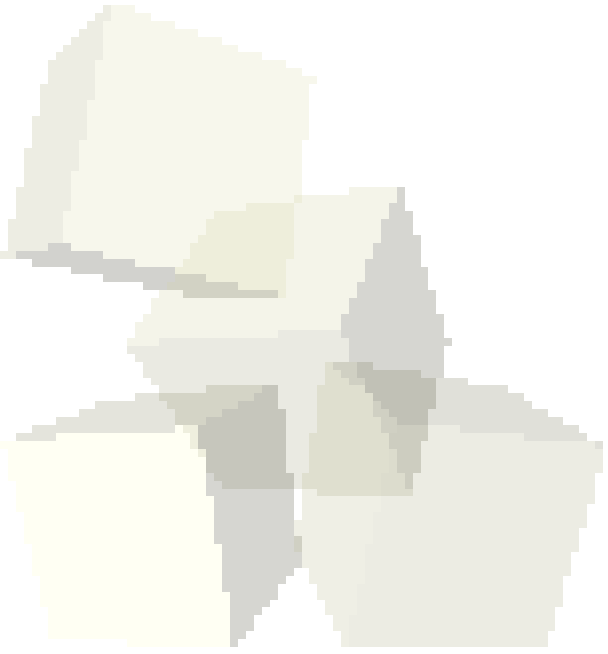
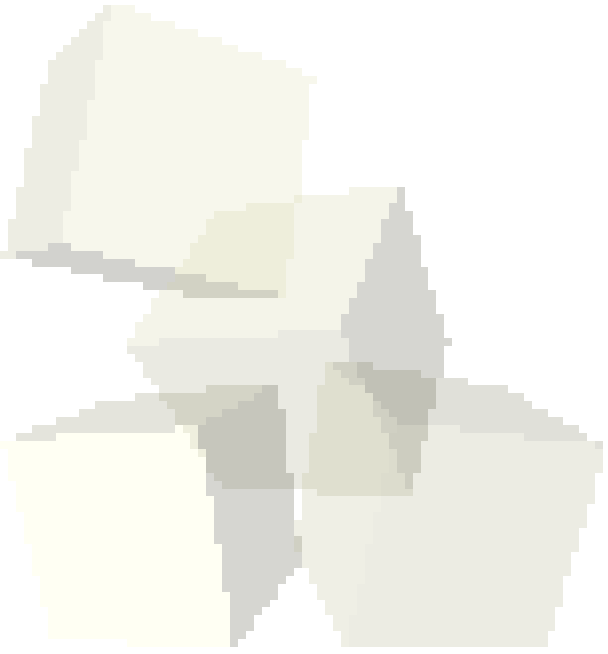
Memoization and Lang. Performance

4-14-2010



Opening Discussion

- What did we talk about last class?



Steps for DP

- To apply DP we first need to develop the recurance relationship. Figuring out the best arguments for this can be a challenge.
- Next we write a solution that fills in an array with the answers looking into the array for earlier solutions.
- If you need to you can also reconstruct the optimal solution by walking back down the array and taking the optimal path.

Memoization

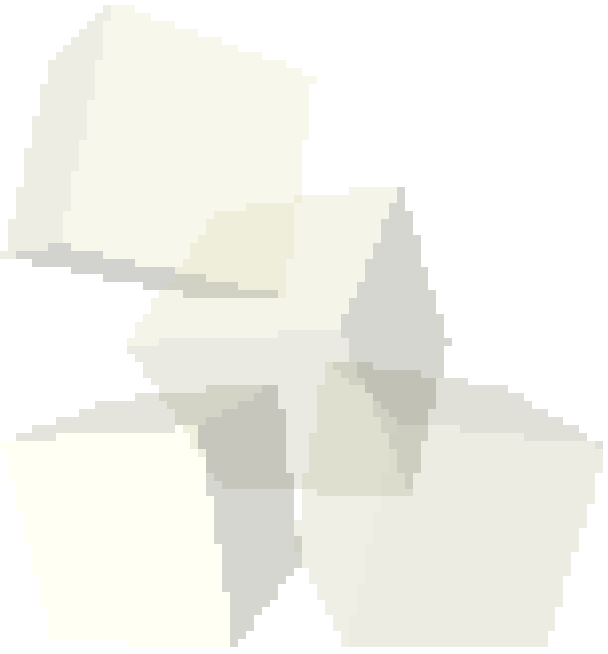
- An alternative to DP that can be almost as fast is memoization. In this approach we write the recursive solution, but pass in an extra argument that stores solutions we have already found.
- When the function runs it checks the stored values before doing a recursive call so it won't solve subproblems that it has solved previously.
- For some problems this method is a lot easier to think about. Memoization can also be used for problems that don't have optimal substructure so DP can't be applied.

Language Performance Characteristics

- Some types of scientific computing, such as large scale simulation, are computationally expensive and hence sensitive to the performance of the tools used.
- Different languages strive to achieve different goals. Performance isn't always high on the list of goals. Goals also impact the level of detail of control one has over the machine.

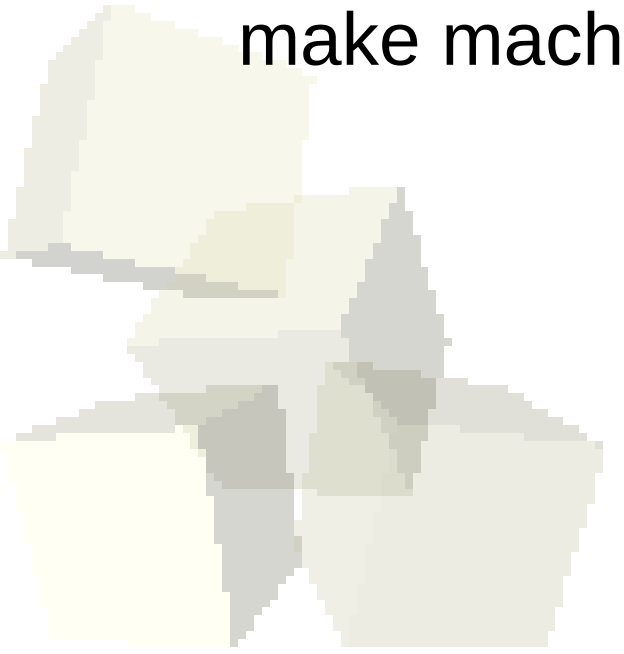
Understanding the Machine

- To know how to make a program run faster we need to know some things about the machines we are working on.
  - Vector machines/SIMD instructions
  - Memory hierarchies
  - Pipelining
  - Out of order instructions
  - MIMD shared memory parallelism



Compilers, Interpreters, and JITs

- Machines don't understand the code you write. Something most convert it to something the machine understands.
  - Compilers take your code to machine language (often with a step are assembly in between).
  - Interpreters are programs that parse text and execute it on the fly. Interpreters are slow.
  - JIT (Just In Time) Compilers take text or bytecode and make machine language at runtime.



Optimization

- Because of the complexity of modern architectures, the performance of machine code can be greatly impacted by optimizations used by the compilers.
- Poor optimization can produce slow code from any language.
- Poor programming practice can give you code that is even slower.

Closing Remarks

- I am supposed to give you a quiz next class.