Timsort est en $O(n\log(n))$

On a une suite de "runs" de diverses longueur que l'on fusionne deux par deux. La fusion de deux runs de longueur a et b donne un run de longueur $a+b$ en un temps $a+b$. Pour que le tri soit stable, on fusionne toujours deux runs consécutifs. L'algorithme de Tim Peters utilise une pile de runs, contenant initialement trois runs fictifs de taille $8n$, $4n$ et $2n$ en notant n la somme des longueurs des runs à fusionner. Si les longueurs x_2, x_1 et x_0 des trois runs en sommet de pile vérifient $x_2 > x_1 + x_0$ et $x_1 > x_0$ alors on empile le run suivant (pris dans la liste initiale), sinon on fusionne x_1 avec le plus petit de ses deux voisins. Autrement dit le haut de la pile x_2, x_1, x_0 est remplacé par $x_2 + x_1$, x_0 si $x_2 < x_0$ et sinon par x_2, $x_1 + x_0$. On recommence jusqu'à ce que la pile contienne $8n$, $4n$, $2n$, n, $2n$, en rajoutant évidemment un run fictif de taille $2n$ à la fin de la liste des runs.

Les longueurs des runs sont strictement positives. Donc $x_2 > x_1 + x_0$ et $x_1 > x_0$ entraine $x_2 > x_1 > x_0$. On peut en déduire que le contenu de la pile, sans les deux éléments du sommet, est toujours strictement décroissant. De plus, juste avant d'empiler un nouveau run, la pile complète est strictement décroissante.

Variante $2x_{i+1} > 3x_i$

L'algorithme est un peu plus rapide et beaucoup plus facile à analyser en remplaçant la condition

$$x_2 > x_1 + x_0 \qquad \text{et} \qquad x_1 > x_0 \tag{1}$$

par

$$2x_2 > 3x_1 \qquad \text{et} \qquad 2x_1 > 3x_0 \tag{2}$$

Avec cette condition au début de la boucle principale on a :

$$\forall i \geq 2,\ 2x_{i+1} > 3x_i \qquad \text{et} \qquad x_1 < \max((2/3)x_2, \min(2x_2, 3x_0)) \tag{3}$$

Si le sommet de pile x_0 est un nouveau run, ou provient de $x_0{+}{=}x_1$, on a $2x_2 > 3x_1$ et il n'y a pas de contrainte sur x_0, qui peut être très petit ou très grand. Mais x_1 peut être plus grand que $(2/3)x_2$ s'il résulte d'un ou de plusieurs $x_1{+}{=}x_2$. Tous les termes additionnés sont majorés par une progression géométrique de raison 2/3.

$$x_1 < \frac{2}{3}x_2 + \frac{2^2}{3^2}x_2 + \frac{2^3}{3^3}x_2 + \cdots = 2x_2$$

Mais alors le sommet de pile doit être supérieur à chacun des termes additionnés. Donc $x_0 > x_1/3$. C'est pourquoi $x_1 < \min(2x_2, 3x_0)$ juste après un $x_1{+}{=}x_2$ et $x_1 < (2/3)x_2$ dans les autres cas.

On pourrait aussi démontrer ces conditions par récurrence. Par exemple si la pile est $\ldots d,\ c,\ b,\ a$ alors on a $b < 2c$ et $3c < 2d$. Si de plus $c < a$, alors la pile devient $\ldots d,\ c+b,\ a$ et on obtient $c + b < 3c < 2d$ et $c + b < 3c < 3a$.

De (3) on déduit que la taille de la pile est en $\log(n)/log(3/2) + O(1)$ et aussi que l'on fusionne toujours

– deux runs voisins de tailles proches (de rapport inférieur à 3)

– ou un run coincé entre deux runs plus grands, avec le plus petit de ses deux voisins.
On en déduit que lors d'une fusion $b+c$, son coût est majoré par la variation du potentiel

$$P = f(x_0) + \sum_{i\geq 0} f(x_i + x_{i+1}) \qquad \text{avec} \qquad f(x) = x\log(x) \tag{4}$$

Le coût total est donc inférieur au potentiel final qui est en $O(n\log(n))$. On va détailler cela.

Par exemple si $\dots a, b, c, d \dots$ avec $b < c < a$ devient $\dots a, b+c, d \dots$, c'est-à-dire quand un run b plus court que ses deux voisins a et b est fusionné avec le plus petit des deux, alors la variation du potentiel est

$$\Delta P = f(a+b+c) + f(b+c+d) - f(a+b) - f(b+c) - f(c+d)$$

On a $f'(x) = 1 + \log(x)$. Donc la dérivée de $f(b+x) - f(x)$ est $\log(b+x) - \log(x) > 0$ car $b > 0$. Donc $f(b+x) - f(x)$ est une fonction croissante de x et $f(b+c+d) - f(c+d)$ est minimal pour $d = 0$. Donc $f(b+c+d) - f(c+d) \geq f(b+c) - f(c)$. Donc

$$\Delta P \geq f(a+b+c) - f(a+b) - f(c)$$

De même pour b et c fixés, ce minorant est une fonction croissante de a qui atteint son minimum pour a minmum, c'est-à-dire $a = c$. Donc

$$\Delta P \geq f(b+2c) - f(b+c) - f(c) = f(s+c) - f(s) - f(s-c)$$

en posant $s = b + c$. Pour s fixé, ce minorant est une fonction croissante de c qui atteint son mininum pour c minimum, c'est-à-dire $c = s/2$ car $b < c$ et $s = b + c < 2c$. Donc

$$\begin{aligned}
\Delta P > f(3s/2) - f(s) - f(s/2) &= (3s/2)\log(3s/2) - s\log(s) - (s/2)\log(s/2) \\
&= (3s/2)\log(3/2) - s\log(1) - (s/2)\log(1/2) \\
&= (s/2)\log((3/2)^3/(1/2)) = ((b+c)/2)\log(27/4)
\end{aligned}$$

Le coût de la fusion $b+c$ est donc pris en compte par l'augmentation du potentiel.

Lors d'une fusion des deux runs en sommet de pile $\dots a, b, c$ avec $c < a$ devient $\dots a, b+c$.

$$\Delta P = f(a+b+c) - f(a+b) - f(c)$$

Ou bien $2b \leq 3c$, ou bien $2a \leq 3b$ et alors $b < 3c$. Dans les deux cas on a $b < 3c$. Si $b < c$ alors $b < c < a$ et b est plus petit que ses deux voisins et fusionne avec le plus petit des deux. On a déjà étudié ce cas. On peut donc supposer que $b \geq c$. Comme précédemment on minimise ΔP en prenant $a = 0$. Donc

$$\Delta P \geq f(b+c) - f(b) - f(c) \qquad \text{et} \qquad c \leq b < 3c$$

A $s = b + c$ fixé, $c \leq b < 3c$ se traduit en $c \in]c/4, c/2]$ et $f(b + c) - f(b) - f(c) = f(s) - f(c) - f(s - c)$ est une fonction strictement croissante de c qui atteint donc son minimum en $c/4$. Donc

$$\begin{aligned}\Delta P > f(s) - f(s/4) - f(3s/4) &= s\log(s) - (s/4)\log(s/4) - (3s/4)\log(3s/4) \\ &= s\log(1) - (s/4)\log(1/4) - (3s/4)\log(3/4) \\ &= (s/4)(\log(4) + 3\log(4/3)) = ((b + c)/4)\log(256/27)\end{aligned}$$

Lors d'une fusion $x_1 + x_2$, $\ldots a, b, c, d$ avec $b < d$ devient $\ldots a, b + c, d$. Si $c < b$ alors $c < b < d$ et c est plus petit que ses deux voisins et fusionne avec le plus petit des deux. On a déjà étudié ce cas. On peut donc supposer que $c \geq b$. Mais $c < 2b$. Comme précédemment on peut minorer ΔP en prenant $d = 0$ et $a = 0$.

$$\Delta P \geq f(b + c) - f(b) - f(c) \qquad \text{et} \qquad b \leq c < 2b$$

En posant $s = b + c$ on a

$$\Delta P > f(s) - f(s/3) - f(2s/3) = (s/3)(\log(3) + 2\log(3/2)) = ((b + c)/3)\log(27/4)$$

On a démontré que lors d'une fusion $b + c$, on a toujours $b + c < C_i \Delta P$ où selon les cas C_i vaut $C_1 = 2/\log(27/4)$ ou $C_2 = 3/\log(27/4)$ ou $C_3 = 4/\log(256/27)$. Puisque $C_1 < C_2 < C_3$ on a donc toujours $b + c < C_3 \Delta P$.

Initialement la pile contient $4n$, $2n$ et P vaut $P_0 = f(4n) + f(6n) + f(2n) = 4n\log(4n) + 6n\log(6n) + 2n\log(2n)$. A la fin la pile contient $4n$, $2n$, n, $2n$ et P vaut $P_1 = P_0 + 2f(3n)$. Donc le coût total des fusions, la somme des $b + c$ est majoré par $C_3(P_1 - P_0) = 6n\log(3n)4/\log(256/27) = O(n\log(n))$.

Analyse de Timsort

Toute l'analyse précédente de la variante de Timsort, peut s'adapter à Timsort car Stijn de Gouw1, Jurriaan Rot, Frank S. de Boer, Richard Bubel et Reiner Hähnle ont démontré dans leur article "OpenJDKr's java.utils.Collection.sort() is broken: The good, the bad and the worst case" que lors du Timsort, non seulement la pile était décroissante, mais aussi pour tout i on a $x_i > x_{i-1} + x_{i-2}$ ou $x_{i+1} > x_i + x_{i-1}$. On peut en déduire que la pile décroît plus vite qu'une progession géométrique de raison $1/\sqrt{2}$. Donc la taille de la pile est majorée par $2\log(n)/\log(2) + O(1)$. De plus la fusion de deux runs fait toujours intervenir un run plus petit que ses deux voisins, qui fusionne avec le plus petit des deux, ou deux runs dont le rapport des tailles reste majoré par une constante. Donc le temps de calcul de Timsort est en $O(n\log(n))$.