COP 4516 Spring 2018 Week 7 Final Individual Contest Solution Sketches

Daisy Chaining Bonanza

The key realization in this problem is that for each group of daisy chained power strips, there is only one power strip that fully gets utilized. The rest are "losing" the use of one plug for a power strip. Thus, in "most" cases, if there are p plugs on each strip, there are p - 1 of them that actually get used. This gives us a baseline answer of $(p - 1)n$, where n is the number of power strips. To this value, we just need to add the number of groups of power strips, each group provides one extra plug, as per our previous observation. The number of groups is simply $\left\lceil \frac{n}{g} \right\rceil$. We use the ceiling function since the last group may be incomplete. The values are large enough that longs must be used.

Dot Game Dominator

This problem has a greedy solution. If we want to eat the fewest dots possible, we simply want to eat the largest dot that is strictly smaller than us. To find this efficiently, a TreeSet will do, since TreeSet has a method called lower, which does exactly this. The intended solution is simply to put each dot in a TreeSet, and iteratively eat dots in the fashion described above (so you have to remove a dot from the tree set once you eat it) until either there are no dots left to eat, or, you are the largest dot. Several students utilized a sorted array and a double loop structure to solve the problem. For the data, this ran in time, but it might be possible to make data for which this technique exceeds the allowed run time. (Though, I believe there might be a complicated proof that shows that you never eat more than a logarithmic number of dots in terms of the sizes of the dots, so perhaps the worst case for this alternate algorithm is $O(n \lg m)$, where n is the number of dots and m is the largest value of any dot.) Either way, the TreeSet implementation is probably easier. One detail that needs to be hashed out about the TreeSet implementation is that multiple dots can have the same value, so one needs to create dot objects with IDs so that two dots with the same size can be stored as separate entities in the TreeSet.

Busy Pirates

The maximum value for n in this problem is 6. We can visit the treasure locations in at most 6! ways. We follow this by visiting the burying locations in at most 6! ways. This means that the total number of possible paths we must consider is just $6!^2 = 720^2 = 518{,}400$. Just noting that 720^2 is less than 1000^2 we can see that there are less than a million possible permutations to consider given the restriction. Thus, we can solve this problem by writing a modified permutation solution, where, during the first half of our permutation, we only try to go to the first half of locations and in the last half of our permutation, we only try to go the second half locations. Notice that the bounds are such that if you tried to do all 12! permutations, you would get a time limit exceeded. (12! ~ 479 million)

Mission Unpossible

The problem is asking you to find a topological sort. Since it's given that there are no cycles in the dependencies, it's known that at least one topological sort will exist. In addition, you must find the first lexicographical topological sort. It's best to use the iterative algorithm and any time you have multiple choices of vertices with an in degree of 0, just choose the one with the smallest value. The bounds are small enough that a "slow" $O(n^2)$ solution easily runs in time.