2.3 Lecture Summary

2.3 Spanning Tree Example

Lecture Summary: In this lecture, we learned how to use object-based isolation to create a parallel algorithm to compute *spanning trees* for an undirected graph. Recall that a spanning tree specifies a subset of edges in the graph that form a tree (no cycles), and connect all vertices in the graph. A standard recursive method for creating a spanning tree is to perform a depth-first traversal of the graph (the *Compute(v)* function in our example), making the current vertex a parent of all its neighbors that don't already have a parent assigned in the tree (the *MakeParent(v, c)* function in the example).

The approach described in this lecture to parallelize the spanning tree computation executes recursive *Compute(c)* method calls in parallel for all neighbors, *c*, of the current vertex, *v*. Object-based isolation helps avoid a data race in the *MakeParent(v,c)* method, when two parallel threads might attempt to call *MakeParent(v1, c)* and *MakeParent(v2, c)* on the same vertex *c* at the same time. In this example, the role of object-based isolation is to ensure that all calls to *MakeParent(v,c)* with the same *c* value must execute the object-based isolated statement in mutual exclusion, whereas calls with different values of *c* can proceed in parallel.

Optional Reading:

1. Wikipedia article on Spanning Trees

Mark as completed