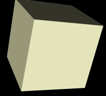
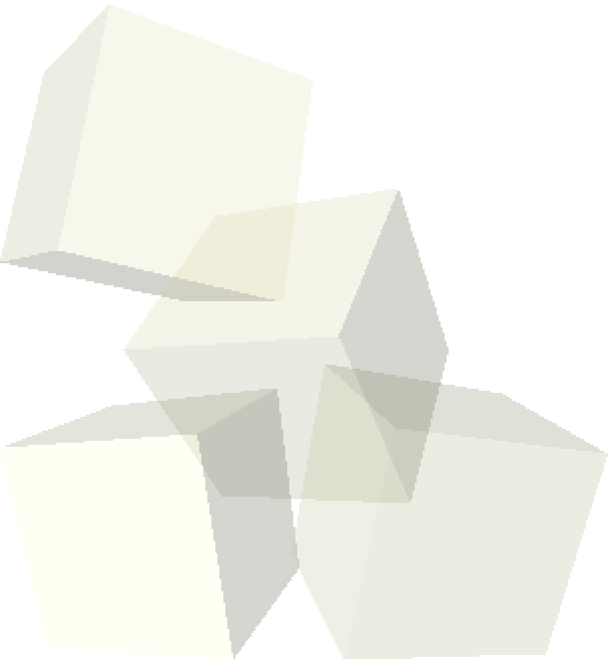
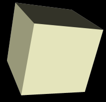


Dynamic Programming

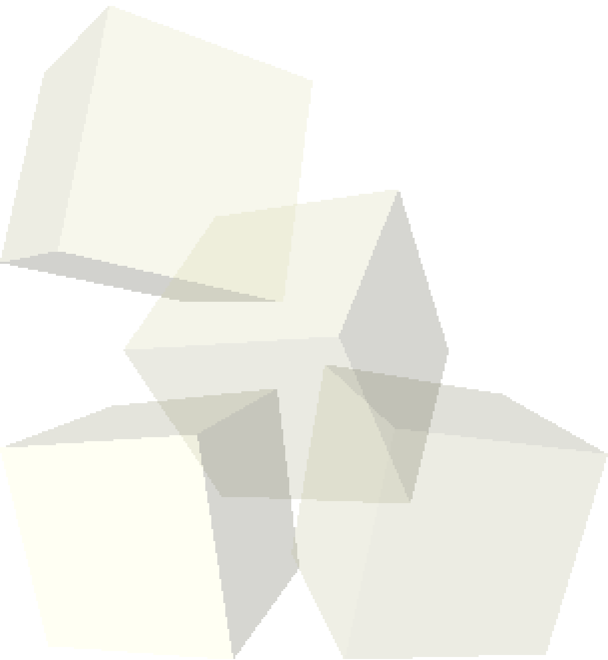
2-23-2006

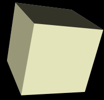




Opening Discussion

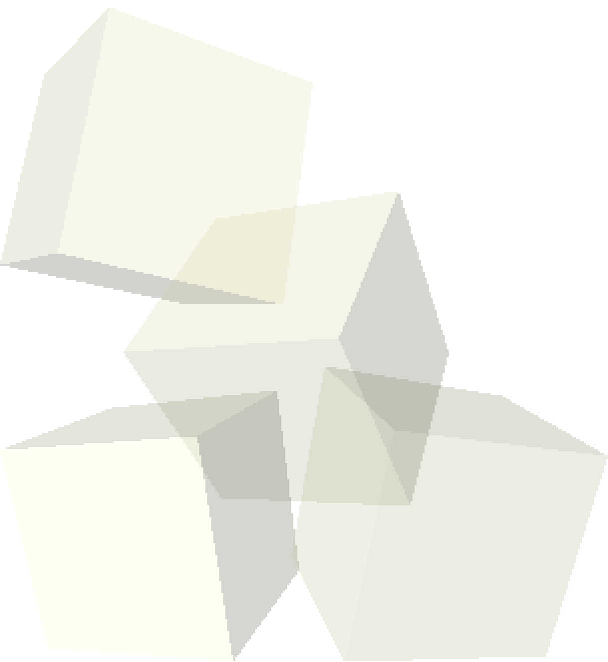
- What did we talk about last class?

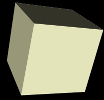




0/1 Knapsack

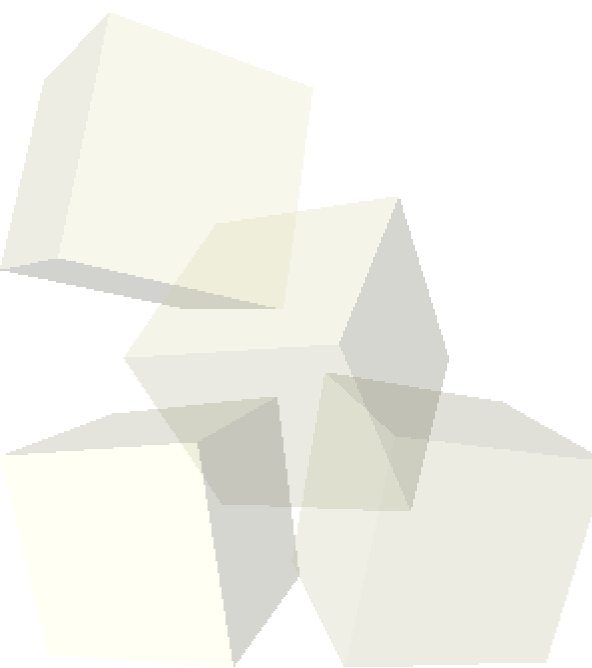
- This is another classic DP problem. Again we will begin by writing the recursive function.
- What happens to us if weight isn't an integer? How must we change our code to make it work?
- Let's write code for the integer weight version then look into changing it.





Segmented Least Squares

- Here the problem is that we want to do a linear fit to some data, but potentially in a piecewise manner. That is to say that the data might be well fit by more than one line. The question is, what is the fewest number of lines that fit the data well.
- A single least squares fit with points (x_i, y_i) is given by y=ax+b where a and b are given by.



$$a=\frac{n\sum x_i y_i-\left(\sum x_i\right)\left(\sum y_i\right)}{n\sum x_i^2-\left(\sum x_i\right)^2}$$

$$b=\frac{\sum y_i-a\sum x_i}{n}$$

Solution

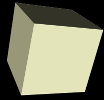
- We give any solution partition a penalty equal to the number of segments in the solution times some constant C plus the errors of the best fit on all the partitions.
- We let opt(i) denote the optimal solution for $p_1..p_i$ and $e_{i,j}$ is the error for the best fit for points $p_i..p_j$.
- If the last segment of the partition goes from i to n then $opt(n)=e_{i,n}+C+opt(i-1)$.
- More generally, $opt(j)=min(e_{i,j}+C+opt(i-1)$ for all i between 1 and j.

RNA Secondary Structure

- Now we will switch to a biology application of dynamic programming. We consider RNA molecules as a string of bases from the set {A,C,G,U}. The bases like to form pairs, but A only pairs with U and C only pairs with G.
- The long chair will bend around to make a secondary structure so that bases can pair up. There are a few rules we will follow on how things can pair up. Elements i and j can pair up if
  - They want to match A-U or C-G.
  - $i<j-4$ (no sharp bending)
  - No base is in more than one pairing.
  - If i,j and k,l are pairs then you can't have $i<k<j<l$. This is the no crossing rule.

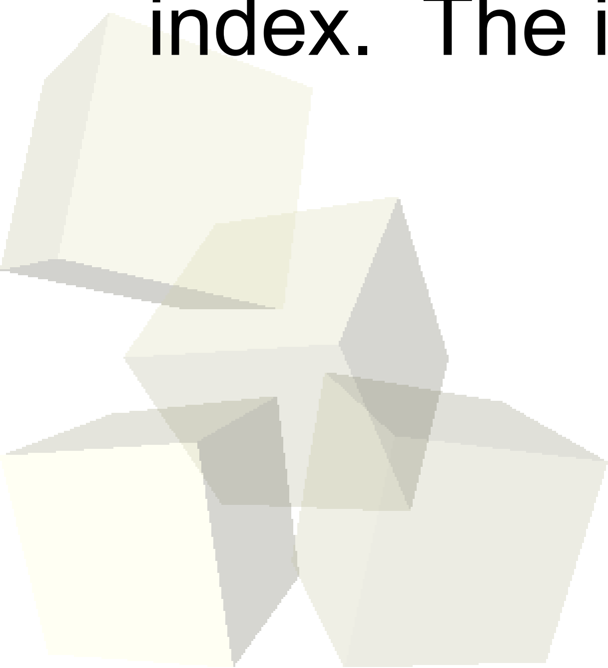
Solution

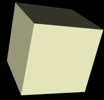
- We want to simply maximize the number of bases in pairs.
- The no crossing rule forces us do a recursion on two arguments. The reason is that once we include a pair i,j we divide the problem into two pieces, one with bases between i and j and one with pieces outside i and j.
- This gives the following recursion. opt(i,j)=max(opt(i,j-1),max(1+opt(i,t-1)+opt(t+1,j-1))) where t is taken over all bases from i to j-4 that fit the rules for matching with j. Use a value of 0 if j-i<4



Coding

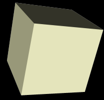
- Given that recursion there is still a trick of figuring out how to code the solution with DP. Basically, how do we go about filling in our array? It turns out that we are always making calls with smaller intervals so we can easily fill up the array by starting with the smallest intervals and working up. Make the outer loop be the interval size, not the index. The inner loop can be the first index.





Sequence Alignment

- This is the big brother of LCS and it is also has biology applications.
- Here we have two strings and we want to find an optimal matching between the strings. For LCS, optimal just meant we matched as much as possible. Here we put a cost, δ, on gaps where we don't match between strings. We also have a mismatch penalty α_{pq} for pairing a p with a q. So we could match things that aren't perfect matches, but there is a penalty for it.
- We solve this the same way as LCS, but minimize the penalty.



Reminders

- I don't have a description up for the next assignment, but you will be writing a data structure for doing disjoint sets with the disjoint sets forest approach. If you don't feel like taking the weekend off or spending it on other tasks you could begin looking at that.