**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе № 1**

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

Тема: «Множества»

|  |  |  |
| --- | --- | --- |
| Студент гр. 3311 |  | Шарпинский Д. А. |
| Преподаватель |  | Манирагена Валенс. |

Санкт-Петербург

2024

**Цель работы**

Исследование четырёх способов хранения множеств в памяти ЭВМ.

**Задание (вариант 33)**

Универсум: десятичные цифры.

Составить множество E, содержащее цифры, общие для множеств А и В, но не встречающиеся ни в С, ни в D.

**Формула для вычисления нового множества**

Для вычисления множества Е в общем случае можно воспользоваться формулой: E = A & B & !(C | D) или ее аналогом E = A & B & !C & !D.

Теперь рассмотрим 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.

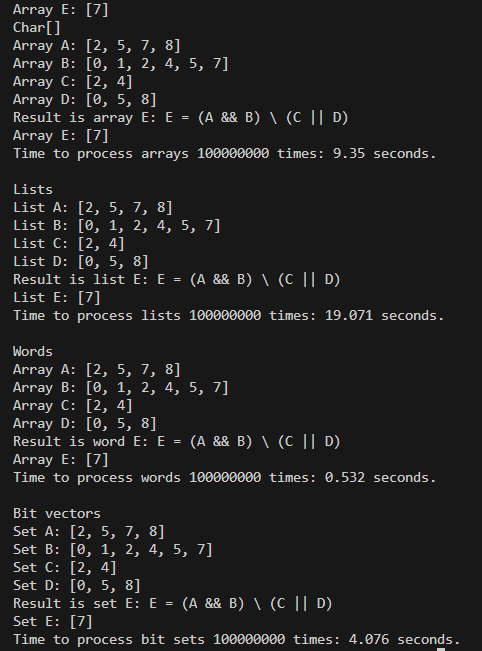
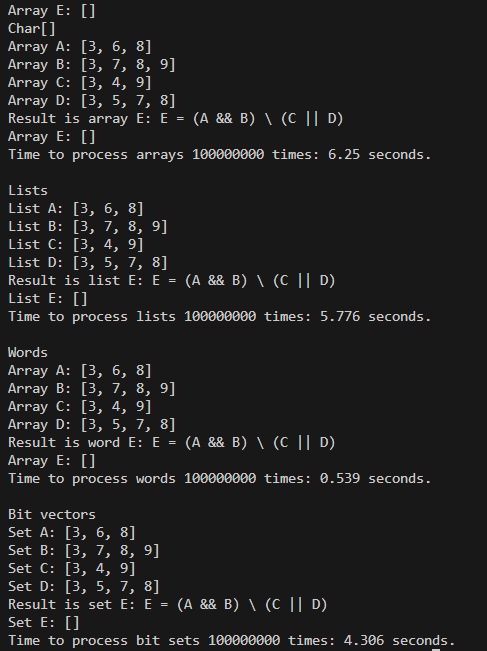
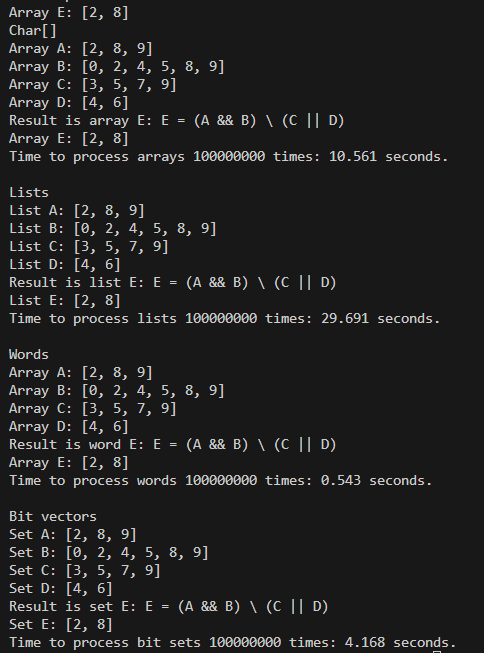
Этот способ позволяет динамически изменять размер множества, но требует дополнительных затрат на управление памятью.

3. Битовые векторы

Множества можно хранить в виде битовых векторов, где каждая цифра представляется отдельным битом. Если элемент присутствует в множестве, соответствующий бит устанавливается в 1. Операции пересечения и объединения выполняются с помощью побитовых операций, что делает этот способ хранения очень эффективным с точки зрения времени выполнения операций.

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**

**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 небольшие дополнительные накладные расходы, связанные с динамическим выделением памяти и управлением указателями, что может увеличивать фактическое время выполнения.

3. Битовые векторы:

Битовые векторы представляют собой фиксированную структуру, где каждый элемент множества представлен одним битом. Операции пересечения, объединения и исключения элементов выполняются с помощью побитовых операций над целыми числами, что делает эти операции очень быстрыми.

Для каждого множества из 10 возможных цифр требуется всего одна операция побитового "И" (&) и "ИЛИ" (|), что выполняется за O(1) для каждого бита.

Общая сложность для битовых векторов: O(1)

Так как операции над битами выполняются за фиксированное время, сложность операций для битовых векторов практически не зависит от размера данных, что делает этот способ хранения множеств очень эффективным.

4. Слова (Word):

Подобно битовым векторам, множества в виде слов (целых чисел) представляют собой набор битов, где каждая цифра множества кодируется отдельным битом. Операции пересечения и исключения выполняются с помощью побитовых операций.

Все операции над множествами A, B, C и D выполняются за постоянное время O(1), поскольку побитовые операции над словами выполняются за фиксированное время.

Общая сложность для операций над словами: O(1)

Побитовые операции выполняются быстро и не зависят от размера множества, что делает этот способ также очень эффективным.

**Результаты измерения времени**

Рассматривались измерения времени для 100000000 итераций.

Массивы: 5.268 sсекунд.

Списки: 6.407 секунд.

Битовые векторы: 3.854 секунд.

Слова: 0.537 секунд.

**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, посколько 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 с большим размером вектора в памяти и тем, что работа с массивом все еще сложнее, чем простое битовое сравнение. Важно понимать, что подобные выводы применимы только к данному уневрсуму.