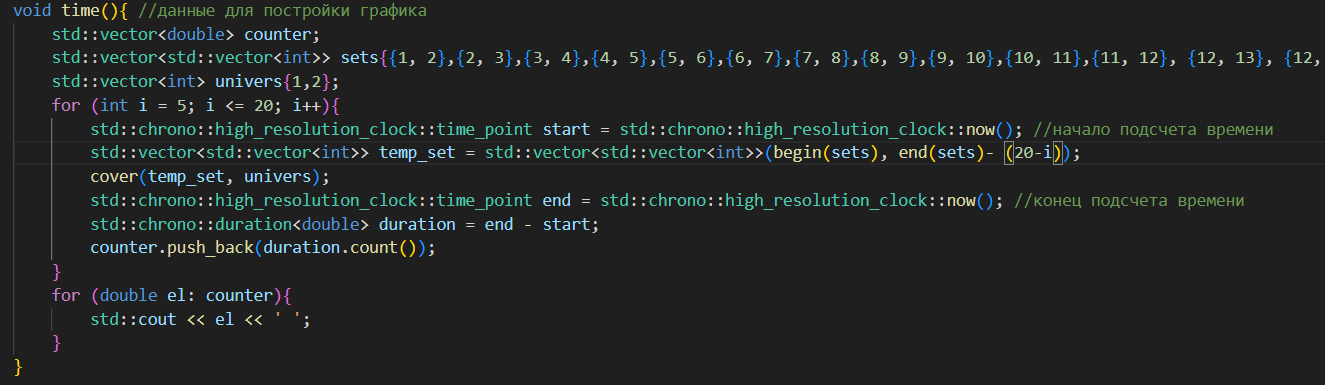
set.cpp

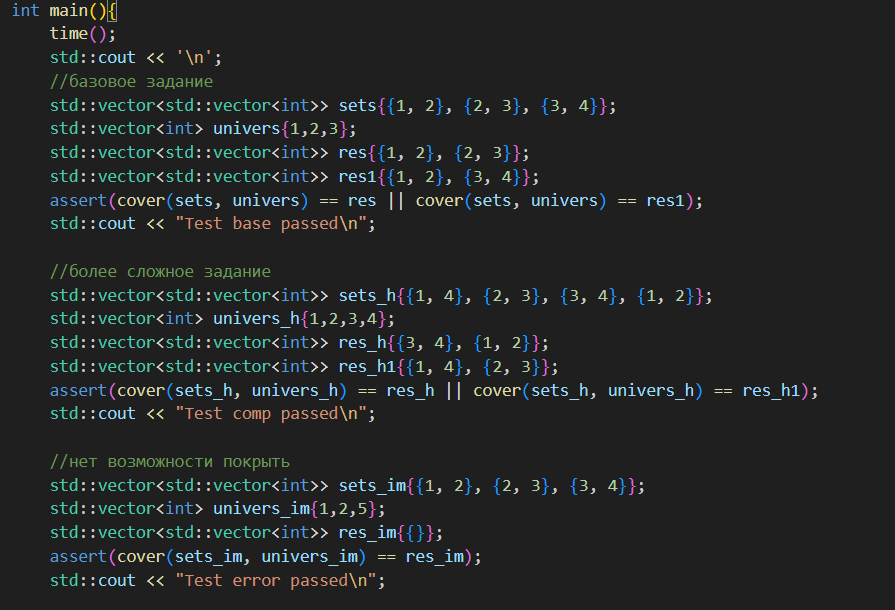


****









****