CSC148H Week 6

Ilir Dema, Michael Miljanovic

Summer 2021

What is Recursion?

- *Recursion*: solving a problem by reducing it to subproblems, then combining the subproblem solutions to solve the original problem
- Subproblems must have the same structure as the original problem and be easier to solve
- Some subproblems are so simple that they can be solved directly (without reducing them further)

Base Case

- The *base case* is the simplest case of a problem
- We can solve it directly, without subdividing further
- e.g. when summing a nested list: a single integer

Recursive Case

- When the problem is too tough to solve directly, we use recursion
- e.g. when summing a nested list: a list of sublists
- It’s critical that recursion brings us closer to the base case, or we will recurse indefinitely
- e.g. when summing a nested list: we recurse on problems whose depth is decreased by 1

Worksheet 1

Worksheet 1, tracing recursive functions

Binary Codes

- A binary code of length *r* is a string of *r* bits (0 or 1)
- There are 2 binary codes of length 1, 4 binary codes of length 2, 8 binary codes of length 3 *. . .*
- Given integer *r* , our task is to generate a list of all binary codes of length *r*

Binary Codes, Base Case

- First, what if r were 0?
- Can we write a function that generates a list of all 0-length binary codes?
- The correct return value is [”], because the only binary code of length 0 is the empty string

Binary Codes, Recursive Structure

- Given a list of all length $r - 1$ binary codes, how can you construct a list of all length r binary codes?
- Remember that when the length of the desired binary codes increases by 1, the number of binary codes doubles
- Each binary code of length $r - 1$ yields **two** binary codes of length r
- Strategy
  - Take each binary code of length $r - 1$ and append a 0 to it
  - Take each binary code of length $r - 1$ and append a 1 to it
  - Combine all of these into a new list and return it

Tracing the Binary Codes Function

- We already know what the function does with argument 0
- When tracing with argument 1, substitute [’’] when a call with argument 0 is made
- Then you know what the function does with argument 1, so you can trace it for argument 2 using a similar process
- And so on *. . .*

Worksheet 2

Worksheet 2, writing recursive functions Let’s start with nested_list_contains.

Permutations

- For a string of *n* characters, there are *n*! permutations
- A permutation is an ordering of the elements

e.g. the permutations of abc are abc, acb, bac, cab, bca, cba

Permutations

Let’s write a recursive function to generate all permutations of a string.
What is the base case?
What is the recursive structure of permutations?

Worksheet 3

Worksheet 3, Mutating Nested Lists