Recursion

•••

Data Structures Lab – Fall 2020
Faisal Khan: faisal.khan@nu.edu.pk

What is recursion?

1. The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function.

OR

1. The process of defining an object in terms of smaller versions of itself is called recursion.

   E.g Apple slicing

Fundamentals

Any recursive definition has two parts:

1. Base
2. Recursion

Base: An initial simple definition which can not be expressed in terms of smaller version.

Recursion: The part of the definition which can be expressed in terms of smaller versions of itself.

Example

```
def cutting(apple):
    if check_if_can_be_sliced(apple):
        - cut the apple into slice
        - apple = remaining apple
        - cutting(apple)
    else return done
```

Forward and backward code execution

Forward execution: Portion of code which comes before recursive function will be executed in forward pass.

Backward execution: Portion of code which comes after recursive function will be executed in backward pass.

Example 2: Printing numbers in forward and backward order

```
void reverse(int n){
  if (n == 0)
     return;
  else {
     cout<<n;
     reverse(n-1);
     cout<<n;  }
}
int main(){
  reverse(5);
}
```

```
void reverse(int n){
  if (n == 0)
     return;
  else {
     cout<<n;
     reverse(n-1);
     cout<<n;  }
}
```

```
void reverse(int n){
  if (n == 0)
     return;
  else {
     cout<<n;
     reverse(n-1);
     cout<<n;  }
}
```

```
void reverse(int n){
  if (n == 0)
     return;
  else {
     cout<<n;
     reverse(n-1);
     cout<<n;   }
}
```

```
void reverse(int n){
  if (n == 0)
     return;
  else {
     cout<<n;
     reverse(n-1);
     cout<<n;   }
}
```

```
void reverse(int n){
  if (n == 0)
     return;
  else {
     cout<<n;
     reverse(n-1);
     cout<<n;   }
}
```

Example 3: Powers

```
int power(int x, int exp) {
  if (exp == 0)
    return 1;
  else
    return x * power(x, exp-1);
}

power(5,3)
```

```
int power(int x, int exp) {
  if (exp == 0)
    return 1;
  else
    return x * power(x, exp-1);
}
```

```
int power(int x, int exp) {
  if (exp == 0)
    return 1;
  else
    return x * power(x, exp-1);
}
```

```
int power(int x, int exp) {
  if (exp == 0)
    return 1;
  else
    return x * power(x, exp-1);
}
```

Extra Examples

- Tree traversal – pre-order, in-order, post-order
- Graph traversal – cycles, connected components

Two Musts for Recursion

- Base case is a must, that makes no recursive calls
- When you make a recursive call it should be to a simpler instance and make forward progress towards the base case.

Recap

- Break a problem into smaller subproblems of the same form and call the same function again on that smaller problem.
- Powerful programming tool
- Not always a wise choice, but often a good one
- Some problems are solved easily by recursion than by iteration