MergeSort and time complexity of Divide-and-Conquer algorithms
Outline
Closed form solution for linear recurrence functions

If a recurrence function is of this form:

\[ f(n) = \begin{cases} 
  b_1 & n = 0 \\
  b_2 & n = 1 \\
  c_1 f(n - 1) + c_2 f(n - 2) & n > 1
\end{cases} \]

The closed form solution can be found by solving the equation \( r^2 - c_1 r - c_2 = 0 \) and finding the roots \( r_1, r_2 \). Then, the closed form solution is of the form

\[ f(n) = a_1 r_1^n + a_2 r_2^n \]
Recurrence for MergeSort

MergeSort(A,b,e):
    if b == e: return
m = (b + e) / 2
MergeSort(A,b,m)
MergeSort(A,m+1,e)
# merge sorted A[b..m] and A[m+1..e] back into A[b..e]
for i in [b,...,e]: B[i] = A[i]
c = b
d = m+1
for i in [b,...,e]:
    if d > e or (c <= m and B[c] < B[d]):
        A[i] = B[c]
c = c + 1
else:  # d <= e and (c > m or B[c] >= B[d])
    A[i] = B[d]
d = d + 1
Merge Sort

Complexity of MergeSort
Divide and Conquer: General Case

Class of algorithms: partition problem into $b$ roughly equal subproblems, solve, and recombine:

$$T(n) = \begin{cases} 
  k & \text{if } n \leq B \\
  a_1 T(\lceil n/b \rceil) + a_2 T(\lfloor n/b \rfloor) + f(n) & \text{if } n > B 
\end{cases}$$

where $B, k > 0$, $b > 1$, $a_1, a_2 \geq 0$, and $a = a_1 + a_2 > 0$. $f(n)$ is the cost of splitting and recombining.
Master Theorem

If $f$ from the previous slide has $f \in \theta(n^d)$, then

$$T(n) \in \begin{cases} 
\theta(n^d) & \text{if } a < b^d \\
\theta(n^d \log n) & \text{if } a = b^d \\
\theta(n^{\log_b a}) & \text{if } a > b^d
\end{cases}$$