Master Theorem, & Greedy

- Master Theorem
  \[ a \geq 1 \]
  \[ b > 1 \]
  \[ f(n) \text{: asymptotically positive} \]
- Ignore floors and ceilings for \( n/b \)

1. If \( f(n) = \Theta(n^{k \cdot \log b}) \) for some constant \( k > 0 \), then \( T(n) = \Theta(n^{k \cdot \log b}) \).
2. If \( f(n) = \Theta(n^{k \cdot \log b}) \), then \( T(n) = \Theta(n^{k \cdot \log b \cdot \log n}) \).
3. If \( f(n) = \Omega(n^{k \cdot \log b}) \) for some constant \( k > 0 \) and if \( a f(n/b) \leq c f(n) \) for some constant \( c < 1 \) and all sufficiently large \( n \), then \( T(n) = \Theta(f(n)) \).

Key idea: Compare \( n^{k \cdot \log b} \) with \( f(n) \)

- \( T(n) \) is determined by the larger one
- \( \begin{align*}
  f(n) = O(n^{\log a - \epsilon}) & \iff f(n) \leq c \cdot n \cdot n^{\log a - \epsilon} & f(n) \leq n^{\log a - \epsilon} & f(n) \leq n \implies f(n)n \leq cn
\end{align*} \)
\[ T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n) \quad a \geq 1 \quad b \geq 1 \quad f(n) \text{ is asymptotically positive} \]

- number of sub-problems
- size of the sub-problem
- the cost of the work done outside recursive calls
- dividing + combination

\begin{itemize}
  \item e.g. \( T(n) = 2T(n-1) + 1 \)
  \item \( T(n) = 9T(\frac{n}{3}) + n \quad a = 9 \quad b = 3 \quad f(n) = n \)
  \end{itemize}

\[ \log_9 a = n \quad \log_9 9 = n^2 \]

\[ f(n) = O\left(n^{\log_b a - \epsilon}\right) \quad n = O\left(n^{2-\epsilon}\right) \]

\[ T(n) = \Theta\left(n^{\log_b a}\right) = \Theta(n^2) \]
3. $T(n) = 3T\left(\frac{n}{4}\right) + n \quad a = 3 \quad b = 4 \quad n^{\log_a b} = n^{\log_a 3} \quad f(n) = n$

   1. $f(n) = \Omega\left(n^{\log_a b + \varepsilon}\right) \iff n = \Omega\left(n^{\log_a b + \varepsilon}\right)$

      $\iff n \geq c \cdot n^{\log_a 3 + \varepsilon}$

      $\log_a 3 + \varepsilon < 1$

   2. $a + f(n/b) \leq c \cdot f(n)$ for some constant $c$.

      $f(n) = n \quad a = 3 \quad b = 4 \quad 3f\left(\frac{n}{4}\right) \quad \underline{\quad c \cdot n \quad}$

      $3 \cdot \frac{n}{4} \leq c \cdot n$ ?

      $T(n) = \Theta\left(f(n)\right) = \Theta\left(n\right)$

4. $T(n) = T\left(\frac{2}{3}n\right) + 1 \quad a = 1 \quad b = \frac{3}{2} \quad n^{\log_b a} = n^{\log_{\frac{3}{2}} 1} = n^0 = 1$

      $T(n) = \Theta\left(n^{\log_{\frac{3}{2}} (gh)}\right) = \Theta(n)$
Some other functions

1. $T(n) = 2^n T\left(\frac{n}{2}\right) + n^n$  

2. $T(n) = 2 T\left(\frac{n}{2}\right) + \frac{n}{\log_2 n}$

3. $T(n) = 0.5 T\left(\frac{n}{2}\right) + n$  

4. $T(n) = 64 T\left(\frac{n}{8}\right) - n^2 \log n$  

$a \geq b \Rightarrow \eta^{\log_b a} = n$

$f(n) = \frac{n}{\log n}$

$f(n) = O\left(n^{\log_b a - \epsilon}\right)$

$\frac{n}{\log n} = O\left(n^{1-\epsilon}\right)$

$c \cdot n^{1-\epsilon} \geq \frac{n}{\log n}$

$c \geq \frac{n}{\log n \cdot n^{1-\epsilon}} = \frac{n}{\log n}$
Greedy algorithm design technique
- used to solve optimization problems

Greedy algorithm:
- Make the choice that looks the best at the moment
- Make locally optimal choice at each step in expectation of achieving global optimum
- Doesn't always yield optimal solution
- Usually fast & simple in implementation

Examples:
1. Dijkstra's algorithm
2. Prim's - MST
3. Kruskal's - MST
Two key ingredients

1. Greedy choice property
   - A global optimal solution can be achieved by making a locally optimal (greedy) choice

2. Optimal Substructure
   - A problem exhibits optimal substructure if an optimal solution to the problem contains within it optimal solution to subproblem.

Example: Shortest path between $S_0$ and $S_n$

\[
\text{cost}(P_{0,n}) = 1 + \text{cost}(P_{L,n})
\]
Design greedy algorithms according to the following sequence of steps:

1. Cast the optimization problem as one in which we make a choice and leave with one subproblem to solve.

2. Prove that there is always an optimal solution to the original problem that makes the greedy choice, so that the greedy choice is always safe.

3. Demonstrate optimal substructure by showing that, having made the greedy choice, what remains is a subproblem with the property that if we combine optimal solution to the subproblem with the greedy choice we have made, we arrive an optimal solution to the original problem.