**Реализация и анализ алгоритмов поиска элементов на языке Си**

Одной из популярных проблем является поиск повторяющегося элемента в целочисленном массиве. Эта задача имеет множество методов, предлагающих различный подход. Часто ее предлагают программисту при прохождении собеседования на работу, чтобы проверить его кругозор

Для решения этой задачи мы будем использовать язык программирования C. Это самый быстрый высокоуровневый язык программирования, потому он будет крайне эффективен для данной задачи. В качестве компилятора использовался GNU Compiler Collection в Ubuntu 20.04. В качестве машины использовался Oracle VM VirtualBox версии 6.1.32, было задействовано 8192 мегабайта оперативной памяти и 4 потока процессора intel core i5 8300h.

Были рассмотрены следующие алгоритмы:

1. Метод грубой силы

Этот метод заключается в последовательном переборе элементов и сравнением их с каждым следующим в массиве. Сложность алгоритма Θ(n2)

1. Метод сортировки и сравнения

В данном методе мы находим повторяющийся элементом путем сортировки массива и последующим попарным сравнением элементов. Его временная сложность составляет Θ(log2n + n)

1. Поиск через сумму

Данный алгоритм состоит в поиске разницы между суммой элементов массива и суммой ряда. Временная сложность составляет Θ(n)

1. Составление массива с индексами

В этом методе мы создаем второй массив размера n - 1, который заполняем числом -1. Затем последовательно проходим по первому массиву, и каждый встреченный элемент увеличивает элемент во втором массиве, индекс которого равен значению элемента из первого массива – 1. После инкрементирования происходит проверка, и если значения истинно, то возвращается значение искомого элемента. Временная сложность составляет Θ(n)

1. Метод с использованием исключающего «или»

Данный алгоритм построен на использовании бинарного сложения. Для этого мы в первый элемент массива последовательно xor’им все элементы, затем создаем отдельную переменную, в которую xor’им все значения от 0 до n – 1. Результат бинарного сложения между этими двумя числами и будет искомым элементом. Временная сложность составляет Θ(n)

Результаты использования программ приведены на таблице:

Таблица

Изображение выглядит как стол

Автоматически созданное описание

Вывод: метод с использованием исключающего «или» является приоритетным. При достаточно больших n скорость этого алгоритма меньше, чем при использовании сортировки и сравнения, к тому же он не вызывает сбой в программе