Week 09 Lab: Sort Detective

Griffin Doyle - z5311098

Introduction:

The purpose of the report is to analyse two unseen sorting functions by testing their response to various sets of data.

Sorting Algorithms:The various sorting algorithms are as follows:

Bubble Sort:

- Passes through the list of data and swaps adjacent elements if they are in the wrong order. May take multiple passes to completely sort the list. (stable)
- **Time Complexity:**
  - **Worst:** N^2
  - **Average:** N^2
  - **Best:** N
- **Resources:**
  - https://www.geeksforgeeks.org/bubble-sort/ (Accessed: 14/04)
  - https://en.wikipedia.org/wiki/Bubble_sort (Accessed: 14/04)

Insertion Sort:

- Passes through the list of data and compares the current item with its predecessors to find where it should be placed into the list, only needs one pass. (stable)
- **Time Complexity:**
  - **Worst:** N^2
  - **Average:** N^2
  - **Best:** N
- **Resources:**
  - https://www.geeksforgeeks.org/insertion-sort/ (Accessed: 14/04)

Selection Sort:

- Finds the minimum of list [j:n] (where n is the number of elements and j iterates from 0 to n) and places it at the start of that interval, only one pass required. (not stable)
- **Time Complexity:**
  - **Worst:** N^2
  - **Average:** N^2
  - **Best:** N^2
- **Resources:**
  - https://www.geeksforgeeks.org/selection-sort/ (Accessed: 14/04)

Shell Sort with gaps 1, 8, 23, 77, 281, 1073, 4193, ...:

- Based on the insertion sort, starts by swapping corresponding pairs if they are not in order that are a specific number apart. The gap gets smaller and smaller until it is 1, where we just do a standard insertion sort. (Not stable)
- **Time Complexity:**
  - **Worst:** depends on the gaps
  - **Average:** $N \log(N)^2$ or $n^{\frac{3}{2}}$
  - **Best:** N
- **Resources:**
  - https://en.wikipedia.org/wiki/Shellsort
  - https://www.geeksforgeeks.org/shellsort/
  - https://www.tutorialspoint.com/data_structures_algorithms/shell_sort_algorithm.htm

Merge sort:

- Recursively splits up the list of data and then re-merges those split lists in order. (stable)
- **Time Complexity:**
  - **Worst:** $N \log(n)$
  - **Average:** $N \log(n)$
  - **Best:** $N \log(n)$
- **Resources:**
  - https://www.geeksforgeeks.org/merge-sort/

Naive Quicksort (leftmost pivot, median-of-three, random):

- Uses a pivot item to split the list of input data such that lower than the pivot is less and higher than the pivot is greater (not stable)
- Pivot items are either the first item, median of first, middle and last item or random item
- **Time Complexity:**
  - **Worst:** N^2
  - **Average:** $N \log(N)$
  - **Best:** $N \log(n)$
- **Resources:**
  - https://www.geeksforgeeks.org/quick-sort/

Bogosort:

- Generates a random permutation of the input list until it is sorted (stable)
- **Time Complexity:**
  - **Worst:** ∞
  - **Average:** $n \times n!$
  - **Best:** n
- **Resources:**

    - https://en.wikipedia.org/wiki/Bogosort

Stable:

- Bubble sort
- Insertion sort
- Merge sort
- Bogosort

Unstable:

- Selection sort
- Shellsort
- Quicksort

Data lists:

Stability:

- In order repetition - 10
- Reverse repetition - 10
- Random repetition - 10

Complexity:

- Random - no repetition - 1000
- Random - no repetition - 2000
- Random - no repetition - 5000
- Random - no repetition - 10000
- In Order- no repetition - 1000
- In Order- no repetition - 2000
- In Order- no repetition - 5000
- In Order- no repetition - 10000
- Reverse- no repetition - 1000
- Reverse - no repetition - 2000
- Reverse - no repetition - 5000
- Reverse - no repetition - 10000

Results:

Sort A:

- Sort A is a stable sorting algorithm

- The execution time of the sorting algorithm grows dramatically as input size increases
- Reverse ordered input is the worst case, being marginally longer than random input. In order input is almost instantly sorted.

Sort B:

- Sort B is an unstable sorting algorithm
- The execution also increases dramatically as the input data increases in size although not to the extent of Sort A
- The reverse ordered input is also the worst case in this sorting algorithm with in order input just behind it. Random ordered input is by far the quickest and is almost instant

Conclusion:

Sort A:

As sort A is a stable sorting algorithm, we are able to cut down the possible algorithms to just Bubble Sort, Insertion Sort, Merge Sort and Bogosort. We can immediately eliminate Bogosort as we achieved a sensible execution time. Sort A cannot be Merge sort as the time complexity is a constant $O(n \log n)$ which is not true for the case of in order sort being drastically faster than random or reverse input. Insertion sort and Bubble sort are very similar and my best guess is bubble sort. As the reversed and random cases have similar execution times, I imagine it is because the number of comparisons in bubble sort is going to be the same for the same size data so even if it is a better order in the random order than the reversed order, the amount of operations remains basically the same. Final answer: **Bubble Sort**

Sort B:

Sort B is an unstable sorting algorithm so our possible sorting algorithms could be selection sort, shell sort or quicksort. Looking at the speed of the worst case scenario in reversed data, we can see that it is far lower than Sort A. This leads me to believe that the worst case is still very efficient, i.e. not $$ O(N^2) $$. We can also deduce that since the random dataset takes a very short amount of time that the algorithm cannot be selection sort. By considering all of this information, I have come to the conclusion that Sort B is **Shell Sort**.

Appendix:

Sort A:

Stability:

Input	Output
1 a first 2 b second 3 c third 4 d fourth 5 e fifth	first second third fourth fifth 1 a 2 b 3 c 4 d 5 e
3 c first second 4 d 1 a third 2 b 5 e fourth fifth	first second third fourth fifth 1 a 2 b 3 c 4 d 5 e
5 e first 4 d second 3 c third 2 b fourth 1 a fifth	first second third fourth fifth 1 a 2 b 3 c 4 d 5 e

Complexity:

Order	Size	Time (average of five)
random	10000	0.43
sorted	10000	0.00
reversed	10000	0.48
random	20000	1.81
sorted	20000	0.00
reversed	20000	1.96
random	40000	7.54

sorted	40000	0.00
reversed	40000	7.84
random	80000	30.79
sorted	80000	0.00
reversed	80000	31.36
random	160000	124.03
sorted	160000	0.02
reversed	160000	125.58

Sort B:

Stability

Input	**Output**
1 a first 2 b second 3 c third 4 d fourth 5 e fifth	first fifth second fourth third 1 a 2 b 3 c 4 d 5 e
5 e first 4 d second 3 c third 2 b fourth 1 a fifth	first fifth second fourth third 1 a 2 b 3 c 4 d 5 e
3 c first second 4 d 1 a third	first second fifth fourth third 1 a

2 b 5 e fourth fifth	2 b 3 c 4 d 5 e

Complexity:

Order	**Size**	**Time (average of five)**
random	10000	0.00
sorted	10000	0.13
reversed	10000	0.15
random	20000	0.00
sorted	20000	0.56
reversed	20000	0.60
random	40000	0.01
sorted	40000	2.25
reversed	40000	2.40
random	80000	0.02
sorted	80000	8.99
reversed	80000	9.60
random	160000	0.05
sorted	160000	35.95
reversed	160000	38.35