CS280 – Recursion, ADT
January 27, 2016

http://azrael.digipen.edu/~mmead/www/Courses/CS280/Recursion2.html

B-List Notes

- For the assignment operator: Should we use existing nodes or clear the list and re-allocate?
  - It shouldn't matter, but it could be optimized
- How do you remove from a node?
  - It is like an array, everything must be shifted (if removing from the front)
  - Just decrement the count (if removing from the end)
- When a node is empty, you must get rid of the node
- You can only use binary search on the arrays inside of the node

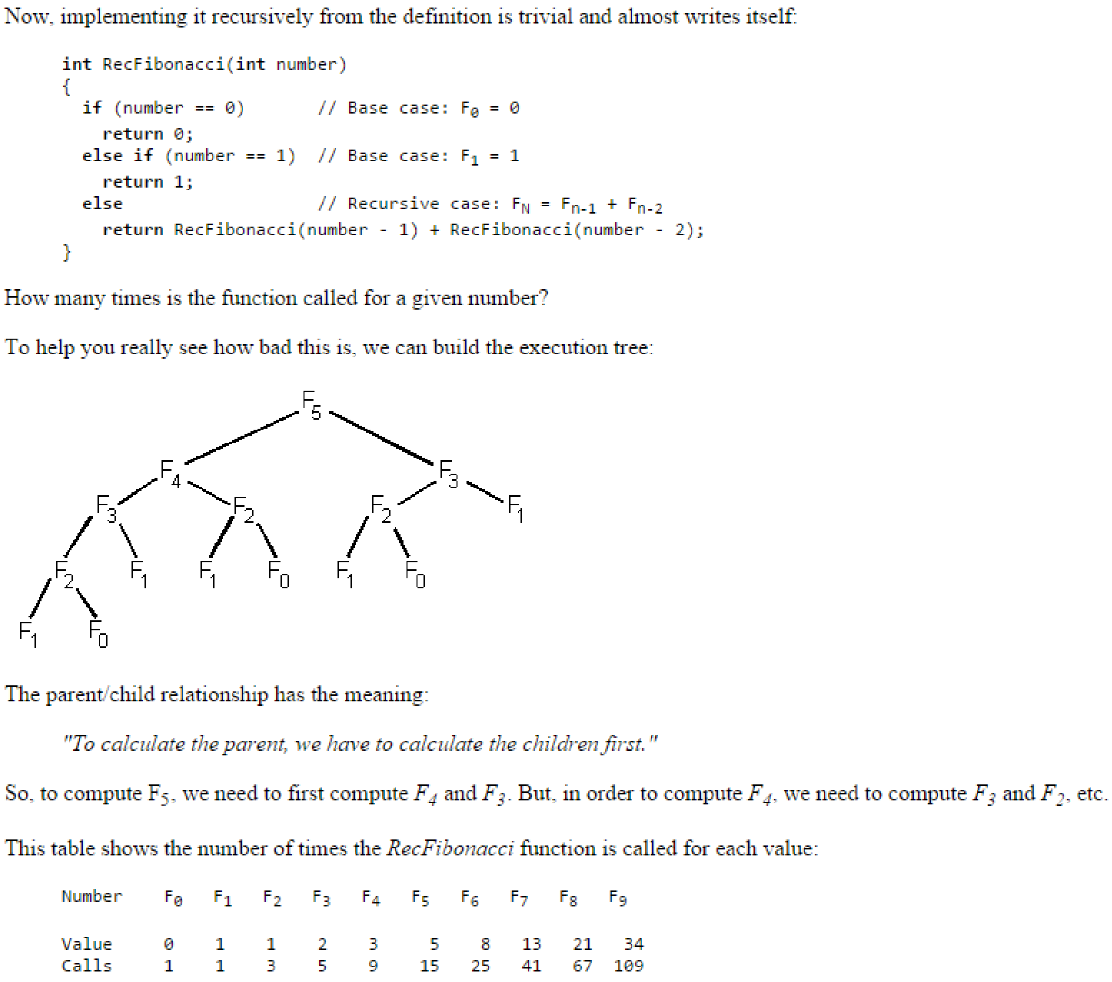
Misc. Things

- If we are following good programming techniques, we will be fine, if not we will struggle
  - HELPER FUNCTIONS!
  - Roughly 15 helper functions for Blist
- Try to keep functions less than the height of the screen
- If we are asked to write code on a test, it will probably be a recursive function
- Try messing with compilers for our games

Recursion

- Recursive function read very similarly to their mathematical definitions
- Doesn't take long to crash the stack if you mess up recursion
- Impicit stack is given to us to use in our programs, but it is more apparent in assembly because you are actually modifying it

- Reversing a string
  - "Swap the ends, reverse the middle"
- Some languages are purely based on recursion, they have no iteration
- We use iteration because even though recursion is elegant, iteration is generally more efficient
- How many times is a recursive function called?
  - Execution tree!



Now, implementing it recursively from the definition is trivial and almost writes itself:

```
int RecFibonacci(int number)
{
  if (number == 0)       // Base case: F0 = 0
    return 0;
  else if (number == 1)  // Base case: F1 = 1
    return 1;
  else                   // Recursive case: FN = Fn-1 + Fn-2
    return RecFibonacci(number - 1) + RecFibonacci(number - 2);
}
```

How many times is the function called for a given number?

To help you really see how bad this is, we can build the execution tree:

The parent/child relationship has the meaning:

> *"To calculate the parent, we have to calculate the children first."*

So, to compute F_5, we need to first compute F_4 and F_3. But, in order to compute F_4, we need to compute F_3 and F_2, etc.

This table shows the number of times the *RecFibonacci* function is called for each value:

Number	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
Value	0	1	1	2	3	5	8	13	21	34
Calls	1	1	3	5	9	15	25	41	67	109

15 calls to recursive sequence. Note: This is generally more expensive than iteration.

- Saving recursive results for future recursive calls is called dynamic programming
  - Note how in the above diagram, *F1* is called 5 times! Save this result the first calculation and use it in future recursive calls

ADT (Abstract Data Types)

http://azrael.digipen.edu/~mmead/www/Courses/CS280/AbstractDataTypes.html

- Abstract data types are always accessed through an interface
- 64 bit has brought us more CPU registers, making things potentially much faster!
    - 8 byte pointers however, so if there was no register increase it could actually slow things down
- Move constructors can detect if you aren't using the right side anymore.
    - Will simply use a pointer and 'steal' the data instead of copying. Much faster!