Lecture 3: Binary search, Recurrences

Some slides courtesy from George Bebis
Recap: $O$, $\Omega$, and $\Theta$

The definitions imply a constant $n_0$ beyond which they are satisfied. We do not care about small values of $n$. 
Recurrences and Running Time

- An equation or inequality that describes a function in terms of its value on smaller inputs.

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

- Recurrences arise when an algorithm contains recursive calls to itself

- What is the actual running time of the algorithm?

- Need to solve the recurrence
  - Find an explicit formula of the expression
  - Bound the recurrence by an expression that involves \( n \)
Example Recurrences

• $T(n) = T(n-1) + n \quad \Theta(n^2)$
  – Recursive algorithm that loops through the input to eliminate one item

• $T(n) = T(n/2) + c \quad \Theta(\log n)$
  – Recursive algorithm that halves the input in one step

• $T(n) = T(n/2) + n \quad \Theta(n)$
  – Recursive algorithm that halves the input but must examine every item in the input

• $T(n) = 2T(n/2) + 1 \quad \Theta(n)$
  – Recursive algorithm that splits the input into 2 halves and does a constant amount of other work
Recurrent Algorithms

BINARY-SEARCH

- for an ordered array A, finds if \( x \) is in the array \( A[lo...hi] \)

**Alg.:** BINARY-SEARCH (A, lo, hi, x)

```
if (lo > hi)
    return FALSE

mid ← ⌊(lo+hi)/2⌋
if x = A[mid]
    return TRUE
if ( x < A[mid] )
    BINARY-SEARCH (A, lo, mid-1, x)
if ( x > A[mid] )
    BINARY-SEARCH (A, mid+1, hi, x)
```
Example

- \( A[8] = \{1, 2, 3, 4, 5, 7, 9, 11\} \)
  - \( \text{lo} = 1 \quad \text{hi} = 8 \quad x = 7 \)

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 7 & 9 & 11 \\
\end{array}
\]

mid = 4, \( \text{lo} = 5 \), \( \text{hi} = 8 \)

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 7 & 9 & 11 \\
\end{array}
\]

mid = 6, \( A[\text{mid}] = x \)
Found!
Another Example

- \( A[8] = \{1, 2, 3, 4, 5, 7, 9, 11\} \)
  
  - low = 1  hi = 8  \( x = 6 \)

  \[
  \begin{array}{cccccccc}
  1 & 2 & 3 & 4 & 5 & 7 & 9 & 11 \\
  \end{array}
  \]

  mid = 4, lo = 5, hi = 8

  \[
  \begin{array}{cccccccc}
  1 & 2 & 3 & 4 & 5 & 7 & 9 & 11 \\
  \end{array}
  \]

  mid = 6, \( A[6] = 7 \), lo = 5, hi = 5

  \[
  \begin{array}{cccccccc}
  1 & 2 & 3 & 4 & 5 & 7 & 9 & 11 \\
  \end{array}
  \]

  mid = 5, \( A[5] = 5 \), lo = 6, hi = 5

  NOT FOUND!
Analysis of BINARY-SEARCH

**Alg.:** BINARY-SEARCH (A, lo, hi, x)

- if \( (lo > hi) \)
  - return \text{FALSE}
  
  \[
  \text{mid} \leftarrow \lfloor (lo+hi)/2 \rfloor
  \]

- if \( x = A[mid] \)
  - return \text{TRUE}

- if \( x < A[mid] \)
  - BINARY-SEARCH (A, lo, mid-1, x) \hspace{1cm} \text{same problem of size n/2}

- if \( x > A[mid] \)
  - BINARY-SEARCH (A, mid+1, hi, x) \hspace{1cm} \text{same problem of size n/2}

- \[ T(n) = c + T(n/2) \]
  
  - \( T(n) \) – running time for an array of size \( n \)
Methods for Solving Recurrences

- Iteration method
- Substitution method
- Recursion tree method
- Master method
The Iteration Method

• Convert the recurrence into a summation and try to bound it using known series
  – Iterate the recurrence until the initial condition is reached.
  – Use back-substitution to express the recurrence in terms of $n$ and the initial (boundary) condition.
The Iteration Method

\[ T(n) = c + T(n/2) \]

\[
T(n) = c + c + T(n/4)
\]

\[
= c + c + c + T(n/8)
\]

Assume \( n = 2^k \)

\[
T(n) = c + c + \ldots + c + T(1)
\]

\[
= clgn + T(1)
\]

\[
= \Theta(lgn)
\]
Iteration Method – Example

\[ T(n) = n + 2T(n/2) \]

Assume: \( n = 2^k \)

\[
\begin{align*}
T(n) &= n + 2T(n/2) \\
&= n + 2(n/2 + 2T(n/4)) \\
&= n + n + 4T(n/4) \\
&= n + n + 4(n/4 + 2T(n/8)) \\
&= n + n + n + 8T(n/8) \\
&= \ldots \quad = in + 2^iT(n/2^i) \\
&= kn + 2^kT(1) \\
&= n\lg n + nT(1) = \Theta(n\lg n)
\end{align*}
\]
The substitution method

1. Guess a solution

2. Use induction to prove that the solution works
Substitution method

• Guess a solution
  – \( T(n) = O(g(n)) \)
  – Induction goal: apply the definition of the asymptotic notation
    • \( T(n) \leq d g(n) \), for some \( d > 0 \) and \( n \geq n_0 \)
  – Induction hypothesis: \( T(k) \leq d g(k) \) for all \( k < n \) (strong induction)

• Prove the induction goal
  – Use the induction hypothesis to find some values of the constants \( d \) and \( n_0 \) for which the induction goal holds
Example: Binary Search

\[ T(n) = c + T(n/2) \]

• **Guess:** \( T(n) = O(\log n) \)
  
  – Induction goal: \( T(n) \leq d \log n \), for some \( d \) and \( n \geq n_0 \)
  
  – Induction hypothesis: \( T(n/2) \leq d \log(n/2) \)

• **Proof of induction goal:**

\[
T(n) = T(n/2) + c \leq d \log(n/2) + c \\
= d \log n - d + c \leq d \log n \\
\text{if: } -d + c \leq 0, \ d \geq c
\]

• **Base case?**
Example 2

$T(n) = T(n-1) + n$

- **Guess:** $T(n) = O(n^2)$
  - Induction goal: $T(n) \leq c n^2$, for some $c$ and $n \geq n_0$
  - Induction hypothesis: $T(n-1) \leq c(n-1)^2$ for all $k < n$

- **Proof of induction goal:**

  $T(n) = T(n-1) + n \leq c(n-1)^2 + n$
  
  $= cn^2 - (2cn - c - n) \leq cn^2$

  if: $2cn - c - n \geq 0 \iff c \geq n/(2n-1) \iff c \geq 1/(2 - 1/n)$

  - For $n \geq 1 \Rightarrow 2 - 1/n \geq 1 \Rightarrow$ any $c \geq 1$ will work
Example 3

\[ T(n) = 2T(n/2) + n \]

- **Guess:** \( T(n) = O(n \log n) \)
  - Induction goal: \( T(n) \leq cn \log n \), for some \( c \) and \( n \geq n_0 \)
  - Induction hypothesis: \( T(n/2) \leq cn/2 \log(n/2) \)

- **Proof of induction goal:**
  
  \[
  T(n) = 2T(n/2) + n \leq 2c \frac{n}{2} \log \left( \frac{n}{2} \right) + n \\
  = cn \log n - cn + n \leq cn \log n
  \]
  
  if: \(- cn + n \leq 0 \Rightarrow c \geq 1\)

- **Base case?**
Changing variables

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

- Rename: \( m = \log n \Rightarrow n = 2^m \)

\[ T\left(2^m\right) = 2T\left(2^{m/2}\right) + m \]

- Rename: \( S(m) = T(2^m) \)

\[ S(m) = 2S(m/2) + m \Rightarrow S(m) = O(m\log m) \]

(demonstrated before)

\[ T(n) = T(2^m) = S(m) = O(m\log m) = O(\log n \log \log n) \]

Idea: transform the recurrence to one that you have seen before