Project 1

Section 1.3 – Sorting Algorithms

Rainbow Martin

Hardware Specs:

- CPU: 11th Gen Intel(R) Core(TM) i7-1195G7 @2.90GHz, 4 cores
- 8 GB of RAM
- 475GB of Storage
- GPU Intel(R) Iris(R) Xe Graphics, 128 MB
- OS: Windows 11

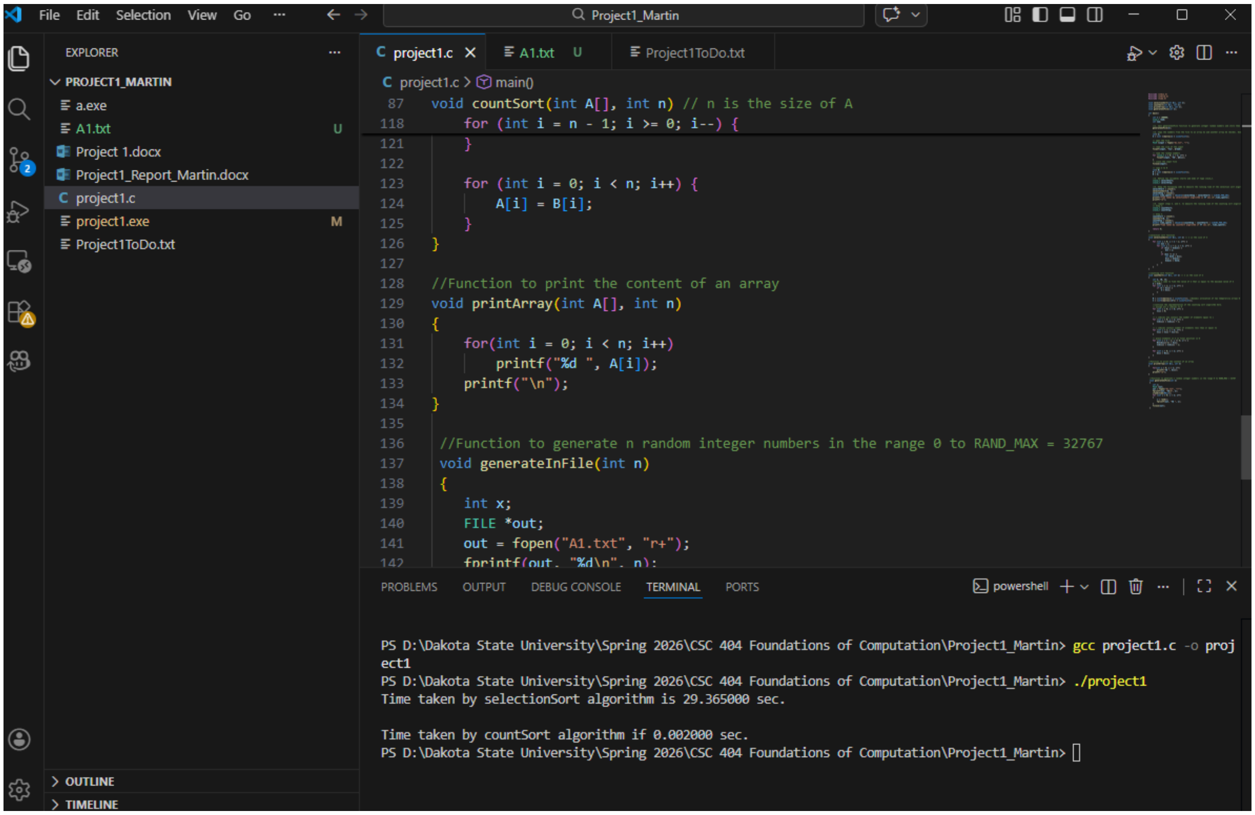
n	**Time taken by Selection sort (in seconds)**	**Time taken by Counting sort (in seconds)**
10^2	0.000000	0.000000
10^3	0.002000	0.000000
10^4	0.280000	0.001000
10^5	29.365000	0.002000
10^6	2293.487000	0.016000
10^7	*	0.145000
10^8	*	2.004000
10^9	*	73.729000

* Algorithm time growth is exponential – will take days to run

Looking at my output, it is obvious that the Selection Sort algorithm's time complexity is n^2 because as the input grows by a factor of 10, the runtime increases rapidly. I originally suspected that an input of size 10^7 would take Selection Sort at least 6 hours to complete. However, that was based on multiplying the runtime of 10^6 by 10. That is incorrect because the time complexity of the algorithm would not produce linear growth. In addition, I researched the limits of my machine (see specs) and found that running 10^7 and above would take longer than a couple days to complete. As you can see from my table, I elected to stop running the Selection Sort algorithm above 10^6 due to the lengthy runtime. These results tell me that Selection Sort is not a good candidate for real-world applications, unless said applications are intended for small datasets (i.e. below 10,000). 29 seconds to sort 100,000 isn't terrible, but I can't imagine that many users would be happy with the pause in application response.

The Counting Sort algorithm is much faster and "better" than the Selection Sort algorithm as the input grows because its time complexity is linear based on (n + k). The runtime increases less dramatically. I suspect that the higher increases in the 10^8 and 10^9 input size datasets could be due to possibly higher max values in these arrays compared to the arrays of lesser input sizes and/or memory issues since I have only 8 GB of RAM. In my research, I found that 16-32 GB of RAM is more adequate for evaluating algorithms with large data sets.

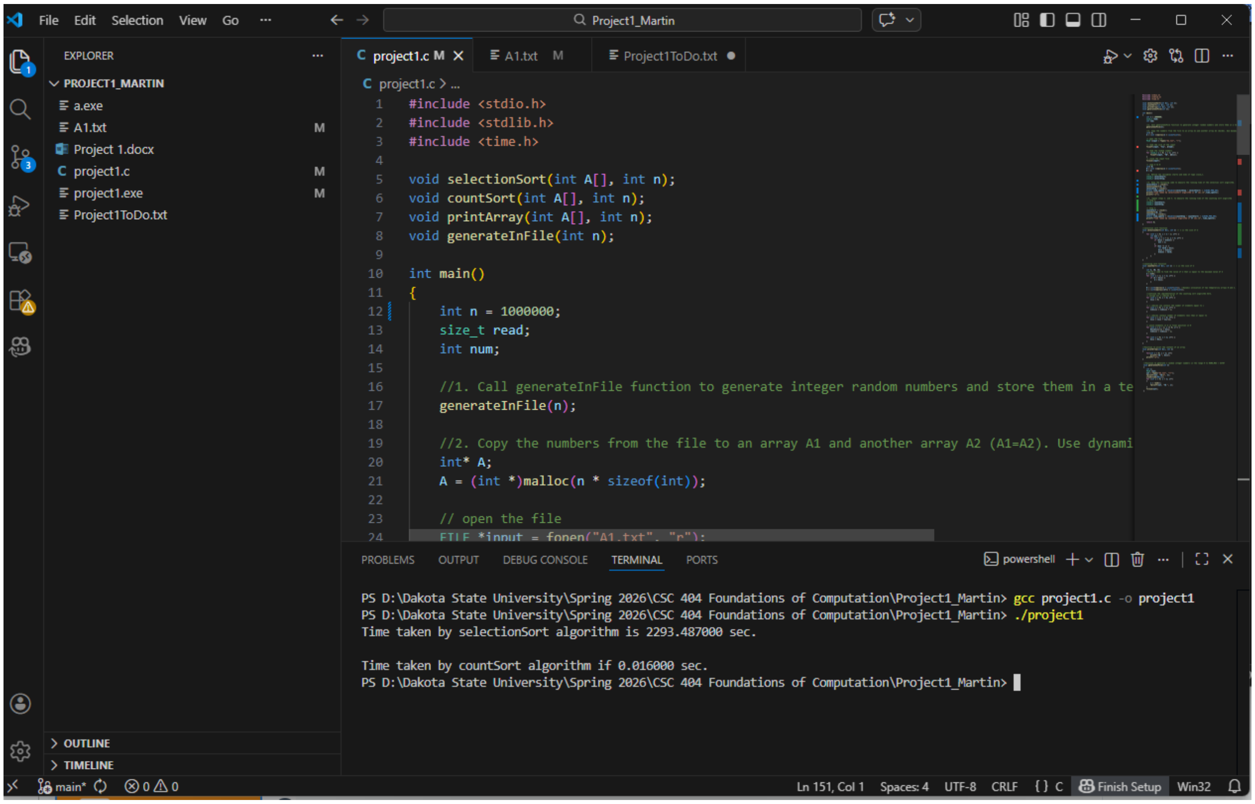
Selection Sort and Count Sort running on 10^5 size input



```c
void countSort(int A[], int n) // n is the size of A
    for (int i = n - 1; i >= 0; i--) {
    }

    for (int i = 0; i < n; i++) {
        A[i] = B[i];
    }
}

//Function to print the content of an array
void printArray(int A[], int n)
{
    for(int i = 0; i < n; i++)
        printf("%d ", A[i]);
    printf("\n");
}

 //Function to generate n random integer numbers in the range 0 to RAND_MAX = 32767
 void generateInFile(int n)
 {
    int x;
    FILE *out;
    out = fopen("A1.txt", "r+");
    fprintf(out, "%d\n", n);
```

```
PS D:\Dakota State University\Spring 2026\CSC 404 Foundations of Computation\Project1_Martin> gcc project1.c -o proj
ect1
PS D:\Dakota State University\Spring 2026\CSC 404 Foundations of Computation\Project1_Martin> ./project1
Time taken by selectionSort algorithm is 29.365000 sec.

Time taken by countSort algorithm if 0.002000 sec.
PS D:\Dakota State University\Spring 2026\CSC 404 Foundations of Computation\Project1_Martin>
```

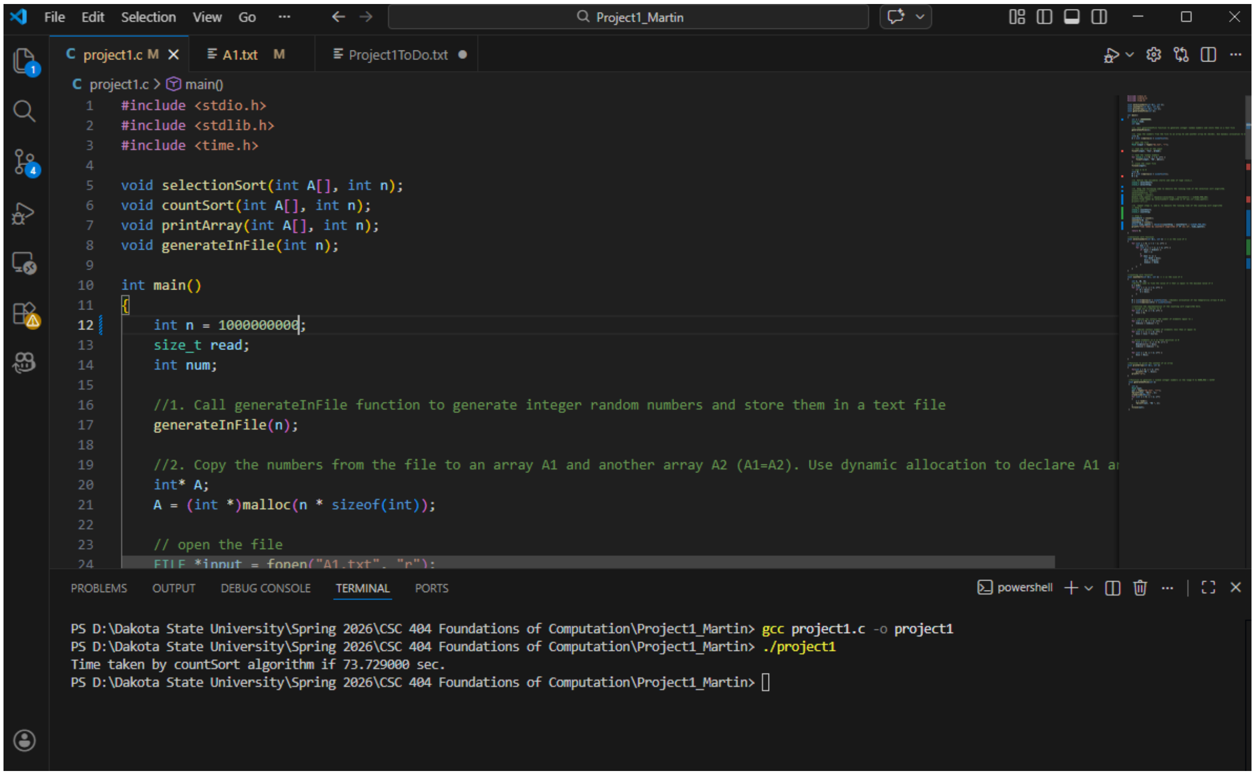
Selection Sort and Count Sort running on 10^6 size input



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void selectionSort(int A[], int n);
void countSort(int A[], int n);
void printArray(int A[], int n);
void generateInFile(int n);

int main()
{
    int n = 1000000;
    size_t read;
    int num;

    //1. Call generateInFile function to generate integer random numbers and store them in a te
    generateInFile(n);

    //2. Copy the numbers from the file to an array A1 and another array A2 (A1=A2). Use dynami
    int* A;
    A = (int *)malloc(n * sizeof(int));

    // open the file
    FILE *input = fopen("A1.txt", "r");
```

```
PS D:\Dakota State University\Spring 2026\CSC 404 Foundations of Computation\Project1_Martin> gcc project1.c -o project1
PS D:\Dakota State University\Spring 2026\CSC 404 Foundations of Computation\Project1_Martin> ./project1
Time taken by selectionSort algorithm is 2293.487000 sec.

Time taken by countSort algorithm if 0.016000 sec.
PS D:\Dakota State University\Spring 2026\CSC 404 Foundations of Computation\Project1_Martin> 
```

Count Sort running on 10^9 size input



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void selectionSort(int A[], int n);
void countSort(int A[], int n);
void printArray(int A[], int n);
void generateInFile(int n);

int main()
{
    int n = 1000000000;
    size_t read;
    int num;

    //1. Call generateInFile function to generate integer random numbers and store them in a text file
    generateInFile(n);

    //2. Copy the numbers from the file to an array A1 and another array A2 (A1=A2). Use dynamic allocation to declare A1 a
    int* A;
    A = (int *)malloc(n * sizeof(int));

    // open the file
    FILE *input = fopen("A1.txt", "r");
```

```
PS D:\Dakota State University\Spring 2026\CSC 404 Foundations of Computation\Project1_Martin> gcc project1.c -o project1
PS D:\Dakota State University\Spring 2026\CSC 404 Foundations of Computation\Project1_Martin> ./project1
Time taken by countSort algorithm if 73.729000 sec.
PS D:\Dakota State University\Spring 2026\CSC 404 Foundations of Computation\Project1_Martin> 
```