codeforces.com

Shortest Path Modelling Tutorial - Codeforces

6-7 minutes

Shortest path problems are really common. There are uncountable problems that can be reduced to some shortest path problem on graph. In this post i will show some different problems that require some extra thinking because they are not the usual shortest path problems (there are additional constraints to the problem). For most high rated coders this is still common stuff but i hope i can help some newcomers out there.

About Graph Modelling

Sometimes we face a problem that asks us to find some shortest path in a graph, but there are additional constraints to the problem. If we are not used to this kind of problem, we may come up with the idea of changing Dijkstra's algorithm so it will work in the specified problem. But usually this will not work. Instead, we must change our graph so that the traditional shortest path algorithm will solve the problem with added constraints. This is called graph modelling.

State Graphs

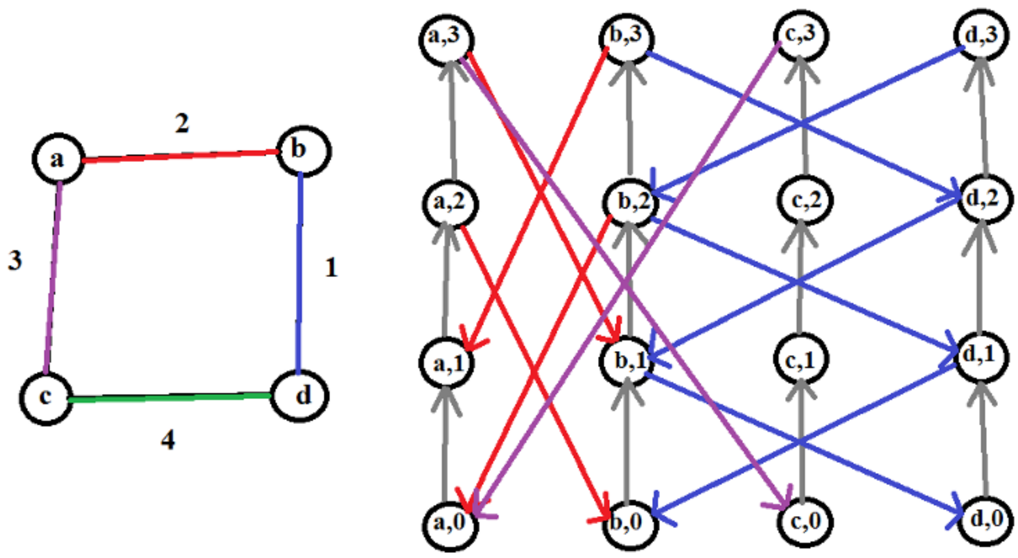
What we want to build is a state graph. In this graph, every vertex will store additional information, representing some state of the problem, and the edges represent transitions between states. We can see it like a finite state machine and we are going to compute the shortest path from the initial state to some final state. The construction of the state graph is usually the challenge in this kind of problem, and after the graph is built we will have a good time using Dijkstra's algorithm. Now we are going to see some examples of shortest path modelling problems and how to build the state graphs.

Problem 1

Statement: You want to travel from city A to city B. There are N cities and M bidirectional roads connecting cities. Your car can store up to C liters of fuel and the tank is initially full. each road (i, j) has a value W_{ij} that represents the amount of liters of fuel to cross this road. Also, in every city you can buy fuel, at a price of C_i dollars per liter. You must compute the minimum amount of dollars spent with fuel to travel from A to B.

Solution: The first thing we must do is to build the state graph. In this problem, a state represents a pair $(city, currentFuel)$. So, we are going to have NC states. There are two kinds of edges in our state graph: at any moment we can charge a liter of fuel (unless the tank is full), so we can go from state (i, f) to state $(i, f + 1)$ with cost C_i, or we can go to another city, so we can go from state (i, f) to state $(j, f - W_{ij})$ for every city j that is adjacent to city i. This second type of edge has cost 0, because remember that we want to compute the amount of dollars spent, not the amount of liters. Now that our graph is done, we can model our problem as finding the shortest path from vertex (A,C) to any vertex (B,*), which can be done with usual Dijkstra's algorithm.

The image below is an example of what is said above. It's a small test case just to show how the state graph is built. The graph on the left is the original graph, and the graph on the right is the state graph. The max fuel capacity C is 3. The gray edges represent recharging the fuel tank. Edge costs are omitted because the drawing is messed up enough (love paint!)



My code for reference: http://pastebin.com/Kv9XdAfq

Now if you look at the code, there is a catch. You don't need to explicitly build the state graph! You can build the adjacency list for the original graph, and model the state graph using the distance array computed in Dijkstra's algorithm. The idea is that you only need the transitions of the original graph in order to be able to traverse in the state graph, so you can build the original graph but instead of traversing the adjacency list in Dijkstra's you will traverse the state graph transitions. The time complexity will not change but we are going to save a lot of memory and the coding will also become easier. This trick will be used in all modelling problems.

Problem 2

This one appeared in a regional contest here in Brazil last year. Unfortunately my team wasn't able to solve it during the contest even though it was an easy problem, and this was my initial motivation to study about graph modelling.

Statement: You are given an usual undirected graph, and you want to find the shortest path from A to B, but there is an additional constraint: you must use an even amount of edges.

Solution: We are going to build the state graph for this problem. Now each state will represent a pair $(vertex, parity)$, meaning we will have $2N$ states. The transitions are really simple: for every vertex v adjacent to vertex u, we can go from state $(u, even)$ to state (v, odd) and from state (u, odd) to state $(v, even)$. Now we can simply find the shortest path from state $(A, even)$ (because we start with a path of length 0) to state $(B, even)$, and once again this is done with usual Dijkstra's algorithm.

My code for reference: http://pastebin.com/gqNJk9yN

Another cool trick that you might not know: if you look at the code i don't do any fancy conditionals to go from an even state to an odd state. We can represent even as 0 and odd as 1, and if you XOR current state with 1, you get to the opposite state! Just look at the XOR truth table and you will see that $0 \oplus 1 = 1$ and $1 \oplus 1 = 0$. This trick is really useful and is used in a lot of algorithms like binary heap and iterative segment tree.

Practice

UVA 10269 is another problem in which you have to compute shortest paths in some state graph : https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=1210

UVA 11492 is a problem from a past ACM ICPC Latin America, a really beautiful problem on this topic : https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=2487

If you know more problems on this topic i will be really grateful if you can post them here!