IIT Bombay

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Recap of Recursive Functions

VIDEO LECTURE RECAP QUIZ

Video Lecture Recap Quiz

IIT Bombay

Q1. A function that calls itself is an example of _______________ function

A. Recursive
B. Iterative
C. Non-terminating
D. Mirror

Video Lecture Recap Quiz

Q2. Which of the following is/are FALSE about recursive functions?

A. Must have at least one parameter

B. Can have only call-by-value parameters

C. May not terminate for some input parameters

D. Cannot call any function other than itself

Video Lecture Recap Quiz

IIT Bombay

Q3. For recursion to terminate, the values of parameters

A. Can change in any order

B. Must move monotonically towards a termination case

C. Must stay unchanged in all calls to the function

D. Must never become negative

Q4. Virahanka numbers can be computed

A. Recursively but not iteratively
B. Iteratively but not recursively
C. Both iteratively and recursively
D. Neither iteratively nor recursively

Video Lecture Recap Quiz

IIT Bombay

Q5. When a function recursively calls itself

A. **The activation record on top of the call stack is popped out**
B. **A new activation record is pushed in the call stack**
C. **The activation record on top of the call stack is overwritten**
D. **No activation records are pushed/popped**

Video Lecture Recap Quiz

IIT Bombay

Q6. Specifying a termination case

A. Guarantees that a recursive function terminates for all inputs
B. May not cause a recursive function to terminate for all inputs
C. Terminates a function if it calls itself
D. Helps the compiler avoid generating code for recursive functions

VIDEO LECTURE RECAP SLIDES

Calling a Function From Itself

IIT Bombay

- Same mechanism of function calls and returns we studied earlier works perfectly !!!

Recursive Function: One that can call itself
Elegant and natural way to solve several problems

Mutually recursive functions
func1 calls func2, which calls func3,
which calls func1

A Program With A Recursive Function

IIT Bombay

```
#include <iostream>
using namespace std;
int newEnc(int q1Marks,int q2Marks);
int main() { …
  for ( … ) {  …
   cipher = newEnc(q1Marks, q2Marks);
  …}
 …
 return 0;
}
```

```
// PRECONDITION:  …
int newEnc(int q1Marks,
                 int q2Marks)
{ switch(q2Marks) {
    case 1:
      if (q1Marks == 1) {return 6;}
      else {return
             2*newEnc(q1Marks – 1, 1);
      }
      break;
    default: … }
}
// POSTCONDITION:  …
```

Caveats Using Recursive Functions

IIT Bombay

- Must specify how to terminate the recursion

  Otherwise, recursion (calling a function from itself) can go on forever

- Must ensure … anges parameters i… entually terminates

Changing parameters in an orderly way to ensure termination

encode(m, n) = encode(m, n-1) x 3, if m, n > 1

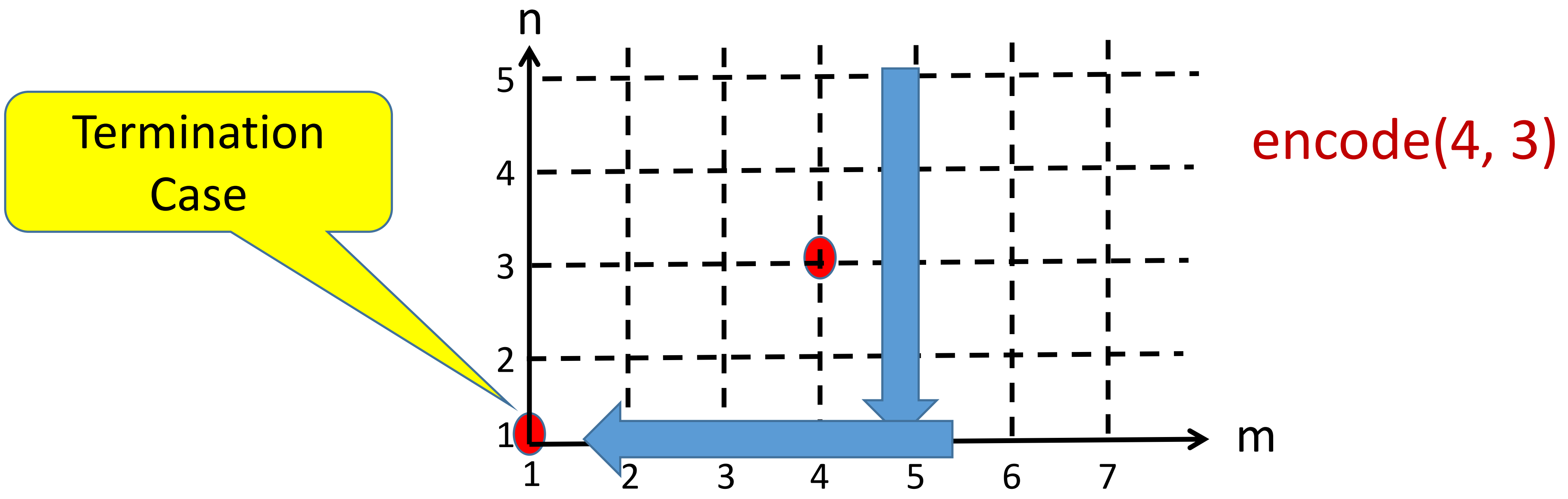
= encode(m-1, 1) x 2, if m > 1, n=1

= 2 x 3 = 6, if m=1, n=1

Termination case

Caveats Using Recursive Functions

- Think of all possible valuations of parameters as ordered with a fixed end (termination case)
- Recursion must change values of parameters so that we move along this order monotonically towards fixed end



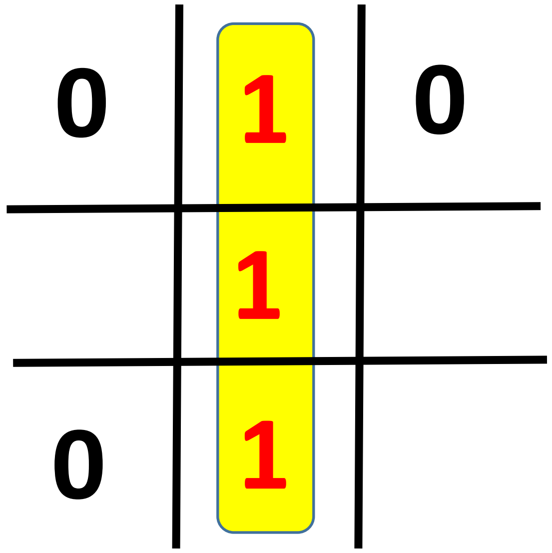
Recursion vs Iteration

IIT Bombay

- Recursive formulation usually clean, intuitive and succinct
  Need to worry about recursion termination (well-founded ordering of parameter values)
  Need to worry about number of recursive calls
- Iterative formulation may be less clean or intuitive (not always!)
  Need to worry about loop invariants, loop variants and termination
  Can be very efficient if formulated correctly
- Best practice: Judicious mix of iteration and recursion

Practice Problem

Let’s build on the problem discussed in last class. Recall the game of tic-tac-toe.



Practice Problem (continued)

IIT Bombay

A configuration of the tic-tac-toe grid is represented by a sequence of 9 integer valued variables x1, x2, … x9

x1	x2	x3
x4	x5	x6
x7	x8	x9

Practice Problem (continued)

IIT Bombay

A configuration of the tic-tac-toe grid is represented by a sequence of 9 integer valued variables x1, x2, … x9

0	1	0
	1	
0	1	

x1 = 0, x2 = 1, x3 = 0

x4 = -1, x5 = 1, x6 = -1

x7 = 0, x8 = 1, x9 = -1

Practice Problem: Subproblem 1

IIT Bombay

Write a C++ function that takes as input an input configuration and determines who ("0" or "1") should move next.

```
int nextTurn(int x1, int x2, … int x9)
{   // Check if configuration is valid
    // Count no. of 0's and 1's
    // Determine who moves next
}
```

Practice Problem (continued)

IIT Bombay

Given a configuration of tic-tac-toe, we want to determine if there is a winning/losing move of the next player.

Winning Move of 0: A move of 0 from which there is at least one way for 0 to win no matter how 1 plays.

Winning Move of 1 similarly defined

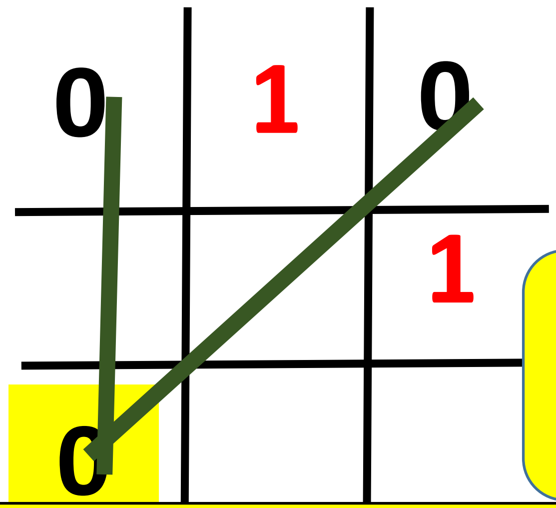
Practice Problem (continued)

IIT Bombay

Example: tic-tac-toe configuration

x1 = x3 = 0, x2 = x6 = 1, Rest are -1

x1	x2	x3
x4	x5	x6
x7	x8	x9



Next move: 0

Winning move of 0: x7 = 0

Guaranteed win for 0 no matter how 1 plays

Practice Problem (continued)

IIT Bombay

Example: tic-tac-toe configuration

x1 = x3 = 0, x2 = x5 = 1, Rest are -1

x1	x2	x3
x4	x5	x6
x7	x8	x9

0	1	0
	1	

Next move: 0

No winning move of 0

Cannot guarantee win for 0 from any next move

Practice Problem (continued)

Losing Move: A move from which there is <u>at least one winning move of the opponent</u>

Practice Problem (continued)

IIT Bombay

Example: tic-tac-toe configuration

x1 = x3 = 0, x2 = 1, Rest are -1

x1	x2	x3
x4	x5	x6
x7	x8	x9

0	1	0
		1
0		

Next move: 1

Losing move of 1: x6 = 1

Winning move of 0 exists after this

Practice Problem (continued)

Useful Observation:

If 0 has a winning move from a configuration, then after this move is taken, 1 cannot have a winning move from the new configuration.

Similarly, with roles of 0 and 1 reversed.

Practice Problem (continued)

Useful Observation:

If 0 has a losing move from a configuration, then after this move is taken, 1 has a winning move from the new configuration.

Similarly, with roles of 0 and 1 reversed.

Practice Problem: Subproblem 2

IIT Bombay

Write mutually recursive C++ functions

winMove and loseMove,

such that each takes as inputs

(i) a configuration, and (ii) next player (0 or 1)

and determines

If next player has at least one winning move

and

If all moves of next player are losing moves

Practice Problem: Subproblem 3

If there is a winning move for the next player, winMove(…) should return the position for the winning move, else it should return -1

If all moves for the next player are losing moves, loseMove(…) should return true, else it should return false.

Practice Problem (continued)

IIT Bombay

Recall example: tic-tac-toe configuration

x1 = x3 = 0, x2 = x6 = 1, Rest are -1

x1	x2	x3
x4	x5	x6
x7	x8	x9

0	1	0
		1
0		

Next move: 0

Winning move of 0: x7 = 0

winMove(0,1,0,-1,-1,1,-1,-1,-1, 0) returns 7

Practice Problem (continued)

IIT Bombay

Example: tic-tac-toe configuration

x1 = x3 = 0, x2 = x5 = 1, Rest are -1

x1	x2	x3
x4	x5	x6
x7	x8	x9

0	1	0
	1	

Next move: 0

No winning move

winMove(0,1,0,-1,1,-1,-1,-1,-1, 0) returns -1

Practice Problem (continued)

IIT Bombay

Recall example: tic-tac-toe configuration

x1 = x3 = 0, x2 = x6 = 1, Rest are -1

x1	x2	x3
x4	x5	x6
x7	x8	x9

0	1	0
		1
0		

Next move: 1

All moves of 1: Losing

loseMove(0,1,0,-1,-1,1,0,-1,-1, 1) returns true

Practice Problem: Subproblem 3

IIT Bombay

```
int winMove(int x1, … int x9, int nextPlayer)
{   // Validate inputs
    // Determine if winning move exists for
    // nextPlayer
}
```

[Hint: Check if opponent has only losing moves after nextPlayer takes a move.
What are the termination cases?]

Practice Problem: Subproblem 3

IIT Bombay

```
int loseMove(int x1, … int x9, int nextPlayer)
{   // Validate inputs
    // Determine if all moves of nextPlayer
    // are losing moves
}
```

[Hint: Check if opponent has a winning move for every next move of nextPlayer]