SACO 2005 Day 2 solutions

Bruce Merry

How not to be seen

Any tree that is at least as tall as both its neighbours (or its one neighbour, for the edges) cannot be destroyed by its neighbours, and hence must be directly bombed. But if a tree has a taller neighbour, either it is directly bombed or it has an even taller neighbour and so on, so no other trees need to be bombed.

Cheese

A brute force approach is to try every possible order of removing the cheeses. This will score 50%, but for the worst cases there are 2^{1999} possible orders — which will take until the end of the universe to check.

A more efficient solution can be found by formulating the problem mathematically. Let $m_{i,j}$ be the amount of money that Mr. Wensleydale can make starting at the point where cheeses i and j are directly available ($i < j$). This can only occur on day $N+2-i-j$ — we abbreviate this to $d_{i,j}$ for convenience.

If Mr. Wensleydale has only one cheese to sell there are no decisions to make, so $m_{i,i} = v_i d_{i,j}$. However, if $i < j$ then he can either sell cheese i or cheese j. If he sells cheese i then he makes $v_i d_{i,j}$ today and $m_{i+1,j}$ in the future; similarly if he sells cheese j then he makes $v_j d_{i,j}$ today and $m_{i,j-1}$ in the future. So

$$m_{i,j} = \max\{v_i d_{i,j} + m_{i+1,j}, v_j d_{i,j} + m_{i,j-1}\}$$

for $i < j$. This can easily be turned into a recursive function that takes i and j as parameters and returns $m_{i,j}$, but this is not much different from the brute force solution. However, you might notice that there are less than N^2 possible pairs (i,j) that can be passed to the function — so if is being called about 2^N times, then there must be a lot of repetition.

Instead of using a function, one can instead compute m as a two-dimensional array indexed by i and j. The array is built in a double for-loop, using the equation above. Some care is needed to get the sequence right: i should iterate downwards and j should iterate upwards, so that the right hand side is always computed before the left hand side.

An alternative idea is to always sell the cheapest available cheese, hoping to let the more expensive cheeses mature. This does not always work because it is a short-sighted approach. The test data is set up so that this solution scores only 20%.

Roads

The property that the Knights want England to have is known as biconnectivity (connectivity means there is at least one path, and biconnectivity means at least two paths)[1].

First consider an easier version of the problem, in which there is initially *exactly* one route between any two castles. Such a network will have $N = M-1$ and is known as a tree. The castles connected to only one other castle as known as leaves. Clearly, any leaf needs another road to it, since otherwise cutting off its only road would leave it isolated. This implies a lower bound of $\lceil \frac{N}{2} \rceil$ on the number of extra roads. What is less easy to prove is that this is always sufficient.

Now consider the original problem. Some roads are already redundant (i.e. they can be cut off without separating England into two unconnected networks). Other roads are critical to the network, and are technically known as *bridges*. The graph can be decomposed into *biconnected components* — subsets of the castles which are internally biconnected but separated from each other by bridges. Each biconnected component can be replaced by a single castle, since the internal routing is adequate and only the external connections need to be considered. This transformation reduces the graph to a tree, and we can apply the algorithm above.

[1]There are actually two variations on biconnectivity — the more conventional one deals with routes that share no castles rather than no roads.

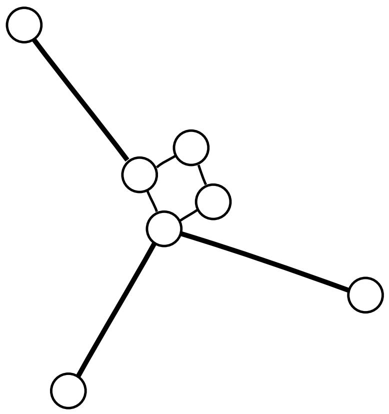

Figure 1: `roads0.in`

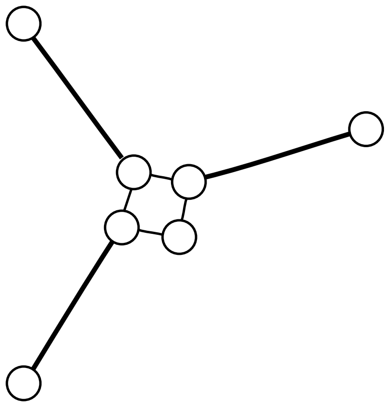

Figure 2: `roads1.in`

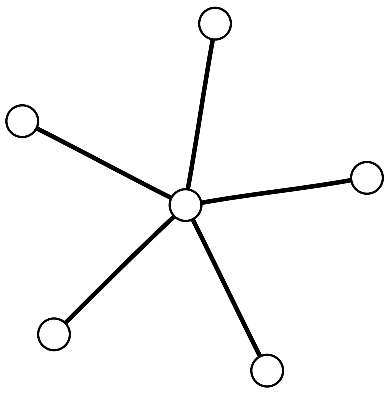

Figure 3: `roads2.in`

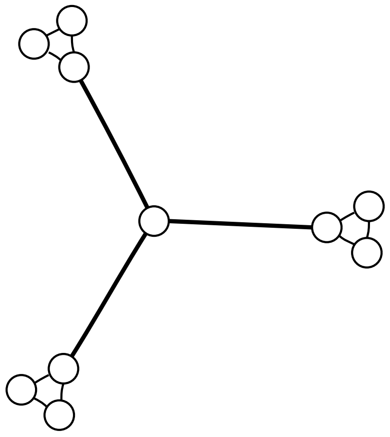

Figure 4: `roads3.in`

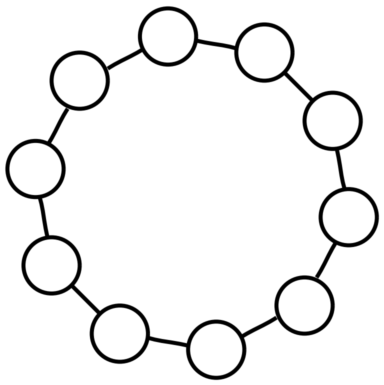

Figure 5: `roads4.in`

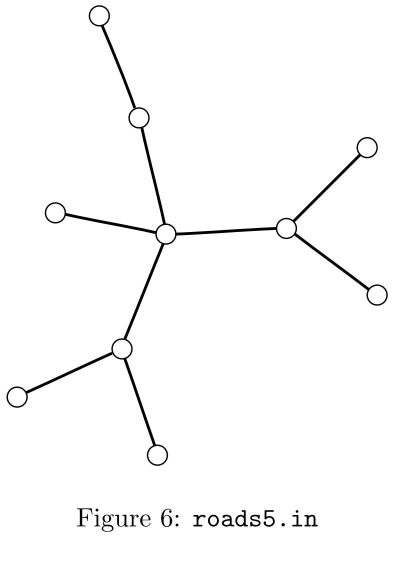

Figure 6: `roads5.in`