George Gabricht
56735102 - ggabrich

ICS 46 - HW 7 Report

```
// O(NlogN -> N^2) Depends on Sorting Algorithm
	double insertAllFromFile(int partition, char *fileName) {
		ifstream ifile(fileName); // 1
		if (!ifile.is_open()) { // 1
			cerr << "File: " << fileName << " couldn't be opened." << endl;
			return -1;
		}

		string curLine;
		for (int ndx = 0; ndx < partition && getline(ifile, curLine); ndx++) { // N
			words[size++] = curLine; // 1
		}

		Timer tme; // 1
		double eTime; // 1

		tme.start(); // 1
		sort(); // Depends on Sorting Algorithm
		tme.elapsedUserTime(eTime); // 1

		ifile.close(); // 1
		return eTime;
	}

// O(N^2)
	virtual void InsertionSorter::sort() override {
		for (int sorted = 1; sorted < size; sorted++) { // N
			for (int ndx = sorted; ndx > 0 && words[ndx - 1] > words[ndx]; ndx--) {//N
				string temp = words[ndx]; // 1
				words[ndx] = words[ndx - 1]; // 1
				words[ndx - 1] = temp; // 1
			}
		}
	}

// O(N^2)
	void insertionSort(int low, int high) {
		for (int sorted = low + 1; sorted < high + 1; sorted++) { // N
			// N
			for (int ndx = sorted; ndx > low && words[ndx - 1] > words[ndx]; ndx--) {
				words[ndx].swap(words[ndx - 1]); // 1
			}
		}
	}
```

George Gabricht
56735102 - ggabrich

```
	// O(1)
	string findPivot(int low, int high) {
		int mid = low + (high - low) / 2; // 1
		string temp; // 1
		if (words[mid] < words[low]) { // 1
			words[mid].swap(words[low]); // 1
		}
		if (words[high] < words[low]) { // 1
			words[high].swap(words[low]); // 1
		}
		if (words[mid] < words[high]) { // 1
			words[mid].swap(words[high]); // 1
		}
		return words[high]; // 1
	}

	// O(N)
	int partition(int low, int high, string piv) {
		int bot = low, top = high - 1; // 1
		while(true) { // 1
			while(words[bot] < piv) { // O(N)
				bot++; // 1
			}
			while(piv < words[top]) { // O(N)
				top--; // 1
			}
			if(bot < top) { // 1
				words[bot++].swap(words[top--]); // 1
			} else { // 1
				break; // 1
			}
		}
		words[bot].swap(words[top]); // 1
		return bot; // 1
	}

// O(N log N)
	void quickSort(int low, int high) {
		if (high - low < 16) { // log N
			insertionSort(low, high); // 1
		} else { // N log N
			string piv = findPivot(low, high); // 1
			int ndx = partition(low, high, piv); // N log N
			quickSort(low, ndx - 1); // log N
			quickSort(ndx + 1, high); // log N
		}
	}
```

George Gabricht
56735102 - ggabrich

```
// O(N log N)
	virtual void QuickSorter::sort() override {
		quickSort(0, size - 1);
	}

// left child
	// O(1)
	int leftChild(int ndx) {
		return 2 * ndx + 1;
	}
	// right child
	// O(1)
	int rightChild(int ndx) {
		return 2 * ndx + 2;
	}
	// convert to binary heap
	// O(log N)
	void heapify(int root, int size) {
		int max = root; // 1
		int left = leftChild(root); // 1
		int right = rightChild(root); // 1
		if (left < size && words[left] > words[max]) { // 1
			max = left; // 1
		}
		if (right < size && words[right] > words[max]) { // 1
			max = right; // 1
		}
		if (max != root) { // 1
			words[root].swap(words[max]); // 1
			heapify(max, size); // log N
		}
	}

// O(N log N)
	virtual void HeapSorter::sort() override {
		// convert array to binary heap
		for (int ndx = size / 2 - 1; ndx >= 0; ndx--) { // N / 2
			heapify(ndx, size); // log N
		}
		for (int ndx = size - 1; ndx >= 0; ndx--) { // N
			words[0].swap(words[ndx]); // 1
			// sift-down to move largest val to root
			heapify(0, ndx); // log N
			// place largest value at end of heap
			// decrease heap size by one
			// continue to root
		}
	}
```

George Gabricht
56735102 - ggabrich

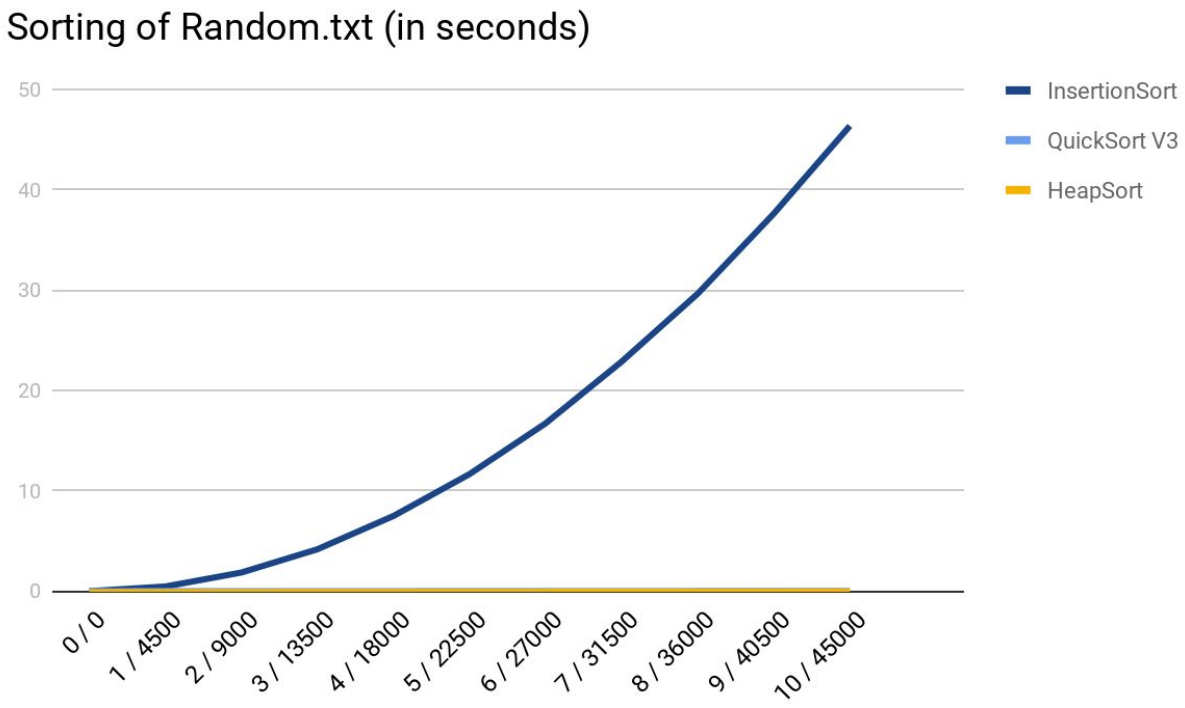
```
$ test_sort
Measuring Sorting of random.txt
Partition: 1    | InsertionSort: 0.46669        | QuickSort: 0.00276    | HeapSort: 0.00528
Partition: 2    | InsertionSort: 1.85933        | QuickSort: 0.00596    | HeapSort: 0.01165
Partition: 3    | InsertionSort: 4.17970        | QuickSort: 0.00944    | HeapSort: 0.01843
Partition: 4    | InsertionSort: 7.49826        | QuickSort: 0.01376    | HeapSort: 0.02585
Partition: 5    | InsertionSort: 11.67076       | QuickSort: 0.01644    | HeapSort: 0.03309
Partition: 6    | InsertionSort: 16.72689       | QuickSort: 0.02006    | HeapSort: 0.04043
Partition: 7    | InsertionSort: 22.87791       | QuickSort: 0.02349    | HeapSort: 0.04823
Partition: 8    | InsertionSort: 29.64529       | QuickSort: 0.02759    | HeapSort: 0.05555
Partition: 9    | InsertionSort: 37.62222       | QuickSort: 0.03339    | HeapSort: 0.06384
Partition: 10   | InsertionSort: 46.36060       | QuickSort: 0.03590    | HeapSort: 0.07177

Measuring Sorting of words.txt
Partition: 1    | InsertionSort: 0.11828        | QuickSort: 0.00378    | HeapSort: 0.00530
Partition: 2    | InsertionSort: 0.51085        | QuickSort: 0.00985    | HeapSort: 0.01147
Partition: 3    | InsertionSort: 0.83941        | QuickSort: 0.01222    | HeapSort: 0.01793
Partition: 4    | InsertionSort: 1.59626        | QuickSort: 0.02081    | HeapSort: 0.02463
Partition: 5    | InsertionSort: 2.64673        | QuickSort: 0.03664    | HeapSort: 0.03150
Partition: 6    | InsertionSort: 5.04753        | QuickSort: 0.05348    | HeapSort: 0.03869
Partition: 7    | InsertionSort: 6.47790        | QuickSort: 0.10019    | HeapSort: 0.04529
Partition: 8    | InsertionSort: 7.76479        | QuickSort: 0.11462    | HeapSort: 0.05256
Partition: 9    | InsertionSort: 9.04191        | QuickSort: 0.09349    | HeapSort: 0.06047
Partition: 10   | InsertionSort: 11.39248       | QuickSort: 0.08194    | HeapSort: 0.06768
ggabrich@andromeda-48 22:58:40 ~/ics46/hw/ggabrich_hw7
$ valgrind test_sort random_small.txt words_small.txt
==30434== Memcheck, a memory error detector
==30434== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==30434== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==30434== Command: test_sort random_small.txt words_small.txt
==30434==
Measuring Sorting of random_small.txt
Partition: 1    | InsertionSort: 0.01915        | QuickSort: 0.01018    | HeapSort: 0.00642
Partition: 2    | InsertionSort: 0.01298        | QuickSort: 0.00121    | HeapSort: 0.00238
Partition: 3    | InsertionSort: 0.01331        | QuickSort: 0.00122    | HeapSort: 0.00248
Partition: 4    | InsertionSort: 0.01348        | QuickSort: 0.00122    | HeapSort: 0.00246
Partition: 5    | InsertionSort: 0.01344        | QuickSort: 0.00124    | HeapSort: 0.00244
Partition: 6    | InsertionSort: 0.01352        | QuickSort: 0.00122    | HeapSort: 0.00245
Partition: 7    | InsertionSort: 0.01244        | QuickSort: 0.00123    | HeapSort: 0.00247
Partition: 8    | InsertionSort: 0.01354        | QuickSort: 0.00121    | HeapSort: 0.00243
Partition: 9    | InsertionSort: 0.02065        | QuickSort: 0.00123    | HeapSort: 0.00243
Partition: 10   | InsertionSort: 0.01384        | QuickSort: 0.00103    | HeapSort: 0.00249

Measuring Sorting of words_small.txt
Partition: 1    | InsertionSort: 0.00081        | QuickSort: 0.00192    | HeapSort: 0.00313
Partition: 2    | InsertionSort: 0.00086        | QuickSort: 0.00192    | HeapSort: 0.00313
Partition: 3    | InsertionSort: 0.00086        | QuickSort: 0.00193    | HeapSort: 0.00315
Partition: 4    | InsertionSort: 0.00086        | QuickSort: 0.00194    | HeapSort: 0.00312
Partition: 5    | InsertionSort: 0.00087        | QuickSort: 0.00194    | HeapSort: 0.00312
Partition: 6    | InsertionSort: 0.00086        | QuickSort: 0.00193    | HeapSort: 0.00316
Partition: 7    | InsertionSort: 0.00087        | QuickSort: 0.00193    | HeapSort: 0.00311
Partition: 8    | InsertionSort: 0.00088        | QuickSort: 0.00193    | HeapSort: 0.00316
Partition: 9    | InsertionSort: 0.00088        | QuickSort: 0.00194    | HeapSort: 0.00318
Partition: 10   | InsertionSort: 0.00087        | QuickSort: 0.00194    | HeapSort: 0.00317
==30434==
==30434== HEAP SUMMARY:
==30434==     in use at exit: 0 bytes in 0 blocks
==30434==   total heap usage: 2,041 allocs, 2,041 frees, 48,470,084 bytes allocated
==30434==
==30434== All heap blocks were freed -- no leaks are possible
==30434==
==30434== For counts of detected and suppressed errors, rerun with: -v
==30434== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

George Gabricht
56735102 - ggabrich

Sorting of Random.txt (in seconds)			
Partition / # Words	InsertionSort	QuickSort V3	HeapSort
1 / 4500	0.46669	0.00276	0.00528
2 / 9000	1.85933	0.00596	0.01165
3 / 13500	4.17970	0.00944	0.01843
4 / 18000	7.49826	0.01376	0.02585
5 / 22500	11.67076	0.01644	0.03309
6 / 27000	16.72689	0.02006	0.04043
7 / 31500	22.87791	0.02349	0.04823
8 / 36000	29.64529	0.02759	0.05555
9 / 40500	37.62222	0.03339	0.06384
10 / 45000	46.36060	0.03590	0.07177



George Gabricht
56735102 - ggabrich

Sorting of Words.txt (in seconds)			
Partition / # Words	InsertionSort	QuickSort V3	HeapSort
1 / 4500	0.11828	0.00378	0.00530
2 / 9000	0.51085	0.00985	0.01147
3 / 13500	0.83941	0.01222	0.01793
4 / 18000	1.59626	0.02081	0.02463
5 / 22500	2.64673	0.03664	0.03150
6 / 27000	5.04753	0.05348	0.03869
7 / 31500	6.47790	0.10019	0.04529
8 / 36000	7.76479	0.11462	0.05256
9 / 40500	9.04191	0.09349	0.06047
10 / 45000	11.39248	0.08194	0.06768