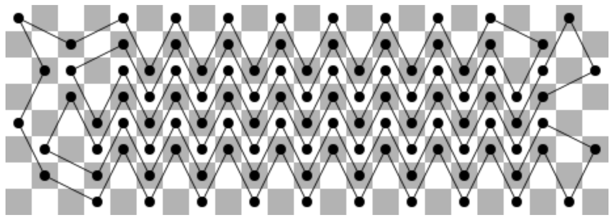
algorithm, where for each line between squares we just add a visited square in the next row, for each visited square with one incoming edge we branch out into the four options for the other edge from this square, and for each empty square we branch out into seven options – either it is really empty, or we visit it with both edges going downwards. We need to do this carefully to make sure the edges we add do not intersect, and that we maintain the connectivity information correctly.

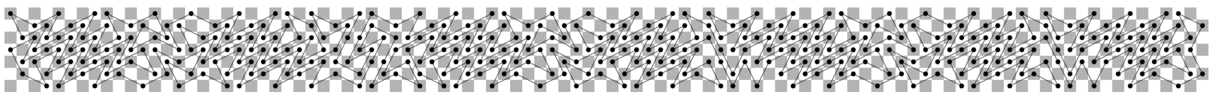
There are many technical details in the implementation of this algorithm (for instance, one has to distinguish the "empty row" state of "we have not started yet, there were no visited squares" from "we have already finished the tour", in order not to allow multiple disconnected tours as a solution). With an efficient implementation, an 8 by 50 chessboard can be solved in several seconds, and an unoptimized program will perhaps run for several minutes.

Now, the crucial observation is that the solutions actually start getting repetitive after a while. For instance, for board size 8 by n, after a while you get a pattern of 6 knights in a row all jumping left, and then all jumping right, adding 12 visited squares with each two added rows, as in the picture below.



This suggests that we should try to run the DP described above for a reasonable range of n, and look for this cyclic behaviour – that adding some k rows to n increases the answer by some a. And indeed – after running up to, say, $n \leq 200$, we will discover cycles for all possible values of m. Thus we can precompute these values locally and then submit a simple solution which has them hard-coded.

We found it interesting that for $m = 7$ one finds a somewhat surprising cycle length of 33 – there is a pattern of size 7×33 that can be repeated over and over in the solution. See, for instance, this diagram:



Problem K: Wireless is the New Fiber

Shortest judge solution: 709 bytes. Shortest team solution (during contest): 851 bytes.

This is one of the easier problems in this problem set. After reading through the statement, we can see the problem is about constructing a tree where as many vertices as possibe have a given degree (note that the only information we care about from the input are the degrees of vertices in the input graph, and not the exact shape of the input graph).

The choice of which vertices preserve their degree can be made greedily. Since we have to construct a tree, the sum of degrees of all the vertices will be $2n - 2$, and each vertex will have degree at least 1. Thus, we are left with $n - 2$ spare degree increases to assign. In order to satisfy as many vertices as possible, we should assign these increments to the vertices with the smallest expected degree first. In this way we arrive with a degree assignment that satisfies as many degrees as possible.

The second part of the problem is to construct a tree with given degree values. This can be done in a number of ways. One such way is to order the vertices by degree, and add them to the tree in order of decreasing degree, starting with the highest degree vertex as the root, and maintaining a list of "outstanding degrees" – that is vertices with a larger expected degree than the number of edges already connected. When adding a new vertex to the tree, we connect it to any of the vertices with positive outstanding degree (and decrease the outstanding degree of both vertices by one). Note that until the very end, there will always be a vertex with positive outstanding degree in the already constructed tree. This is because a tree with k vertices has outstanding degree zero if the sum of expected degrees in the tree is $2k - 2$, and so the average expected degree is $(2k - 2)/k$. However, the average expected degree among all vertices is $(2n - 2)/n$, which is larger than $(2k - 2)/k$ since $k < n$, and the average expected degree in any partially constructed tree is at least equal to the average expected degree among all vertices, because we take the vertices from the largest expected degree.