Recursion and Induction

• Themes
  – Recursion
  – Recursive Definitions
  – Recurrence Relations
  – Induction (prove properties of recursive programs and objects defined recursively)

• Examples
  – Tower of Hanoi
  – Gray Codes
  – Hypercube
Tower of Hanoi

- There are three towers
- 64 gold disks, with decreasing sizes, placed on the first tower
- You need to move the stack of disks from one tower to another, one disk at a time
- Larger disks can not be placed on top of smaller disks
- The third tower can be used to temporarily hold disks
Tower of Hanoi

• Assume one disk can be moved in 1 second
  How long would it take to move 64 disks? N disks?

• To create an algorithm to solve this problem, it is convenient to generalize the problem to the “N-disk” problem, where in our case N = 64.
Recursive Solution
Recursive Solution
Recursive Solution
Recursive Solution
Tower of Hanoi
Tower of Hanoi
Tower of Hanoi
Tower of Hanoi
Tower of Hanoi
Tower of Hanoi
Tower of Hanoi
Tower of Hanoi
Recursive Algorithm

```plaintext
define hanoi (n, from, to, using)
  if n = 1 then
    move(from, to)
  else
    hanoi(n-1, from, using, to)
    move(from, to)
    hanoi(n-1, using, to, from)
```
Correctness

• Use induction to prove that the recursive algorithm solves the Tower of Hanoi problem.

• Let $H(n,a,b,c) = \text{property that } \text{hanoi}(n,a,b,c) \text{ moves } n \text{ disks from tower } a \text{ to } b \text{ using tower } c \text{ without placing larger disks on top of smaller disks}$. 
Correctness

• **Base case:**
  • $H(1,a,b,c)$ works since $hanoi(1, a, b, c) = move(a,b)$

• **Inductive Hypothesis (IH):**
  • Assume $H(n,a,b,c)$ is correct for arbitrary $a b c$

• **Show IH $\Rightarrow H(n+1,a,b,c)$**
  • The first recursive call correctly moves $n$ disks from tower $a$ to tower $c$ [by IH]
  • $move$ moves the largest disk to the empty tower $b$ (all other disks on tower $c$)
  • The second recursive call correctly moves $n$ disks from tower $c$ to tower $b$ on top of largest disk
Cost

• The number of moves $M(n)$ required by the algorithm to solve the n-disk problem satisfies the recurrence relation
  – $M(n) = 2M(n-1) + 1$
  – $M(1) = 1$
Cost

• We’d like to find a closed form, a formula that is not a recurrence relation

• We can do this a couple ways:
  – We can guess at the answer (and then prove it, of course)
  – We can unwind the recurrence (still need to prove it)
Guess and Prove

- Calculate $M(n)$ for small $n$ and look for a pattern.
- Guess the result and prove your guess correct using induction.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$M(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>31</td>
</tr>
</tbody>
</table>
Proof by Induction

• **Base case:** \( M(1) = 1 = 2^1 - 1 \)

• **Inductive Hypothesis (IH)**
  - Assume \( M(n) = 2^n - 1 \)
  
  • **Show IH \( \Rightarrow M(n+1) = 2^{n+1} - 1 \)**
    - \( M(n+1) = 2M(n) + 1 \)
    - \( = 2(2^n - 1) + 1 \) [by IH]
    - \( = 2 \times 2^n - 1 = 2^{n+1} - 1 \)
Substitution Method

- Unwind recurrence, by repeatedly replacing $M(n)$ by the r.h.s. of the recurrence until the base case is encountered.

$$M(n) = 2M(n-1) + 1$$

$$= 2^2 \times [2 \times M(n-2) + 1] + 1 + 2$$

$$= 2^3 \times M(n-3) + 1 + 2 + 2^2$$
Geometric Series

• After k steps: (prove by induction on k)
  \[ M(n) = 2^k \cdot M(n-k) + 1 + 2 + 2^2 + \ldots + 2^{k-1} \]

• Base case, M(1), encountered when n-k=1
  \[ \Rightarrow k = n-1 \]

• Substituting in, to get rid of all of the k’s:
  \[ M(n) = 2^{n-1} \cdot M(1) + 1 + 2 + 2^2 + \ldots + 2^{n-2} \]
  \[ = 1 + 2 + \ldots + 2^{n-1} = \sum_{i=0}^{n-1} 2^i = 2^n - 1 \]

• Use induction to reprove result for M(n) using this sum. Generalize by replacing 2 by x.
Induction

• After k steps: (prove by induction on k)
  \[ M(n) = 2^k \times M(n-k) + 1 + 2 + 2^2 + \ldots + 2^{k-1} \quad (1 \leq k < n) \]

• Base case:
  • Using recurrence after 1 step \( M(n) = 2M(n-1) + 1 \)

• Induction Hypothesis
  \[ M(n) = 2^k \times M(n-k) + 1 + 2 + 2^2 + \ldots + 2^{k-1} \]

• Show IH \( \Rightarrow \)
  \[ M(n) = 2^{k+1} \times M(n-(k+1)) + 1 + 2 + 2^2 + \ldots + 2^k \]
Induction

• Show IH ⇒

\[ M(n) = 2^{k+1} \times M(n-(k+1)) + 1 + 2 + 2^2 + \ldots + 2^k \]

\[ M(n) = 2^k \times M(n-k) + 1 + 2 + 2^2 + \ldots + 2^{k-1} \quad [IH] \]

\[ = 2^k \times (2M(n-k-1)+1) + 1 + 2 + 2^2 + \ldots + 2^{k-1} \quad [recurrence] \]

\[ = 2^k \times 2M(n-k-1) + 2^k + 1 + 2 + 2^2 + \ldots + 2^{k-1} \]

\[ = 2^{k+1} \times M(n-(k+1)) + 1 + 2 + 2^2 + \ldots + 2^k \]
Geometric Series

• Show \(1 + 2 + \ldots + 2^{n-1} = \sum_{i=0}^{n-1} 2^i = 2^n - 1\)

• Base case: \(\sum_{i=0}^{0} 2^i = 1 = 2 - 1\)

• Inductive Hypothesis \(\sum_{i=0}^{n-1} 2^i = 2^n - 1\)

• Show \(\sum_{i=0}^{n} 2^i = 2^n + \sum_{i=0}^{n-1} 2^i = 2^n + (2^n - 1) = 2 \times 2^n - 1 = 2^{n+1} - 1\)

• Generalize by replacing 2 by \(x\). \(\sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1}\)
Worst Case

• Is it possible to solve Hanoi in fewer than $2^n - 1$ moves?

• Let $M^*(n)$ be the fewest moves required to solve $H(n,a,b,c)$

• Claim $M^*(n) \geq 2M^*(n-1) + 1$ [why?]
  • Use induction to show $M^*(n) \geq 2^n - 1$
Iterative Solution to Hanoi

- It is possible to derive an iterative solution to the tower of Hanoi problem, but it is much more complicated than the recursive solution.
- The solution is related to the binary reflected Gray code.
- An iterative method for computing the binary reflected Gray code will be derived and this will lead to an iterative solution of Tower of Hanoi.
Gray Code

• An n-bit Gray code is a 1-1 onto mapping from [0..2^n-1] such that the binary representation of consecutive numbers differ by exactly one bit.

• Invented by Frank Gray for a shaft encoder - a wheel with concentric strips and a conducting brush which can read the number of strips at a given angle. The idea is to encode 2^n different angles, each with a different number of strips, corresponding to the n-bit binary numbers.
Shaft Encoder (Counting Order)

Consecutive angles can have an abrupt change in the number of strips (bits) leading to potential detection errors.
Since a Gray code is used, consecutive angles have only one change in the number of strips (bits).
Binary-Reflected Gray Code

- \( G_1 = [0,1] \)
- \( G_n = [0G_{n-1},1\overline{G}_{n-1}] \), \( \overline{G} \) \( \Rightarrow \) reverse order \( \equiv \) complement leading bit

- \( G_2 = [0G_1,1\overline{G}_1] = [00,01,11,10] \)
- \( G_3 = [0G_2,1\overline{G}_2] = [000,001,011,010,110,111,101,100] \)
- Use induction to prove that this is a Gray code
Iterative Formula

• Let $G_n(i)$ be a function from $[0, \ldots , 2^n-1]$

• $G_n(i) = i \oplus (i \gg 1)$ [exclusive or of $i$ and $i/2$]
  – $G_2(0) = 0$, $G_2(1) = 1$, $G_2(2) = 3$, $G_2(3) = 2$

• Use induction to prove that the sequence $G_n(i)$, $i=0,\ldots,2^n-1$ is a binary-reflected Gray code.
Gray Code & Tower of Hanoi

• Introduce coordinates \((d_0, \ldots, d_{n-1})\), where \(d_i \in \{0, 1\}\)
• Associate \(d_i\) with the \(i\)th disk
• Initialize to \((0, \ldots, 0)\) and flip the \(i\)th coordinate when the \(i\)-th disk is moved
• The sequence of coordinate vectors obtained from the Tower of Hanoi solution is a Gray code (why?)
Tower of Hanoi

(0,0,0)

[000, 001, 011, 010, 110, 111, 101, 100]
Tower of Hanoi

(0,0,1)

[000, 001, 011, 010, 110, 111, 101, 100]
Tower of Hanoi

(0,1,1)

[000,001,011,010,110,111,101,100]
Tower of Hanoi

(0,1,0)

[000,001,011,010,110,111,101,100]
Tower of Hanoi

$(1,1,0)$

$[000,001,011,010,110,111,101,100]$
Tower of Hanoi

(1,1,1)

[000,001,011,010,110,111,101,100]
Tower of Hanoi

(1,0,1)

[000,001,011,010,110,111,101,100]
Tower of Hanoi

(1,0,0)

[000,001,011,010,110,111,101,100]
Hypercube

Graph (recursively defined)

n-dimensional cube has $2^n$ nodes with each node connected to n vertices

Binary labels of adjacent nodes differ in one bit
Hypercube, Gray Code and Tower of Hanoi

A Hamiltonian cycle is a sequence of edges that visit each node exactly once and ends at a node with an edge to the starting node.

A Hamiltonian cycle on a hypercube provides a Gray code (why?)
A Hamiltonian cycle is a sequence of edges that visit each node exactly once and ends at a node with an edge to the starting node.

A Hamiltonian cycle on a hypercube provides a Gray code (why?)

000 001 101 100 110 111
010 011 100 101 110 111
000 001 101 100 110 111
Hypercube, Gray Code and Tower of Hanoi

A Hamiltonian cycle is a sequence of edges that visit each node exactly once and ends at a node with an edge to the starting node.

A Hamiltonian cycle on a hypercube provides a Gray code (why?)
A Hamiltonian cycle is a sequence of edges that visit each node exactly once and ends at a node with an edge to the starting node.

A Hamiltonian cycle on a hypercube provides a Gray code (why?)
A Hamiltonian cycle is a sequence of edges that visit each node exactly once and ends at a node with an edge to the starting node.

A Hamiltonian cycle on a hypercube provides a Gray code (why?)
Hypercube, Gray Code and Tower of Hanoi

A Hamiltonian cycle is a sequence of edges that visit each node exactly once and ends at a node with an edge to the starting node.

A Hamiltonian cycle on a hypercube provides a Gray code (why?)
Hypercube, Gray Code and Tower of Hanoi

A Hamiltonian cycle is a sequence of edges that visit each node exactly once and ends at a node with an edge to the starting node.

A Hamiltonian cycle on a hypercube provides a Gray code (why?)
A Hamiltonian cycle is a sequence of edges that visit each node exactly once and ends at a node with an edge to the starting node.

A Hamiltonian cycle on a hypercube provides a Gray code (why?)
A Hamiltonian cycle is a sequence of edges that visit each node exactly once and ends at a node with an edge to the starting node.

A Hamiltonian cycle on a hypercube provides a Gray code (why?)