Tim Sort

Source - https://www.youtube.com/watch?v=emeME__917E

It's a sorting algorithm made up combining other algos.

Basic sudo code for tim sort:

1. The array is divided into max-size of 64 and min-size of 32.
2. On the arrays(64-32size) we try to identify mini-chunks of sorted array.
3. These sorted arrays can be in increasing or decreasing order .
4. Min size of chunk should be 3 no max size as far as we keep getting sorted values.
5. If the values are not presented in sorted order we make mini sorted arrays of size 3 using insertion sort.
6. Once the whole array is divided into chunks, decreasing arrays are converted into increasing.
7. Final sorting is done using merge sort.
    a. Merge sort is done in a different way here. Here we do inplace merge sort with finding out the smallest chunk out of 2 chunks of arrays and then shifting the bigger array to the right and then comparing and merging the array and chunk using merge sort.
    b. The merging process is even more optimised using stack. In Tim-sort these sorted pieces of chunks are called runs. All runs are loaded into a stack i.e stack contains a collection of chunks.
    c. Now comparison is done in stack values in a specific way. The scope of comparison is 3 stack values from top.

       If len(run[2])>len(run[0])+len(run[1]) {run[i] means chunk=run at index i in stack} , then merge-sort is applied on run[0],run[1] otherwise mergesort applied on run[1],run[2].

    d. This operation is done keeping in mind that stack should always be in increasing run length.
    e. This process is applied till the stack is completed. Thus we have a sorted array.
        i. This sorting of 2 sorted sub-array (A&B) can be optimised even more.
        ii. This is done as if we can find A[0] correct position in B.
        iii. Because if we find the correct position and lets say it is found at index 4, till index 3 we can simply use B.

iv. Similarly find A[n] correct position in B, we can simply use B values after it.
v. This merging of A & B is done using binary search on the array.
    1. But this merging of A&B using Binary search is not efficient in most of the cases. Therefore, we use the concept of Galloping.
    2. Galloping - A threshold value of 7, which when achieved we award and reduce threshold by 1 and if threshold not met we punish by increasing threshold to 8.
    3. If Gallop is less or equal to 7, binary search will be implemented for merging otherwise Normal mergesort merging of 2 sorted arrays.

Galloping -

vi. This sorting of 2 sorted sub-array (A&B) can be optimised even more.
vii. This is done as if we can find A[0] correct position in B.
viii. Because if we find the correct position and lets say it is found at index ‘i’, till index 3 we can simply use sortedarray2.

When we perform this operation, if the index is found after min. 7 indexes in 2nd array i.e i>=7 , then we say that threshold is matched.

Article with same as sudo code - https://hackernoon.com/timsort-the-fastest-sorting-algorithm-youve-never-heard-of-36b28417f399