Prim’s Algorithm to Find a Minimum Spanning Tree

Inner Workings of Prim’s Algorithm

Prim’s algorithm searches through a weighted tree to find a sub-tree with the lowest possible weight, the minimum spanning tree. Depending on the starting tree, there can be multiple possible sub-trees with the minimum weight. The algorithm is designed to be greedy. It works efficiently by only considering what seems optimal in the moment rather than any future or past choices it has made. Like other greedy algorithms, Prim takes three key steps to make each of its decisions: the selection procedure, the feasibility check, and the solution check.

Prim’s algorithm’s selection procedure decides which vertex in the graph is the closest to any vertex in the current set being considered, Y. The feasibility check makes sure that adding a new edge to the minimum spanning tree doesn’t create a cycle. The selection procedure and feasibility check are done at the same time by checking only vertices in V – Y, the set of vertices on the original graph, V, not including those in Y. The vertex in V – Y that is closest to Y is selected. When an edge is added to the minimum spanning tree, the new vertex that connects it is added to Y. The solution check at the end of the algorithm is whether V and Y are the same. Once the algorithm makes it through all three steps, it moves on to its next decision until the minimum spanning tree is complete.

Prim’s algorithm makes decisions using two important arrays: nearest and distance. In nearest, each element holds two vertices, one vertex as the element’s value and one as the element’s index. Nearest is updated whenever a valid edge is found with a smaller weight. Once the algorithm is finished, the indices and elements of nearest represent the edges of the minimum spanning tree. The distance array holds key information for Prim’s algorithm to make decisions. Distance holds information on the most recent lowest edge weight connecting each pair of vertices checked. The weights are compared to determine whether an edge has the lowest possible weight for each iteration of Prim’s algorithm. Nearest also prevents cycles from forming on the minimum spanning tree. When a vertex is added to nearest, the corresponding element in distance is assigned a value of -1, which the algorithm is set to skip over to avoid redundant vertices.

Complexity Discussion

Prim’s algorithm performs two comparison operations which should be considered when finding the algorithm’s complexity. This implementation of Prim’s algorithm runs using two if statements within double-nested for loops. The loops cause an exponential growth of if statements based on the number of vertices on the original graph. Because of the double-nested loops, Prim’s algorithm has an expected complexity of O(n2).

Examples of Minimum Spanning Trees

Minimum spanning trees must see a lot of use in urban development. A civil engineer designing roadways would be interested in minimum spanning trees weighted by the distance between intersections or major highways. The edges could also be weighted by the expected number of cars on the road during rush hour for engineers concerned with the flow of traffic.

Like roads, the biological categorization systems in taxonomy also deal with branching, weighted paths. Trees in taxonomy have edges weighted by time. Categories of life with a common ancestor are biologically more related when they existed closer together in time to each other. A minimum spanning tree could show which species are most related to each other.