This week

- Recurrences
  - Merge-Sort
- Master Theorem
- Quick Sort
### Merge-Sort

- **Problem**: Given a list $S$ of $n$ integers, create a sorted list of elements in $S$.
- **Merge-sort Algorithm**:
  - **Divide**: If $S$ has at least two elements (nothing needs to be done if $S$ is empty or has only one element), remove all the elements from $S$ and put them into two sequences, $S_1$ and $S_2$, each containing about half of the elements of $S$.
  - **Recursion**: Sort sequences $S_1$ and $S_2$.
  - **Conquer**: Put back the elements into $S$ by merging the sorted sequences $S_1$ and $S_2$ into a unique sorted sequence.

### Merging Two Sorted Sequences

- **Problem**: Given two sequences $S_1$ and $S_2$ of sizes $n_1$ and $n_2$, create a (union) sorted list $S$ (of size $n=n_1+n_2$).
- **Algorithm** $\text{Merge}(S_1, S_2, S)$:
  - $\text{top}(S_i) =$ first element in $S_i$, for $i$ in $\{1, 2\}$.
  - While $S_1$ is not empty and $S_2$ is not empty do
    - If $\text{top}(S_1) < \text{top}(S_2)$ then
      - move $\text{top}(S_1)$ at the end of $S$
      - advance $\text{top}(S_1)$
    - Else
      - move $\text{top}(S_2)$ at the end of $S$
      - advance $\text{top}(S_2)$
  - While $S_1$ is not empty do
    - move the remaining of $S_1$ to $S$
  - While $S_2$ is not empty do
    - move the remaining of $S_2$ to $S$
Recurrence for Merge Sort:

- Recurrence Relation:
  \[ T(n) = 2T(n/2) + n \]
  \[ T(1) = 1 \]

- Solution by unfolding:
  \[ T(n) = 2(2T(n/4)+(n/2))+n \]
  \[ = 4T(n/2) + 2n \]
  \[ = 4(2T(n/8)+(n/4))+2n \]
  \[ = 8T(n/8) + 3n = \ldots \]
  \[ = 2^iT(n/2^i) + i.n \]

The expansion stops for \( i = \log n \)

\[ T(n) = 2^{\log n} + n \log n \]

Total Number of moves:

\[ T(n) = n + n \log n = O(n \log n) \]

Iterative recurrences

- Example:
  \[ T(n) = 4T(n/4) + n \]
  \[ = n + 4(n/2 + 4T(n/4)) \]
  \[ = n + 2n + 16T(n/4) \]
  \[ = n + 2n + 16[n/4 + 4T(n/8)] \]
  \[ = n + 2n + 4n + 4T(n/8) \]
  \[ = n + 2n + 4n + \ldots \]
  \[ = n \sum_{i=0}^{\log n-1} 2^i + 4^{\log n}T(1) \]
  \[ = \Theta(n^2) + \Theta(n^2) \]

- Disadvantage:
  - Tedious
  - Error-Prone

- Use to generate initial guess, and then prove by induction.
Solving Recurrences by “Guess and Prove”

- Recurrence relation (for all even \( n \)):
  \[
  T(n) = 2T(n/2) + n \\
  T(1) = 1
  \]

- Step 1: Take a wild guess that
  \[
  T(n) = \log n + n \log n
  \]

- Step 2: Prove it by induction:
  Basis
  \[
  T(1) = 1 + \log 1 = 1
  \]
  Inductive step
  assume \( T(n) = n + n \log n \) and prove it for next case \( (n+2) \):
  \[
  T(n+2) = 2T((n+2)/2) + (n+2) \\
  = 2[(n+2)/2 + (n+2)/2 \log (n+2)/2] + (n+2) \\
  = 2[(n+2)/2 + (n+2)/2[(\log (n+2))-1]] + (n+2) \\
  = (n+2) \log (n+2) + (n+2)
  \]

Initial Condition

- Can initial condition affect the solution?
  \[
  T(n) = [T(n/2)]^2
  \]
  If \( T(1) = 2 \) \( \Rightarrow \) \( T(n) = 2^n \)
  If \( T(1) = 3 \) \( \Rightarrow \) \( T(n) = 3^n \)
  If \( T(1) = 1 \) \( \Rightarrow \) \( T(n) = 1 \)

- \( n \) was assumed to be a power of 2.
Recursion Tree

- Example: \( T(n) = T(n/4) + T(n/2) + n^2 \)

At \( k \)-the level we get a general formula: \( i \) steps right, \( k-i \) left

\[
n^2 \sum_{i} \binom{k}{i} \left( 2^{-i} 4^{-(k-i)} \right)^2 = n^2 \sum_{i} \binom{k}{i} \left( 4^{-i} 16^{-(k-i)} \right)^2 \\
= n^2 \left[ \frac{1}{4} + \frac{1}{16} \right] = n^2 \left[ \frac{5}{16} \right]^k \\
= \Theta(n^2)
\]

Master Method

- Consider the following recurrence
  1. \( f(n) = O(n^{\log_b a - \varepsilon}), \varepsilon > 0 \Rightarrow \Theta(n^{\log_b a}) \)
  2. \( f(n) = O(n^{\log_b a} \log^k n), k > 0 \Rightarrow \Theta(n^{\log_b a} \log^{k+1} n) \)
  3. \( f(n) = O(n^{\log_b a + \varepsilon}), \varepsilon > 0 \Rightarrow \Theta(f(n)) \)

- Let \( Q = n^{\log_b a} \). Then the cases are
  - \( Q \) polynomially larger than \( f \)
  - \( f \) larger than \( Q \) by a polynomial factor
  - \( Q \) polynomially smaller than \( f \)
### Build recursive tree

The tree:

- $f(n)$
- $af(n/b)$
- $a^2f(n/b^2)$

Last row: $\Theta(a^{\log_a a}) = \Theta(n^{\log_a a})$ elements, each one $\Theta(1)$.

Total: $\Theta(n^{\log_a a}) + \sum_{i=1}^{\log_a n - 1} a^i f(n/b^i)$

Which term dominates?

---

### Back to Algorithms

- **Quick Sort**
  - Sort in place
  - Very practical
  - Divide-and-conquer

- **Algorithm**
  - Divide into two arrays around the first element
  - Recursively sort each array
  - Merge/combine-trivial
### Partition Routine

\[
\text{Partition}(A, p, r) \\
\quad x = A(r) \\
\quad i = p - 1 \\
\quad \text{for } j = p \text{ to } r - 1 \\
\quad \quad \text{if } A(j) \leq x \text{ then} \\
\quad \quad \quad i++ \\
\quad \quad \quad \text{exchange}(A(i), A(j)) \\
\quad \text{exchange}(A(i+1), A(r)) \\
\quad \text{return}(i + 1)
\]

\[
\begin{array}{|c|c|c|}
\hline
\leq x & > x & ?? \\
\hline
p & i & j & r \\
\hline
\end{array}
\]

### Quick Sort

\[
\text{Quicksort}(A, p, r) \\
\text{while } (p < r) \\
\quad q = \text{partition}(A, p, r) \\
\quad \text{Quicksort}(A, p, q-1) \\
\quad \text{Quicksort}(A, q+1, r)
\]

- To simplify, assume distinct elements:
  - Lucky always an even element: \( T(n) = 2T(n/2) + \Theta(n) = \Theta(n \log n) \)
  - Unlucky: \( T(n) = 2T(0) + T(n-1) + \Theta(n) = \Theta(n^2) \)

- How to avoid bad case?
  - Partition around middle element (does not work!)
  - Idea: Partition around a random element!
QuickSort (cont'd.)

- Partition around a Randomly chosen element and let \( T(n) \) be the expected time to sort.
- Consider the case where the partition is \((k, n-k-1)\). In this case, the expected time to terminate is:
  \[
  T(k) + T(n-k-1) + \Theta(n)
  \]
- Condition on \( k \) being a specific value, note that any value of \( k \) from 0 to \( n-1 \) is equally likely:
  \[
  T(n) = \sum_k \Pr[(k, n-k-1) \text{split}] T(n \mid (k, n-k-1) \text{split})
  \]
  \[
  = \frac{1}{n} \sum_k [T(k) + T(n-k-1) + \Theta(n)]
  \]
  \[
  = \frac{2}{n} \sum_{k=0}^{n-1} [T(k) + \Theta(n)]
  \]

Solving the recurrence

- Next: We try to prove that \( T(n) \leq an \log n + b \)
  First, Choose \( b \) large enough to satisfy \( T(1) \leq a \log 1 + b = b \)
- Inductive step:
  \[
  T(n) = \frac{2}{n} \sum_{k=1}^{n-1} T(k) + \Theta(n) \leq \frac{2}{n} \sum_{k=1}^{n-1} ak \log k + b + \Theta(n)
  \]
  \[
  = \frac{2a}{n} \sum_{k=1}^{n-1} k \log k + \frac{2}{n} nb + \Theta(n)
  \]
  Need to prove this is \( \leq an \log n + b + \Theta(n) \)
  \[
  \leq \frac{2a}{n} \left( \frac{1}{2} n^2 \log n - \frac{1}{8} n^2 \right) + 2b + \Theta(n)
  \]
  \[
  = an \log n + b + \left( \Theta(n) + b - \frac{an}{4} \right)
  \]
Technical Lemma

- We need to show $n^2 \log n$ bound is true.

$$\sum_{k=1}^{n-1} k \log k = \sum_{k=1}^{\left\lfloor \frac{n-1}{2} \right\rfloor} k \log k + \sum_{k=\left\lceil \frac{n}{2} \right\rceil}^{n-1} k \log k$$

$$\leq \log n \left\{ \sum_{k=1}^{\left\lfloor \frac{n-1}{2} \right\rfloor} k - \sum_{k=1}^{\left\lceil \frac{n}{2} \right\rceil} k \right\}$$

$$\leq \log n \frac{n(n-1)}{2} - \frac{n}{2} \left( \frac{n}{2} - 1 \right)$$

$$\leq \frac{1}{2} n^2 \log n - \frac{n^2}{8}$$