Blair Hagen

Lab 7 Report

Partner: Katrina Francis

CSC-150

5-12-16

In this lab, we analyzed the runtime of six different sorting methods as they sorted five different arrays of integers. Our five array types were random, mostly sorted, reverse sorted, sorted, and constant (all the same integers). We tested each method with varying array sizes, sizing them so the largest number is close to producing a stack or memory error, or running for longer than five minutes. Here are the worst case runtime scenarios for each of the sorting methods:

Method 1:

Worst case runtime comes from inputting a reverse sorted array. The worst case runtime is O(n2) because a doubling of input values equals a quadrupling of operations needed to sort the array, and this is maintained for both small values and large values, ruling out the possibility of it being O(n log(n)).

|  |  |
| --- | --- |
| n | opCount |
| 10000 | 399980000 |
| 20000 | 1599960000 |
| 30000 | 3599940000 |
| 70000 | 19599860000 |
| 80000 | 25599840000 |
| 90000 | 32399820000 |
| 130000 | 67599740000 |
| 140000 | 78399720000 |
| 150000 | 89999700000 |
| 300000 | 3.59999E+11 |

Method 2:

Worst case runtime comes from inputting a mostly sorted array and a sorted array. The worst case runtime is O(n log(n)) because at smaller values it appears to be O(n2), but with larger input that ratio approached O(n).

|  |  |
| --- | --- |
| n | opCount |
| 1000 | 506494 |
| 2000 | 2012994 |
| 3000 | 4519494 |
| 7000 | 24545494 |
| 8000 | 32051994 |
| 9000 | 40558494 |
| 13000 | 84584494 |
| 14000 | 98090994 |
| 15000 | 112597494 |
| 19000 | 180623494 |

Method 3:

Worst case runtime comes from inputting a mostly sorted or sorted array. The runtime is O(n), as every array sorted returned a linear plot. If the input values were doubled, so were the operations needed to sort the array.

|  |  |
| --- | --- |
| n | opCount |
| 1000000 | 102938550 |
| 2000000 | 215877110 |
| 3000000 | 334389304 |
| 7000000 | 821842614 |
| 8000000 | 943508470 |
| 9000000 | 1070646712 |
| 13000000 | 1585423670 |
| 14000000 | 1713685238 |
| 15000000 | 1840653366 |
| 2E+08 | 2.8332E+10 |

Method 4:

Worst case runtime was the same for all array types with a runtime of O(n2). The runtime is O(n2) because a doubling of input values equals a quadrupling of operations needed to sort the array. Again this is maintained for both small and large values.

|  |  |
| --- | --- |
| n | opCount |
| 500 | 127746 |
| 5000 | 12527496 |
| 10000 | 50054996 |
| 15000 | 112582496 |
| 20000 | 200109996 |
| 30000 | 450164996 |
| 50000 | 1250274996 |
| 80000 | 3200439996 |
| 100000 | 5000549996 |
| 200000 | 20001099996 |

Method 5:

Worst case runtime comes from inputting a mostly sorted or sorted array. The runtime is O(n) as every array sorted returned a linear plot. If the input values were doubled, so were the operations needed to sort the array.

|  |  |
| --- | --- |
| n | opCount |
| 1000000 | 102938550 |
| 2000000 | 215877110 |
| 3000000 | 334389304 |
| 7000000 | 821842614 |
| 8000000 | 943508470 |
| 9000000 | 1070646712 |
| 13000000 | 1585423670 |
| 14000000 | 1713685238 |
| 15000000 | 1840653366 |
| 200000000 | 2.8332E+10 |

Method 6:

Worst case runtime comes from inputting a reverse sorted array. The runtime is O(n2) because a doubling of input values equals a quadrupling of operations needed to sort the array. Again this is maintained for both small and large values.

|  |  |
| --- | --- |
| n | opCount |
| 500 | 376747 |
| 5000 | 37517497 |
| 10000 | 150034997 |
| 15000 | 337552497 |
| 20000 | 600069997 |
| 30000 | 1350104997 |
| 50000 | 3750174997 |
| 80000 | 9600279997 |
| 100000 | 15000349997 |
| 200000 | 60000699997 |