**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

отчет

**по лабораторной работе № 1**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: «Множества»

|  |  |  |
| --- | --- | --- |
| Студенты гр. 3311 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Локтионов Т. И.  Осипова Е. Р. |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Преподаватель |  | Манирагена Валенс |

Санкт-Петербург

2024

**Цель работы**

Исследование четырёх способов хранения множеств в памяти ЭВМ.

**Задание (вариант 25)**

Универсум: десятичные цифры.

Множество, содержащее цифры, имеющиеся в каж­дом из множеств ***A*, *B*, *C*, и *D***.

**Формула для вычисления нового множества**

Найти множество, содержащее все цифры данных множеств, по сути, означает найти пересечение всех этих множеств: **E = A ∩ B ∩ C ∩ D**.

Теперь рассмотрим применение этой формулы по отдельности для каждого из способов хранения множеств в памяти.

1. Массивы

Множества могут храниться в виде массивов символов, представляющих цифры от 0 до 9. В этом случае элементы множества представлены в виде строки, а операция пересечения множеств выполняется с помощью линейного поиска элементов в другом массиве.

Операция вычисления множества E для массива реализована с помощью цикла, в котором проверяются условия принадлежности элементов к множествам А, В, С и D.

2. Связные списки

Множества также могут быть представлены в виде связных списков, где каждый элемент множества хранится в узле списка. Операции пересечения и объединения множеств выполняются путём последовательного обхода списков и проверки каждого элемента на наличие в других множествах.

Этот способ позволяет динамически изменять размер множества, но требует дополнительных затрат на управление памятью, то есть является менее выгодным с точки зрения её использования.

3. Массив битов

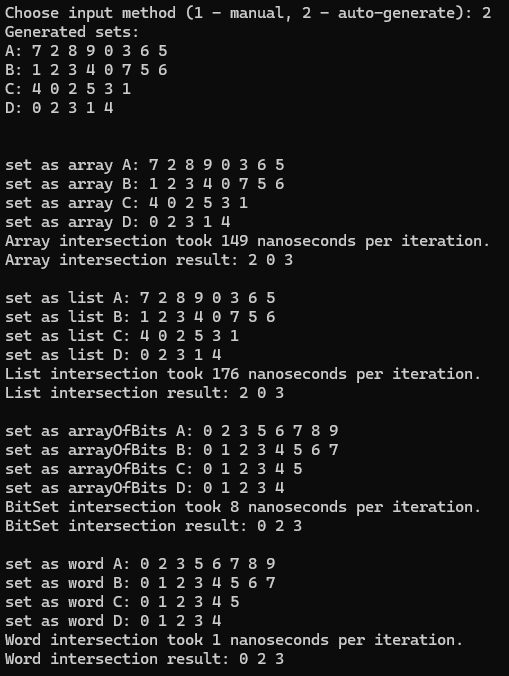
Множества можно хранить в виде массива битов. Операции пересечения и объединения выполняются с помощью проверки соответствующего индекса в массиве, что делает этот способ хранения очень эффективным с точки зрения времени выполнения операций и хранения самих данных, ведь позволяет хранить на каждую единицу данных выделять ровно столько места, сколько нужно (1 бит).

4. Слова (Word)

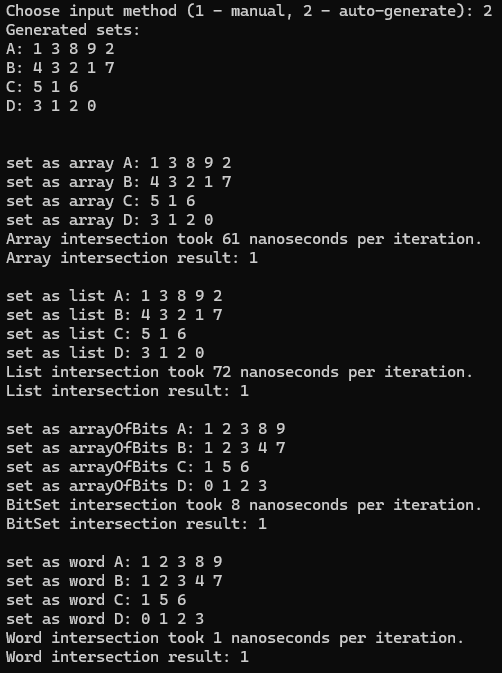
Множество можно хранить в виде целого числа (слова), где каждый бит числа соответствует элементу множества. Это также позволяет выполнять побитовые операции над множествами, обеспечивая высокую производительность.

**Контрольные примеры:**

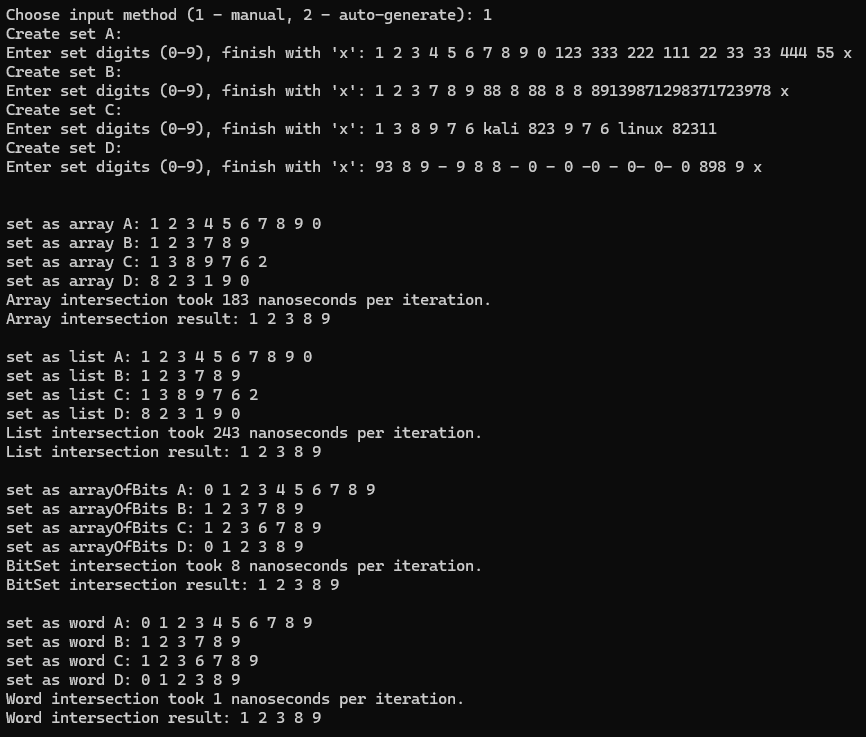
**>> 1.**

****

**>> 2.**

****

**>> 3.**

****

**Оценка сложности**

1. Массивы:

Так как происходит линейный перебор каждого из массивов для формирования нового множества E, сложность будет зависеть от количества элементов в массиве. Предположим, что каждый массив содержит n элементов. Поиск элемента в другом массиве требует линейного времени O(n). Операция пересечения множеств А и В, а также исключение элементов множеств C и D также требует обхода всех элементов.

Итого, сложность операции для массивов: O(n2), где n — размер массивов.

2. Связные списки:

В случае со связными списками для каждого элемента списка A выполняется поиск в списках B, C и D. Каждая операция поиска в списке требует линейного обхода, что аналогично массивам. Для поиска элемента в другом списке требуется O(n) времени.

Таким образом, для пересечения и исключения элементов нужно обрабатывать все элементы каждого списка.

Общая сложность для связных списков:

Итого, сложность операции для списков: O(n2), где n — размер списков.

Связные списки могут иметь дополнительные накладные расходы, связанные с динамическим выделением памяти и управлением указателями, что может увеличивать фактическое время выполнения.

3. Массивы битов:

Массив битов позволяет моментально получить доступ к информации о том, содержится ли в множестве данный элемент. Поскольку алгоритм проверки всегда осуществляется для n = const, где n – размер полученного в задании универсума, то временная сложность от константы будет составлять O(1).

4. Слова (Word):

Машинные слова для данного универсума являются самым эффективным  
способом хранения информации. Поскольку все операции (пересечение, объединение, исключение) делаются за О(1), поскольку требуют единичного побитового сравнения двух чисел.

**Выводы**

По итогам измерений самым эффективным способ – использование машинных слов. Поскольку в данном варианте рассматривается небольшой универсум (10 элементов), то использование short int размером 2 байта позволяет максимально эффективно вычислять пятое множество. Очевидно, что в данном случае, связные списки будут иметь худшую эффективность, поскольку требуют работы с памятью и полного обхода каждого списка. Аналогично с массивами. Битовые векторы показывают себя более эффективными по сравнению с массивами и списками, но имеют худшее время относительно машинных слов. Это связано тем, что работа с массивом все еще сложнее, чем простое битовое сравнение. Важно понимать, что подобные выводы применимы только к данному универсуму и для данного компьютера, на котором проводились эти тесты.