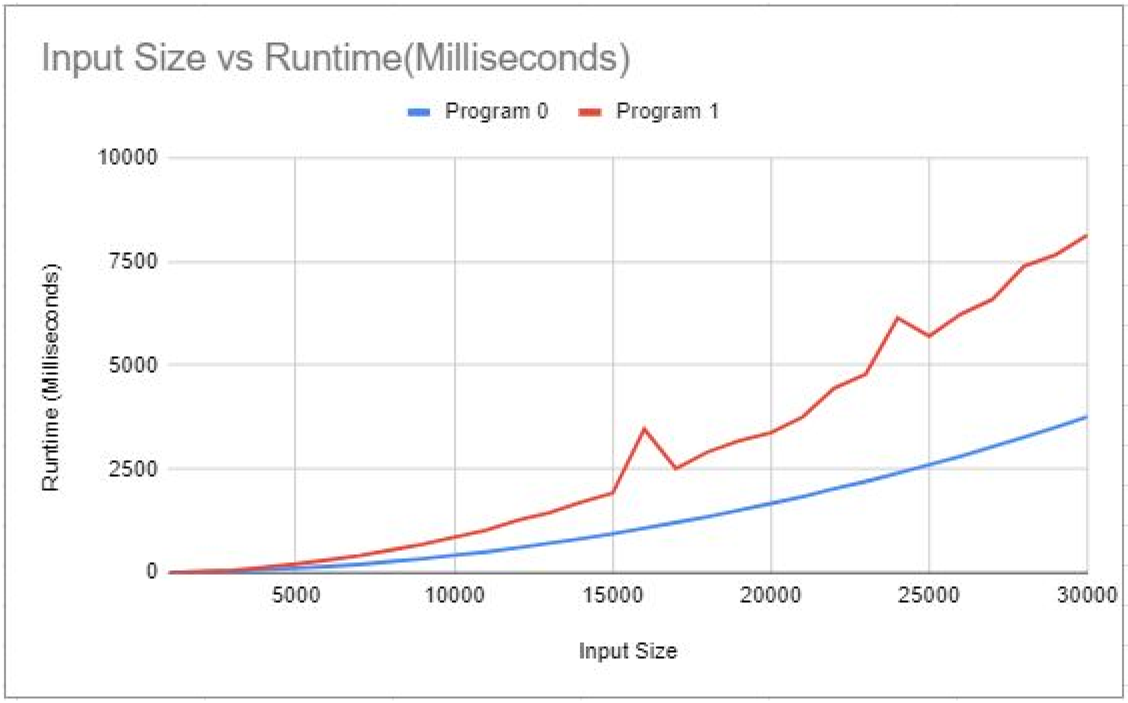
In this lab, we compare two versions of matrix_vector_multiply.
Program0 matrix_vector_multiply

```
void matrix_vector_multiply(long long *y, long long *A, long long *x, long
long size) {
   long long i, j;

   for (i = 0; i < size; ++i) {
      for (j = 0; j < size; ++j) {
         y[i] += A[i*size + j] * x[j];
      }
   }
}
```

Program1 matrix_vector_multiply

```
void matrix_vector_multiply(long long *y, long long *A, long long *x, long
long size) {
   long long i, j;

   for (i = 0; i < size; ++i) {
      for (j = 0; j < size; ++j) {
         y[j] += A[j*size + i] * x[i];
      }
   }
}
```

When we compare the performance on the, program0 beats program1. Program 0 beats Program 1 because multiplication is an expensive operation. In program 0, multiplication happens size^2 times. In program 1, multiplication happens size^3 times. This is because i*size (program 0)only changes in the outer loop, while program(1) (j*size) happens in the inner loop so there is size * size * size multiplications, causing size^3 runtime.


Input Size vs Runtime(Milliseconds)
Program 0
Program 1
Runtime (Milliseconds)
Input Size