Data Structures and Algorithms 1

Declan Doyle

Comparing time-efficient algorithms

Radix

- Copes with large numbers
- Doesn’t need to compare items

Merge

- Recursive
- Divide and Conquer

Data Structure

Array

- No need for dynamically changing list
- Efficient list
- Fixed amount of memory

```
const int ten = 10;
int list[ten];
```

```
const int tenThousand = 10000;
int list[tenThousand];
```

The Program

- User interface to decide the sort, list size and range of numbers in the list
- Generates random list based on parameters
- Sorts list 1,000 times
- Outputs times to a file

```
Declan Doyle Data Structures and Algorithms 1 Coursework Program
Enter -1 to exit any menu screen
Choose a sort:
1: Radix
2: Merge
```

```
Radix Sort
Choose the list size:
1: 10
2: 100
3: 1000
4: 10,000
5: 100,000
```

```
Radix Sort
Choose the list range:
1: 0-10
2: 0-100
3: 0-1000
4: 0-10,000
5: 0-100,000
```

```
List size 10000 selected
List range 100000 selected
Enter 'g' to begin benchmark
```

```
Sorts completed. Data saved to a file called 10000Radix100000.csv
File name is as follows: [List Size][Sort Type][List Range].csv

Enter 'g' to go to previous menu
```

```
void radix();
void radixSize(int menu);
void radixRange(int *list, int size);
void radixInit(int menu, int *list, int size);
void radixExecute(int *list, int size, int range, string file);
void radixSort(int *input, int n);
```

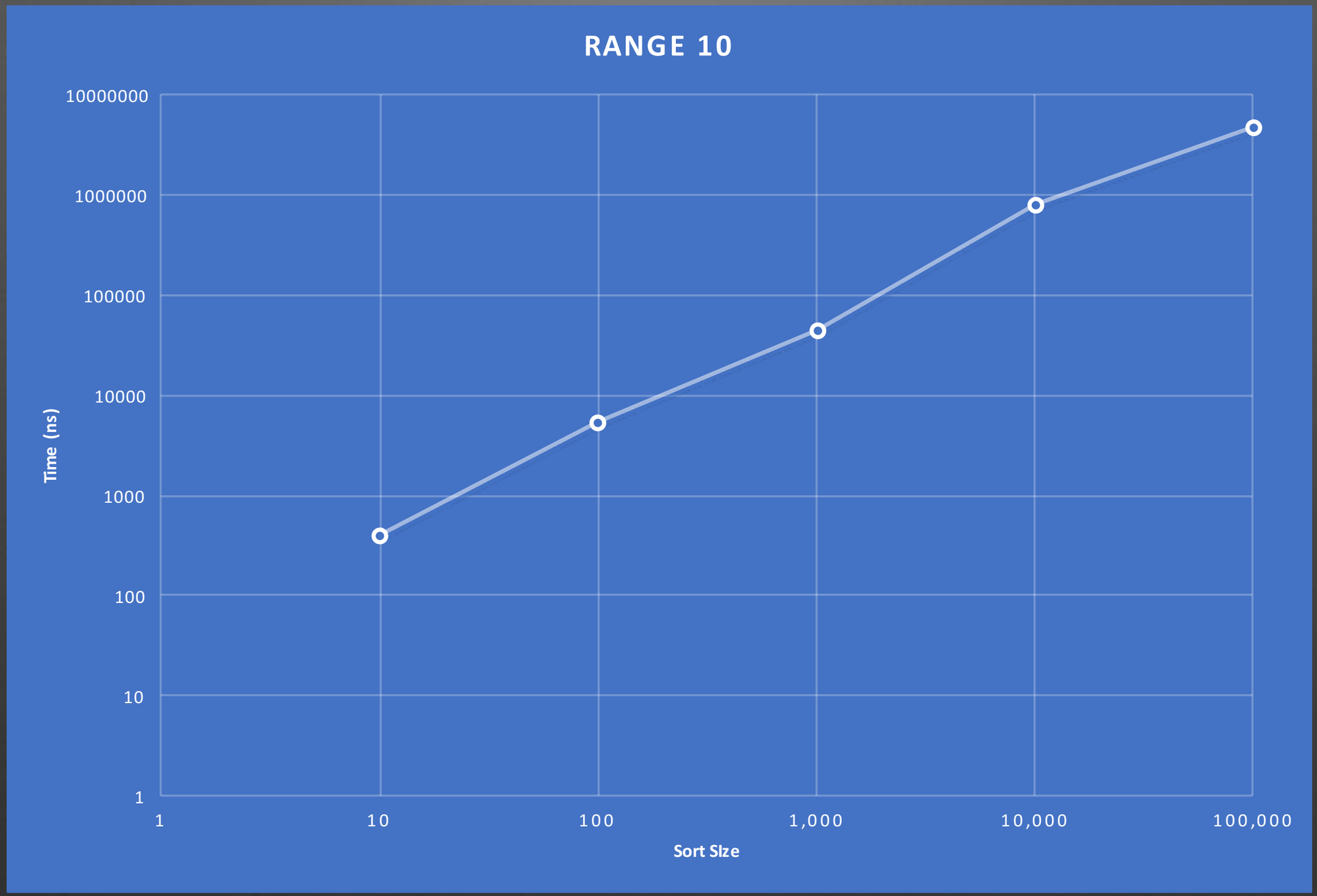
```
else if (menu == 3)
{
   const int oneThousand = 1000;
   int list[oneThousand];
   radixRange(list, oneThousand);
}
```

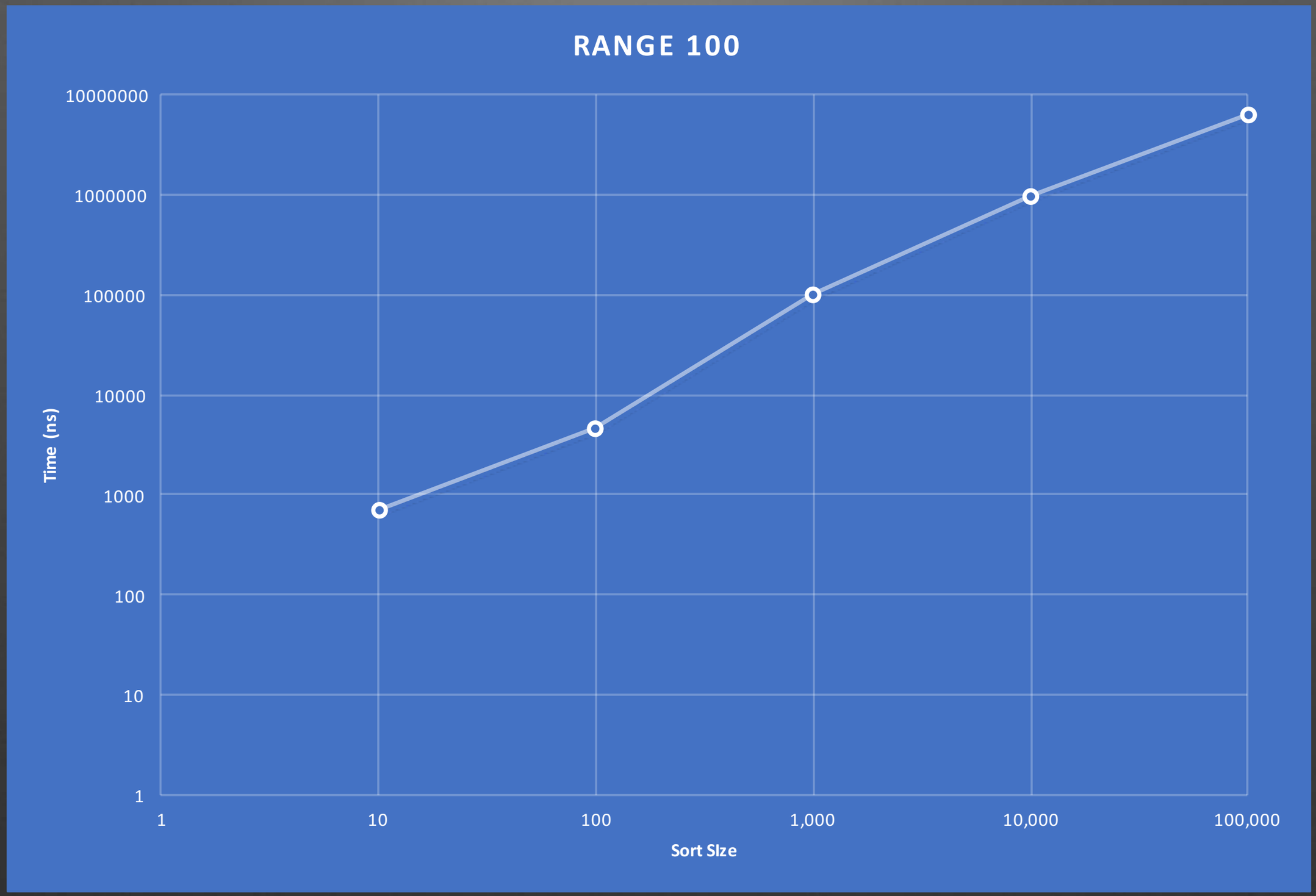
Performance

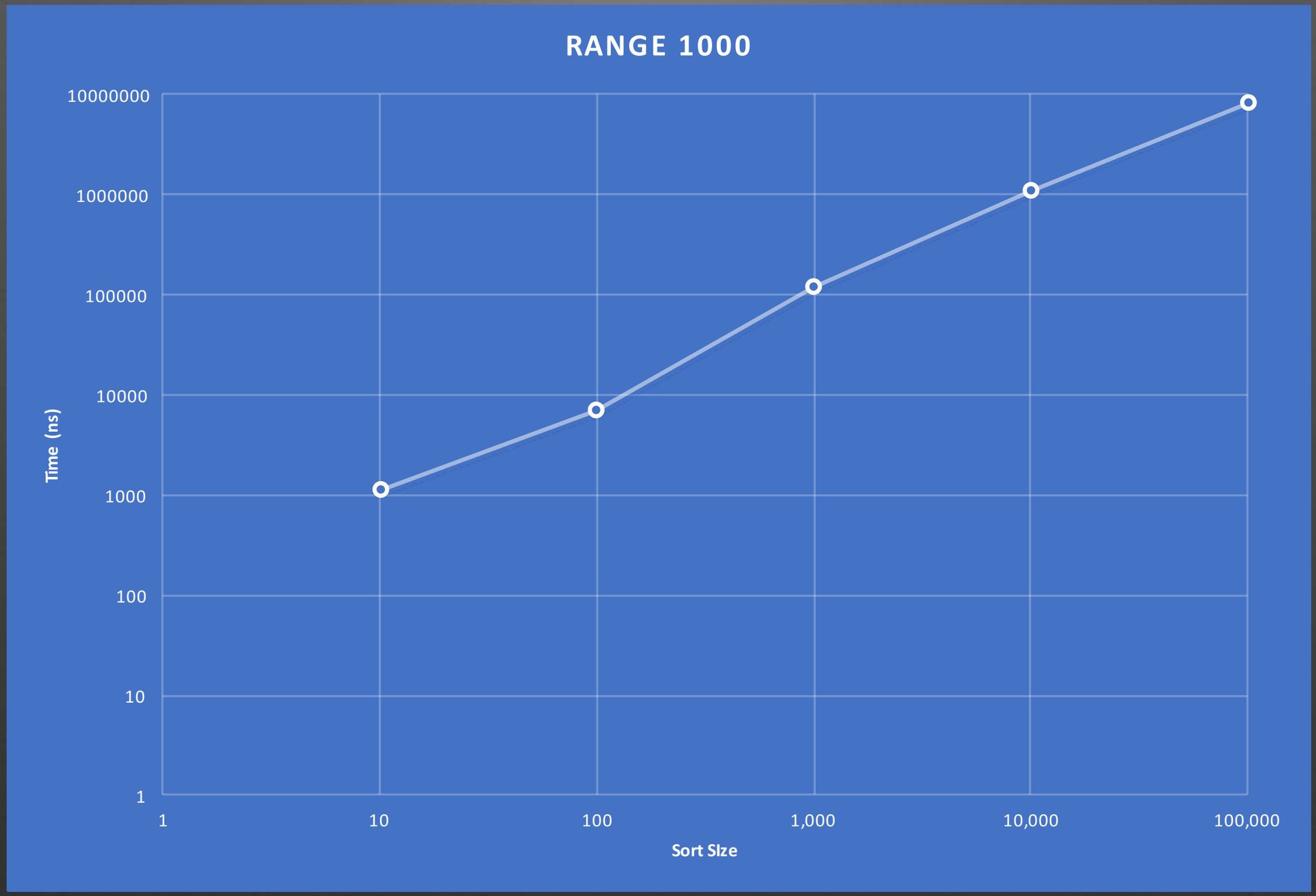
- Program was ran inside a VM
- 2 Processor Cores and 2GB of RAM
- No other processes running inside the VM

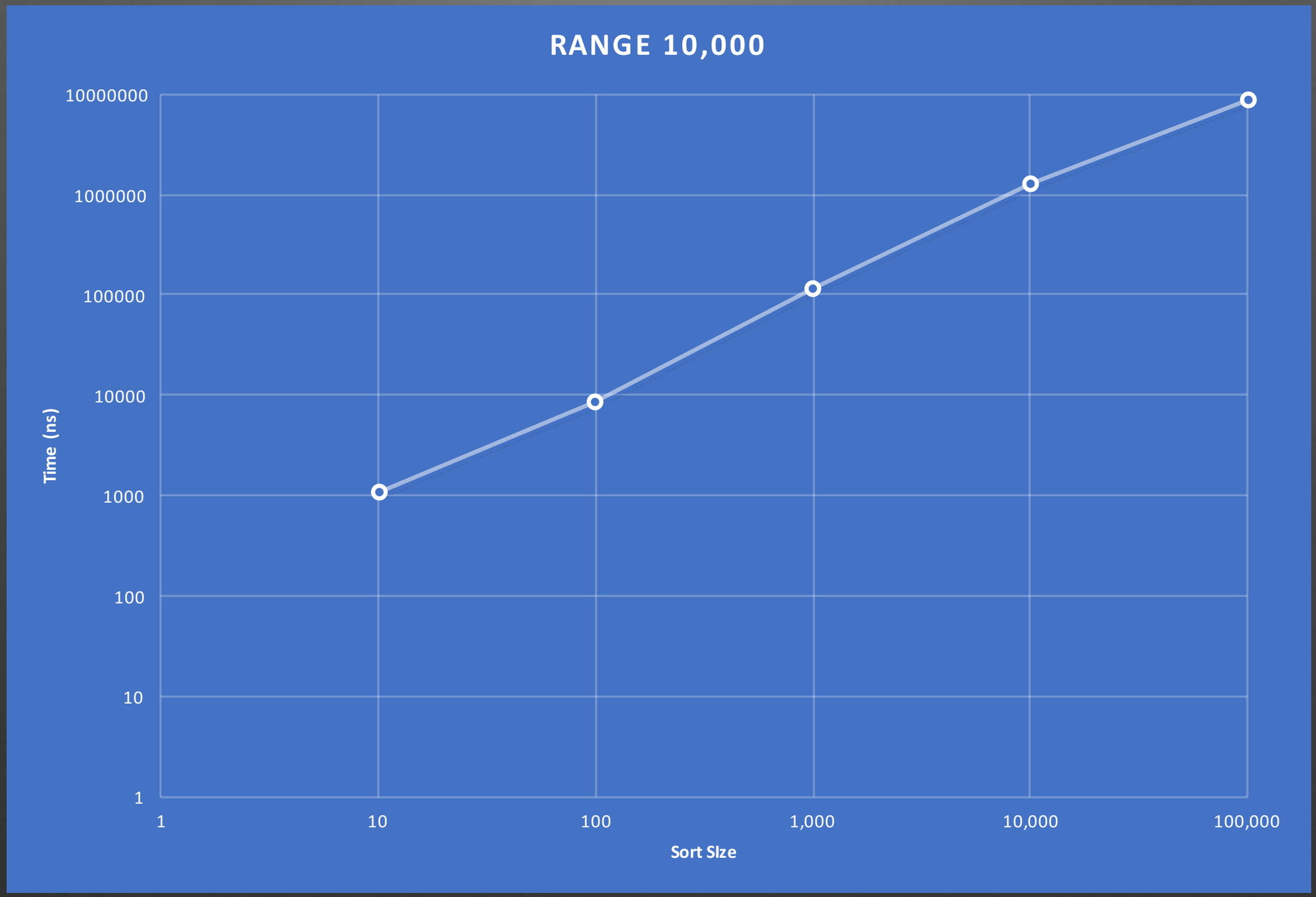
Radix

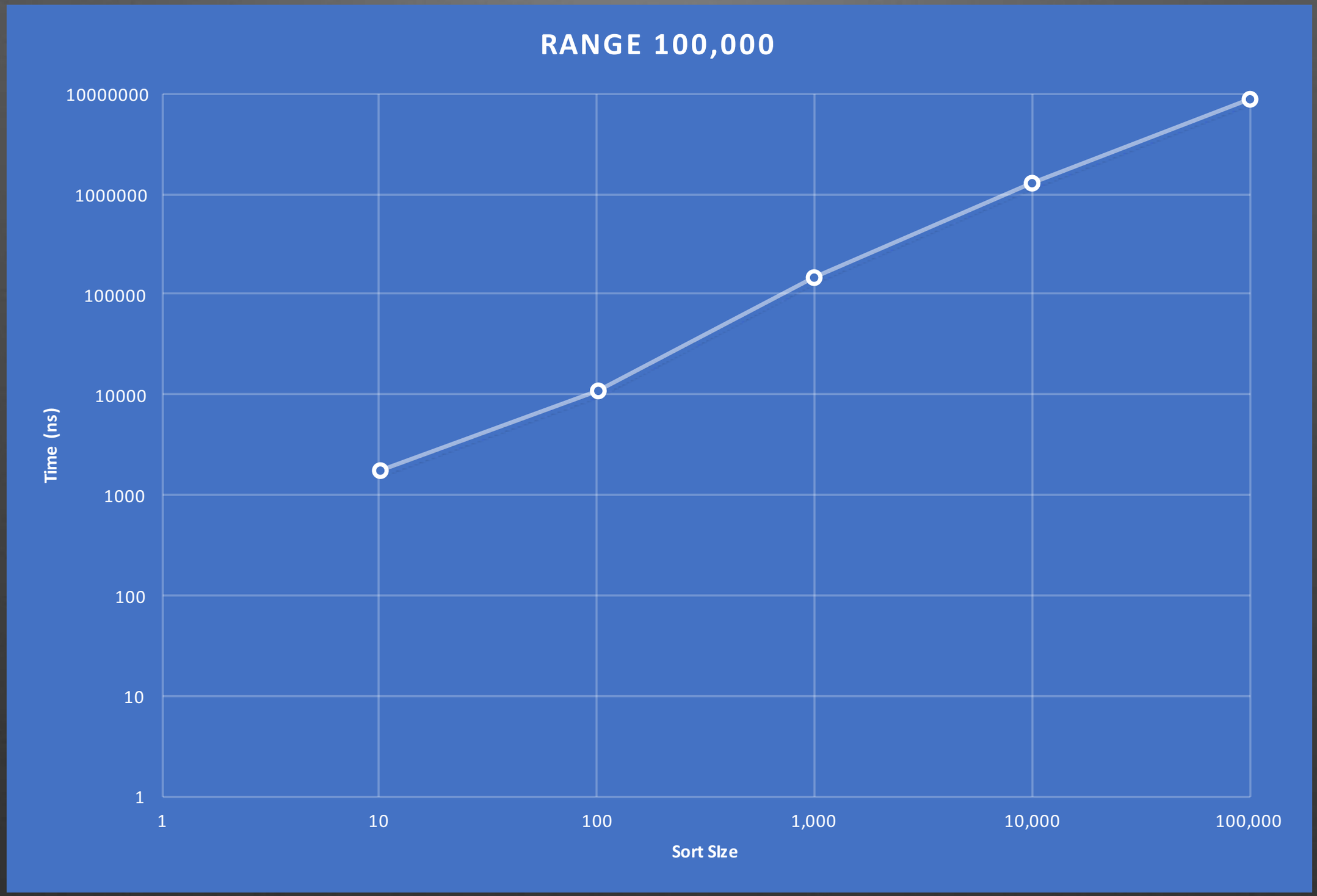
O(*k*n)


RANGE 10
10000000
1000000
100000
10000
1000
100
10
1
Time (ns)
1
10
100
1,000
10,000
100,000
Sort SIze


RANGE 100
10000000
1000000
100000
10000
1000
100
10
1
Time (ns)
1
10
100
1,000
10,000
100,000
Sort SIze

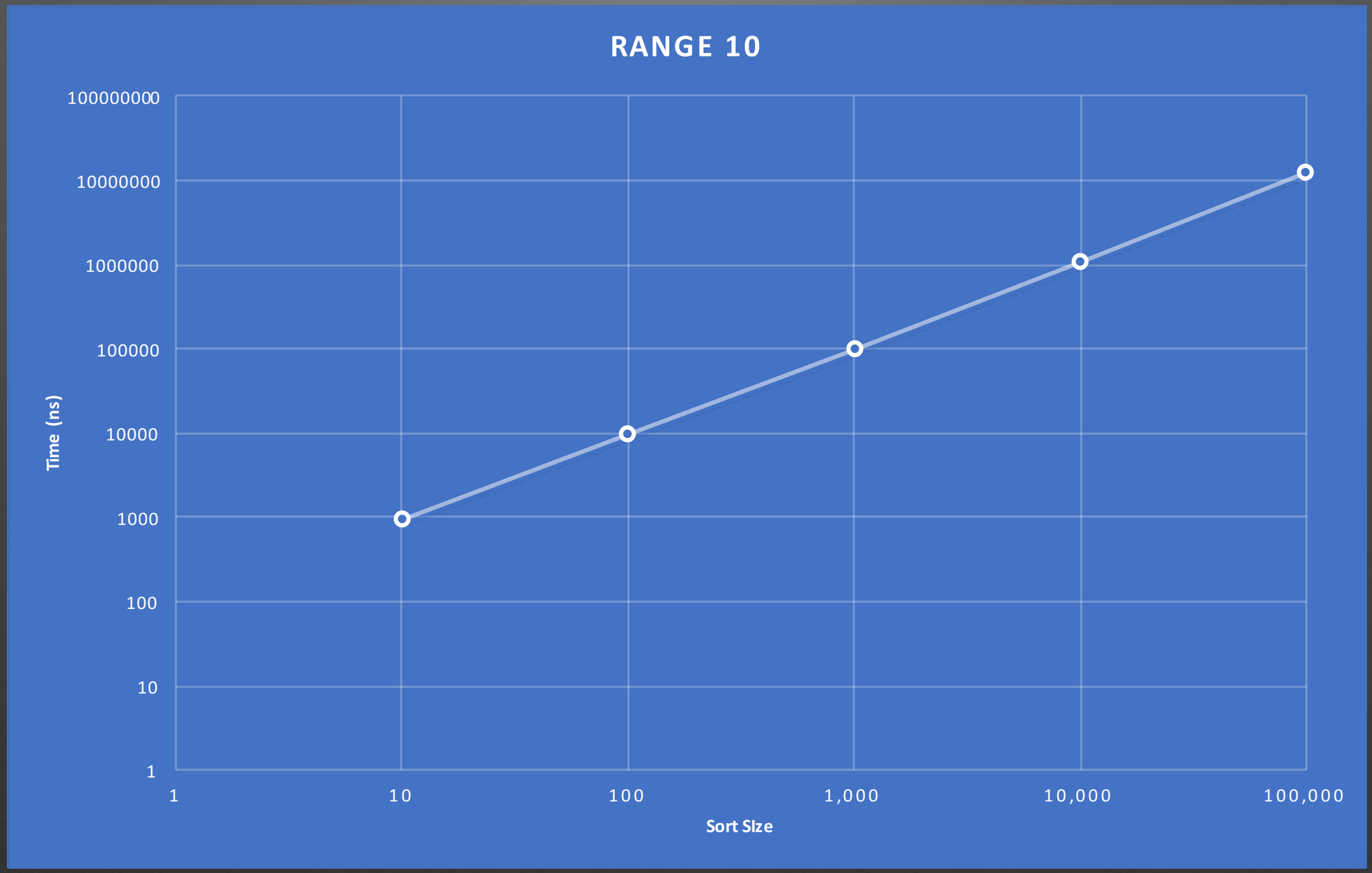

RANGE 1000
10000000
1000000
100000
10000
1000
100
10
1
Time (ns)
1
10
100
1,000
10,000
100,000
Sort SIze


RANGE 10,000
10000000
1000000
100000
10000
1000
100
10
1
Time (ns)
1
10
100
1,000
10,000
100,000
Sort SIze

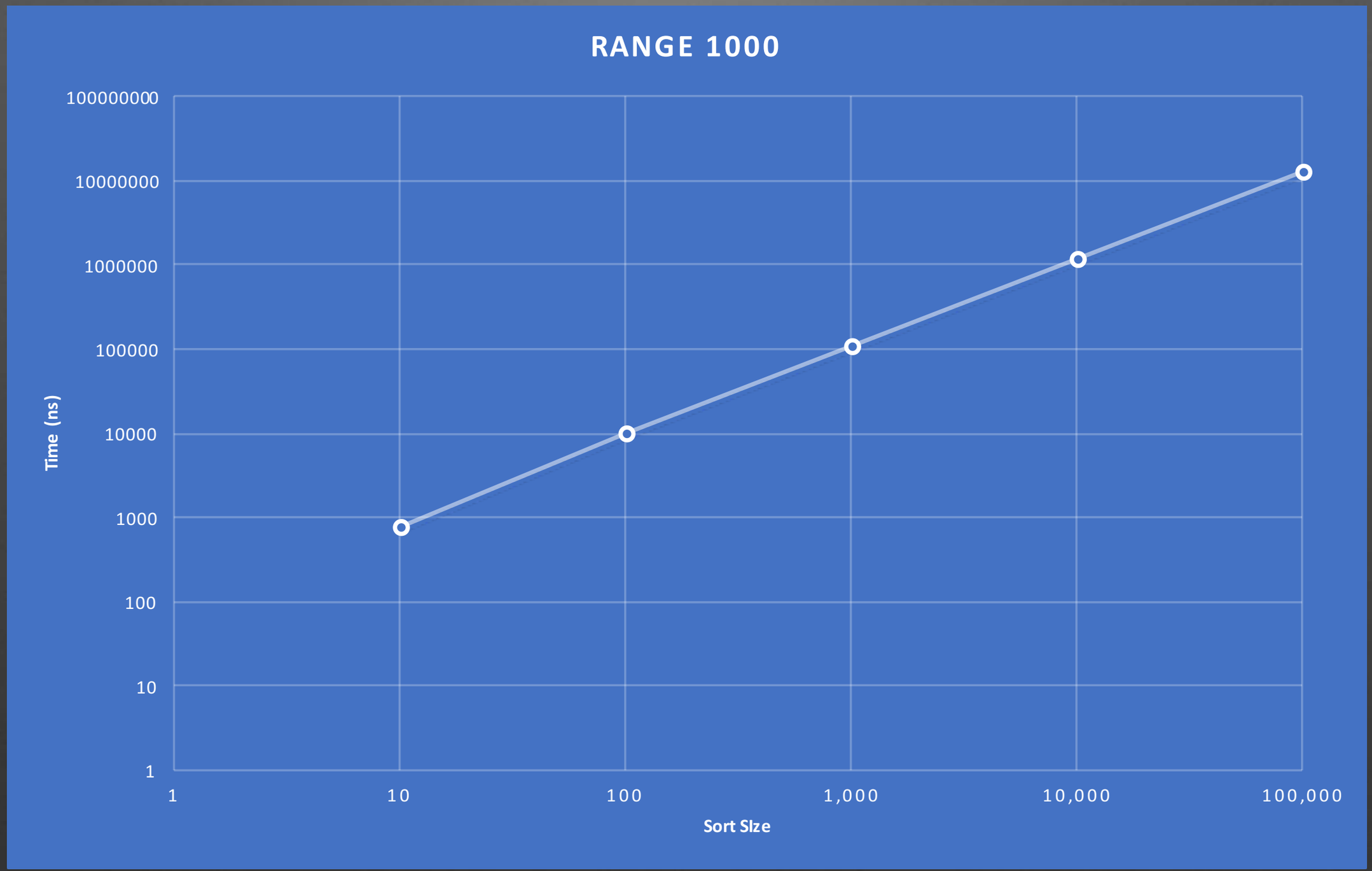

RANGE 100,000
10000000
1000000
100000
10000
1000
100
10
1
Time (ns)
1
10
100
1,000
10,000
100,000
Sort SIze

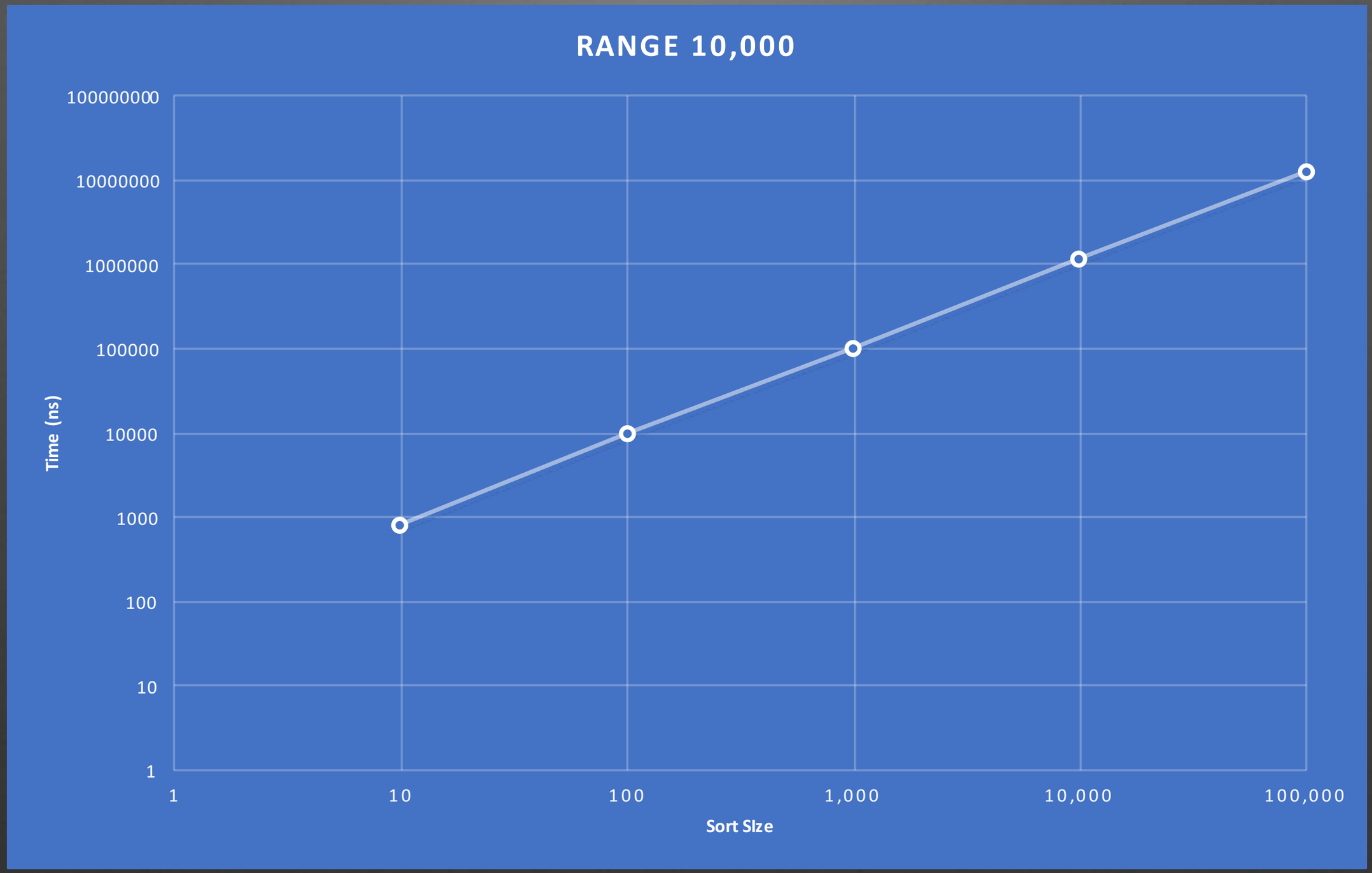
Merge

O(n*log*n)


RANGE 10
100000000
10000000
1000000
100000
10000
1000
100
10
1
Time (ns)
1
10
100
1,000
10,000
100,000
Sort SIze


RANGE 100
100000000
10000000
1000000
100000
10000
1000
100
10
1
Time (ns)
1
10
100
1,000
10,000
100,000
Sort SIze


RANGE 1000
100000000
10000000
1000000
100000
10000
1000
100
10
1
Time (ns)
1
10
100
1,000
10,000
100,000
Sort SIze

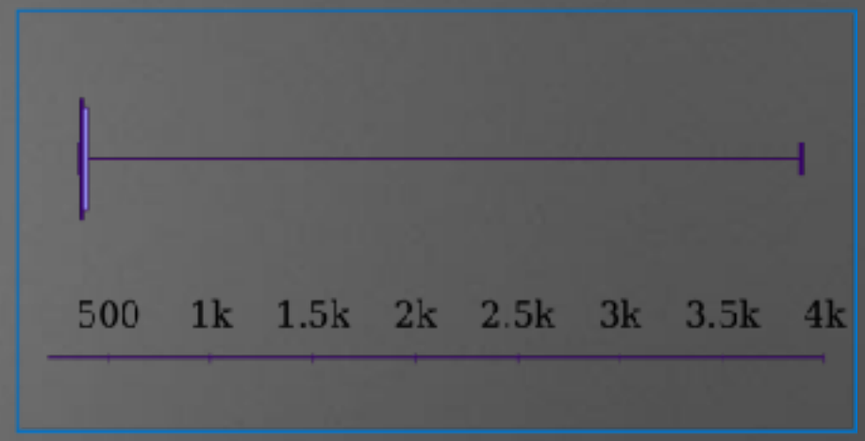

RANGE 10,000
100000000
10000000
1000000
100000
10000
1000
100
10
1
Time (ns)
1
10
100
1,000
10,000
100,000
Sort SIze


RANGE 100,000
100000000
10000000
1000000
100000
10000
1000
100
10
1
Time (ns)
1
10
100
1,000
10,000
100,000
Sort SIze

Error

- Host computer's processes
- Averages
- Clock Precision

Problems Encountered



- Generating random numbers
- Creating box-plots
- Timing

Questions?

http://www.sourcetricks.com/2013/03/radix-sort.html

http://www.sourcetricks.com/2011/06/merge-sort.html