

CS Bridge Module 12 Recursions Part 2

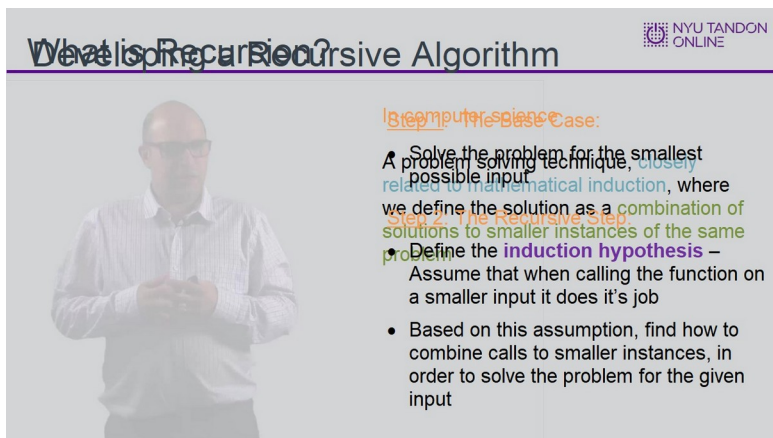
1. Recursions - Part 2

1.1 CS Bridge: Recursions



Notes:

1.2 What is Recursion?



Notes:

1.3 Print Ascending Problem

Print Ascending Problem

NYU TANDON ONLINE

Step 1: The Recursive Step

Definition: This is the problem with a smaller input a solution for the given input

The input's size of: n

"If we have an `printAsc` (implement a smaller function):
`void printAsc(int start, int end);`
 print the numbers in the range in ascending order."
 If a function could print the numbers from `start` to `end`, we could `cout<<start<<endl;`
 or in case (3, 5) for.

Assume that `start ≤ end`

Example: `printAsc(7, 9)`
`printAsc(start, end-1);`
`cout<<end<<endl;`
`printAsc(0, 5)`

$n = \text{start} + 1$
 $n = 3$

 $n = 1$
 end

$n = \text{end} - 1$
 1
 2
 3
 4
 end

$n = (\text{number of integers in the range from start to end})$

1.4 Tracing printAsc

Tracing printAsc

NYU TANDON ONLINE

Tracing Execution

Initial state: `printAsc(1, 4)`

```
void printAsc(int start, int end){
    if (start == end){
        cout<<start<<endl;
    }
    else{
        printAsc(start, end-1);
        cout<<end<<endl;
    }
}
```

1
2
3
4

✓ `printAsc(1, 2)` → 1
2

Notes:

1.5 Tracing printAsc Counted

Tracing printAsc Counted

Tracing Execution using runtime stack

Let's execute `printAsc(1, 4)`

```
void printAsc(int start, int end){
    if (start == end){
        cout<<start<<endl;
    }
    else{
        printAsc(start, end-1);
        cout<<end<<endl;
    }
}
```

1
2
3
4

start = 1
end = 4
start = 1
end = 3
start = 1
end = 2
start = 1
end = 1

1.6 printAsc Alternative Implementations

printAsc Alternative Implementations

Other Solutions: Version 3

```
void printAsc(int start, int end){
    if (start == end){
        cout<<start<<endl;
    }
    else{
        printAsc(start, end-1);
        printAsc(start+1, end);
    }
}
```

start
start+1
start
...
start+1
end-1
end

Notes:

1.7 Developing a Recursive Algorithm

Developing a Recursive Algorithm

NYU TANDON ONLINE

Problem

Step 1: The Base Case

Write a recursive implementation for the function:

- Solve the problem for the smallest possible input.

Step 2: The Recursive Step

Assume that the function `printAscAndDesc` works for all inputs smaller than the current input. Based on this assumption, find how to combine calls to smaller instances, in order to solve the problem for the given input.

```
void printAscAndDesc(int start, int end) {  
    if (start == end) {  
        cout << start << endl; return;  
    }  
    printAscAndDesc(start+1, end);  
    cout << start << endl; return;  
}
```

`printAscAndDesc(3, 5)`

Notes:

1.8 End of Module

NYU TANDON ONLINE

End of Module

Exit