MATH 38 - Graph Theory

Carter Kruse

Homework 5

Section 2.3 - Question 3

There are five cities in a network. The cost of building a road directly between i and j is the entry $a_{i,j}$ in the matrix below. An infinite entry indicates that there is a mountain in the way and the road cannot be built. Determine the least cost of making all the cities reachable from each other.

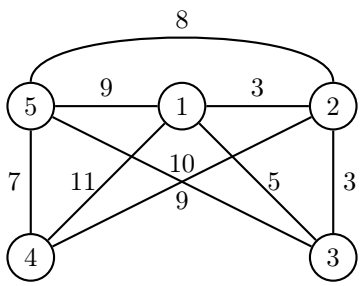
$$\begin{pmatrix} 0 & 3 & 5 & 11 & 9 \\ 3 & 0 & 3 & 9 & 8 \\ 5 & 3 & 0 & \infty & 10 \\ 11 & 9 & \infty & 0 & 7 \\ 9 & 8 & 10 & 7 & 0 \end{pmatrix}$$

A **weighted graph** is a graph with numerical labels on the edges. When building links to connect locations, the costs of potential links yield a weighted graph. The minimum cost of connecting the system is the minimum total weight of its spanning trees.

In a connected weighted graph of possible communication links, all spanning trees have $n-1$ edges; we seek one that minimizes or maximizes the sum of the edge weights For these problems, the most naive heurisitic quickly produces an optimal solution.

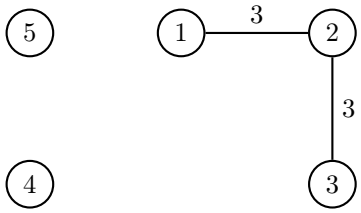
Kruskal's Algorithm (Minimum Spanning Trees)
Input: A weighted connected graph.
Idea: Maintain an acyclic spanning subgraph H, enlarging it by edges with low weight to form a spanning tree. Consider edges in non-decreasing order of wight, breaking ties arbitrarily.
Initialization: Set $E(H) = \varnothing$
Iteration: If the next cheapest edge joins two components of H, then include it; otherwise, discard it. Terminate when H is connected.

The matrix corresponds to the following weighted graph:

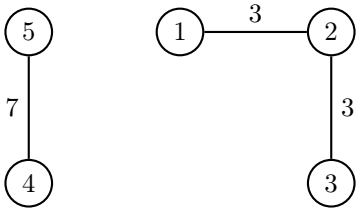


Using Kruskal's algorithm, we select the least expensive/costly edge that does not create a cycle for each iteration.

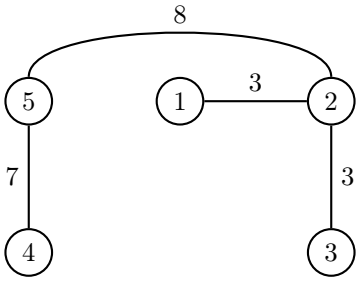
We start using the two edges of weight 3.



Now, the edge of weight 5 cannot be used, as it would create a cycle, so we use the edge of weight 7.



The edge of weight 8 completes the minimum spanning tree, as follows.



Thus, the total weight of the minimum spanning tree is 21. So, we may conclude that given five cities in a network, with the cost of building a road directly between i and j as the entry $a_{i,j}$ in the corresponding matrix, the least cost of making all the cities reachable from each other is 21.

Section 2.3 - Question 10

Prim's algorithm grows a spanning tree from a given vertex of a connected weighted graph G, iteratively adding the cheapest edge from a vertex already reached to a vertex not yet reached, finishing when all the vertices of G have been reached. (Ties are broken arbitrarily.) Prove that Prim's algorithm produces a minimum-weight spanning tree of G. (Jarnik [1930], Prim [1957], Dijkstra [1959])

To prove that Prim's algorithm produces a minimum-weight spanning tree of G, we use a similar argument to that for Kruskal's algorithm (in a connected weighted graph G).

We show first that the algorithm produces such a tree. It never chooses an edge that completes a cycle, as edges are added from a vertex already reach to a vertex not yet reached. If the final graph has more than one component, then we considered no edge joining two of them, because such an edge would be accepted. Since G is connected, some such edge exists and we considered it. Thus the final graph is connected and acyclic, which makes it a tree.

Let T be the resulting tree, and let T^* be a spanning tree of minimum weight (an "optimal" tree). If $T = T^*$, we are done. If $T \neq T^*$, let e be the first edge chosen for T that is not in T^*. Let U be the set of vertices in the sub-tree of T prior to the addition of e. Adding e to T^* creates a cycle C. Since e links U to $\overline{U}$, C has an edge e' from U to $\overline{U}$.

Consider the spanning tree $T^* + e - e'$, which indicates (by the optimality of T^*) that $w(e') \leq w(e)$. Since e' is incident to U, e' is available (for consideration) when the algorithm chooses e, and hence $w(e) \leq w(e')$.

This implies $w(e) = w(e')$, so $T^* + e - e'$ is a spanning tree with the same weight as T^*. This contradicts the choice of T^*, as $T^* + e - e'$ is an "optimal" spanning tree. $\square$

Section 2.3 - Question 16

Four people must cross a canyon at night on a fragile bridge. At most two people can be on the bridge at once. Crossing requires carrying a flashlight, and there is only one flashlight (which can cross only by being carried). Alone, the four people cross in $10, 5, 2, 1$ minutes, respectively. When two cross together, they move at the speed of the slower person. In 18 minutes, a flash flood coming down the canyon will wash away the bridge. Can the four people get across in time? Prove your answer without using graph theory and describe how the answer can be found using graph theory.

Let us consider the four people labeled according to the number of minutes they take to cross the canyon at night alone (on the fragile bridge), so $10, 5, 2, 1$. The following is a solution, which indicates that the four people can get across in time, if in 18 minutes, a flash flood coming down the canyon will wash away the bridge.

The following solution considers that at most two people can be on the bridge at once, that crossing requires carrying a flashlight, and there is only flashlight (which can cross only by being carried), and that when two cross together, they move at the speed of the slower person.

Step 1: Send 1 and 2 across. (2 Minutes)
Step 2: Send 1 back with the flashlight. (1 Minute)
Step 3: Send 5 and 10 across. (10 Minutes)
Step 4: Send 2 back with the flashlight. (2 Minutes)
Step 5: Send 1 and 2 across. (2 Minutes)

The total time is 17 minutes, which is less than the cutoff time of 18 minutes. The key is to maximize the use/crossing of 1 and 2, while minimizing that of 5 and 10.

To determine the solution using graph theory, create a vertex for every possible state, which consists of a partition of people (along with the location the flashlight). This completely describes the scenario. There exists an edge from one state to another if and only if the initial state may be obtained by moving one or two people (along with the flashlight) from one side to another, as represented by the partition.

In this sense, the problem is reduced to Dijkstra's algorithm, which may be used to find a minimum path between the various vertices and edges that exist, representing the graph states.

Section 3.1 - Question 8

Prove Or Disprove: Every tree has at most one perfect matching.

To prove that every tree has at most one perfect matching, we use an inductive proof, as follows.

Base Case: While a tree with a single vertex does not have a perfect matching, a tree with two vertices has exactly one perfect matching.

Inductive Step: Consider an arbitrary tree T with $n \geq 2$ vertices. Let v represent a leaf of the tree. In a perfect matching, v must be matched to its sole neighbor u. Thus, a matching will consist of the matchings of $T - \{u, v\}$ and u, v. Since each perfect matching in T contains the edge uv, the number of perfect matching in T is equivalent to the number of perfect matchings in $T - \{u, v\}$.

By the (implied) induction hypothesis, as each component of $T - \{u, v\}$ is a tree, each component has at most one perfect matching. $\square$

Section 3.1 - Question 24

A permutation matrix P is a $0, 1$-matrix having exactly a single 1 in each row and column. Prove that a square matrix of non-negative integers can be expressed as the sum of k permutation matrices if and only if all row and column sums equal k.

If S is the sum of k permutation matrices, then each matrix adds 1 to the sum in each row and column, thus each row or column of S has sum k.

Conversely, let S be a square matrix with rows and columns that sum to k. By induction on k, we may express S as the sum of k permutation matrices.

Base Case: For $k = 1$, S is simply a permutation matrix. This is trivial.

Induction Step: For $k > 1$, let us create a bipartite graph G with vertices $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, such that the number of edges joining x_i and y_j is $a_{i,j}$, the adjacency matrix entries for S. As the graph G is bipartite and regular, it has a perfect matching.

Let $b_{i,j} = 1$ if $x_i y_i$ belongs to this matching and $b_{i,j} = 0$ otherwise, the adjacency matrix entries for R are $b_{i,j}$. Thus, the matrix R is a permutation matrix, as each row and column of R has exactly a single 1.

Thus, $Q = R - S$ is a non-negative integer matrix whose rows and columns sum to $k - 1$. By the induction hypothesis, this indicates that there are k permutation matrices that sum to the square matrix S. $\square$