Analysis of Recursions

- The following is the sloppy recurrence for time complexity of merge sort:

\[
T(n) = \begin{cases} 
2 \ T \left( \frac{n}{2} \right) + cn & \text{if } n > 1 \\
\text{d} & \text{if } n = 1.
\end{cases}
\]

- We can find the solution using alternation method:

\[
T(n) = 2 \ T(n/2) + cn \\
= 2(2 \ T(n/4) + cn/2) + cn = 4 \ T(n/4) + 2cn \\
= 4(2 \ T(n/8) + cn/4) + 2cn = 8 \ T(n/8) + 3cn \\
= \ldots \\
= 2^k \ T(n/2^k) + kcn \\
= 2^{\log n} \ T(1) + \log ncn = \Theta(n \log n)
\]
**Substitution method**

- **Guess** the growth function and prove an upper bound for it using induction.
  - For merge-sort, prove $T(n) < Mn \log n$ for some value of $M$ (that we choose).
  - This holds for $n = 2$ since we have $T(2) = 2d + 2c$, which is less than $2M$ as long as $M \geq c + d$ (base of induction).
  - Fix a value of $n$ and assume the inequality holds for smaller values. we have $T(n) = 2T(n/2) + cn \leq 2M(n/2)(\log n/2) + cn = Mn \log n - Mn + cn \leq Mn \log n$ as long as $M$ is selected to be no less than $c$ (the inequality comes from the induction hypothesis)

- This shows $T(n) \in O(n \log n)$
Master theorem

\[ T(n) = \begin{cases} 
  a \ T \left( \frac{n}{b} \right) + f(n) & \text{if } n > 1 \\
  d & \text{if } n = 1.
\end{cases} \]

\((a \geq 1, \ b > 1, \text{ and } f(n) > 0)\)

- Compare \(f(n)\) and \(n^{\log_b a}\)
- Case 1: if \(f(n) \in O(n^{\log_b a - \epsilon})\), then \(T(n) \in \Theta(n^{\log_b a})\)
- Case 2: if \(f(n) \in \Theta(n^{\log_b a})\) then \(T(n) \in \Theta(n^c \log n)\)
- Case 3: if \(f(n) \in \Omega(n^{\log_b a + \epsilon})\) and if \(af(n/b) \leq cf(n)\) for some constant \(c < 1\) (regularity condition), then \(T(n) \in \Theta(f(n))\)
Master theorem examples

- \( T(n) = 2T(n/2) + \log n \)?
Master theorem examples

- \( T(n) = 2T(n/2) + \log n \?)
  - case 1: \( T(n) \in \Theta(n) \)
Master theorem examples

- \( T(n) = 2T(n/2) + \log n \)
  - case 1: \( T(n) \in \Theta(n) \)
  - \( T(n) = 4T(n/4) + 100n \)
Master theorem examples

- \( T(n) = 2T(n/2) + \log n \)?
  - case 1: \( T(n) \in \Theta(n) \)
- \( T(n) = 4T(n/4) + 100n \)
  - case 2: \( T(n) \in \Theta(n \log n) \)
Master theorem examples

- $T(n) = 2T(n/2) + \log n$
  - case 1: $T(n) \in \Theta(n)$
- $T(n) = 4T(n/4) + 100n$
  - case 2: $T(n) \in \Theta(n \log n)$
- $T(n) = 3T(n/2) + n^2$
Master theorem examples

- \( T(n) = 2T(n/2) + \log n \)?
  - case 1: \( T(n) \in \Theta(n) \)
- \( T(n) = 4T(n/4) + 100n \)
  - case 2: \( T(n) \in \Theta(n \log n) \)
- \( T(n) = 3T(n/2) + n^2 \)
  - Case 3, check whether regularity condition holds, i.e., whether \( af(n/b) \leq cf(n) \).
  - Since we have \( 3(n/2)^2 = 3/4n^2 \) the regularity condition holds, i.e., \( f(n) \in \Theta(n^2) \)
Master theorem examples

- $T(n) = 2T(n/2) + \log n$?
  - case 1: $T(n) \in \Theta(n)$
- $T(n) = 4T(n/4) + 100n$
  - case 2: $T(n) \in \Theta(n \log n)$
- $T(n) = 3T(n/2) + n^2$
  - Case 3, check whether regularity condition holds, i.e., whether $af(n/b) \leq cf(n)$.
  - Since we have $3(n/2)^2 = 3/4n^2$ the regularity condition holds, i.e., $f(n) \in \Theta(n^2)$
- $T(n) = T(n/2) + n(2 - \cos(n))$
  - Case 3, check whether regularity condition holds.
  - For $n = 2k\pi$, we have $\cos(n/2) = -1$ and $\cos(n) = 1$; we have $af(n/b) = n/2(2 - \cos(n/2)) = 3n/2$, which is not within a factor $c < 1$ of $f(n) = n(2 - 1) = n$ [i.e., we cannot say $3n/2 \leq cn$ for any $c < 1$]. So we cannot get any conclusion from Master theorem.
QuickSort

QuickSelect is based on a sorting method developed by Hoare in 1960:

\[
\text{quick-sort1}(A)
\]
\[
A: \text{array of size } n
\]
\[
1. \quad \textbf{if } n \leq 1 \textbf{ then return}
2. \quad p \leftarrow \text{choose-pivot1}(A)
3. \quad i \leftarrow \text{partition}(A, p)
4. \quad \text{quick-sort1}(A[0, 1, \ldots, i - 1])
5. \quad \text{quick-sort1}(A[i + 1, \ldots, \text{size}(A) - 1])
\]

Here pivot is chosen arbitrarily (e.g., it is the first item in the array)
Worst case: \( T^{(\text{worst})}(n) = T^{(\text{worst})}(n - 1) + \Theta(n) \)

The algorithm has a running time of \( \Theta(n^2) \) in the worst case.
**QuickSort**

**Analysis of Quick-sort**

**Worst case:** \( T^{(\text{worst})}(n) = T^{(\text{worst})}(n - 1) + \Theta(n) \)

The algorithm has a running time of \( \Theta(n^2) \) in the worst case.

**Best case:** \( T^{(\text{best})}(n) = T^{(\text{best})}(\lfloor \frac{n-1}{2} \rfloor) + T^{(\text{best})}(\lceil \frac{n-1}{2} \rceil) + \Theta(n) \)

Similar to Merge-sort; \( T^{(\text{best})}(n) \in \Theta(n \log n) \)
QuickSort

Analysis of Quick-sort

**Worst case:** $T^{(\text{worst})}(n) = T^{(\text{worst})}(n - 1) + \Theta(n)$

The algorithm has a running time of $\Theta(n^2)$ in the worst case.

**Best case:** $T^{(\text{best})}(n) = T^{(\text{best})}([\frac{n-1}{2}]) + T^{(\text{best})}([\frac{n-1}{2}]) + \Theta(n)$

Similar to Merge-sort; $T^{(\text{best})}(n) \in \Theta(n \log n)$

**Any other best case?** $T(n) = T(n/100) + T(99n/100) + cn$

which belongs to $\Theta(n \log n)$
In a comparison-based sorting the running time is proportional to the total number of comparisons performed during partitioning.
QuickSort

Average-case analysis of quick-sort

- In a **comparison-based sorting** the running time is proportional to the total number of comparisons performed during partitioning.

- Let $X_n$ be a random variable denoting the number of comparisons made by quicksort on an array of size $n$.

  \[ E[X_n] = \text{expected no. of comparisons} = \sum_{i,j \in 0,\ldots,n-1} \text{prob}(\text{the } i\text{'th and } j\text{'th elements are compared}) \]

The expected time complexity will be

\[ \sum_{i,j \in 0,\ldots,n-1} \frac{8}{9} \]
QuickSort

**Average-case analysis of quick-sort**

- In a **comparison-based sorting** the running time is proportional to the total number of comparisons performed during partitioning.

- Let $X_n$ be a random variable denoting the number of comparisons made by quicksort on an array of size $n$.
  
  $$E[X_n] = \text{expected no. of comparisons} = \sum_{i,j \in 0,\ldots,n-1} \text{prob}(\text{the i’th and j’th elements are compared})$$
  
  Elements $i$ and $j$ are compared iff one of them is selected as a pivot at some point before any other element in $\{i+1, i+2, \ldots, j-1\}$. This occurs with probability $\frac{2}{j-i+1}$.

  - In sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, what is the chance that 3 and 7 are compared? → the chance that 4, 5, 6 are Not selected as pivot before 3, 7 → $2/5$
QuickSort

Average-case analysis of quick-sort

- In a **comparison-based sorting** the running time is proportional to the total number of comparisons performed during partitioning.

- Let $X_n$ be a random variable denoting the number of comparisons made by quicksort on an array of size $n$.
  
  \[ E[X_n] = \text{expected no. of comparisons} = \sum_{i,j \in 0,\ldots,n-1} \text{prob}(\text{the } i'th \text{ and } j'th \text{ elements are compared}) \]

- Elements $i$ and $j$ are compared iff one of them is selected as a pivot at some point before any other element in $\{i + 1, i + 2, \ldots, j - 1\}$. This occurs with probability $\frac{2}{j-i+1}$.
  
  - In sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, what is the chance that 3 and 7 are compared? $\rightarrow$ the chance that 4, 5, 6 are Not selected as pivot before 3, 7 $\rightarrow 2/5$

- The expected time complexity will be
  
  \[ \sum_{i,j \in 0,\ldots,n-1,j>i} \frac{2}{j-i+1} \]
For the expected time complexity of Quicksort, we have:

\[ E[X_n] = \sum_{i,j \in \{0,\ldots,n-1\}, j > i} \frac{2}{j - i + 1} \]

\[
= \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \frac{2}{j - i + 1} \\
\]

\[
= \sum_{i=0}^{n-2} \sum_{k=2}^{n-i} \frac{2}{k} < \sum_{i=0}^{n-2} \sum_{k=2}^{n} \frac{2}{k} \\
\]

\[
= \sum_{i=0}^{n-2} \Theta(\log n) = \Theta(n^2) 
\]

Note that we used the fact that the sum of Harmonic series belongs to \( \Theta(\log n) \).