Sequence Alignment

4-9-2010

Opening Discussion

- Do you have any questions about the assignment?
- Do you have any questions about the reading?
- Student presentations!
  - I would like to have you all do a round of lectures. Each person should do 1-2 classes on a topic mostly of your choosing though I have some suggestions.
    - Sci Comp Languages, past, present, future
      - X10
      - Fortress
      - Fortran
    - Grand Challenge Problems and computation
    - Internet 2?
    - Anything else you find of interest.

Comparing Sequences

- Once biologists have sequences of DNA or RNA from difference species, they want to be able to compare them.
- This comparison can't be a simple equality comparison. The desired comparison is one of homology, could one have evolved from the other or could they have shared a common ancestor.
- The details of how these comparison are done is a bit beyond the scope of this class. However, we can speculate on it a bit to see what actually matters.

Longest Common Subsequence

- A standard problem in computer science that can give us some insight into how sequence comparisons work is the longest common subsequence problem.
- You are given two strings and are asked what is the longest string that appears in proper order in both of the strings.
- Let's look at some examples and consider different ways that we might go about trying to solve this problem.

Recursive Solution

- The most intuitive solution to the longest common subsequence problem is a recursive one. First we can build up a recurance relationship in mathematical terms.
- Once we have it in this form we can convert it into code fairly easily.

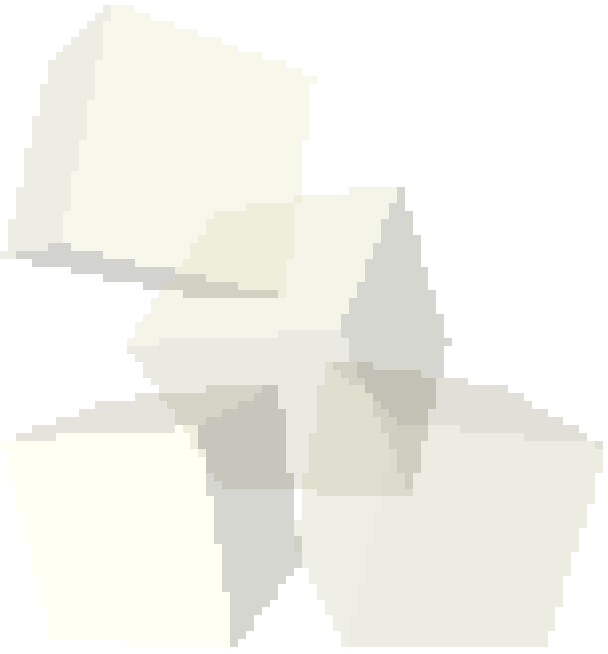
$$lcs(m,n)=\left\{\begin{array}{c} 0\,if\ m=0\vee n=0 \\ lcs(m-1,n-1)+1\,if\ s1(m)=s2(n) \\ max(lcs(m-1,n),lcs(m,n-1)\,otherwise) \end{array}\right\}$$

Speed Issues

- What is the order of the code we just created? Will we be able to apply this to DNA sequences with many thousands of base pairs?

Longest Common Subsequence

- While a recursive function works, it has the downfall of scaling exponentially with string size. Because of this, our method isn't really practical to use.
- Let's look at some alternate methods of solving this problem and other optimization problems that will make them tractable.



Dynamic Programming (DP)

- The problem with our recursive solution is that it winds up solving the same problem over and over again. This is fairly common in optimization problems.
- When things do this they exhibit optimal substructure. That implies that the optimal solution for a problem is built from optimal solutions to smaller problems. Only when this is true you can use dynamic programming.
- The idea of dynamic programming is to fill in solutions starting with very small problems and build up instead of recursing down.

Applying DP

- To build a DP solution we must first find the recursive solution, then devise a way to fill in solutions from the bottom-up.
- All recursive solutions have termination conditions. In a DP solution that is where you start. Often you fill in an array with these values. Then you run through the rest of the array filling in values. The trick is that you compute each new value by looking up solutions in the parts of the array you are already filled in.
- Using our recurance relation from last class we can do this for LCS.

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.

Closing Remarks

- Assignment #7 is due Monday.