CS 4080 Final Project Presentation

Team: Home United
Jennifer Chiang, Rachel Lewis, Mai Luu, Sofia Pineda

Introduction

- Array is one of the most common types of data storage and almost every modern programming language uses arrays.
- So creating an array within a program or object's scope should be easy and fast, read and write operations should be quick, and the array's data and contents should be volatile so that when a program terminates
- Our project is to study whether the arrays in Java, Javascript, C++, and Python meet the stated requirements.
- Also, we compare arrays between the four languages by researching how design decisions impact an array's effectiveness as a data structure for each language.

Java - Arrays Overview

- Always dynamically allocated
- Size must an an int value
- Each item is an element
- Built-In Functions in java.util.Arrays
    - Binary Search
    - Copy
    - Fill
    - Sort

Java - Array Experiments

- 2D Arrays that equal 1D Arrays use same computation time
  - JavaScript
  - C++
- Binary Search is directly proportional to size
  - 3,000,000 -> 26,900 ns
  - 15,000,000 - > 18,800 ns

Javascript Arrays Overview

- Inheritance hierarchy: null → Object.prototype → Array.prototype
- Heterogeneous data types are allowed in Javascript arrays
  - Type-checking is not done by Javascript on arrays
- Javascript arrays are dynamically sized
  - Object.prototype's length value is not fixed
  - IndexOutOfBounds errors don't exist, Javascript simply changes the array's length value

Javascript Array Experiments

Addition between large 2 arrays

- Test 1: Two 1D arrays of size 29,997,529
- Test 2: Two 2D arrays of size 5,477x5,477

Conclusion: Adding 2 large 2D arrays faster than adding large 1D arrays.

Binary Search on one large array

- Test 1: Performing binary search on 5 arrays of size: 30 million, 15 million, 7.5 million, 3.75 million, & 1.825 million

Conclusion: Array size has strong correlation to binary search time

C++ Research of Arrays

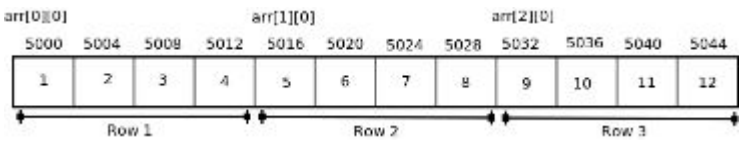
Array in C++ is homogeneous type, and it is a collection of variables of the same type stored at contiguous memory locations.

Multidimensional Array: Arrays of an array. For example:

- Int arr[m][n] represents ‘m’ arrays of 1D arrays, which contain ‘n’ elements.
- Three Dimensional Array: int arr[2][3][4]: 2 arrays of 3x4 arrays.
- The concept of storing multidimensional arrays in memory: it is stored in row-major order so rows are placed next to each other. Each row is considered as a 1D array.

Multidimensional Arrays:

- The 2D array: int arr[][]
- The pointer-to-array: int (*arr)[]
- The pointer-to-pointer: int **arr
- The array-of-pointers: int *arr[]
- Use array pointer to access element inside multidimensional arrays.



C++ Experiment on Arrays

- First and Second Experiments: compare the result of adding two single dimensional arrays versus two multidimensional arrays
  - The results show that the performance of adding two single dimension arrays for the size below 50,000 are faster than adding two 2D arrays.
  - However, when the size reaches 50,000 the performance of adding two multidimensional arrays are faster than adding two single dimension arrays.
  - Since the array storing as an array of arrays in C++, and the pointers are pointed to other arrays, so the performance of adding 2D arrays are faster than 1D arrays when the size grows bigger.
- Third Experiment: the performance of binary search on the array of size from 100 - 1,000,000.
  - Result for this experiment: as the array size goes bigger but the time for searching a value through a binary search algorithm is nearly constant with regard to the size, this matches with the original expectation.

Arrays in Python

- There is no “array” type
  - Lists
  - Tuples
  - Dictionaries
  - Sets

- Can include mixed types
- Index starts at 0, not 1
  - FORTRAN & APL start at 1
- Multidimensional arrays are lists within lists
- Lists are dynamic

Python built in methods and experiment results

- append()
- len()
- pop()
- clear()
- index()
- sort()
- copy()

- Experiment 1
  - 2D array took longer to accomplish the same task
- Experiment 2
  - Follows trend as input size grows time taken also grows
  - Python took a longer time to complete the search than Java and C++

Conclusion

- Out of the four languages javascript and python took the longer to run the search algorithm
  - This is because javascript and python are interpreted languages whereas Java and C++ are compiled
- Experiment 1 results
  - Java
    - Both dimension's performance was around the same
  - Javascript & C++
    - Multidimensional arrays performed better than one dimensional arrays
      - Javascript: Size of two dimensional array was actually smaller in memory than one dimensional arrays
    - For C++ this happened after hitting a threshold size of 50,000
  - Python
    - One dimensional arrays performed better
- Experiment 2 results
  - As size of array to search upon increased, the runtime also increased
- All languages have their own built in array methods