34 The magic factor t entered equation (2). The series for $e^{kt} - e^{ct}$ starts with $1 + kt + \frac{1}{2}k^2t^2$ minus $1 + ct + \frac{1}{2}c^2t^2$. Divide by $k - c$ and set $k = c$ to start the series for te^{ct}.

35 Find four exponentials $y = e^{\lambda t}$ for $d^4y/dt^4 - y = 0$.

36 Find a particular solution to $d^4y/dt^4 + y = e^t$.

37 The solution is $y = Ae^{-2t} + Bte^{-2t}$ when $d = 4$ in Figure 16.2. Choose A and B to match $y_0 = 1$ and $y_0' = 0$. How large is $y(2\pi)$?

38 When d reaches 5 the quadratic for Figure 16.2 is $\lambda^2 + 5\lambda + 4 = (\lambda + 1)(\lambda + 4)$. Match $y = Ae^{-t} + Be^{-4t}$ to $y_0 = 1$ and $y_0' = 0$. How large is $y(2\pi)$?

39 When the quadratic for Figure 16.2 has roots $-r$ and $-4/r$, the solution is $y = Ae^{-rt} + Be^{-4t/r}$.

(a) Match the initial conditions $y_0 = 1$ and $y_0' = 0$.

(b) Show that y approaches 1 as $r \to 0$.

40 In one sentence tell why $y'' = 6y$ has exponential solutions but $y'' = 6y^2$ does not. What power $y = x^n$ solves this equation?

41 The solution to $dy/dt = f(t)$, with no y on the right side, is $y = \int f(t)\,dt$. Show that the Runge-Kutta method becomes Simpson's Rule.

42 Test all methods on the logistic equation $y' = y - y^2$ to see which gives $y_\infty = 1$ most accurately. Start at the inflection point $y_0 = \frac{1}{2}$ with $h = \frac{1}{10}$. Begin the multistep method with exact values of $y = (1 + e^{-t})^{-1}$.

43 Extend the tests of Improved Euler and Runge-Kutta to $y' = -y$ with $y_0 = 1$. They are *stable* if $|y_1| \leq 1$. How large can h be?

44 Apply Runge-Kutta to $y' = -100y + 100 \sin t$ with $y_0 = 0$ and $h = .02$. Increase h to .03 to see that instability is no joke.

16.3 Discrete Mathematics: Algorithms

Discrete mathematics is not like calculus. *Everything is finite.* I can start with the 50 states of the U.S. I ask if Maine is connected to California, by a path through neighboring states. You say *yes*. I ask for the ***shortest path*** (fewest states on the way). You get a map and try all possibilities (not really all—but your answer is right). Then I close all boundaries between states like Illinois and Indiana, because one has an even number of letters and the other has an odd number. *Is New York still connected to Washington*? You ask what kind of game this is—but I hope you will read on.

Far from being dumb, or easy, or useless, discrete mathematics asks good questions. It is important to know the fastest way across the country. It is more important to know the fastest way through a phone network. When you call long distance, a quick connection has to be found. Some lines are tied up, like Illinois to Indiana, and there is no way to try every route.

The example connects New York to New Jersey (7 letters and 9). Washington is connected to Oregon (10 letters and 6). As you read those words, your mind jumps to this fact—there is no path from New York with 7 letters to Washington with 10. Somewhere you must get stuck. There might be a path between all states with an odd number of letters—I doubt it. Graph theory gives a way to find out.

GRAPHS

A model for a large part of finite mathematics is a ***graph***. It is not the graph of $y = f(x)$. The word "graph" is used in a totally different way, for a collection of ***nodes*** and ***edges***. The nodes are like the 50 states. *The edges go between two nodes*—the neighboring states. A network of computers fits this model. So do the airline connections between cities. A pair of cities may or may not have an edge between them—depending on flight schedules. The model is determined by V and E.

DEFINITION A ***graph*** is a set V of *nodes* (or *vertices*) and a set E of *edges*.

EXAMPLE 1 How many edges are possible with n nodes, in a ***complete graph***?

The first node has edges to the $n-1$ other nodes. (*An edge to itself is not allowed.*) The second node has $n-2$ new edges. The third node has another $n-3$. The total count of edges, when none are missing, is the sum from Section 5.3:

$$1+2+\cdots+(n-1)=\tfrac{1}{2}n(n-1) \quad \text{edges in a complete graph.}$$

Fifty states have $25 \cdot 49 = 1225$ possible edges. The "neighboring states graph" has less than 200. A line of 6 nodes has 5 edges, out of $\frac{1}{2} \cdot 6 \cdot 5 = 15$ possible.

EXAMPLE 2 Which states with an odd number of letters are reachable from New York? Boundaries to states like Pennsylvania (12 letters) are closed.

Method of solution Start from New York (7). There is an edge to Connecticut (11). That touches Massachusetts (13), which is a neighbor of Vermont (7). But we missed Rhode Island, and how do we get back? *The order depends on our search method*—and two methods are specially important.

Depth first search **(DFS)** "From the current state, go to *one* new state if possible." But what do we do from Vermont, when New Hampshire (12) is not allowed? The answer is: *backtrack to Massachusetts.* That becomes the next current state.

We label every state as we reach it, to show which state we came from. Then VT has the label MA, and we easily cross back. From MA we go to RI. Then backtrack to MA and CT and NY. At every step I searched for a new state with no success. From NY we see NJ (9). Finally we are in a corner.

The depth first search is ended, by a barrier of even states. Unless we allow Ontario and keep going to Minnesota.

Breadth first search **(BFS)** "From the current state, add *all* possible new states to the bottom of the list. But take the next current state from the top of the list." There is no need to backtrack.

From NY we reach VT and MA and CT and NJ. What comes next?

Where **DFS** moves from the *last* possible state, breadth first search moves from the *first* possible state. No move from VT is possible—so we "scan" from Massachusetts. We see Rhode Island (barely). That ends **BFS**.

The same six states are reached both ways. Only the order is different. **DFS** is ***last in-first out***. **BFS** is ***first in-first out***. You have the same choice in drawing a family tree—follow a path as far as it goes and backtrack, or list all brothers and sisters before their children. The **BFS** graph in Figure 16.3 is a tree. So is the DFS graph, using forward edges only.

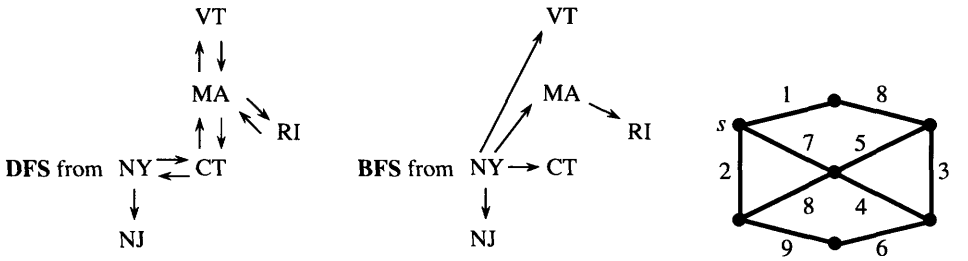


Fig. 16.3 Search trees from New York. The minimum spanning tree.

DEFINITION A ***tree*** is a connected graph with ***no loops***. Its N nodes are connected by $N - 1$ edges. If $N = n$, so every node is in the tree, it is a ***spanning tree***.

The path from VA to KY to TN to NC to VA is a *loop* (or *cycle*). If one of those four edges is removed, we have a tree. If two edges are removed, we have two trees (a small forest).

EXAMPLE 3 Allow an edge between neighboring states only when *one state is even and the other is odd.* Are the lower 48 states connected?

Start anywhere—say California. Apply either type of search—maybe DFS. Go to Arizona (7) then Utah (4) then WY (9) then CO (8) then NM then OK then TX. (I am writing this on an airplane, looking at the map.) We will never get to Florida! It is blocked by Alabama and Georgia.

The search creates a tree, but not a spanning tree. This graph is not connected.

An odd-to-even graph is special and important. It is called "***bipartite***," meaning *two parts*. The odd states are in one part, the even states are in the other. All edges go *between* parts. No edges are within a part.†

EXAMPLE 4 Is there a "***complete matching***" between 25 even and 25 odd states? This requires neighboring states to be paired off (with no repetition).

Method 1 Start pairing them off: CA–AZ, UT–WY, NV–ID, NE–SD, WA–MT. What about Oregon? Maybe it should have been paired with Idaho. Then Nevada could pair with Arizona. Trial and error goes nowhere fast.

Method 2 Think first. The four states CA–OR–WA–NV are even. This whole group is only connected to three odd states (AZ, ID, MT). The matching is impossible.

16B A complete matching is impossible if a group of nodes in one part is connected to fewer nodes in the other part. If every group has enough connections, the matching can be achieved.

This is Hall's Theorem. In a course on graphs, it would be proved. Our purpose here is to see the ideas and questions in discrete mathematics, more than the proofs.

THE GREEDY ALGORITHM

Put back all edges between neighboring states. The nodes could be provinces of Canada or states of Australia. If they are countries of Europe–Asia–Africa (or the Americas), we need a new map. The essential thing is the new problem.

In a network each edge has a "length." A positive number c_{ij} is assigned to the edge from node i to node j. In an economics problem, c_{ij} is the *cost*. In a flow problem it is the *capacity*, in an electrical circuit it is the *conductance*. We look for paths that minimize these "lengths."

PROBLEM ***Find the minimum spanning tree.*** Connect all nodes by a tree with the smallest possible total length.

The six cheapest highways connecting seven cities form a minimum spanning tree. It is cheapest to build, not cheapest to drive—you have to follow the tree. Where there

†Exactly half the states have an even number of letters (a real trivia question). This is the little-known reason for admitting Alaska and Hawaii.

is no edge we set $c_{ij} = \infty$ (or an extremely large value, in an actual code). Then the algorithm works with a complete network—all $n(n-1)/2$ edges are allowed. How does it find the minimum spanning tree in Figure 16.3c?

Method 1 *Always add the shortest edge that goes out from the current tree.*

Starting from node s, this rule chooses edges of length 1, 2, 7, 4, 3. Now it skips 5, which would close a loop. It chooses 6, for total length 23.

Method 2 *Add edges in order, from shortest to longest.* Reject an edge that closes a loop. Several trees grow together (a forest). At the end we have a minimum spanning tree.

This variation chooses edge lengths in the order 1, 2, 3, 4, 6 (*rejecting* 5), 7. In our network both methods produce the same tree. When many edges have equal length, there can be many shortest trees.

These methods are examples of the ***Greedy Algorithm*: *Do the best thing at every step.*** Don't look ahead. Stick to a decision once it is made. In most network problems the Greedy Algorithm is not optimal—in this spanning tree problem it is.

Method 2 looks faster than Method 1. Sort the edges by length, and go down the list. Just avoid loops. But sorting takes time! It is a fascinating problem in itself—bubble sort or insertion sort or heapsort. We go on to a final example of discrete mathematics and its algorithms.

PROBLEM *Find the **shortest path** from the source node s to each other node.*

The shortest path may not go along the minimum spanning tree. In the figure, the best path going east has length 1 + 8. There is a new *shortest path tree*, in which the source plays a special role as the "root."

How do we find shortest paths? Listing all possibilities is more or less insane. A good algorithm builds out from the source, selecting one new edge at every step. After k steps we know the distances $d_1, \ldots, d_k$ to the k nearest nodes.

Algorithm: Minimize $d_i + c_{ij}$ over all settled nodes i and all remaining nodes j.

The best new node j is a distance c_{ij} from a settled node, which is a distance d_i from the source. In the example network, the first edges are 1, 2, 7. *Next is* 8. The northeast node is closest to the source at this step. The final tree does not use edges 3, 5, 6—even though they are short.

These pages were written to show you the algorithmic part of discrete mathematics. The other part is *algebra*—permutations, partitions, groups, counting problems, generating functions. There is no calculus, but that's fair. The rest of the book was written to show what calculus can do—***I hope very much that you enjoyed it.***

Thank you for reading, and thinking, and working.

16.3 EXERCISES

Read-through questions

A graph is a set V of __a__ and a set E of __b__. With 6 nodes, a complete graph has __c__ edges. A spanning tree has only __d__. A tree is defined as __e__, and it is spanning if __f__. It has __g__ path between each pair of nodes.

To find a path from node i to node j, two search methods are __h__. As nodes are reached, DFS looks out from the __i__ node for a new one. BFS looks out from __j__. DFS must be prepared to __k__ to earlier nodes. In case of fire, BFS locates all doors from the room you are in before __l__.

In a bipartite graph, all edges go from one part to __m__. A matching is impossible if k nodes in one part are connected to __n__ nodes in the other part. The edges in a *network* have __o__ c_{ij}. A minimum spanning tree is __p__. It can be found by the __q__ algorithm, which accepts the shortest edge to a new node without worrying about __r__.

1 Start from one node of a hexagon (six nodes, six edges). Number the other nodes by (a) breadth first search (b) depth first search.

2 Draw two squares with one node in common (7-node graph). From that node number all others by **DFS** and **BFS**. Indicate backtracks.

3 How many spanning trees in the hexagon graph?

4 Draw a spanning tree in the two-square graph. How many spanning trees does it have?

5 Define a connected graph. If a graph has 7 edges and 9 nodes, prove that it is not connected.

6 Define a loop. If a connected graph has 8 edges and 9 nodes, prove that it has no loops.

7 Find the shortest path (minimum number of edges) from Maine to California.

8 Which state is farthest (how many edges are needed) from the state you are in? Why would it come last in **BFS**?

9 List the steps of **BFS** from your state to Georgia or Colorado or New Jersey. (*There are edges Hawaii–California and Alaska–Washington.*)

10 With edges between odd neighboring states and between even neighbors, what is the largest connected set of states? Map required.

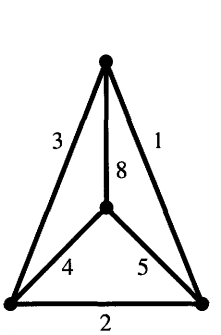
11 With edges only from odd to even neighbors, how many states can be matched? (Answer unknown to author—please advise.)

12 A matching is a forest of two-node trees. Give another description.

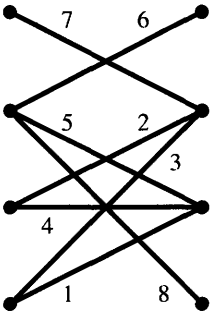
13 Find the minimum spanning tree for network **A**.

14 Find the shortest path tree from the center of network **A**.

15 Is there a complete matching between left and right nodes in graph **B**? If not, which group of nodes has too few connections?



A



B

16 Find the loop in network **B**. Then find a minimum spanning tree by Method 1 and Method 2.

17 How many spanning trees in graph **B**? It has one loop.

18 Show that a graph cannot have 0, 1, 2, 3, and 4 edges going into its five nodes.

19 If the only edges into a node have lengths 6 and 8, can they both be in a minimum spanning tree?

20 In Problem 19, prove that a minimum spanning tree contains edge (6) if it contains edge (8).

21 *True or false*, with reason or example.

(a) In a complete network, the minimum spanning tree contains the $n-1$ shortest edges.

(b) If a graph has 9 nodes and 9 edges, it has a loop.

(c) A graph with a complete matching must be connected.

22 Draw a tree that is perfect for (a) DFS; (b) BFS.

23 The *adjacency matrix* has $a_{ij}=1$ if there is an edge from node i to node j. Write down this matrix for graphs A and B.

24 In a complete network start with $d_{ij}=c_{ij}$. Show that the d_{ij} at the end of this program are shortest distances:

for $i=1$ to n do
for $j=1$ to n do
for $k=1$ to n do
$d_{ij}=\max(d_{ij},\ d_{ik}+d_{kj})$

25 How many spanning trees in graph **A**?

26 A maximum spanning tree has greatest possible length. Give an algorithm to find it.

27 Write a code that will find a spanning tree (or stop), given a list of edges like (1, 2), (1, 3), (4, 7), ...

MIT OpenCourseWare
http://ocw.mit.edu

Resource: Calculus Online Textbook

Gilbert Strang

The following may not correspond to a particular course on MIT OpenCourseWare, but has been provided by the author as an individual learning resource.

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.