Anna Karlin, 426C Sieg

Homework # 2
Due by 4pm, FRIDAY, October 13

Reading: Skiena, chapters 1–4. You also may want to look at chapters 16, and 23– 26 in Cormen, Leiserson, Rivest.

Problems:

1. Dynamic Programming:

   - Consider the following data compression technique. We have a table of m text strings, each of length at most k. These are our code words. We want to encode a data string D of length n using as few code words as possible. For example, if our table contains the four code words $(a, ba, abab, b)$ and the data string is $bababbaababa$, then the best way to encode it is $(b, abab, ba, abab, a)$ – a total of five code words. Give the most efficient algorithm you can that, given an input data string D, produces an optimal encoding for D. You may assume that D has an encoding in terms of the table. What is the complexity of your algorithm in terms of n, k and m?
   - (Surface reconstruction problem that we discussed in class.) Let P and Q be polygons with m and n vertices $\{p_0, p_1, \ldots p_m\}$ and $\{q_0, q_1, \ldots q_n\}$ in 3 dimensions, where P and Q are planar and parallel. A *tiling* of P and Q is a set of triangles, called *tiles*, whose vertices are of the form $\{p_i, p_{i+1}, q_j\}$ or $\{q_j, q_{j+1}, p_i\}$, that is, one side of each tile is an edge of one polygon and the remaining tile vertex is on the other polygon (throughout $i + 1$ is taken modulo m and $j+1$ is taken modulo n). The tile edges that have a vertex from each polygon are called cross edges. The set of tiles constructed must satisfy two properties: (1) Every edge of each polygon is a side of exactly one tile. (2) Every cross edge is a side of exactly two tiles.

     Give a dynamic programming algorithm, which given polygons P and Q, constructs a tiling of P and Q of minimum surface area (the sum of the areas of all the triangles in the tiling). What is the complexity of your algorithm in terms of m and n?

2. Graph representation: Consider an unweighted graph with n vertices and m edges. The two common data structures used to represent such a graph are adjacency matrices and adjacency lists. An adjacency matrix is an n x n matrix M, where $M[i, j] = 1$ if there is an edge from vertex i to vertex j and $M[i, j] = 0$ otherwise. An adjacency list consists of an n-element array of pointers, where the i-th element points to a linked list of the edges incident on vertex i.

   - For which of adjacency lists and adjacency matrices is it faster to test if a given edge (v, w) is in the graph?
   - For which of adjacency lists and adjacency matrices is it faster to find the degree of a vertex?
   - How much memory is used by each of the two data structures (in big Oh notation)?

- What is the complexity of edge deletion/edge insertion for the two methods?
- What is the complexity of traversing the graph for the two methods?

3. Depth-first search on a *directed* graph: Write pseudo-code for depth first sirch on a directed graph, so that it prints out, for each edge examined, whether the edge is a tree edge, a back edge, a forward edge or a cross edge.

4. We discussed (sometimes briefly) the following in the first lecture.

   - Traversing a graph using depth-first search
   - Traversing a graph using breadth-first search.
   - Finding the connected components of an undirected graph.
   - Finding the strongly connected components of a directed graph.
   - Determining if a directed or undirected graph has a cycle.
   - Topologically sorting a directed acylic graph.
   - Testing a graph for planarity.
   - Finding a minimum spanning tree for a weighted graph.

   For each of the following problems, describe how it could be modeled as a graph problem and solved with the aid of one or more of the above known algorithms, or ideas involved therein. I have purposely left these problems somewhat vague....

   (a) Given a set of processes that need to be scheduled, and dependencies that tell you which processes need to be completed in order for a given process to execute, how would you schedule the processes so as not to violate any of the dependencies?

   (b) How would you find your way out of a maze using a very large collection of pennies?

   (c) DNA sequences – you are given experimental data consisting of small DNA fragments. For each fragment f, it has been determined experimentally that certain fragments lie to the left of f on the chromosome, certain fragments lie to the right of f, and the rest can be on either side. How would you go about finding a consistent ordering of the fragments from left to right that satisfies all the constraints?

   (d) Broadcast – suppose a given processor on a network wishes to broadcast a message to all other processors. How would you go about routing the message to all the other processors as efficiently as possible? You can assume that you have information about the bandwidth and congestion on the various links in the network.

   (e) Circuit simulation – Given a circuit layout and input/output information for each gate in the circuit (where its outputs go and where its inputs come from) how would you schedule the simulation of the gates in the circuit?

   (f) Graphics– A triangle strip is a sequence of triangles, $t_1, \dots, t_n$ such that t_i shares exactly one edge with t_{i+1} and exactly one (other edge) with t_{i-1}, for each $1 < i < n$. Rendering engines are often able to very efficiently render a "triangle strip", much more efficiently than they can render each of the n individual triangles. Given a large set of triangles (given by the 3D coordinates of the three vertices of each triangle), how would you go about constructing a set of maximal triangle strips to feed to the renderer.