**Timsort**

Recently, I learnt that the default sorting algorithm in Python has been updated from Timsort to Powersort. Timsort is a hybrid sorting algorithm that combines merge-sort and insertion sort, like the problem in homework 1. In an email correspondence Tim Peters introduced a "non-recursive adaptive stable natural mergesort / binary insertion sort hybrid" [3], later named Timsort. He explains that the point of this sort was to "waste" some extra memory to have performant standard sorting algorithm. There are three factors that make Timsort a good choice for the default sort algorithm in Python. First, raising the worst case. When implementing a general sort, that will be used by novice programmers and built into a programing language, it is important to make sure that the worst case is as close to optimal as possible. Timsort has an worst-case runtime of O(nlogn), which, as Tim puts it, "It (Timsort) has no bad cases"[3]. Another factor in Timsort's performance is that it performs better on semi-sorted lists, than other mergesort based algorithms. This better reflects real world data which usually contains runs of increasing or decreasing values\*. Lastly, Timsort is stable, which means that equal elements persevere their original order. This makes it useful in many real-world scenarios, such as sorting by last name and then first name. While Timsort performs well on many real-world data patterns, it does not have the advantage of quickly sorting already sorted data. I think this is a fine tradeoff because in most cases we don't need to sort sorted array. The implementation details are outlined in [listsort.txt](https://github.com/python/cpython/blob/main/Objects/listsort.txt) [4] and pseudocode of the algorithm is taken from [5]:

A screenshot of a math test

AI-generated content may be incorrect.

Timsort can be broken into three steps. First

As an aside the python email mailing list is a really great source of programing information and comedic content. It's cool to read the ideas and problems the python community ran into building the language.

\* I’m interested if anyone has done a comprehensive study to show if this is true. Tim takes it for granted in his email and I’ve seen similar arguments in papers. I don’t know if this is a result of some heuristics, a gut feeling or based on how he generated random sequences.   
  
Citations:   
[[1] https://en.wikipedia.org/wiki/Powersort](https://en.wikipedia.org/wiki/Powersort)  
[2] https://en.wikipedia.org/wiki/Timsort  
[3] <https://mail.python.org/pipermail/python-dev/2002-July/026837.html>  
[4] <https://github.com/python/cpython/blob/main/Objects/listsort.txt>

[5] Auger, Nicolas, et al. "On the worst-case complexity of TimSort." *arXiv preprint arXiv:1805.08612* (2018).