Recurrences

Binary Search
- If array $A[1..n]$ is sorted, and key $x$ is in $A$, then $bsearch(x, A, 1, n)$ returns the index of $X$ in $A$.
- Algorithm
  
  ```
  bsearch(x, A, l, r)
  m = ⌊(l+r)/2⌋;
  if (x = A[m])
      then return(m);
  elseif (x < A[m])
      then return(bsearch(x, A, l, m))
  else return(bsearch(x, A, m+1, r))
  ```

Merge Sort
- Given array $A[1..n]$, $Merge-Sort(A, 1, n)$ sorts items in $A$.
- Algorithm
  
  ```
  Merge-Sort(A, l, r)
  if (l < r)
      then
          m = ⌊(l+r)/2⌋;
          Merge-Sort(A, l, m);
          Merge-Sort(A, m+1, r);
          Merge(A, l, m, r);
  ```

- Iteration
  - expand the recurrence
  - express as a sum of terms dependent on $n$ and the initial conditions (but not on $T$).
  - apply techniques for evaluating sums
- example

\[
T(1) = 1 \\
T(n) = T(n - 1) + n
\]

- example

\[
T(1) = 1 \\
T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n^2
\]

- Recursion Trees
  - visual, 2D display
  - each tree node represents a recursive call
  - each node is labeled with its complexity, exclusive of recursion
  - the children represent the recursive calls
  - sum each level in the tree, then sum the levels

- Substitution
  - guess the form
    * calculate first few terms
    * look for regularity
  - use induction to find constants and show the guess is correct
- example
  * note base case: $T(2)$ and $T(3)$
  * note form of induction: inequality
    $$T(1) = 1$$
    $$T(n) = 2T(\lfloor \frac{n}{2}\rfloor) + n$$

- example
  $$T(1) = 1$$
  $$T(n) = 2T(\lfloor \sqrt{n}\rfloor) + \log n$$

  - let $m = \log n$
  - then $n = 2^m$, $\lfloor \sqrt{n}\rfloor = 2^{m/2}$
  - and $T(2^m) = 2T(2^{m/2}) + m$
  - Let $S(m) = T(2^m)$
  - then $S(m) = 2S(m/2) + m$
  - This recurrence is similar to $T(n) = 2T(\lfloor n/2\rfloor) + n$ which has solution $T(n) = O(n \log n)$.
  - So $S(m) = O(m \log m) = O(\log n \log \log n)$

- Change of Variable
  - transform a problem to one with a known solution

- Master Method
  - consider recurrences of the form
    $$T(n) = aT(n/b) + f(n)$$
  where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function.
– Case 1. If, for some constant $\epsilon > 0$, 
\[ f(n) = O(n^{\log_b a - \epsilon}), \]
then 
\[ T(n) = \Theta(n^{\log_b a}) \]

– Example
\[
\begin{align*}
T(1) &= 1 \\
T(n) &= 4T(n/2) + n
\end{align*}
\]

– Case 2. If 
\[ f(n) = \Theta(n^{\log_b a}), \]
then 
\[ T(n) = \Theta(n^{\log_b a} \lg n) \]

– Example
\[
\begin{align*}
T(1) &= 1 \\
T(n) &= 4T(n/2) + n^2
\end{align*}
\]

– Case 3. If, for some constant $\epsilon > 0$, 
\[ f(n) = \Omega(n^{\log_b a + \epsilon}), \]
and if $af(n/b) \leq cf(n)$ for some $c < 1$ and sufficiently large $n$, then 
\[ T(n) = \Theta(f(n)) \]

– Example
\[
\begin{align*}
T(1) &= 1 \\
T(n) &= 4T(n/2) + n^3
\end{align*}
\]