SACO 2005 Day 1 solutions

Bruce Merry

Hungarian phrasebook

The brute force approach is to take something said by a Hungarian and try to match it to any of the phrase-book phrases. This solves the 50% constraints, but is too slow for the 100% constraints.

A more efficient implementation uses a data structure called a hash table. This structure has similar properties to an array, but the lookup is done on a string or other object rather than an integer. All the prefixes of phrases are first inserted into the hash table. The things said by Hungarians are then looked up in the hash table to determine whether they are prefixes.

Pascal does not provide a hash table class, making this a difficult solution to implement. An alternative solution is to sort the phrases. If something said by a Hungarian is a prefix, then the corresponding phrase will be the one immediately after this in a dictionary order. So instead of doing a linear search through all the phrases, one need only do a binary search to identify the phrase that is just after what was said in the list of phrases.

Scales

The brute force approach tries every possible combination of weights, adds them up, and checks against the best current combination and the upper bound. For 40 weights, there are 2^{40} sets — roughly 10^{12}, which is far too slow.

The unusual property of the weights (that $w_{i+2} \geq w_{i+1} + w_i$) allows one to speed up the algorithm. If we write

$$\begin{aligned} w_3 - w_2 &\geq w_1 \\ w_4 - w_3 &\geq w_2 \\ w_5 - w_4 &\geq w_3 \\ &\vdots \\ w_{i+2} - w_{i+1} &\geq w_n \end{aligned}$$

and then add up the equations, we obtain $w_{i+2} - w_2 \geq \sum_{j=1}^{i} w_j$. In other words, using the first i weights will never give as much as just using weight $i+2$. Now consider the following cases:

- $w_{N-1} + w_N \leq C$: If we take everything except w_N, we get less than $w_{N-1} + w_N$, in which case we might as well just take w_{N-1} and w_N instead. So either way, we must take w_N.
- $w_N \leq C < w_{N-1} + w_N$: We cannot take both w_{N-1} and w_N. If we take neither then we get less than w_N and should just take w_N instead. So we must take exactly one of them.
- $C < w_N$: obviously we cannot take w_N.

In the first and third cases, we know whether to take w_N, and repeat the problem using only the first $N-1$ weights and any remaining capacity. In the second case, we have to branch and consider both cases, in each case solving the problem for the remaining $N-2$ weights. In the worst case we will always have the second case, causing us to branch two ways at each level. However, there are only $\frac{N}{2}$ levels of recursion (because we eliminate two, rather than one, weight each time), so the running time is proportional to $2^{N/2}$ rather than 2^N.

Ni

Conceptually, this is a problem about shortest paths. For each shrubbery, we need the shortest path from Arthur to the shrubbery and from the shrubbery to the Knights who say Ni. The shortest path in a maze can be found using a technique called breadth first search (BFS). Initially, the starting point is added to a queue. Then, while the queue is not empty, the front of the queue is removed and processed. Any neighbours of this point that can be used but haven't yet been visited are added to the back of the queue. Because a queue is first-come-first-serve, the points nearest the start are processed first.

Figure 1: `ni0.in`

Figure 2: `ni1.in`

For each point, we also keep track of where it was reached from so that we can reconstruct the path.

An interesting feature of BFS is that it gives you the shortest distance to *all* other points, not just the target point of interest. We can exploit this to speed up the solution. Instead of finding the shortest path from each shrubbery to the Knights who say Ni, we can find the shortest path from them to all the shrubberies, and later just reverse the path.

Figure 3: `ni2.in`

Figure 4: `ni3.in`

Figure 5: `ni4.in`

Figure 6: `ni5.in`