CS311H: Discrete Mathematics

Recursive Definitions

Instructor: Işıl Dillig

Recursive Definitions

- Should be familiar with recursive functions from programming:

```
public int fact(int n) {
   if(n <= 1) return 1;
   return  n * fact(n - 1);
}
```

- Recursive definitions are also used in math for defining sets, functions, sequences etc.

Recursive Definitions in Math

- Consider the following sequence:

$$1, 3, 9, 27, 81, \ldots$$

- This sequence can be defined recursively as follows:

$$\begin{aligned} a_0 &= 1 \\ a_n &= 3 \cdot a_{n-1} \end{aligned}$$

- First part called base case; second part called recursive step
- Very similar to induction; in fact, recursive definitions sometimes also called inductive definitions

Recursively Defined Functions

- Just like sequences, functions can also be defined recursively
- Example:

$$\begin{aligned} f(0) &= 3 \\ f(n+1) &= 2f(n) + 3 \quad (n \geq 1) \end{aligned}$$

- What is $f(1)$?
- What is $f(2)$?
- What is $f(3)$?

Recursive Definition Examples

- Consider $f(n) = 2n + 1$ where n is non-negative integer
- What's a recursive definition for f?
- Consider the sequence $1, 4, 9, 16, \ldots$
- What is a recursive definition for this sequence?
- Recursive definition of function defined as $f(n) = \sum_{i=1}^{n} i$?

Recursive Definitions of Important Functions

- Some important functions/sequences defined recursively
- Factorial function:

$$\begin{aligned} f(1) &= 1 \\ f(n) &= n \cdot f(n-1) \quad (n \geq 2) \end{aligned}$$

- Fibonacci numbers: $1, 1, 2, 3, 5, 8, 13, 21, \ldots$

$$\begin{aligned} a_1 &= 1 \\ a_2 &= 1 \\ a_n &= a_{n-1} + a_{n-2} \quad (n \geq 3) \end{aligned}$$

- Just like there can be multiple bases cases in inductive proofs, there can be multiple base cases in recursive definitions

Inductive Proofs for Recursively Defined Structures

- Recursive definitions and inductive proofs are very similar
- Natural to use induction to prove properties about recursively defined structures (sequences, functions etc.)
- Consider the recursive definition:

$$f(0) = 1$$
$$f(n) = f(n-1) + 2$$

- Prove that $f(n) = 2n + 1$

Example

- Let f_n be n'th element in the Fibonacci sequence ($n \geq 1$)
- Prove: For $n \geq 3$, $f_n > \alpha^{n-2}$ where $\alpha = \frac{1+\sqrt{5}}{2}$
- Proof is by strong induction on n with two base cases
- Intuition 1: Definition of f_n has two base cases
- Intuition 2: Recursive step uses f_{n-1}, f_{n-2} $\Rightarrow$ strong induction
- Base case 1 (n=3): $f_3 = 2$, and $\alpha < 2$, thus $f_3 > \alpha$
- Base case 2 (n=4): $f_4 = 3$ and $\alpha^2 = \frac{(3+\sqrt{5})}{2} < 3$

Example, cont.

Prove: For $n \geq 3$, $f_n > \alpha^{n-2}$ where $\alpha = \frac{1+\sqrt{5}}{2}$

- Inductive step: Assuming property holds for f_i where $3 \leq i \leq k$, need to show $f_{k+1} > \alpha^{k-1}$
- First, rewrite α^{k-1} as $\alpha^2\alpha^{k-3}$
- α^2 is equal to $1 + \alpha$ because:

$$\alpha^2 = \left(\frac{1+\sqrt{5}}{2}\right)^2 = \frac{\sqrt{5}+3}{2} = \alpha + 1$$

- Thus, $\alpha^{k-1} = (\alpha+1)(\alpha^{k-3}) = \alpha^{k-2} + \alpha^{k-3}$

Example, cont.

- $\alpha^{k-1} = \alpha^{k-2} + \alpha^{k-3}$
- By recursive definition, we know $f_{k+1} = f_k + f_{k-1}$
- Furthermore, by inductive hypothesis:

$$f_k > \alpha^{k-2} \qquad f_{k-1} > \alpha^{k-3}$$

- Therefore, $f_{k+1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}$

$\square$

Recursively Defined Sets and Structures

- We saw how to define functions and sequences recursively
- We can also define sets and other data structures recursively
- Example: Consider the set S defined as:

$$3 \in S$$
$$\text{If } x \in S \text{ and } y \in S, \text{ then } x + y \in S$$

- What is the set S defined as above?

More Examples

- Give a recursive definition of the set E of all even integers:
  - Base case:
  - Recursive step:
- Give a recursive definition of the set I of inverses of 2 mod 5:
  - Base case:
  - Recursive step:

Strings and Alphabets

- Recursive definitions play important role in study of strings
- Strings are defined over an alphabet Σ
  - Example: $\Sigma_1 = \{a, b\}$
  - Example: $\Sigma_2 = \{0\}$
- Examples of strings over Σ_1: $a, b, aa, ab, ba, bb, \ldots$
- Set of all strings formed from Σ forms language called Σ^*
  - $\Sigma_2^* = \{\epsilon, 0, 00, 000, \ldots\}$

Recursive Definition of Strings

- The language Σ^* has natural recursive definition:
  - Base case: $\epsilon \in \Sigma^*$ (empty string)
  - Recursive step: If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$
- Since ϵ is the empty string, $\epsilon s = s$
- Consider the alphabet $\Sigma = \{0, 1\}$
- How is the string "1" formed according to this definition?
- How is "10" formed?

Recursive Definitions of String Operations

- Many operations on strings can be defined recursively.
- Consider function $l(w)$ which yields length of string w
- Example: Give recursive definition of $l(w)$
  - Base case:
  - Recursive step:

Another Example

- The reverse of a string s is s written backwards.
- Example: Reverse of "abc" is "bca"
- Give a recursive definition of the $\text{reverse}(s)$ operation
  - Base case:
  - Recursive step:

Palindromes

- A palindrome is a string that reads the same forwards and backwards
- Examples: "mom", "dad", "abba", "Madam I'm Adam", ...
- Give a recursive definition of the set P of all palindromes over the alphabet $\Sigma = \{a, b\}$
- Base cases:
- Recursive step:

Bitstrings

- A bitstring is a string over the alphabet $\{0, 1\}$
- Give a recursive definition of the set S of bitstrings that contain equal number of 0's and 1's.
- Base case:
- Recursion: