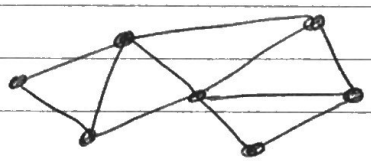
11.4 Spanning Trees

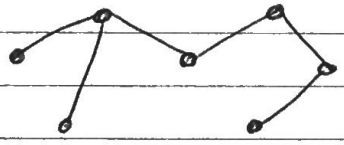
Suppose these 7 towns in Maine are connected by the road network shown in this graph:



After a big snowfall, what is the fewest number of roads that can be plowed to keep a path between any two towns?

Idea: if roads make simple circuits then at least one doesn't need to be plowed.

One solution, plowing 6 roads:



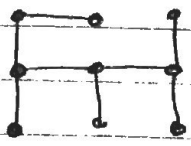
answer is a tree

Definition: A <u>spanning tree</u> of a graph G is a subgraph that is a tree containing every vertex of G.

It is easy to show that a simple graph has a spanning tree if and only if it is connected.

Example ① Find a spanning tree for:

one solution:



There were many possible spanning tree solutions.

Cayley's formula says that K_n (with all vertices labeled) has n^{n-2} spanning trees.

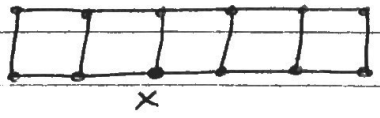
Depth-first search

This is one of the main methods to construct a spanning tree - also called backtracking.

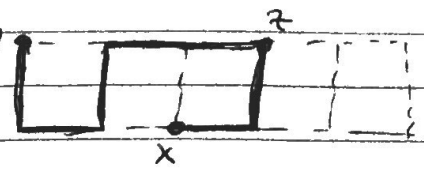
It works by starting with any vertex of the graph. This will be the root of the spanning tree. Make the longest possible simple path in the graph starting at the vertex and making sure the path has no simple circuits.

If there are vertices not on the path then move back towards the root until you can continue the path in another direction. Repeat until all vertices included.

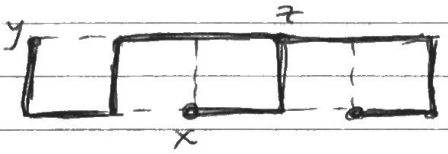
Example (2) Use a depth-first search to find a spanning tree for this graph, starting at x



Solution. There are many solutions. Our first path could be



and we get stuck at vertex y. Adding another edge at y would make a circuit. From y we must backtrack towards x. At vertex z we see we can continue the path to get new vertices:



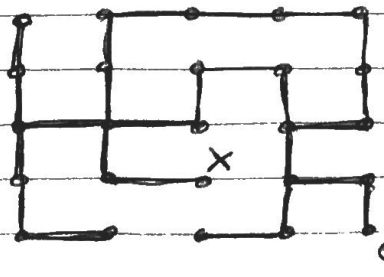
We have finished the spanning tree.

Applications of Backtracking

These depth-first searches give a systematic way to go through all possibilities when looking for a solution.

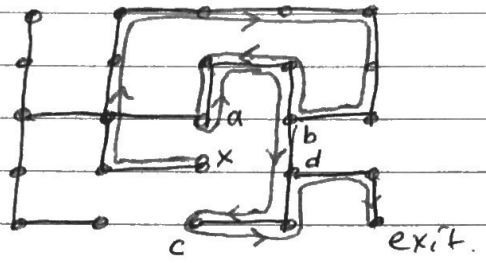
- OA pages 791-792 backtracking is used to decide if a graph can be colored using n colors.

- Escaping from a maze. Suppose you are inside a maze like this example where the edges represent paths you can walk along:



Start at the vertex marked x and try to find the vertex marked exit.

A search using backtracking is certain to eventually find a way to the exit (you just need a way to mark your paths, eg. with pebbles). For example, your depth-first search might look like this:

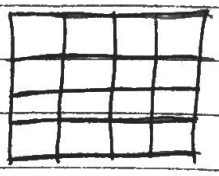


You make a path from x to a and at a you have to stop or you've made a circuit. Backtrack to b and make a new path to c. Backtrack from c to d and then continue to the exit.

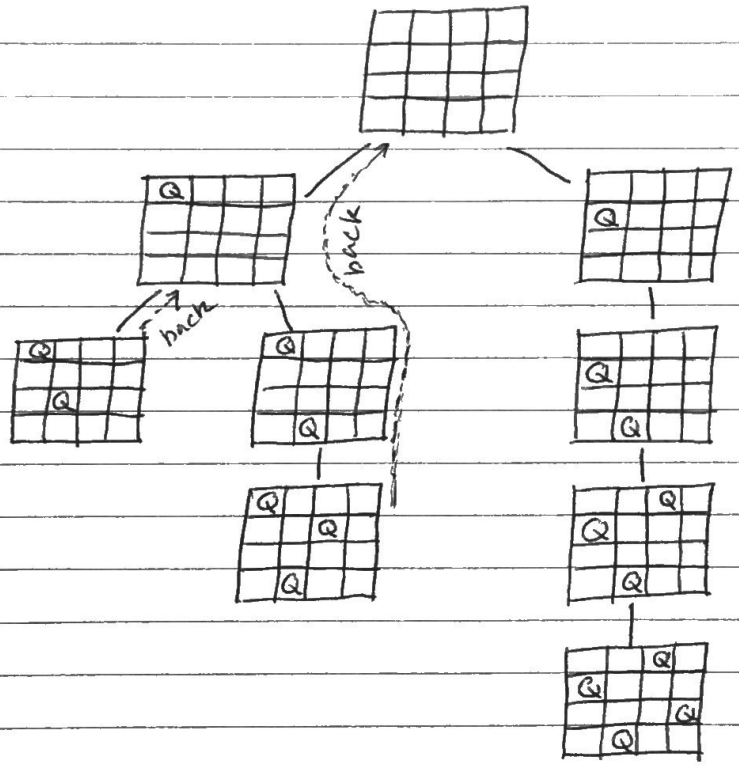
- The n-queens problem. We want to place n queens on an nxn chess board so that no two are attacking each other - see example 7 p 792.

We can solve this problem using backtracking, working column by column and trying to put the queen as high as possible in each column.

For example with n=4, start with empty board

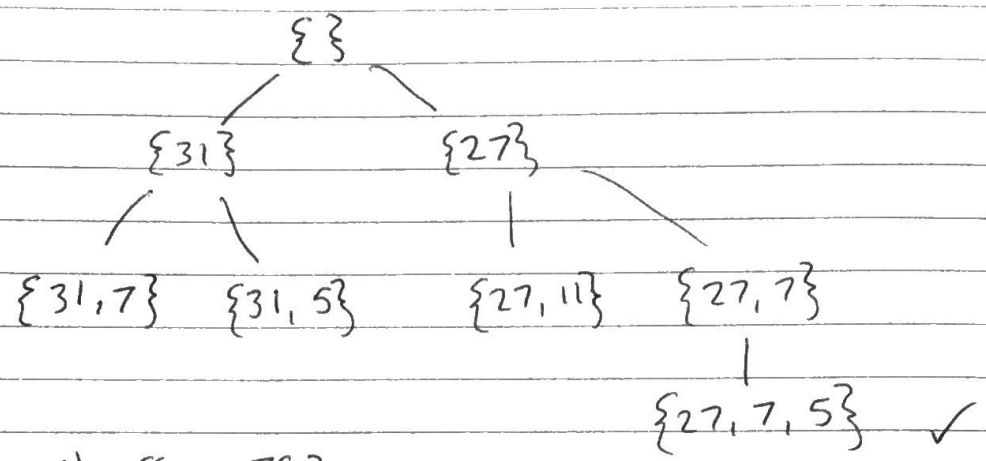


In the first column, try the queen Q as high as possible. Then in the next column the highest Q can be is the third row. In the next column we have a problem – all squares are attacked so we must backtrack. See the following tree



Solution.

- Sums of subsets. Given a set of numbers, find a subset that adds to a particular number. For example, find a subset of $\{31, 27, 15, 11, 7, 5\}$ that adds to 38. As in the last example we make a tree of the possibilities, starting with the numbers in their decreasing order. If the sum gets bigger than 38 we must backtrack:



✓

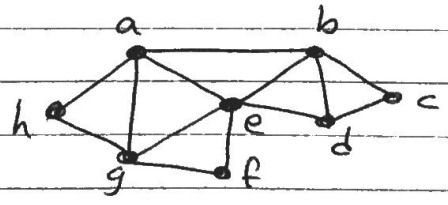
See example 8, p793.

Breadth-first search

A second important method to make a spanning tree for a graph is called breadth-first search.

This works by choosing a vertex of the graph. Add paths to all the adjacent vertices (making level 1 on our tree). Next add all the adjacent vertices to these level 1 vertices - making sure there are no circuits. Repeat until all vertices are included.

Example. Use a breadth-first search to find a spanning tree for this graph, starting at a.



Solution.
In the first step add paths from a to b, e, g, h.
In the second step add paths from b to c, d and from e to f.