5. **Divide And Conquer I**

- mergesort
- counting inversions
- randomized quicksort
- median and selection
- closest pair of points
Divide-and-conquer paradigm

Divide-and-conquer.

• Divide up problem into several subproblems (of the same kind).
• Solve (conquer) each subproblem recursively.
• Combine solutions to subproblems into overall solution.

Most common usage.

• Divide problem of size $n$ into two subproblems of size $n/2$. $O(n)$ time
• Solve (conquer) two subproblems recursively.
• Combine two solutions into overall solution. $O(n)$ time

Consequence.

• Brute force: $\Theta(n^2)$.
• Divide-and-conquer: $O(n \log n)$. 

attributed to Julius Caesar
5. **Divide and Conquer**

- **mergesort**
- **counting inversions**
- **randomized quicksort**
- **median and selection**
- **closest pair of points**
Problem. Given a list $L$ of $n$ elements from a totally ordered universe, rearrange them in ascending order.
Sorting applications

Obvious applications.
- Organize an MP3 library.
- Display Google PageRank results.
- List RSS news items in reverse chronological order.

Some problems become easier once elements are sorted.
- Identify statistical outliers.
- Binary search in a database.
- Remove duplicates in a mailing list.

Non-obvious applications.
- Convex hull.
- Closest pair of points.
- Interval scheduling / interval partitioning.
- Scheduling to minimize maximum lateness.
- Minimum spanning trees (Kruskal’s algorithm).
- ...
Mergesort

- Recursively sort left half.
- Recursively sort right half.
- Merge two halves to make sorted whole.

input

```
  A   L   G   O   R   I   T   H   M   S
```

sort left half

```
A   G   L   O   R   I   T   H   M   S
```

sort right half

```
A   G   L   O   R   H   I   M   S   T
```

merge results

```
A   G   H   I   L   M   O   R   S   T
```
Merging

**Goal.** Combine two sorted lists $A$ and $B$ into a sorted whole $C$.

- Scan $A$ and $B$ from left to right.
- Compare $a_i$ and $b_j$.
- If $a_i \leq b_j$, append $a_i$ to $C$ (no larger than any remaining element in $B$).
- If $a_i > b_j$, append $b_j$ to $C$ (smaller than every remaining element in $A$).
Mergesort implementation

**Input.** List $L$ of $n$ elements from a totally ordered universe.

**Output.** The $n$ elements in ascending order.

\[ \text{MERGE-SORT}(L) \]

**IF** (list $L$ has one element)

**RETURN** $L$.

Divide the list into two halves $A$ and $B$.

\[ A \leftarrow \text{MERGE-SORT}(A). \quad T(n/2) \]
\[ B \leftarrow \text{MERGE-SORT}(B). \quad T(n/2) \]
\[ L \leftarrow \text{MERGE}(A, B). \quad \Theta(n) \]

**RETURN** $L$. 
A useful recurrence relation

**Def.** \(T(n) = \text{max number of compares to mergesort a list of length } n.\)

**Recurrence.**

\[
T(n) \leq \begin{cases} 
0 & \text{if } n = 1 \\
T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n & \text{if } n > 1
\end{cases}
\]

between \(\lfloor n / 2 \rfloor\) and \(n - 1\) compares

**Solution.** \(T(n)\) is \(O(n \log_2 n)\).

**Assorted proofs.** We describe several ways to solve this recurrence. Initially we assume \(n\) is a power of 2 and replace \(\leq\) with \(=\) in the recurrence.
Proposition. If $T(n)$ satisfies the following recurrence, then $T(n) = n \log_2 n$.

$$T(n) = \begin{cases} 
0 & \text{if } n = 1 \\
2T(n/2) + n & \text{if } n > 1 
\end{cases}$$

assuming $n$ is a power of 2
Proof by induction

**Proposition.** If $T(n)$ satisfies the following recurrence, then $T(n) = n \log_2 n$.

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ 2T(n/2) + n & \text{if } n > 1 \end{cases}$$

**Pf.** [by induction on $n$]

- **Base case:** when $n = 1$, $T(1) = 0 = n \log_2 n$.
- **Inductive hypothesis:** assume $T(n) = n \log_2 n$.
- **Goal:** show that $T(2n) = 2n \log_2 (2n)$.

Given the recurrence

$$T(2n) = 2T(n) + 2n$$

By inductive hypothesis,

$$= 2n \log_2 n + 2n$$

$$= 2n (\log_2 (2n) - 1) + 2n$$

$$= 2n \log_2 (2n). \quad \blacksquare$$
Which is the exact solution of the following recurrence?

\[
T(n) = \begin{cases} 
0 & \text{if } n = 1 \\
T([n/2]) + T([n/2]) + n - 1 & \text{if } n > 1 
\end{cases}
\]

- A. \( T(n) = n \lceil \log_2 n \rceil \)
- B. \( T(n) = n \lceil \log_2 n \rceil \)
- C. \( T(n) = n \lfloor \log_2 n \rfloor + 2^\lfloor \log_2 n \rfloor - 1 \)
- D. \( T(n) = n \lceil \log_2 n \rceil - 2^\lceil \log_2 n \rceil + 1 \)
- E. Not even Knuth knows.
Analysis of mergesort recurrence

**Proposition.** If $T(n)$ satisfies the following recurrence, then $T(n) \leq n \lceil \log_2 n \rceil$.

\[
T(n) \leq \begin{cases} 
0 & \text{if } n = 1 \\
T([n/2]) + T([n/2]) + n & \text{if } n > 1 
\end{cases}
\]

**Pf.** [ by strong induction on $n$ ]

- **Base case:** $n = 1$.
- **Define** $n_1 = [n/2]$ and $n_2 = [n/2]$ and note that $n = n_1 + n_2$.
- **Induction step:** assume true for $1, 2, \ldots, n-1$.

\[
T(n) \leq T(n_1) + T(n_2) + n
\]

- **Inductive hypothesis**

\[
\leq n_1 \lceil \log_2 n_1 \rceil + n_2 \lceil \log_2 n_2 \rceil + n
\]

\[
\leq n_1 \lceil \log_2 n_2 \rceil + n_2 \lceil \log_2 n_2 \rceil + n
\]

\[
= n_2 \lceil \log_2 n_2 \rceil + n
\]

\[
\leq n_2 ([\log_2 n] - 1) + n
\]

\[
= n \lceil \log_2 n \rceil. \quad \blacksquare
\]

\[
n_2 = [n/2]
\]

\[
\leq \left\lfloor 2^{\lceil \log_2 n \rceil} / 2 \right\rfloor
\]

\[
= 2^{\lceil \log_2 n \rceil} / 2
\]

\[
\log_2 n_2 \leq \lceil \log_2 n \rceil - 1
\]

an integer