Code Jam 2017 - World Finals

Analysis: Spanning Planning

Spanning Planning: Analysis

In general, for any value of X greater than 2, it is possible to construct a graph that has X spanning trees: connect X vertices to form a single cycle. Then there are X ways to delete a single edge, and each of these deletions leaves behind a different spanning tree. However, in this problem, we are only allowed up to 22 vertices, so we can only use that strategy for values of **K** up to 22.

The statement guarantees that an answer exists for every **K** in the range [3, 10000], so we just need to find those answers. Since there are so few possible testcases, we can prepare ourselves by finding one answer for each possible **K** before ever downloading a dataset.

A good way to start exploring the problem is to create random graphs and count their spanning trees. We can use the beautiful Kirchhoff matrix tree theorem to reduce the problem of counting spanning trees in a graph to the problem of finding the determinant of a matrix based on the graph. We can either carefully implement our own function for finding determinants, or use a library function, e.g., from SciPy. We must take care to avoid overflow; a complete 22-node graph has 22^{20} trees! Precision loss is also a potential concern if working outside of rational numbers, but one of our internal solutions successfully used Gaussian elimination and doubles.

It turns out that we cannot find all the answers we need via a purely random search, but we can find most of them, which provides some hope that we can reach the others with some tweaking. For example, we can change the graph size and the probability that each edge exists; empirically, 13 nodes and an existence probability of 1/4 work well.

Another strategy is to apply small modifications to a graph, hoping to converge on one of the desired numbers of trees. Specifically, we can remember what numbers of trees we still need to get, pick one of them as a "target", and then repeatedly add edges if we have fewer trees than the target or remove them (taking care not to disconnect the graph) if we have more trees than that target. Once we reach our goal, we pick another still-unreached number as the new goal, and so on. To make this finish quickly, we can mark all visited numbers as reached even if we reach them while in pursuit of a different goal.

In both of the above approaches, generating solutions for **K** = 13 and **K** = 22 can take a very long time, since those numbers apparently require very specific graph structures. However, since we're allowed 22 vertices, we can get the answers from those numbers by building a cycle with 13 or 22 vertices, as described earlier.

This problem is inherently experimental, and requires research on a computer; you cannot just write down a solution on paper. (If you do know of a tractable construction-based solution, we'd love to hear about it!) However, this sort of exploratory research skill is valuable in real-world programming, and we like to reward it in Code Jam, particularly in problems in which it is surprising that a randomized strategy works at all.