JOSHUA JAISTEIN, ACADEMIC AI RESEARCHER

PROBLEM - 1

SQUARE-MATRIX MULTIPLICATION

OPTIMIZATION STEPS DONE :

Unit test : Catch2 ; Machine Specs : i5 8th Generation with 8 GB Memory ; N = 5000

1. C++ Arrays :
   The matrix multiplication logis is coded and is a very basic methodology of matrix multiplication. It proved to be very slow at this stage.The running time tends to be large so the program was stopped after 23 hours.

2. Multi-threading :
   Multi-threading concept was introduced and less runtime was expected but the code is still time consuming and slow. The program was halted after 19 hour of running.The program achieved 17% speedup relatively.

3. CUDA C :
   The same matrix logic is coded on CUDA C to run in parallel and the results were significant to almost 99 percent relative decrease in code runtime.The runtime framework CUDA 10.2 on NVIDIA 1060 CARD with 6gb vRAM was used.

4. STRASSEN - DIVIDE AND CONQUER CUDA C : (O(n)^2.8)
   Divide and Conquer rule provided the most optimal solution while running in parallel on CUDA. The program achieved 100 percent relative decrease in code runtime.

RUNNING THE CODE :

- Please place the catch.hpp in the running directory
- The test cases are built in to the programs

The above steps are coded into three files namely :

MatrixOne.cpp

```
$ g++ MatrixOne.cpp
$ ./a.out
```

MatrixTwo.cpp

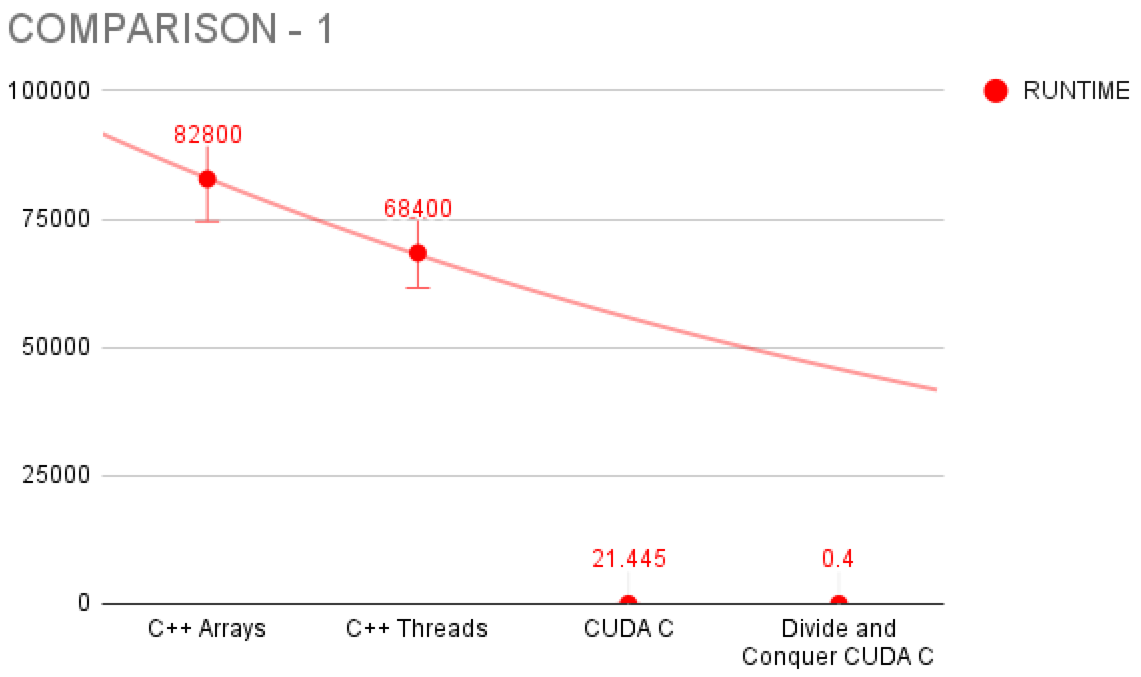
```
$ g++ MatrixTwo.cpp
$ ./a.out
```

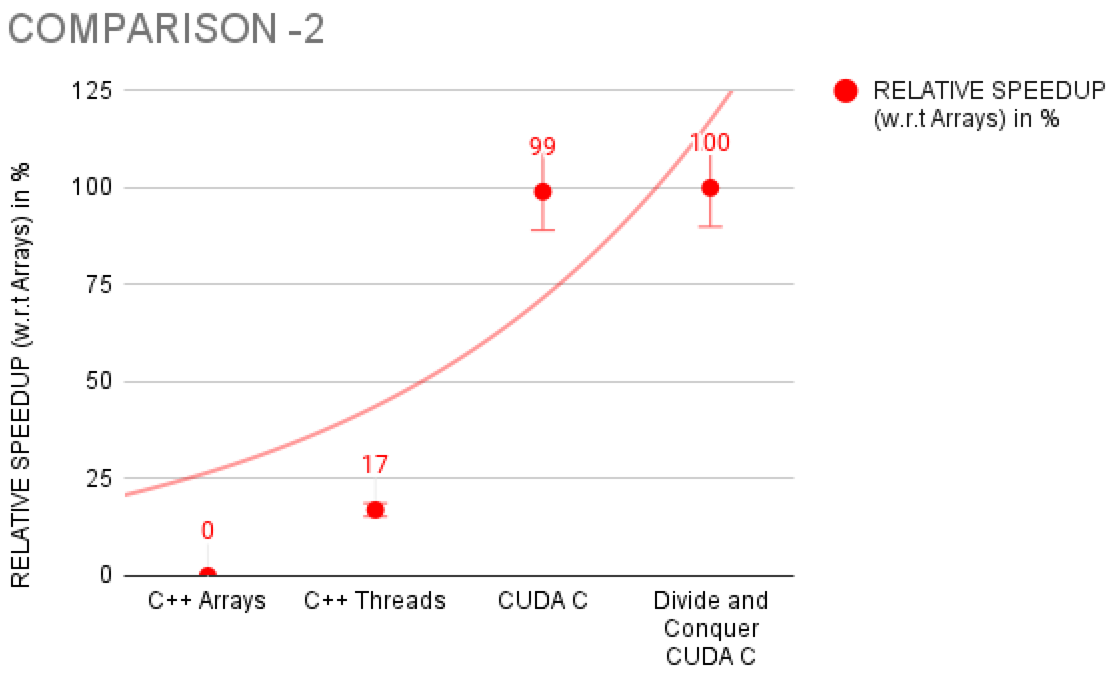
MATRIX-CUDA3.cpp

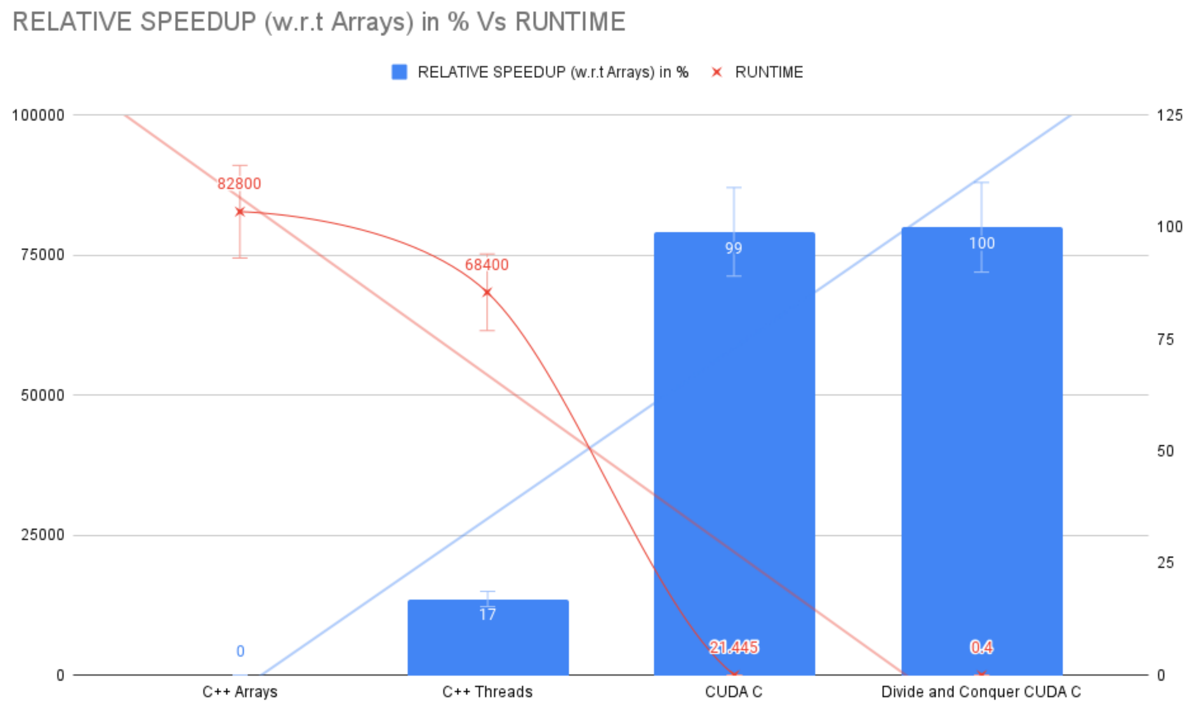
```
$ g++ MATRIX-CUDA3.cu
$ ./a.out
```

MATRIX-STRASSES-CUDA4.cpp

```
$ nvcc -arch=sm_52  MATRIX-STRASSES-CUDA4.cu cudaTimer.cc -o outputCUDA -lcublas
$ ./outputCUDA
```







PROBLEM - 2

BINOMIAL COEFFICIENTS

OPTIMIZATION STEPS DONE :

COMPARISON VALUES : N = 34 ; k = 15

1. C++ RECURSION :

   The binomial coefficient function is coded as a recursive function that calls itself over and over.

2. C++ DYNAMIC PROGRAMMING APPROACH :

   The Dynamic programming approach was used and was highly efficient than the previous implementation.

RUNNING THE CODE :

Please place the catch.hpp in the running directory

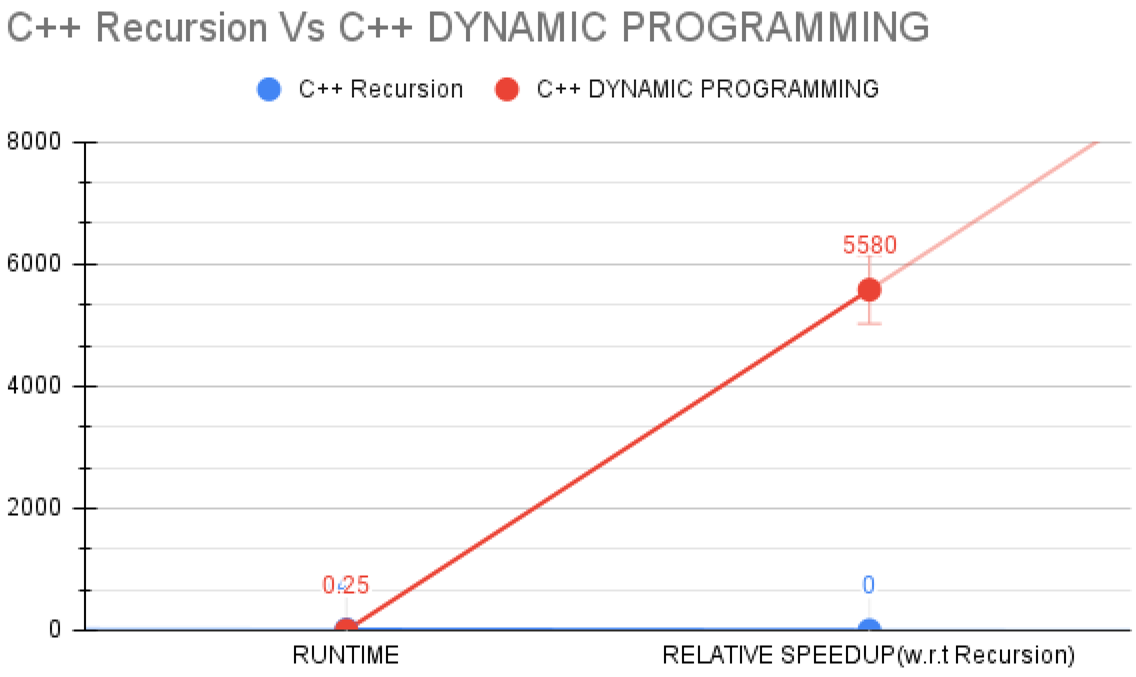
The above steps are coded into two files namely :

BCRecursion.cpp (use of BITS library)

```
$ g++ BCRecursion.cpp
$ ./a.out
```

BCDynamic.cpp

```
$ g++ BCDynamic.cpp
$ ./a.out
```



PROBLEM - 3

CONSOLE SPREADSHEET

OPTIMIZATION STEPS DONE :

Tests included in the code : Catch2

C++ MATRIX ALLOCATION :

a. **User input by reading 'input.txt' file in the directory**
b. **From the input stream the variables are initialised into a 2x2 matrix**
c. **User output is filed as a .csv spreadsheet file**

RUNNING THE CODE :

Please place the catch.hpp in the running directory

The above steps are coded into a single file namely :
CS10.cpp

```
$ g++ CS10.cpp
$ ./a.out
```