Uzair Akram

Algorithms

Assignment 1

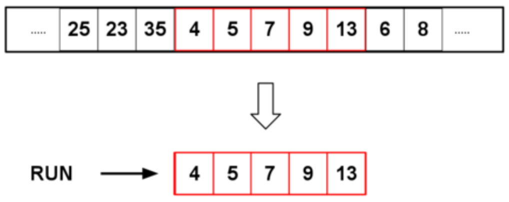
Part 1 Sorting Algorithms - TimSort

TimSort is a hybrid stable sorting algorithm, designed to perform well on many kinds of real world data. The algorithm is a hybrid of merge sort and insertion sort. The algorithm is adaptive to the data, it handles the sorting computation differently based on various factors such as the size of data etc. The Algorithm was created by Tim Peter and is named after him “TimSort”. The algorithm was derived and optimized based on empirically testing of the sorting algorithm. TimSort is designed to take advantage of the best case time complexity of insertion sort. TimSort is built for practical use dealing with real world data which is usually sorted.

Algorithm Complexities

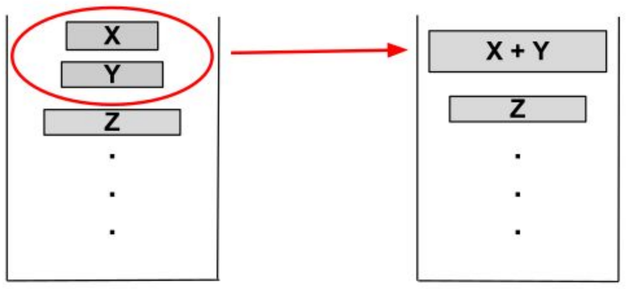
Worst-case performance	*O(n log n)*
Best-case performance	*O(n)*
Average Performance	*O(n log n)*
Worst-case space complexity	*O(n)*

The algorithm divides the array of data into runs, each run is sorted using insertion sort, since insertion sort performs well on small arrays. The size of the run can vary from 32 to 64 depending on the size of the data, reason being the merge function performs well when size of subarray are powers of 2, or slightly less. The merge function is less efficient when runs are larger than a power of 2. The algorithm iterates over the data looking for natural runs the runs are either descending or non-descending, the descending runs are reversed. The runs are then merged using merge sort.



Timsort algorithm searches for minimum-size ordered sequences, minruns, to perform its sort, the minruns are sorted using insertion sort.(Image from Wikipedia)

Uzair Akram



The runs are inserted in a stack. If $|Z| \leq |Y| + |X|$, then X and Y are merged and replaced on the stack. In this way, merging is continued until all runs satisfy i. $|Z| > |Y| + |X|$ and ii. $|Y| > |X|$.(image from Wikipedia)

TimSort optimizes for merging in the galloping mode, runs are repeatedly merged two at a time. Inorder to form balanced merges TimSort considers three runs on top of the stack.

i. $|Z| > |Y| + |X|$
ii. $|Y| > |X|$

When merging the algorithm performs a binary search to find the location of the first element of the run and starts MergeSorting two runs from that location. To improve the efficiency the algorithm uses a galloping mode, in the galloping mode the algorithm checks the first element of each run to get a sense of where the first element will be placed.

Citation:

Wikipedia:
https://en.wikipedia.org/wiki/Timsort
GeeksforGeek
https://www.geeksforgeeks.org/timsort/