**Abstract**

In this experiment, it shows the performance of adding two single dimension arrays versus two multiple dimension arrays. Also, the performance for binary searching of an sorted arrays in nanoseconds.

**Experiment:**

All of the arrays are using in the experiments are randomly generated.

* First Experient is performing the addition between two single dimension arrays with the size range from 100 – 250,000, and the times (in nanoseconds) are presented in the table below:

|  |  |
| --- | --- |
| Size | Time |
| 100 | 695 |
| 10,000 | 48,953 |
| 40,000 | 293,957 |
| 80,000 | 480,584 |
| 100,000 | 668,793 |
| 160,000 | 697,548 |
| 250,000 | 1,627,071 |

* Second Experiment is performing the addition between two two-dimension arrays with the size also range from 100 – 250,000, and the times (in nanoseconds) are presented below:

|  |  |
| --- | --- |
| Size | Time |
| 100 | 818 |
| 10,000 | 64,635 |
| 40,000 | 349,308 |
| 80,000 | 321,034 |
| 100,000 | 511,943 |
| 160,000 | 627,390 |
| 250,000 | 1,250,464 |

After finishing these two experiments and comparing the numbers from the tables:

* 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 multidimension 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 is doing binary search on the various size of sorted single dimensional array. The table below is the time (in nanoseconds):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size | 100 | 1000 | 10,000 | 50,000 | 100,000 | 150,000 | 500,000 | 1,000,000 |
| Time | 3045 | 4109 | 4873 | 4472 | 5803 | 5465 | 6839 | 8287 |

Result for this experiment: as the array size goes bigger but the time for searching a value through a binary search algorithm are nearly constants with regard to the size. This is match with the original expectation.

**Conclusion**

If the array sizes are smaller than 50,000, C++ handles single dimension arrays better than multiple dimension arrays but if the size is bigger than 50,000, C++ handles the add operation between two multidimension arrays better than single dimension arrays.

For Binary Search, even though the size of the array are growing from small to large number, the time for doing binary search are nearly constants with compared to the size.