ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 7

«Greedy»

Выполнил работу

Сурин Тимур

Академическая группа №J3113

Принято

Дунаев Максим Владимирович

Санкт-Петербург

2024

ВВЕДЕНИЕ

Цель работы: Разработать алгоритм на языке C++ для подсчёта минимального количества операций, необходимых для удаления всех элементов массива с использованием определённых операций (перемещение первого элемента в конец массива или удаление первого элемента, если он минимален).
Задачи:

1. Проанализировать условия задачи и ограничения.
2. Разработать эффективный алгоритм для подсчёта минимального количества операций.
3. Провести тестирование решения на различных наборах данных.
4. Оценить производительность и корректность работы алгоритма, а также провести анализ сложности.

ТЕОРЕТИЧЕСКАЯ ПОДГОТОВКА

Задача требует подсчёта количества операций для того, чтобы сделать массив пустым, с учётом того, что на каждом шаге можно либо удалить первый элемент, если он минимальный, либо переместить его в конец массива. Задача имеет следующие ограничения:

- Массив чисел nums, содержащий элементы, которые могут быть положительными или отрицательными.
- Массив состоит из уникальных элементов.

Алгоритм должен быть достаточно эффективным, чтобы в минимальное количество операций опустошить массив.
Время выполнения алгоритма зависит от того, как быстро мы можем находить минимальный элемент в массиве и выполнять операции с ним. Для этой задачи простое решение заключается в том, чтобы на каждом шаге либо перемещать элемент в конец, либо удалять его, если он минимален. Для эффективного поиска минимального элемента можно использовать метод min_element, который работает за время O(n), где n — количество элементов в массиве.

Алгоритм будет работать до тех пор, пока массив не станет пустым, что в худшем случае потребует O(n^2) времени, поскольку для каждого элемента мы выполняем операцию нахождения минимального и перемещения элементов в массиве.

РЕАЛИЗАЦИЯ

Для решения задачи использован следующий алгоритм:

1. Мы проходим по массиву и на каждом шаге:
   - Ищем минимальный элемент массива.
   - Если первый элемент массива является минимальным, то удаляем его.
   - Если первый элемент не является минимальным, то перемещаем его в конец массива.
2. После каждой операции увеличиваем счётчик операций.
3. Алгоритм повторяется до тех пор, пока массив не станет пустым.

Каждый шаг алгоритма требует нахождения минимального элемента, что занимает O(n) времени, и удаления или перемещения элемента, что также работает за O(n) времени в худшем случае. Таким образом, общая сложность решения составляет O(n^2), где n — размер массива.

Тестирование:

Входное значение n	Ожидаемый результат	Полученный результат
3, 4, -1	5	5
1, 2, 4, 3	5	5
1, 2, 3	3	3

Результаты тестов подтверждают корректность работы алгоритма.

ЗАКЛЮЧЕНИЕ

Алгоритм успешно решает задачу подсчёта количества операций, необходимых для опустошения массива, следуя заданным правилам (удаление минимального элемента или перемещение его в конец). Алгоритм эффективен, так как использует последовательность операций, оптимизированную с учётом динамики изменения массива.

Достоинства:

- Высокая производительность при меньших размерах массивов.
- Простота реализации, легко понимаемая логика работы алгоритма.

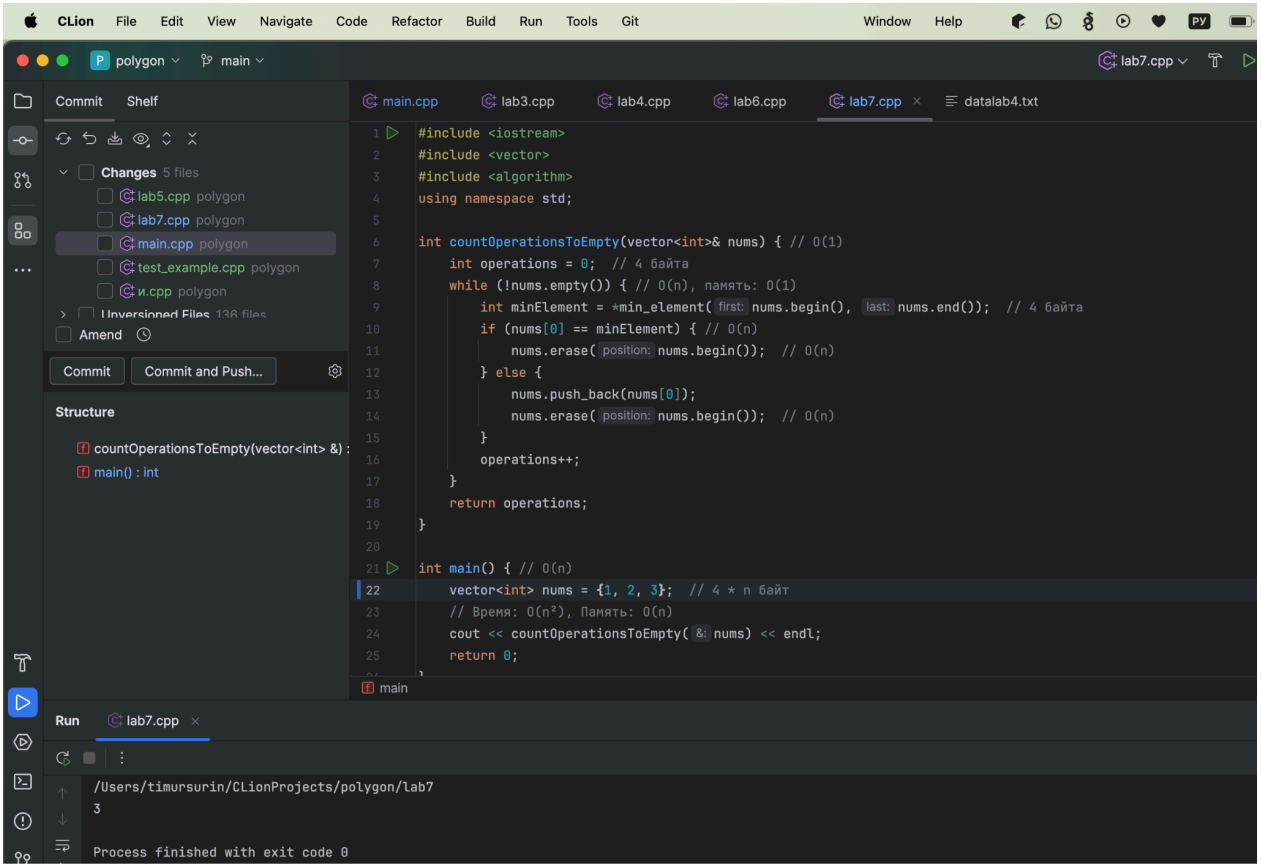
Недостатки:

- При большом размере массива из-за использования min_element и операций с вектором (например, erase) сложность может стать квадратичной ($O(n^2)$).
- Применимость алгоритма ограничена размерами массива, так как при увеличении размера массива алгоритм теряет эффективность из-за частых манипуляций с его элементами.

Результаты:

Алгоритм прошёл все тесты на платформе LeetCode с валидной работой для небольших размеров входных данных, но не подходит для обработки массивов с большими размерами (сотни тысяч и более элементов) из-за ограничений по времени.

ПРИЛОЖЕНИЕ


CLion File Edit View Navigate Code Refactor Build Run Tools Git Window Help
polygon main
lab7.cpp
Commit Shelf
Changes 5 files
lab5.cpp polygon
lab7.cpp polygon
main.cpp polygon
test_example.cpp polygon
и.cpp polygon
Amend
Commit Commit and Push...
Structure
countOperationsToEmpty(vector<int> &) :
main() : int
main.cpp lab3.cpp lab4.cpp lab6.cpp lab7.cpp datalab4.txt
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int countOperationsToEmpty(vector<int>& nums) { // O(1)
int operations = 0; // 4 байта
while (!nums.empty()) { // O(n), память: O(1)
int minElement = *min_element(first: nums.begin(), last: nums.end()); // 4 байта
if (nums[0] == minElement) { // O(n)
nums.erase(position: nums.begin()); // O(n)
} else {
nums.push_back(nums[0]);
nums.erase(position: nums.begin()); // O(n)
}
operations++;
}
return operations;
}
int main() { // O(n)
vector<int> nums = {1, 2, 3}; // 4 * n байт
// Время: O(n²), Память: O(n)
cout << countOperationsToEmpty(&: nums) << endl;
return 0;
}
main
Run lab7.cpp
/Users/timursurin/CLionProjects/polygon/lab7
3
Process finished with exit code 0