Math 181 - Section X1
Midterm I
Review
Spring 2009

Chapter 1: Urban Services - Euler Circuits

Terms: Graph, vertex, edge, path, circuit, Euler circuit, valence, connected graph, Eulerization, directed graph (digraph)

Results:

- A graph has an Euler circuit if and only if the graph is connected and every vertex has even valence.
- Any connected graph can be "Eulerized" by doubling existing edges. In terms of planning routes for postal workers, etc., this corresponds to doubling back along part of the route.

Chapter 2: Business Efficiency - Hamiltonian Circuits & Spanning Trees

Terms: Hamiltonian circuit, spanning tree, order-requirement digraph, critical path

The Traveling Salesman Problem (Algorithms for finding "good" Hamiltonian circuits)

- "Brute force" method
  - just list *all* Hamiltonian circuits and pick the cheapest one
  - this will always find the cheapest Hamiltonian circuit, but it is not efficient
- Nearest Neighbor Algorithm
  - starting at a particular vertex, first pick the edge to the "nearest" nieghbor. then do the same, starting at this neighbor. keep going until all vertices have been reached (don't return to a vertex you've previously visited until the very end)
  - this method is *not* guaranteed to find the cheapest Hamiltonian circuit
  - the resulting circuit depends on the choice of initial vertex

- Sorted-Edges Algorithm
  - start by listing all edges, in order of increasing cost. choose the cheapest edges until all vertices have been reached, making sure to never (1) close up any part of the circuit before all vertices have been reached or (2) pick a third edge connecting to any vertex
  - this method is *not* guaranteed to find the cheapest Hamiltonian circuit

Kruskal's Algorithm (for finding a minimal-cost spanning tree)
- This is just like the sorted-edges algorithm above, except that you *never* close up the circuits

Chapter 3: Planning and Scheduling - Task Scheduling & Bin Packing Algorithms

Terms: Processor, priority list, chromatic number of a graph

Task-scheduling Algorithms (fixed number of processors)

- List-Processing Algorithm
  - given a priority list, assign to the next idle processor the next "ready" task on the priority list
  - this does not always find an optimal schedule

- Critical-Path Scheduling Algorithm
  - use critical paths to devise a new priority list, and then apply the list-processing algorithm with this new priority list
  - this does not always find an optimal schedule

- Decreasing-Time-List Algorithm
  - this is just the Critical-Path algorithm when the tasks are all independent (so the order-requirement digraph has no edges)

Bin-Packing Algorithms

- can think of this as scheduling independent tasks on an unspecified number of processors, but within a given time limit

- Next Fit
  - stick an item into the current open bin; if it doesn't fit, close the bin and move on to the next bin
  - don't need to keep track of any extra information

- First Fit
  - stick an item into the first open bin into which it fits
  - only close a bin when it's full - need to keep track of which bins are open and how much room is left in each

- Best Fit
  - put item into the bin into which it fits "best", that is, with the least room left over
  - only close a bin when it's full

- Worst Fit
  - put item into the bin into which it fits "worst", that is, with the most room left over
  - only close a bin when it's full

- Decreasing versions of all of the above
  - first resort items into decreasing order in terms of size and then use the above algorithms
  - these tend to perform better than then above versions, but you need to know the complete list of items in order to begin

Conflict Avoidance

- Can think of this in terms of graphs with labeled, or colored, vertices

- **4 Color Theorem**: If a graph can be drawn without any of the edges intersecting each other, then it can be colored using four colors or less

- The 4 Color Theorem says that, in particular, any map can be colored using at most four colors