time complexity

- merge sort as a whole

$$T(n) = T(n/2) + T(n/2) + T(\text{merge}(n))$$

$n-1$

$$T(n) = 2T(n/2) + n-1$$

$\vdots$

$$T(1) = 0$$

$$T(n) = 2\left(2T(n/4) + \frac{n}{2} - 1\right) + n-1$$

$$= 4T(n/4) + n - 2 + n - 1$$

$$= 4\left(2T(n/8) + \frac{n}{4} - 1\right) + 2n - 2 - 1$$

$$= 8T(n/8) + n - 4 + 2n - 2 - 1$$

$$= 8T(n/8) + 3n - 4 - 2 - 1$$

$\log_2 n$ terms

$$= nT(1) + n\log_2 n - \sum_{i=0}^{\log_2 n - 1} 2^i$$

geometric series

$-(1 + 2 + 4 + \dots 2^{\log_2 n - 1})$

$$= n\log_2 n - (2^{\log_2 n} - 1)$$

$$= n\log_2 n - (n-1)$$

should also be worst case no. of comparisons

insertion sort as a whole :

- average case for n elements : $\frac{1}{2}\left(\frac{(n-1)(n+2)}{2}\right)$ $\Theta(n^2)$
- worst case $\frac{(n)(n-1)}{2}$

hybrid algorithm $\longrightarrow$ mergesort : n elements
insertion sort : m

$$T(n) = T(n/2) + T(n/2) + T(\underbrace{merge(n)}_{n-1})$$

$$T(n) = 2T(n/2) + n-1$$

worst $\rightarrow$ $T(m) = \frac{m(m-1)}{2}$ avg $\rightarrow$ $T(m) = \frac{1}{2}\left(\frac{(m-1)(m+2)}{2}\right)$

let $m = \frac{n}{k} = \frac{n}{2^x}$ $\quad 2^x = k \quad x = \log_2 k$

$$T(n) = 2T(n/2) + n-1$$

$$T(n) = 2(2T(n/4) + n/2 - 1) + n-1$$

$$T(n) = 4T(n/4) + n - 2 + n - 1$$

$\vdots$

$$T(n) = x\underline{\underline{T(n/2^x)}} + xn - \sum_{i=0}^{x-1} 2^i$$

$\downarrow$ $T(m) = \frac{m(m-1)}{2}$ $\qquad \sum_{i=0}^{x-1} 2^i \rightarrow 2^x - 1$

worst case $T(n) = x\left(\frac{m(m-1)}{2}\right) + xn - 2^x + 1$

where $x = \log_2\left(\frac{n}{m}\right)$

avg case $T(n) = x\left(\frac{(m-1)(m+2)}{4}\right) + xn - 2^x + 1$

best case

$$T(n) = x\left(\frac{(m-1)(m+2)}{4}\right) + xn - 2^{x} + 1$$

a) $\frac{1000}{100} = 10$ $\qquad 2.3\left(\frac{99 \times 102}{4}\right) + (2.3)(1000) - 2^{2.3} + 1$

$x = 2.3$ $\qquad 5812 + 2300 - 5 + 1 = 8108$

best case

$$T(n) = 2T(n/2) + n/2$$
$$T(n) = 2(\ 2T(n/4) + n/4\ ) + n/2$$
$$T(n) = 4T(n/4) + 2(n/2)$$
$$T(n) = 4(\ 2T(n/8) + n/8\ ) + 2(n/2)$$
$$T(n) = 8T(n/8) + 3\,(n/2)$$
$$T(n) = nT(n/n) + \log_2 n\ (n/2)$$
$$T(n) = \frac{n}{2}\log_2 n$$

worst case $= T(n) = n\log_2 n - (n-1)$

$\{$

$O(n\log_2 n)$

insertion sort best : $O(n)$ worst : $O(n^2)$

hybrid

$T(m) = O(n)$ $\qquad \frac{n}{2^k} = S \qquad \frac{n}{S} = 2^k \qquad k = \log_2\left(\frac{n}{S}\right)$

where $n = 8$

$T(n)$ 5 6 7 8 3 4 2 1

$\uparrow n/2$

$T(n/2)$ 5 6 7 8 $T(n/2)$ 3 4 2 1

$\uparrow n/4$ $\uparrow n/4$

$T(n/4)$ 56 78 34 21

$\uparrow n/8$ $\uparrow n/8$ $\uparrow n/8$ $\uparrow n/8$

$T(n/8)$ 5 6 7 8 3 4 2 1

$$n/2 + n/2 + n/2 = 3n/2 = \log_2 n \times \frac{n}{2}$$

$$\log_2 11 \times 5.5 = 19.$$

stop when the array size = s

when $\frac{n}{2^m} = s$, we stop

$\therefore \boxed{\frac{n}{s} = 2^m}$

$s = 2^{k-m}$

$m = \log_2(n/s)$

let $n = 2^k$

$$T(n) = 2T(n/2) + \overset{n/2}{T(\text{merge})}$$

$$T(n) = 2^m T(s) + \log_2(2^m)(n/2)$$

$$T(n) = 2^m \underline{\underline{T(s)}} + m \cdot (n/2)$$

↳ time for insertion sort : best case $= (s-1)$

$$T(n) = 2^m(s-1) + m(n/2)$$

$$T(n) = \left(\frac{n}{s}\right)(s-1) + \frac{n}{2}\log_2\left(\frac{n}{s}\right)$$

$$= \left(n - \frac{n}{s}\right) + \frac{n}{2}\log_2\frac{n}{s}$$

$$= O(n) + O\left(n\log_2\left(\frac{n}{s}\right)\right)$$

$$= O\left(n + n\log_2\frac{n}{s}\right)$$

---

key :

total elements $= n = 2^k$

threshold $= s = \frac{n}{2^m}$

2^m = number of split arrays

$k = \log_2 n$

$m = \log_2\left(\frac{n}{s}\right) \quad \because 2^m = \frac{n}{s}$

worst case

$2^m = \frac{n}{s}$

$s = \frac{n}{2^m}$

$$T(n) = 2T(n/2) + (n-1)$$

$$T(n) = 4T(n/4) + n + n - 2 - 1 = 4T(n/4) + 2n - 2 - 1$$

$$T(n) = 8T(n/8) + 3n - \underbrace{4 - 2 - 1}_{\to\, 2^{3-1} - 1}$$

$$\vdots$$

$$T(n) = 2^m \underbrace{T(s)}_{\to\, \frac{s(s-1)}{2}} + \log_2(2^m)\, n - (2^m - 1)$$

$$T(n) = \frac{n}{s}\left(\frac{s(s-1)}{2}\right) + mn - 2^m + 1$$

$$= \frac{n}{s}\left(\frac{s(s-1)}{2}\right) + n\log_2\left(\frac{n}{s}\right) - \frac{n}{s} + 1$$

$$= \left(\frac{ns}{2} - \frac{n}{2} - \frac{n}{s} + 1\right) + n\log_2\left(\frac{n}{s}\right)$$

$$= O(ns) + O\left(n\log_2 \frac{n}{s}\right)$$

$$= O\left(ns + n\log_2 \frac{n}{s}\right)$$

$$\frac{16 \times 5}{2} - 8 - 3 + 1 + 16\log_2(3.2)$$

$$40 - 8 - 3 + 1 + \cancel{16}\,5$$

$.5$ $\quad .5$

$46 \leftarrow \underline{\underline{47}}$

1k	1000
10k	10000
50k	50000
100k	100000
200k	2
300k	
400k	
500k	
600k	
700k	
800k	
900k	
1M	1000000
2 5M	
5M	
7.5M	
10M	

Let's suppose that once the array size reaches k, you switch from merge sort to insertion sort. We want to work out the time complexity of this new approach. To do so, we'll imagine the "difference" between the old algorithm and the new algorithm. Specifically, if we didn't make any changes to the algorithm, merge sort would take time $\Theta(n \log n)$ to complete. However, once we get to arrays of size k, we stop running mergesort and instead use insertion sort. Therefore, we'll make some observations:

- There are $\Theta(n / k)$ subarrays of the original array of size k.
- We are skipping calling mergesort on all these arrays. Therefore, we're avoiding doing $\Theta(k \log k)$ work for each of $\Theta(n / k)$ subarrays, so we're avoiding doing $\Theta(n \log k)$ work.
- Instead, we're insertion-sorting each of those subarrays. Insertion sort, in the worst case, takes time $O(k^2)$ when run on an array of size k. There are $\Theta(n / k)$ of those arrays, so we're adding in a factor of $O(nk)$ total work.

Overall, this means that the work we're doing in this new variant is $O(n \log n) - O(n \log k) + O(nk)$. Dialing k up or down will change the total amount of work done. If k is a fixed constant (that is, $k = O(1)$), this simplifies to

$O(n \log n) - O(n \log k) + O(nk)$

$= O(n \log n) - O(n) + O(n)$

$= O(n \log n)$

and the asymptotic runtime is the same as that of regular insertion sort.

It's worth noting that as k gets larger, eventually the $O(nk)$ term will dominate the $O(n \log k)$ term, so there's some crossover point where increasing k starts decreasing the runtime. You'd have to do some experimentation to fine-tune when to make the switch. But empirically, setting k to some modest value will indeed give you a big performance boost.