CS316 Project 4 Running time Report, April 2015

|  |  |  |
| --- | --- | --- |
| Name of the image processed: Buchtel Hall | | |
|  |  |  |
| Sorting algorithm | Window size | Recorded Running time |
| Insertion sort | 5 | 29.877 ms |
| Insertion sort | 11 | 427.109 ms |
| Insertion sort | 23 | 6,510.52 ms |
| Insertion sort | 47 | 105,105 ms |
| Insertion sort | 97 | 1,371,410 ms |
|  |  |  |
| Sorting algorithm | Window size | Recorded Running time |
| Quicksort sort | 5 | 33.857 ms |
| Quicksort sort | 11 | 224.914 ms |
| Quicksort sort | 23 | 1,155.48 ms |
| Quicksort sort | 47 | 5,883.99 ms |
| Quicksort sort | 97 | 31,602.2 ms |
|  |  |  |
| Sorting algorithm (list the name) | Window size | Recorded Running time |
| Radix | 5 | 60.01 ms |
| Radix | 11 | 267.818 ms |
| Radix | 23 | 1,060.16 ms |
| Radix | 47 | 4,105.36 ms |
| Radix | 97 | 14,151.6 ms |
|  |  |  |
|  |  |  |
|  |  |  |

The asymptotic running time of the three algorithms:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Time complexity | | |
|  | Best case | Average case | Worst case |
| Insertion sort | O(n) | O(n^2) | O(n^2) |
| Quicksort sort | O(n\*log(n)) | O(n\*log(n)) | O(n^2) |
| Your alg (name it) |  |  |  |
| Radix | O(nk) | O(nk) | O(nk) |

Did your implementation meet your expectation? Why?

Overall out implementation did meet our expectations because the time complexities ran comparably to the expected best, average and worst cases listed in the table above. Insertion sort was by far the slowest algorithm at the highest window size, taking about 22 minutes for a window size of 97. On the contrary, radix sort was the fastest with large window sizes, taking only 14 seconds for a window size of 97. Despite insertion sort's poor performance on large window sizes, it outperforms both quick and radix sort with a window size of 5. Being that insertion sort is an n^2 algorithm, the actual times reflect that. The same is true for quicksort (n\*log(n)) and radix sort (n\*k).