Recursion

Recursion is defining something in terms of itself.

- We define many data structures recursively
  - A linked list node contains a pointer to a node
  - A binary tree node contains two pointers to nodes
- Many functions can be defined recursively:
  - Factorial: \( n! = n(n-1)! \)
  - Differentiation (chain rule): \( \frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx} \)
  - The binomial coefficient: \( \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1} \)
- Euclid's algorithm for gcd is recursive!

Recursive Functions in C++

- Most modern programming languages allow recursion in functions;
- In C++, you simply call a function from within itself, e.g.:
  ```cpp
  unsigned int factorial(unsigned int n) {
    if (n == 0) return 1;
    return n * factorial(n - 1);
  }
  ```

The Base Case

Note the first line of the `factorial` function:
```cpp
unsigned int factorial(unsigned int n) {  
    if (n == 0) return 1;  
    return n * factorial(n-1);  
}
```

What would happen without that line?

When the input \( n \) is 0 we call it the base case. The test for the base case must come first!

Example: Palindrome

- A palindrome is a recursive object; it is:
  - Empty, or
  - A single character, or
  - A palindrome between two of the same character
- Here's a recursive test function:
  ```cpp
  bool is_palindrome(const string &s, int start, int end) {
    if (end <= start) return true;
    return s[start] == s[end] &&
           is_palindrome(s, start+1, end-1);
  }
  ```

Example: Binomial Coefficient

```cpp
unsigned int nchoosek(unsigned int n, unsigned int k) {
    assert(n >= k);
    if (k == 0 || k == n) return 1;
    return nchoosek(n-1,k) + nchoosek(n-1,k-1);
}
```

Note - more than one base case!

Note - two recursive calls!
Common Mistakes

- No base case:
  ```
  void infinite(int n) {
    cout << n << endl;
    infinite(n-1);
  }
  ```

- Recursion step doesn’t reduce problem:
  ```
  void infinite2(int n) {
    if (n < 0) return;
    cout << n << endl;
    infinite2(n);
  }
  ```

Recursion vs. Iteration

Recursion is often the simplest approach.

However, recursion can often be replaced by iteration plus some storage for intermediate results.

```c
unsigned int factorial(unsigned int n) {
    unsigned int ans = 1;
    for (int j = n; j > 1; j--) ans = ans * j;
    return ans;
}
```