Clare Minnerath

CSC 225 – Dr. Linda Wilkens

November 15, 2017

1. **StopWatch function:**

```
struct stopWatch{
	clock_t start;
	clock_t stop;
};
void startTimer(stopWatch *timer) {
	timer->start = clock();
}
void stopTimer(stopWatch *timer) {
	timer->stop = clock();
}
double getElapsedTime(stopWatch *timer) {
	clock_t elapsedInClocks = timer->stop - timer->start;
	// This returns the time in seconds
	return (double(elapsedInClocks) / double(CLK_TCK));

}
```

2. **Time four "efficient" comparison sorts (Shell sort, quicksort, merge sort, heap sort)**

**Shell sort:**

```
startTimer(&mytimer);
Shellsort<unsigned long>(data, dataSize);
stopTimer(&mytimer);
```

```
table[0][dsi] += getElapsedTime(&mytimer);
```

**Quick sort:**

```
startTimer(&mytimer);
quicksort<unsigned long>(data, dataSize);
stopTimer(&mytimer);
```

```
table[1][dsi] += getElapsedTime(&mytimer);
```

**Merge sort:**

```
startTimer(&mytimer);
mergesort<unsigned long>(data, dataSize);
stopTimer(&mytimer);
```

```
table[2][dsi] += getElapsedTime(&mytimer);
```

**Heap sort:**

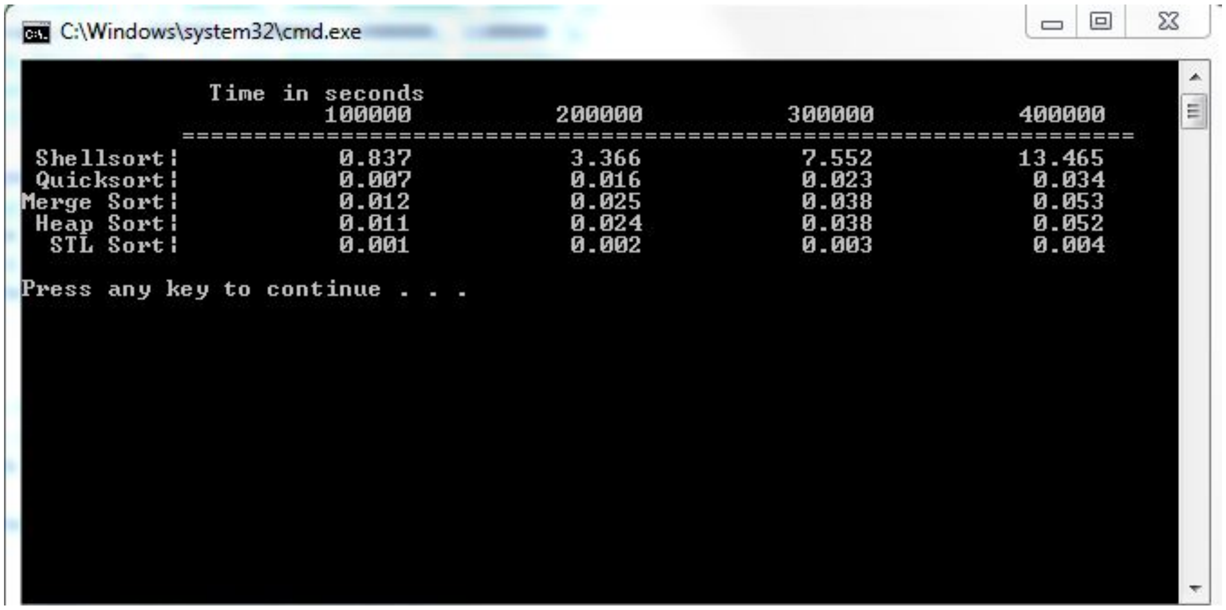
```
startTimer(&mytimer);
heapsort<unsigned long>(data, dataSize);
stopTimer(&mytimer);
```

```
table[3][dsi] += getElapsedTime(&mytimer);
```

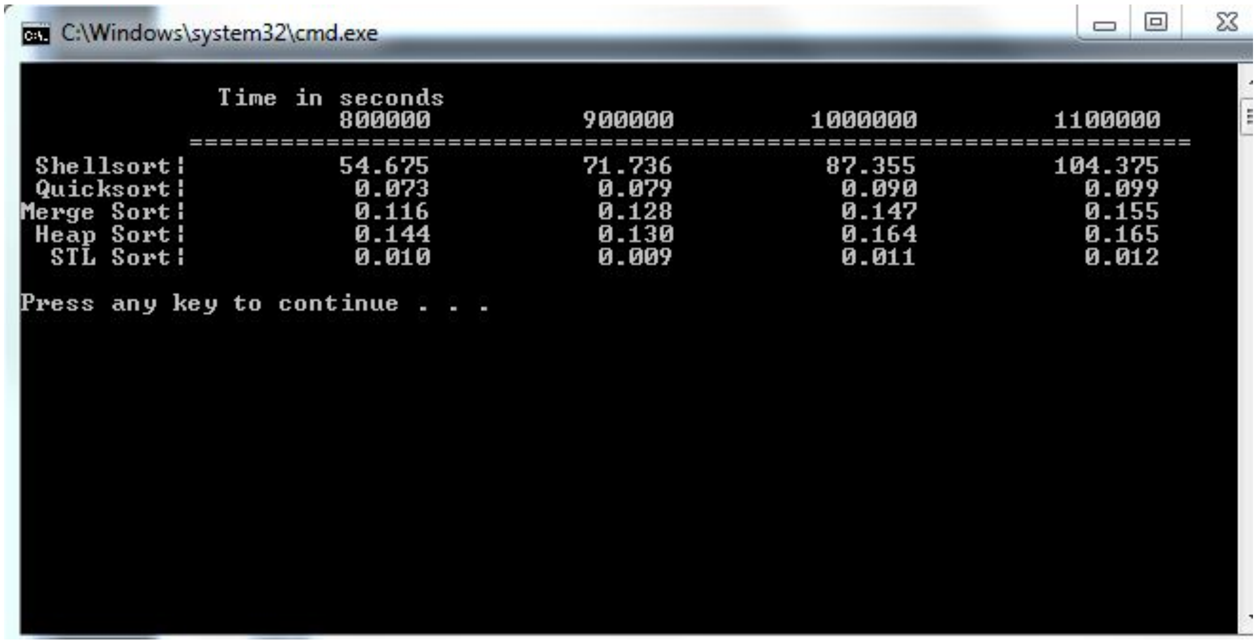
3. **STL sort:**

```
vector<unsigned long> a (dataSize);
	for (i = 0; i < dataSize; i++)
		a[i] = data[i];
	startTimer(&mytimer);
	sort(a.begin(), a.end());
	stopTimer(&mytimer);
	if (issorted(data, dataSize))
		table[4][dsi] += getElapsedTime(&mytimer);
	else
		cout << "Error occurred with STL Sort\n";
```

4. **Screenshots of the output:**

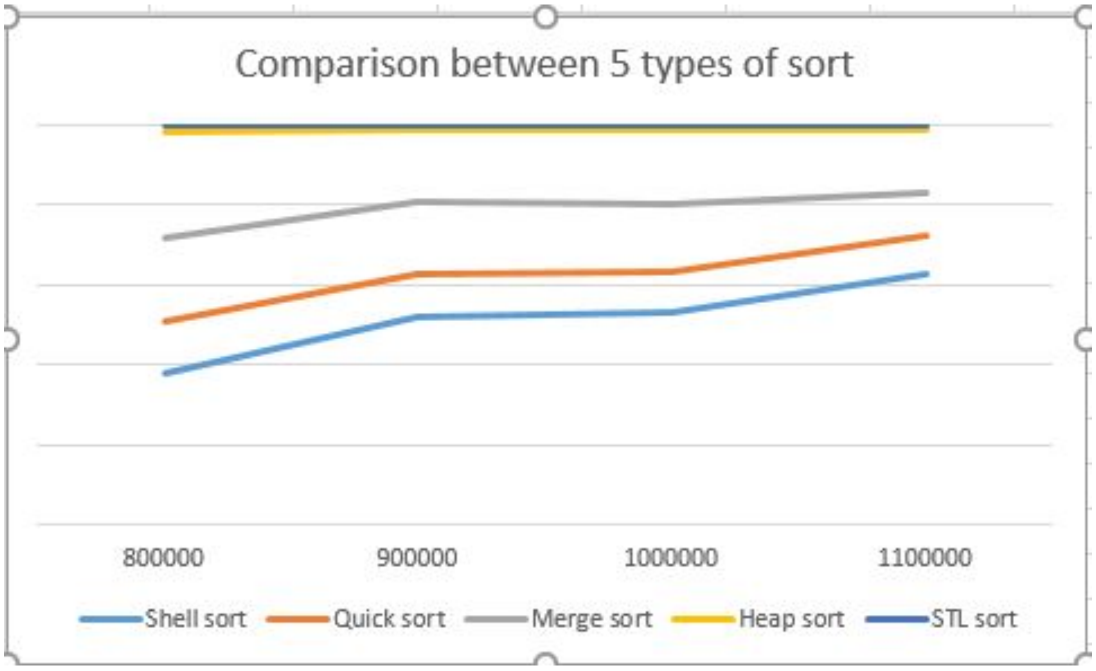


```
             Time  in  seconds
                      100000          200000          300000          400000
    ================================================================
 Shellsort|             0.837           3.366           7.552          13.465
 Quicksort|             0.007           0.016           0.023           0.034
Merge Sort|             0.012           0.025           0.038           0.053
 Heap Sort|             0.011           0.024           0.038           0.052
  STL Sort|             0.001           0.002           0.003           0.004

Press any key to continue . . .
```



```
             Time in seconds
                     800000           900000          1000000          1100000
           ==============================================================
 Shellsort|           54.675           71.736           87.355          104.375
 Quicksort|            0.073            0.079            0.090            0.099
Merge Sort|            0.116            0.128            0.147            0.155
 Heap Sort|            0.144            0.130            0.164            0.165
  STL Sort|            0.010            0.009            0.011            0.012

Press any key to continue . . .
```

5. **Excel illustrations and analysis:**



According to the above graph as well as the data table, we can see that Merge sort, Quick sort and especially STL sort are three most efficient ones. Meanwhile, Heap sort and especially the Shell sort take a long time to process.