TCSS 343 - Week 7

Jake McKenzie

August 31, 2018

Graph Algorithms and some NP Completeness

"Finding a needle in a haystack is actually quite easy. It's finding a very specific piece of hay in a haystack that happens to be hard."
...
Avi Wigderson

In his first computer science class Ryan Williams was not afraid to tell his instructor that he thought it was too easy. Eventually, his frustrated teacher pulled a heavy white book off of a shelf, dumped it dramatically on Williams' desk, and told him to look up the problem described in the final chapter. "If you can solve that," he said, "then you can complain." That book was CLRS' Introduction to Algorithms and the problem was P vs. NP.

"It is not enough to be in the right place at the right time. You should also have an open mind at the right time."
...
Paul Erdős

0. With this problem I want to present a problem to you and ask you why the greedy algorithm fails.

Imagine we have a wizard that knows a few spells. Each spell has 3 attributes: Damage, cooldown time, and a cast time.

Cooldown time: the amount of time (t) it takes before being able to cast that spell again. A spell goes on "cooldown" the moment it begins casting.

Cast time: the amount of time (t) it takes to use a spell. While the wizard is casting something another spell cannot be cast and it cannot be canceled.

The question is: **How would you maximize damage given different sets of spells?**

It is easy to calculate the highest damage per cast time. But what about in situations where it is better to wait then to get "stuck" casting a low damage spell when a much higher one is available...for example consider the two sets of spells:

Chill Touch: 100 damage at a rate of 1 second per cast with a 10 second cooldown.

Mage Hand. 10 damage at a rate of 4 second per cast with a 0 second cooldown.

Optimal spell ordering $\Sigma = \{$Chill Touch, Mage Hand, Wait, Repeat$\}$

0. Given an arbitary amount of time t what is the maximum amount of spells we can cast S?

Now imagine that there is one spell, henceforth called the Eldritch Blast, which does a very, very large amount of damage, has 0 casting time, and has some positive cooldown n. If all the other spells do much less damage than the Eldritch Blast, it will clearly be optimal to cast the Eldritch Blast every n seconds and then optimize the cooldown time with the weaker spells.

1. Why does the greedy algorithm work in this case?

Now, assume all the other spells also have cooldown n. If one optimizes a given n-second downtime with these spells, then the same spell-sequence will also be possible in the next n-second downtime, and so we can assume the solution is n-periodic.

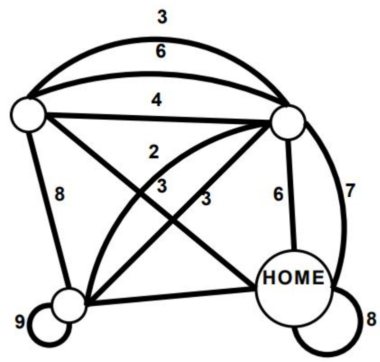
2. Why does the greedy algorithm work in this case?

3. Would you say this problem is NP-Complete? NP-Complete problems have the highest complexity of any problem in NP, which are the class of problems which can be quickly checked to be true.

For this problem and many problems, we can abstract out the details like we just did and reduce the problem to it's key constituent parts. When you employ this design technique you'll begin to notice that many problems you have reduce to problems to classes of problems. This is the most powerful design technique I know of when it comes to algorithms.

4. Of the problems you've covered in class, which does this problem degenerate too? Describe for which cases this problems degrenates to that problem you've already covered.

5. Can you produce an informal algorithm that attempts to solve this problem?



6. You're given a weighted, undirected graph G with loops(edges that loop back on themselves), multiple edges and only positive edge weights. There is a special node named "Home" and you're given a positive integer $i \geqslant 0$. Can you find a path for the jogger j that starts from home, travels the distance i and returns home without repeating an edge while notes can be repeated?

Instead of solving the problem let's explore this problem, first off: Is the Jogger problem in NP? Give a brief explanation why or why not.

7. We've talked previously in this packet about a problem that was disguised as a problem that we already knew. Reduce the jogger problem into the subset sum problem. Since we can use jogger to solve subset sum the following must be true for the jogger algorithm J and subset sum algorithm SS:

"cost of SS" = "times we use J" $\times$ "cost of J" + "cost of the reduction"

8. Prove or disprove the following lemma: If each edge weight for a graph is increased by 1, then the minimum spanning tree does not change.

9. Prove or disprove the following lemma: If each edge weight for a graph is decreased by 1, then the minimum spanning tree does not change.

A. Show that there's a unique minimum spanning tree (MST) in case the edges' weights are pairwise different ($w(e) \neq w(f)$ for $e \neq f$)

(**Hint:** If you employ Prim, Kruskal, or any of the other greedy minimum spanning tree algorithms, you can find that the weights needn't be added, only compared. What does this imply about collection of edge weights that make up each minimum spanning tree? What do we know about the weights of both graphs?)

B. Assume that G is a positive weighted graph. Under what condition does Prim's and Kruskal's algorithm on G yield the same minimum spanning trees? If they never do, explain why.

For the next few problems state whether these desiderata are true or false and justify your work.

C. In an undirected graph, the shortest path between two nodes lies on some minimum spanning tree.

D. If the edges in a graph have different weights, then the minimum spanning tree is unique.

E. If $P \neq NP$ then every problem in NP requires exponential time.

F. Let Σ be an algorithm that operates on a list of n objects, where n is a power of two. Σ spends $\Theta(n^2)$ time dividing its input list into two equal pieces and selecting one of the two pieces. It then calls itself recursively on that list of $\frac{n}{2}$ elements. Then Σ's running time on a list of n elements is $O(n)$.

10. If problem ψ is NP-hard and $\sigma \leqslant \psi$ then σ is NP-hard.

11. If problem σ is NP-hard and $\sigma \leqslant \psi$ then ψ is NP-hard.