Graph Theory and Combinatorial Optimization

Sunny Liang

2023-10-22

1 Introduction to Graphs

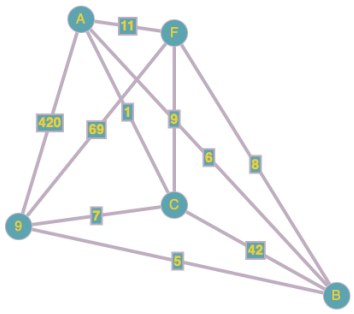


Figure 1: This is a graph.

A *graph* consists of two sets: a set of *vertices* (AKA *nodes*) and a set of *edges* which connect two vertices. After the first few problems, we will be dealing with *complete undirected weighted graphs*. *Complete* meaning that every pair of vertices has an edge between them. *Undirected* meaning that edges have no notion of direction. *weighted* meaning each edge has an associated value called a *weight*.

Imagine walking along the edges of a graph. A *cycle* is a subset of the edges of a graph such that each vertex is visited no more than once. A *Hamiltonian cycle* is a cycle such that each **vertex** is visited exactly once (except where you started). An *Eulerian cycle* visits each **edge** exactly once.

The *degree* of a vertex is the number of edges connected to it.

2 Warm-up

1. What is the relationship between the sum of the degrees of every vertex and the number of edges?
2. If there is a party where people shake hands, prove that the number of people who shake an odd number of people's hands is even. What does this have to do with unweighted graphs? (Handshake Lemma)
3. Is it possible to cross each of 7 bridges in Figure 2 exactly once? No swimming.
4. *(Hard)* Come up with some graphs that have an Eulerian cycle. Come up with some graphs that don't. What do you notice about the degree of the vertices? In general, how can you know whether a graph will have an Eulerian cycle?
5. How many unique Hamiltonian cycles can be found on a complete undirected graph with n vertices?

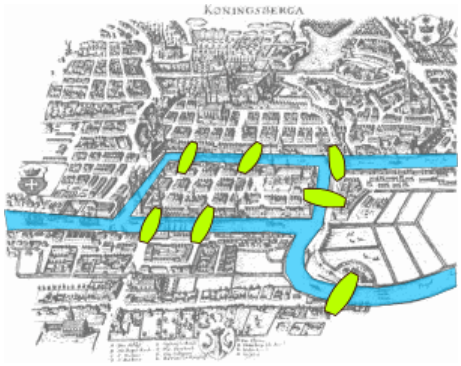
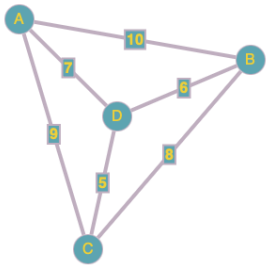
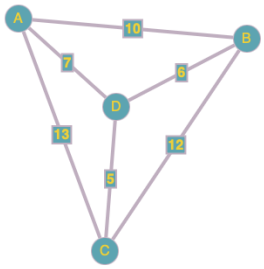


Figure 2: The seven bridges of Konigsberg, famously solved by Leonhard Euler.

3 Minimum Spanning Trees



(a) A graph.



(b) Another graph.

Figure 3: Two graphs.

Thomas the Tank Engine must build a rail network between four cities A, B, C, and D. Due to construction costs, the railways may not branch or merge except at a city. The distances between each pair of cities are given as the weights in the graph in Figure 3a.

6. Which subset of edges form the least expensive rail network? What is the length of this network?

This optimal rail network that minimizes the sum of weights is called the *minimum spanning tree* (MST) of the graph. Finding the MST of the graph above was easy because there were only 4 nodes. It's a bit harder when there are more nodes, but still doable reasonably quickly.

7. Find the MST of Figure 1 and the sum of weights. Notice that I chose numbers that do not correspond to the positions that I placed the vertices, you can think of this as being analogous to the traffic in Los Angeles.

8. Is the MST of a graph unique? If not, what conditions might make it unique?

9. You can obviously still find an MST on an incomplete graph. How might you complete an incomplete weighted graph without affecting the MST?

10. Work with the people around you to devise a simple algorithm for finding the MST of an arbitrary graph. Then, determine the *time complexity* of your algorithm[1].

[1]Time complexity is a way of approximating how much the time it takes to run an algorithm increases as you increase the scale of a problem. For example, the time complexity of a program that enumerates a square grid of width n is just $O(n^2)$. The letter O tells you that we're talking about time complexity. We only care about the highest order term, so $O(n^2 + n + 99^{99})$ can be simplified to $O(n^2)$.

4 Travelling Salesman Problem: Introduction

Thomas the Tank Engine doesn't want to be a tank engine anymore. So, he becomes a travelling salesman, selling herbal supplements of questionable efficacy. Using Figure 3a as a map, Thomas starts at A and visits all the other cities, returning to A. This is analogous to a Hamiltonian cycle on a graph.

11. What path should Thomas the Travelling Salesman take? Check your answer by listing all possible paths and their lengths.

12. Does it actually matter which node Thomas starts at? Why or why not?

13. What's wrong with figure 3b? How might it be possible to have a physical arrangement of cities that corresponds to this graph? (Hint: Think outside the plane)

14. Given n cities. How many paths would Thomas have to check to find the optimal one? (Hint: You've already done this problem)

15. If Thomas has a super cool gaming computer that computes the length of a path in 1 nanosecond, approximately how many years would it take to compute the lengths of all the paths with 21 cities? How about 29?

16. Thomas's super cool gaming computer gets really hot when you try to run this brute force algorithm to solve the 9-node TSP in Figure 4. Work with the people around you to find a faster algorithm. If you can't come up with one that works, try to think of a heuristic, that is, an algorithm that works 'well enough'. Determine the time complexity of your new algorithm.

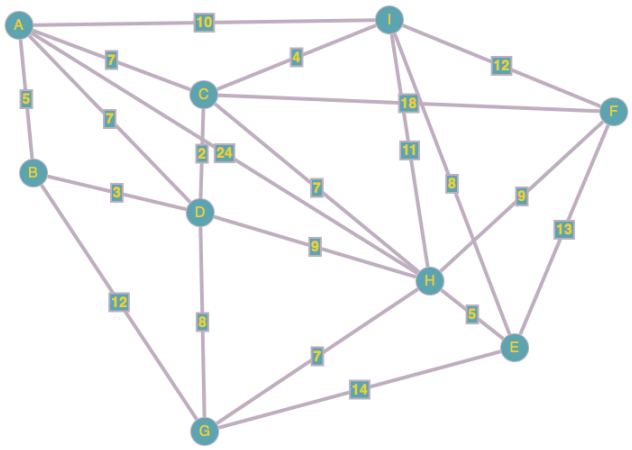


Figure 4: This graph has 9 nodes so it would take you a while to check every cycle. I left this as an incomplete graph to make it easier to see what's going on.

5 Travelling Salesman Problem: Heuristics

The last problem is really hard if you want an algorithm that finds you the optimal solution every single time. Exact algorithms faster than brute force all use something called *dynamic programming* to achieve approximately $O(2^n)$, that is, as the number of nodes n increases, the number of operations required increases exponentially base 2. Though not *polynomial time*, this is much faster than the $O(n!)$ of our brute force algorithm. I would be very impressed if you managed to come up with one of these. We won't discuss

dynamic programming here, but if you're interested, we could explore them in the future. Hopefully, you managed to come up with a simpler algorithm that works 'well enough'.[2]

You probably came up with what is called the *Nearest Neighbour algorithm*:

- Start at any node.
- Go to the nearest unvisited node and mark it as visited.

17. What is the time complexity of the Nearest Neighbour algorithm?
18. Can you come up with a graph where the Nearest Neighbour algorithm fails to find the optimal solution?
19. Come up with a graph such that Nearest Neighbour gives you the **worst** possible route.

Now, if you were a bit smarter, you may have come up with the slightly improved *Multi-fragment algorithm* (AKA *Greedy algorithm*[3]):

- Mark the shortest edge.
- Mark the shortest unmarked edge that **does not**:
  - complete a cycle that doesn't include all the nodes.
  - create a node with degree greater than 2.

20. What is the time complexity of the Multi-fragment algorithm?
21. Try this algorithm on the graph you drew for problems 18 and 19. Can you come up with a graph where Nearest Neighbour and Multi-fragment give different solutions?

6 Steiner Tree Problem

The *Steiner tree problem* (STP) is very similar to finding MSTs. In the Steiner tree problem, we are given a set of nodes with no edges and we must connect them with lines while minimizing the length of those lines. Our lines can branch off, reconnect, they can even be curvy and spirally, though, this may not help you solve the problem.

(a) 3 cities to be connected.

(b) An allowed but incorrect solution.

Figure 5: Two graphs.

[2] I encourage you to research the various exact algorithms and heuristics used to solve the TSP, this is a deep rabbit hole. For example, visit `tspvis.com` to see a visualization of solving the TSP using a *convex hull*.

[3] I prefer not to use this term since *greedy* more generally refers to any algorithm that makes locally optimal choices, so Nearest Neighbour would also count as a greedy algorithm.

22. Consider Figure 5a depicting 3 equidistant cities. Try to solve the Steiner tree problem for these nodes. That is, draw lines however you want to connect them such that you minimize the length of those lines. What happens when you vary the height of the triangle?

23. Now do the same for 4 equidistant cities. How about 5? How about a random scattering of cities? What do you notice? Compare your attempts with those of your classmates, you can use a ruler to see who is closer.

24. Nature is very good at optimization[4]. Take soap bubbles, for example, they form perfect spheres without even thinking about it! This is because the surface tension of soap bubbles naturally causes them to minimize surface area. Devise a physical construction that uses soapy water to solve arbitrary Steiner tree problems.

25. Here's a hard problem in case you got this far. $\triangle ABC$ is equilateral. Let P be a point in the interior. PD, PE, and PF are the perpendiculars from P to the sides of the triangle.

    (a) Find the point(s) P such that $PD + PE + PF$ is minimized.

    (b) *(Fermat-Toricelli point)* Find the point(s) Q such that $QA + QB + QC$ is minimized. What if the triangle is not equilateral?

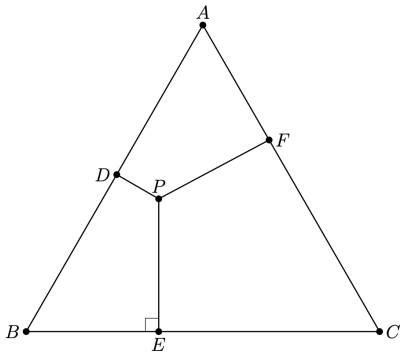


Figure 6: Now what might this problem have to do with Steiner trees?

7 Bonus

It turns out MSTs are very useful in trying to solve the TSP heuristically. Can you see why?

26. Explain why an MST on a complete graph is always going to have less total length than even the optimal TSP solution. Thus, we can use the MST as a lower-bound and starting point for our attempts at solving the TSP.

27. One of the best heuristics for the TSP is called the *Christofides algorithm*. Don't look it up just yet. See if you can come up with it on your own, or at least something close. Use your knowledge of MSTs and Eulerian cycles.

[4]Have you ever wondered why honeycombs are hexagonal?