CSC 212: Data Structures and Abstractions

12: Linked Lists (part 2)

Prof. Marco Alvarez

Department of Computer Science and Statistics
University of Rhode Island

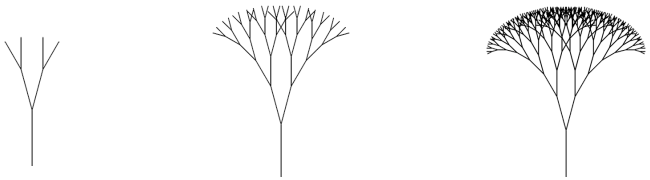
Fall 2025

Recursion

- Definition
  - method of solving problems that involves breaking a problem into smaller and smaller subproblems (of the same structure) until reaching a small enough problem that can be solved trivially
- Recursive functions
  - technically, a recursive function is one that invokes itself
  - must contain at least one base case and one recursive call
  - **base case**: a terminating condition that halts the recursion
  - **recursive case**: a condition that perpetuates the recursion by calling the function again

Why recursion?

- Can we live without it?
  - yes, every recursive function has an equivalent iterative solution
- However …
  - some formulas are inherently recursive in nature
  - some problems naturally lend themselves to recursive solutions



https://courses.cs.washington.edu/courses/cse120/17sp/labs/11/tree.html

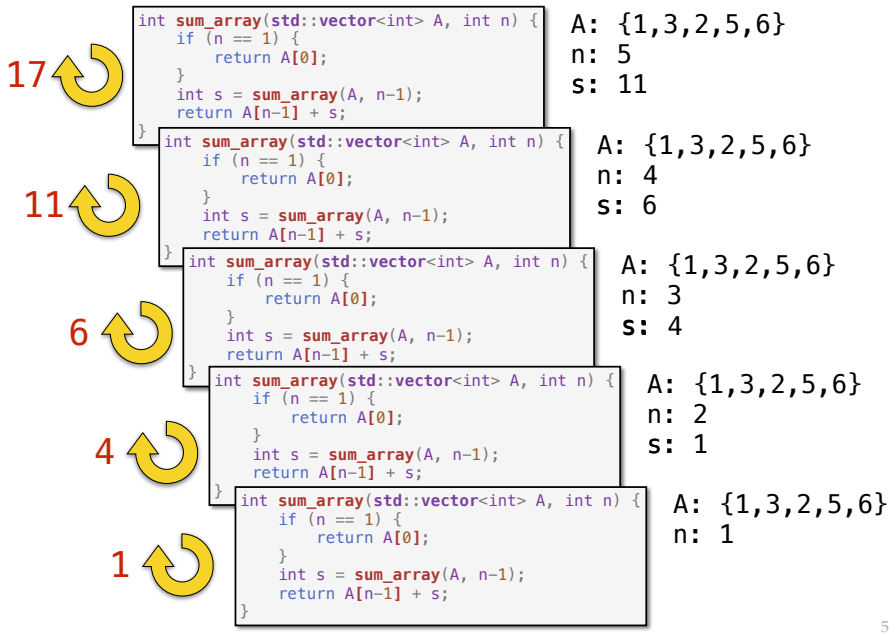
Practice

- Write a recursive function to add all elements in a vector
  - trace the call sequence with an input array {1,3,2,5,6}, including the parameters passed at each step

```
int sum_array(std::vector<int>& A, int n) {
    // base case
    if (n == 1) {
        return A[0];
    }

    // recursive call
    int s = sum_array(A, n-1);

    // return sum
    return A[n-1] + s;
}
```



SLL Methods (recursive implementation)

- print()
  - ✓ uses a temporary pointer to traverse the list starting from the head
  - ✓ prints the value stored in each node during traversal
- clear()
  - ✓ traverse the list and deletes each node
  - ✓ resets head and tail to nullptr and size to zero
- search(value)
  - ✓ uses a temporary pointer to traverse the list starting from the head
  - ✓ compares each node's value with the target value
  - ✓ returns true if the value is found; otherwise, returns false

Some recursive methods need **helpers** to maintain a clean public interface while the helper handles the extra parameters that recursion requires internally

Methods (recursive implementation)

- at(index)
  - ✓ returns the element at the specified index (starting from 0)
- reverse()
  - ✓ traverses the list and reverses the direction of all node pointers
  - ✓ swaps the head and tail pointers