Lecture C6: Graphs

Response to 'Muddiest Part of the Lecture Cards'

(8 respondents)

1) *Could the graphs be analogous to Markov chains and then would the weights be analogous to the transitions probabilities*?
The Markov chain can be represented as an infinite graph, with probabilities on the transitions instead of weights.

2) *Could the shortest path problem also be approached with Markov chains? Maybe for systems with more uncertain weights? Or for systems where future weights are unknowns but estimates exist*?
Yes you can.

3) *Can you find the shortest path (or least costly) using the adjacent matrix*?
Yes, there exists many different kind of algorithms to solve the shortest path problem in graphs. It is common that the algorithms take an NxN adjacency matrix as input and calculates an NxN matrix S, where $S_{i,j}$ is the length of the shortest path from v_i to v_j, or a special value, e.g., '∞' if there is not path.

4) *What is an example of a graph with negative weights*?
An interesting question, there are some paths that are undesirable but still have to be considered (say for instance in the case of a traffic jam), in that case, the weight assigned to the edge is negative (the goal in this case would be maximize the cost of traversing the graph).

5) *How do you represent '∞' in Ada*?
The Ada language does not define a representation for infinity. That is left upto the implementation. You can define your own value for infinity. In the case of Dijkstra's shortest path algorithm that we saw in class today, you can use a –ve value to represent infinity (because the algorithm only works for positive weights and tries to minimize the total weight).

6) *No idea what the problem was with the goose and grain*.
An astronaut wants to move herself, a goose, a grain and a fox across a river, from the north side to the south side. Unfortunately her Rover is so small that she can at most bring one of the fox, grain or goose across on any trip. Even worse, the fox will eat the goose if they are left alone on one side of the river, and just as bad, the goose will eat the grain if they are left alone on one side of the river, so the Astronaut must not leave the fox alone with the goose nor the goose alone with the grain. How should the Astronaut solve the problem?

A solution to the problem is shown in the lecture slides.

7) *How do you implement these graphs in Ada? Linked lists*?
A very popular representation of a graph is the adjacency matrix. The graph is then represented by a matrix where the rows and columns are indexed by the vertices. A drawback with the adjacency matrix is that it requires NxN space even when there are very few edges. A matrix where most of the positions are *zero* is called a *sparse matrix*. Sparse matrices can very effectively be implemented using multi-linked lists.

8) *From slide 23 → I am lost, can we go over this in recitation*?
Dijkstra's algorithm solves the problem of finding the shortest path from one node in the graph (the source) to a destination node in the graph. The algorithm actually finds the shortest path from the source node to all the other nodes in the graph at the same time.

Dijkstra's algorithm keeps two sets of vertices

S	the set of vertices whose shortest paths from the source have already been determined
Q	The remaining vertices

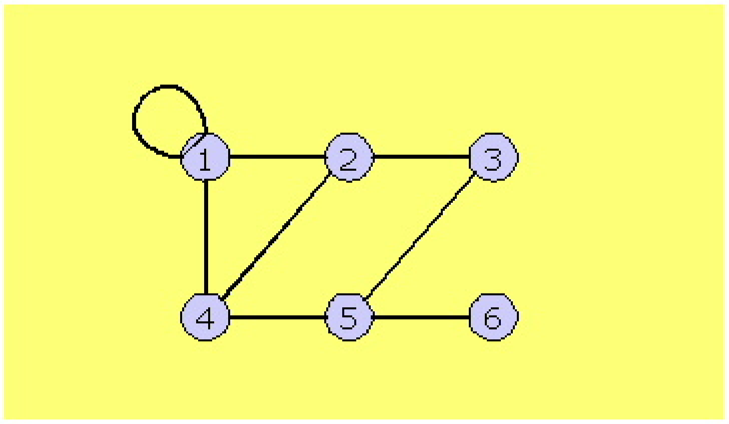
The other data structures needed are

d	array or best estimates of shortest path to each vertex
prev	array of predecessors for each vertex (a predecessor list is a structure for storing a path through a graph)

The algorithm is shown in the lecture slides. The relaxation process is used to update the costs of all the vertices, v, connected to a vertex, u, if we could improve the best estimate of the shortest path to v by including (u, v) in the path to v.

9) *What does degree mean*?
A degree *deg(v)* of a vertex $v \in V$ is the total number of edges incident with it. Note that each loop is counted twice.

10) *What does a loop look like*?



A **loop** in a graph is an edge e in E whose endpoints are the same vertex. In figure, the edge that starts and ends at vertex 1.

11) *What are the readings*?
A separate handout that is available via the CP web page.

12) *No mud* (1 student)