22C Fall 2017 Homework 5 Report

Velly Simeonov

Testing between heapsort and selection sort shows just how much more effective heapsort is than selection sort on large arrays, but for smaller arrays (1000 or less) selection sort is perhaps better. Below is a table showing time results from Visual Studio’s debugger for different size arrays.

|  |  |  |
| --- | --- | --- |
| Number of Elements | Heap Sort ( <= ms) | Selection Sort ( <= ms) |
| 10 | 1 + 1 = 2 | 1 |
| 100 | 1 + 1 = 2 | 1 |
| 1,000 | 1 + 3 = 4 | 2 |
| 10,000 | 5 + 30 = 35 | 111 |
| 100,000 | 50 + 389 = 439 | 11,249 |
| 1,000,000 | 487 + 4,849 = 5,336 | >300,000 |

Downfalls for the data table results is the minimum measurable time span is 1ms, so any calculations that take less time than that are rounded up. Heap Sort works in two stages, it builds a heap out of an array, which for 100 items or less may take less than 1ms, then it sorts the array, which also may take less than 1ms, so summing these two stages may erroneously give a time span far longer than the actual heap sort took. Perhaps next time I’ll make a subroutine to do both steps in one line, and the debugger will treat them as one operation.

But regardless of the downfalls, it becomes obvious which sorting method is more effective on arrays of size 10,000 or more.

For 1,000,000 elements selection sort was taking enough time that I figured to stop it and show how long I waited, the real time it would have taken to sort all the elements is unnecessary to prove the point.

Data Table below shows a different run and the operations for each sort.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Heap Sort | | Selection Sort | |
| Elements | Swaps | Comparisons | Swaps | Comparisons |
| 10 | 28 | 92 | 7 | 130 |
| 100 | 571 | 2,146 | 96 | 10,300 |
| 1,000 | 9081 | 34,862 | 981 | 1,003,000 |
| 10,000 | 123,707 | 480,768 | 9,939 | 100,030,000 |
| 100,000 | 1,568,070 | 6,133,201 | 99,499 | 1,410,365,408 |

From the data in the table above, one can conclude that swapping, which involves 3 assignments is significantly faster than a single conditional statement. So I modified the selection sort not to check if the current pointed to item is the same one as the one it will be swapped to (because it is the smallest value in the remaining array). The next table shows the results from the next run with the new selection sort.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Heap Sort | | Selection Sort | |
| Elements | Swaps | Comparisons | Swaps | Comparisons |
| 10 | 29 | 95 | 10 | 120 |
| 100 | 574 | 2,159 | 100 | 10,200 |
| 1,000 | 9,056 | 34,781 | 1,000 | 1,002,000 |
| 10,000 | 123,792 | 481,150 | 10.000 | 100,020,000 |
| 100,000 | 1,568,634 | 6,135,307 | 100,000 | 1,410,265,408 |

As the array size increases the benefit to the changed selection sort becomes relatively less in which case heap sort should be used any way.