Arbitrage, or – How to Make Money Instantly!

In economics – taking advantage of an imbalance between two or more markets. Buy low in one market, sell high in another – risk-free profit.

Example – taking advantage of currency exchange rates. Given three currencies, US (USD) dollars, Euros (EUR) and Canadian dollars (CAD). Say 1 USD buys 0.741 EUR, 1 EUR buys 1.366 CAD and 1 CAD buys 0.995 USD.

Here is a subgraph describing these exchange rates (based on the slides. Only the relevant subgraph is included here for simplicity).

**Question:** Given such a graph, how do you detect an arbitrage?

**Answer:** Notice that if you start with 1000 USD you can buy 741 EUR, use them to buy 1,012.206 CAD and then buy 1,007.145 with them! So, just by cycling through these three exchanges you can end up with more money than you started. Theoretically, you could make an infinite amount of money like that! (practically it’s impossible of course, if only because you’d crash the money market long before it happened).

From the graph point of view it’s easy to see that there is an arbitrage in the graph iff there is a cycle whose edge product is $> 1$. We only learned to deal with edge sums, though, not products. However, we have a simple way to convert products into sums... Just take a log of the original values! Remember that:

\[
\prod_i x_i > 1 \Leftrightarrow \sum_i \log x_i > 0.
\]

This means that if we create a new graph $G'$ with the same vertices, but replace the rates by their logs, we will have an arbitrage in the graph iff there is a cycle whose edge sum is $> 0$. See example here: (I used ln following the book’s example, but it can be any log). In the graph below the sum of the cycle is 0.007.

It would be easier if we could detect a negative cycle, though, because that’s what Bellman-Ford’s
algorithm knows how to do. Luckily, we can use another known property of logarithms: We know that $\log x > 0 \Leftrightarrow \log \frac{1}{x} < 0$.

So finally, we can construct a graph $G' = V, E'$ with the same set of vertices and edges, but the weights on the edges are $\log \frac{1}{w}$, where $w$ is the original weight. Or alternatively, just use $-\log w$ (it’s the same). To sum up, there is an arbitrage in the original graph iff there is a negative cycle in the graph shown here.

We know how to find a negative cycle – just run Bellman Ford’s algorithm...

This kind of trick is called a reduction – we convert an instance of problem $a$ to an instance of problem $b$ that we know how to solve, and use the solution to solve $a$. It works if the conversion is efficient enough and also if the two problems are related such that a result on $a$ is true iff a result on $b$ is true.