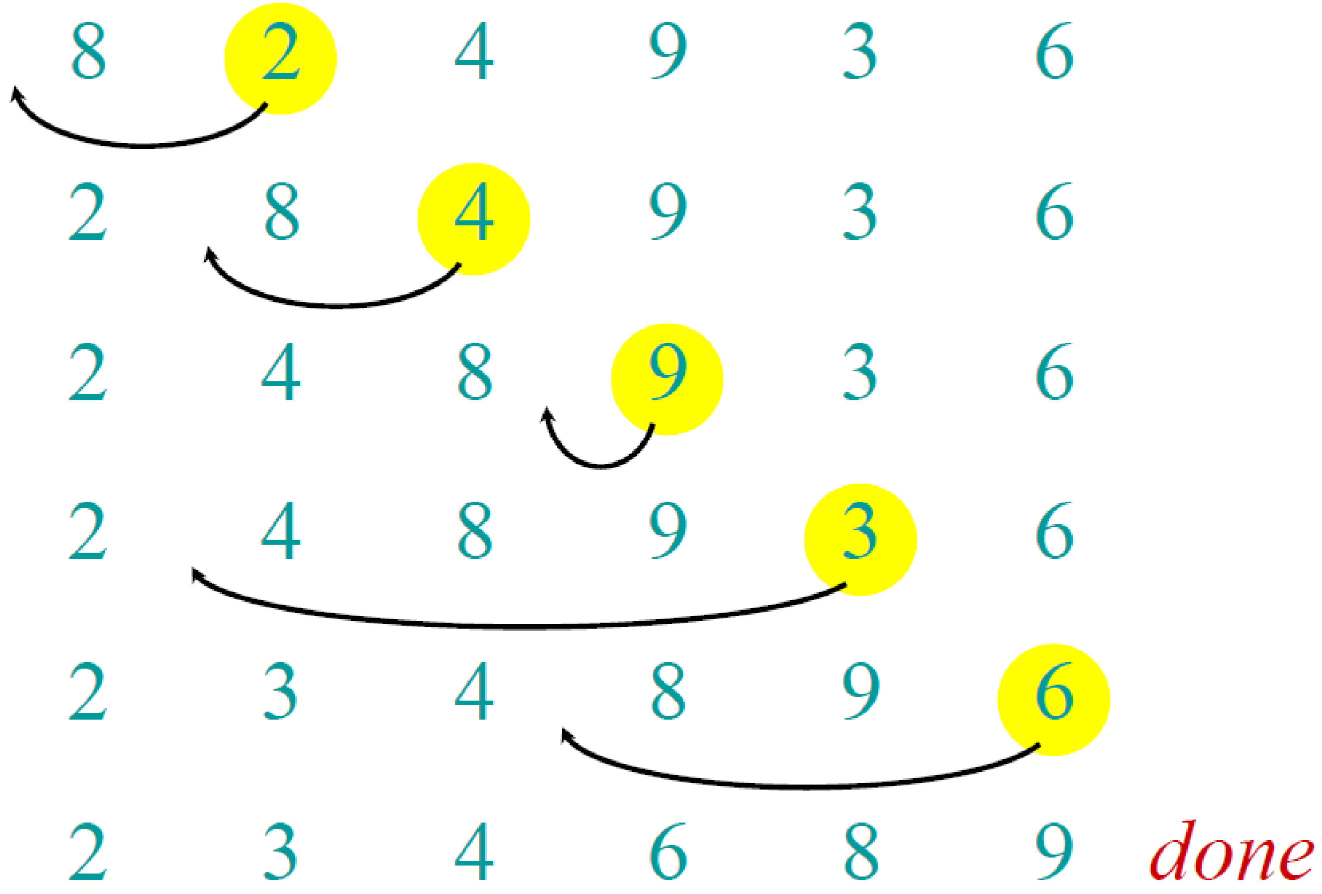
Performance Analysis

SWE2016-44

Insertion Sort



Insertion Sort

INSERTION-SORT (A, n)	T(n)
for j ← 2 to n	*n-1*
do key ← A[j]	*n-1*
i ← j – 1	*n-1*
while i > 0 and A[i] > key	Best: $\sum_{j=2}^{n} 1$, Worst: $\sum_{j=2}^{n} j$, Avg: $\frac{1}{2}\sum_{j=2}^{n}(j+1)$
do A[i+1] ← A[i]	Best: $\sum_{j=2}^{n}(1\text{-}1)$, Worst: $\sum_{j=2}^{n}(j\text{-}1)$, Avg: $\frac{1}{2}\sum_{j=2}^{n}(j\text{-}1)$
i ← i – 1	Best: $\sum_{j=2}^{n}(1\text{-}1)$, Worst: $\sum_{j=2}^{n}(j\text{-}1)$, Avg: $\frac{1}{2}\sum_{j=2}^{n}(j\text{-}1)$
A[i+1] = key	*n-1*

- Best Case: $T(n) = 4(n-1) + (n-1) = 5n - 5$
- Worst Case: $T(n) = 4(n-1) + \frac{n(n-1)}{2} + 2\left(\frac{n(n-1)}{2} - (n-1)\right) = \frac{3}{2}n^2 + \frac{1}{2}n - 2$
- Average Case: $T(n) = 4(n-1) + \frac{n-1}{2} + \frac{n(n-1)}{4} + \left(\frac{n(n-1)}{2} - (n-1)\right) = \frac{3}{4}n^2 + \frac{11}{4}n - \frac{7}{2}$

Insertion Sort

- Best Case: $\mathrm{T(n) = 5n - 5}$
  - Θ notation: $\Theta(\mathrm{n})$ 🡐 $\mathrm{4n \leq T(n) \leq 5n}$ for $\mathrm{n \geq 5}$
  - O notation: $\mathrm{O(n)}$ 🡐 $\mathrm{T(n) \leq 5n}$ for $\mathrm{n \geq 1}$

- Worst Case: $\mathrm{T(n)} = \frac{3}{2}\mathrm{n}^2 + \frac{1}{2}\mathrm{n} - 2$
  - Θ notation: $\Theta(\mathrm{n}^2)$ 🡐 $\frac{3}{2}\mathrm{n}^2 \leq \mathrm{T(n)} \leq 2\mathrm{n}^2$ for $\mathrm{n \geq 4}$
  - O notation: $\mathrm{O(n^2)}$ 🡐 $\mathrm{T(n)} \leq 2\mathrm{n}^2$ for $\mathrm{n \geq 1}$

- Average Case: $\mathrm{T(n)} = \frac{3}{4}\mathrm{n}^2 + \frac{11}{4}\mathrm{n} - \frac{7}{2}$
  - Θ notation: $\Theta(\mathrm{n}^2)$ 🡐 $\frac{3}{4}\mathrm{n}^2 \leq \mathrm{T(n)} \leq \mathrm{n}^2$ for $\mathrm{n \geq 6}$
  - O notation: $\mathrm{O(n^2)}$ 🡐 $\mathrm{T(n)} \leq \mathrm{n}^2$ for $\mathrm{n \geq 6}$

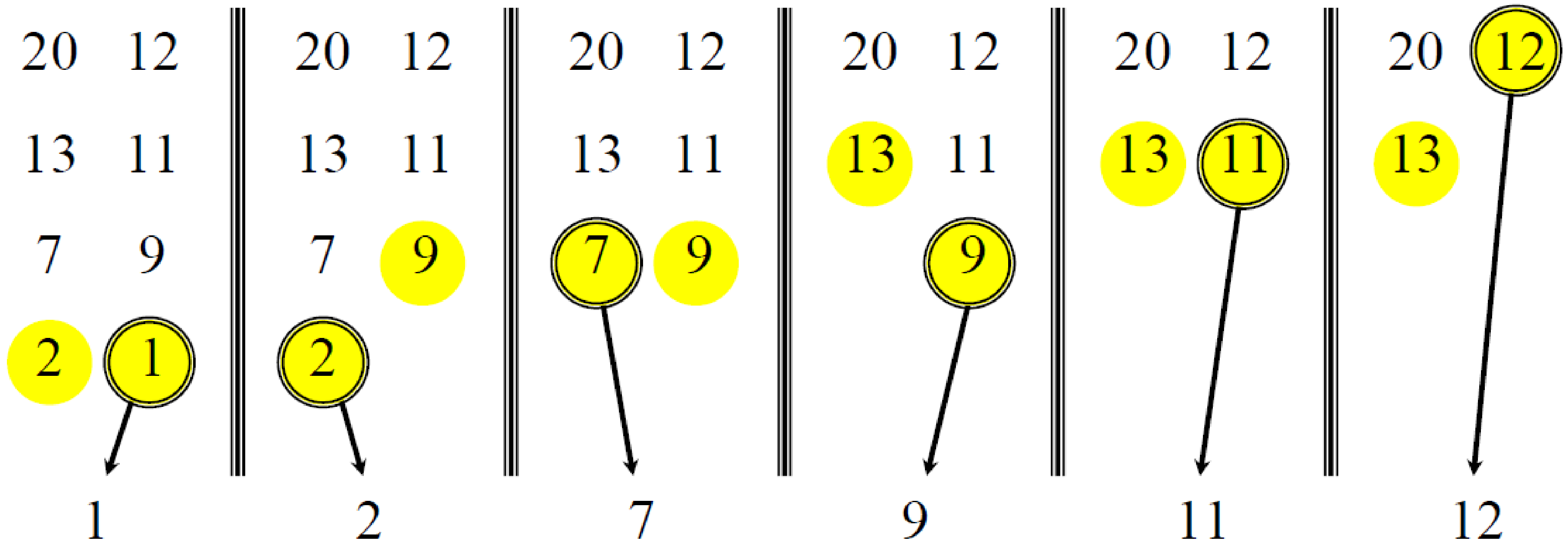
Insertion Sort

- $T(n) = \frac{3}{4}n^2 + \frac{11}{4}n - \frac{7}{2}$
  - $T(n) = \frac{3}{4}n^2 + \frac{11}{4}n - \frac{7}{2} \le \frac{3}{4}n^2 + \frac{11}{4}n \le \frac{3}{4}n^2 + \frac{11}{4}n^2 = \frac{7}{2}n^2$
  - $T(n) = \frac{3}{4}n^2 + \frac{11}{4}n - \frac{7}{2} \ge \frac{3}{4}n^2 - \frac{7}{2} = \frac{3}{8}n^2 + \frac{3}{8}n^2 - \frac{7}{2} \ge \frac{3}{8}n^2$ for all $n \ge 4$

➔ $\frac{3}{8}n^2 \le T(n) \le \frac{7}{2}n^2$ for $n \ge 4$

➔ $T(n) = O(n^2)$

Merge two sorted arrays



Merge two sorted arrays

* Best Case Assumption: $A[i] < B[j]$ for all i, j

MERGE (A, B, m, n)	T(m, n)
$i = 1, j = 1$	*1*
while $i < m + 1$ *and* $j < n + 1$	Best: $m + 1$, Worst: $m + n$, Avg: $\frac{1}{2}(m+1) + \frac{1}{2}(m+n)$
if $A[i] < B[j]$	Best: $3m$, Worst: $3(m+n\text{-}1)$, Avg: $\frac{3}{2}m + \frac{3}{2}(m+n\text{-}1)$
do $Z[i + j - 1] = A[i]$	
$i = i + 1$	
else	
do $Z[i + j - 1] = B[j]$	
$j = j + 1$	
for $k = i$ to m	Best: n, Worst: 1, Avg: $\frac{1}{2}n + \frac{1}{2}$
$Z[k + n] = A[k]$	
for $k = j$ to n	
$Z[m + k] = B[k]$	

- Best Case: $T(n) = 4m + n + 2$
- Worst Case: $T(n) = 4m + 4n\text{-}1$
- Average Case: $T(n) = 4m + \frac{5}{2}n + \frac{1}{2}$

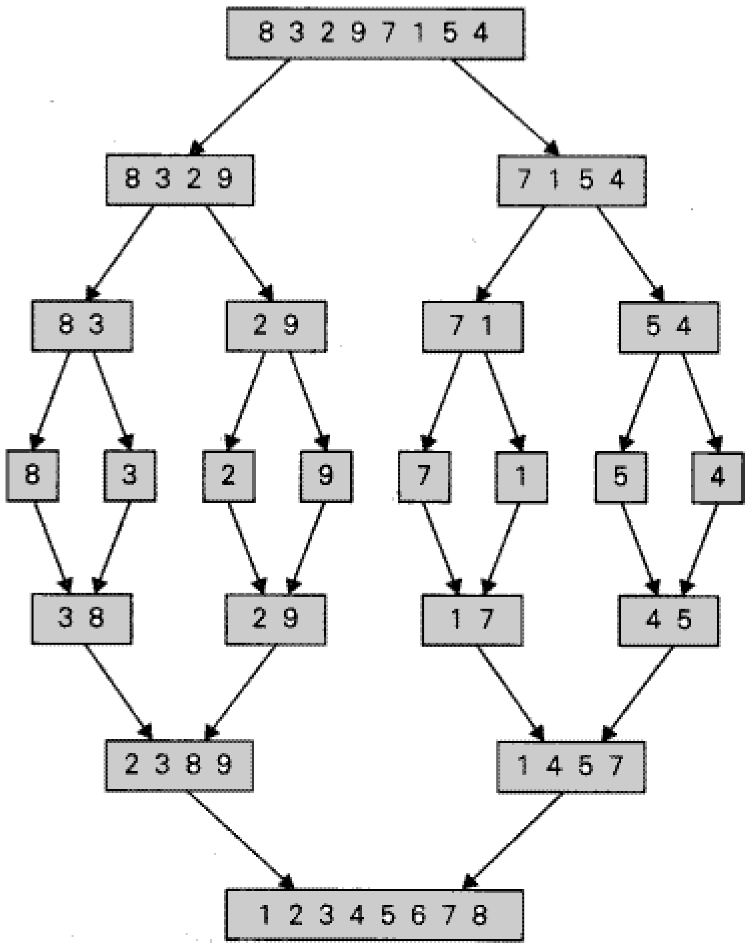
Merge two sorted arrays

- Best Case: $T(n) = 4m + n + 2$
  - Θ notation: $\Theta(m+n)$ $\Leftarrow$ $m + n \le T(n) \le 4(m+n)$ for $m \ge 1,\ n \ge 1$
  - O notation: $O(m+n)$ $\Leftarrow$ $T(n) \le 4(m+n)$ for $m \ge 1,\ n \ge 1$

- Worst Case: $T(n) = 4m + 4n - 1$
  - Θ notation: $\Theta(m+n)$ $\Leftarrow$ $m + n \le T(n) \le 4(m+n)$ for $m \ge 1,\ n \ge 1$
  - O notation: $O(m+n)$ $\Leftarrow$ $T(n) \le 4(m+n)$ for $m \ge 1,\ n \ge 1$

- Average Case: $T(n) = 4m + \frac{5}{2}n + \frac{1}{2}$
  - Θ notation: $\Theta(m+n)$ $\Leftarrow$ $2(m+n) \le T(n) \le 4(m+n)$ for $m \ge 1,\ n \ge 1$
  - O notation: $O(m+n)$ $\Leftarrow$ $T(n) \le 4(m+n)$ for $m \ge 1,\ n \ge 1$

Merge Sort



from: https://freefeast.info/general-it-articles/merge-sort-pseudo-code-of-merge-sort-merge-sort-in-data-structure-divide-and-conquer-approach/

Merge Sort

```
func mergesort( var a as array )
     if ( n == 1 ) return a

     var l1 as array = a[0] ... a[n/2]
     var l2 as array = a[n/2+1] ... a[n]

     l1 = mergesort( l1 )
     l2 = mergesort( l2 )

     return merge( l1, l2 )
end func
```

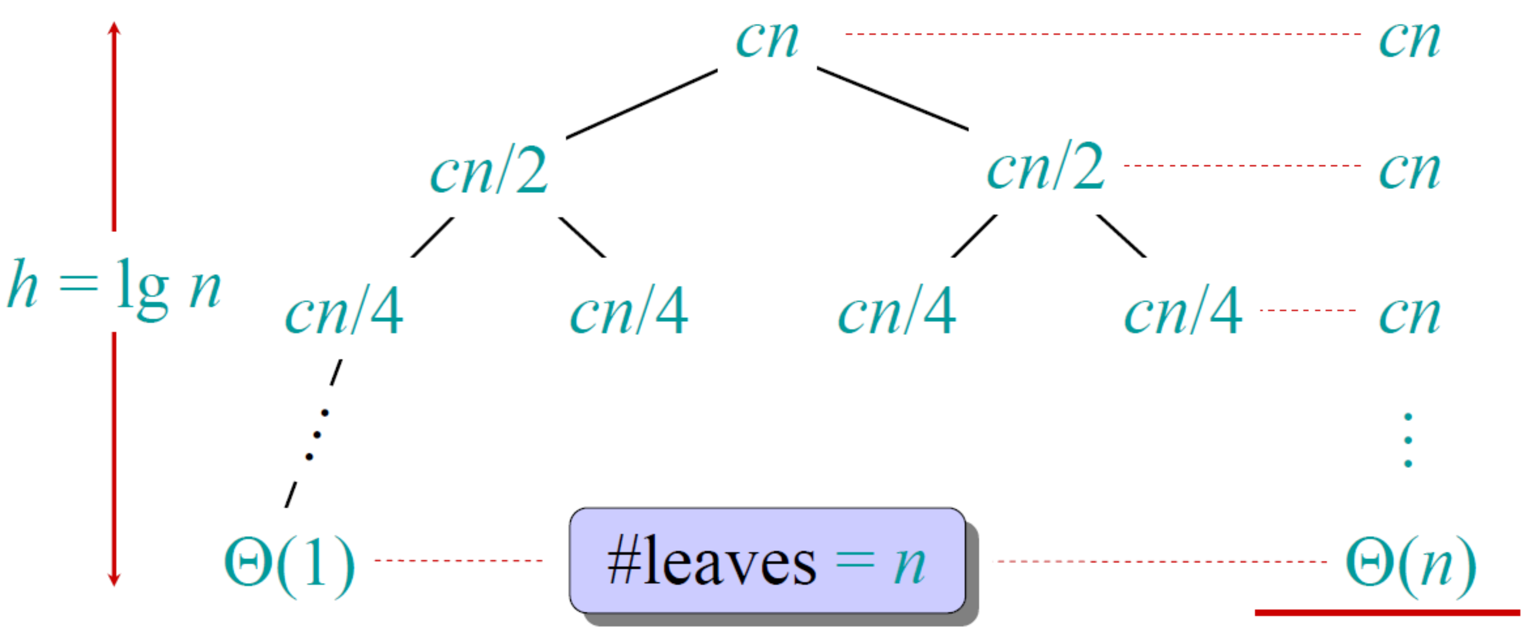
```
func merge( var a as array, var b as array )
     var c as array

     while ( a and b have elements )
          if ( a[0] > b[0] )
               add b[0] to the end of c
               remove b[0] from b
          else
               add a[0] to the end of c
               remove a[0] from a
     while ( a has elements )
          add a[0] to the end of c
          remove a[0] from a
     while ( b has elements )
          add b[0] to the end of c
          remove b[0] from b
     return c
end func
```

from: https://www.algorithmist.com/index.php/Merge_sort

Merge Sort

$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1; \\ 2T(n/2) + \Theta(n) \text{ if } n > 1. \end{cases}$$

Solve $T(n) = 2T(n/2) + cn$, where $c > 0$ is constant.



Total $= \Theta(n \lg n)$

Merge Sort

$$
\begin{aligned}
\mathrm{T(n)} &= 2\mathrm{T}\left(\frac{\mathrm{n}}{2}\right) + \mathrm{cn} \\
&= 2\left\{2\mathrm{T}\left(\frac{\mathrm{n}}{4}\right) + \frac{\mathrm{cn}}{2}\right\} + \mathrm{cn} = 4\mathrm{T}\left(\frac{\mathrm{n}}{4}\right) + 2\mathrm{cn} \\
&= 4\left\{2\mathrm{T}\left(\frac{\mathrm{n}}{8}\right) + \frac{\mathrm{cn}}{4}\right\} + 2\mathrm{cn} = 8\mathrm{T}\left(\frac{\mathrm{n}}{8}\right) + 3\mathrm{cn} \\
&= \cdots \\
&= \mathrm{nT(1)} + \mathrm{cn} \ln \mathrm{n} \\
&= \Theta(\mathrm{n} + \mathrm{n} \ln \mathrm{n}) \\
&= \Theta(\mathrm{n} \ln \mathrm{n})
\end{aligned}
$$