February 10, 2025

LEC 14 - Recursion

What Is Recursion?

- A recursive algorithm is one that calls itself on a smaller version of a problem
- With each call the problem becomes simpler
- At some point, the problem becomes trivial
    - Base case

Types of Recursion

- **N-1 approach** → Handle one entity, then call the recursion for N-1 entities
- **Divide and conquer** → Apply recursion to each half, quarter, etc.
- Other ways out of scope for this course

Programmer's Perspective

- Recursion is when a function calls itself directly
    - There are indirect recursions that won't be covered
- The goal is to solve a smaller part of the problem using the same function
- Some cases require us to combine the solution

Recursion Steps

1. Base case
    - Simplest problem (can't be broken up any further)
    - We stop recursing here and return a specific value
2. Recursive decomposition step
    - Breaks the problem down into smaller subproblems
    - Must guarantee to eventually get to the base case

February 10, 2025

LEC 14 - Recursion

Example

How do we sum all the elements of an n-depth list?

- Using for loops won't work because we don't know the depth, it could be infinite

```python
def sum_list(L):
      # Recursive Step
      if isinstance(L, list):
            s = 0
            for elem in L:
                  # Calculate the sum of each sublist recursively
                  s += sum_list(elem)
            return s
      # Base Case
      else:
            return L
```

Partial Tracing

- Attempting to fully trace recursive code is time-consuming and error prone
- When tracing recursive code, dont trace into the recursive calls
    - Assume each call is correct and make sure the code uses those calls correctly