

Analysis: Pegman

If Pegman walks off the map, there must have been some arrow that points towards an edge of the grid, with no other arrows in between. So we must take every such arrow and point it towards another arrow. We simply count how many of these arrows there are, and if any of them can't be pointed towards another arrow — which is when there are no other arrows in the same row or column — then the answer is `IMPOSSIBLE`.

Analysis: Kiddie Pool

While the pool is being filled, each source of water is turned on for some fraction of the time. This is equivalent to having each source turned on for the whole time at a fixed rate which is between zero and R_i . So the problem is to find what rate each source should be set to in order to maximize the total flow and get the correct temperature. Then the fill time is the capacity of the pool divided by the total flow rate.

Find the temperature of the water we get when setting each source's rate to the maximum. If the temperature is exactly the desired temperature, then we have the solution.

If the temperature is too hot, then we need to lower the temperature by lowering the flow rate of some of the sources. To lower the total flow by as little as possible, we decrease flow starting with the hottest source, moving to the cooler sources if necessary, until the average temperature of the flowing water equals the desired temperature. If we never reach the desired temperature, then no solution was possible.

If the temperature is too low when all the sources are set to their maximum rate, then the solution is similar — we raise the temperature by reducing the flow rate of sources, starting from the coldest source.

Analysis: Bilingual

Construct a graph with a node for each sentence and word. For each sentence S , add edges in the graph between the node for S and each of the nodes for the words in S .

Now consider a path in the graph from sentence 0 to sentence 1. The nodes on the path will alternate between sentences and words. Since the language of the sentences in the path must at some point change from English to French, one of the words along the path must belong to both languages. We need to find a minimal set of words such that every path from sentence 0 to sentence 1 in the graph goes through one of the words in the set. This is equivalent to finding a minimal vertex cut in the graph, where the cut vertices can only include the nodes for words.

We can solve this by transforming the problem into an edge cut problem in a directed graph. For each word w , create two nodes A_w and B_w . Add a directed edge from A_w to B_w with capacity 1, and for each sentence S which contains w , add directed edges from S to A_w and from B_w to S , both with infinite capacity.

Now we find the size of the minimum cut using a maximum flow algorithm, and the [max-flow min-cut theorem](#). Each edge in the cut will cut the edge from A_w to B_w for some word w , since those are the only finite-capacity edges. These words form a minimal set of bilingual words which solves the problem.

Analysis: Drum Decorator

Some important initial insights are:

- A cell can only take three values: 1, 2, or 3. 4s would have to keep propagating until they hit an edge cell, and edge cells only border three other cells.
- If a 3 appears, it must be in a band of 2 complete rows. To see this, place a 3 along the border of the top of the drum, and note that the 3 adjacent cells must all contain 3s. This must continue all the way around the top, which also fills up the row below. Suppose that some other finite arrangement of 3s was valid — then there must be a highest 3 in the pattern, above which no other 3 can be placed, but then that 3 can only be part of a set of 2 rows of 3s, by the same argument above.
- If a 2 appears, it must be either in a 2x2 square of 2s, or in a (possibly winding) line of 2s that goes all the way around the drum and reconnects with itself.
- 1s can only appear in adjacent "dominoes" of two.

Besides the bands of 3s, there are four patterns using 1s and 2s:

I. one-thick bands of 2s:

2...

II. alternating upright dominoes of 1s and 2x2 squares of 2s:

122...

122...

III. a line of 2s winding around horizontal dominoes of 1s:

222112...

112222...

IV. a line of 2s winding around vertical dominoes of 1s:

2212...

1212...

1222...

None of these patterns can border each other, but they can be separated by bands of 3s. The number of columns in a drum can rule out some patterns -- II, III, and IV require multiples of 3, 6, and 4, respectively.

Now we can use dynamic programming to assign patterns to the drum. Our state is how many rows we have filled, and whether the previous pattern was a pair of rows of 3s.

Each of the five patterns repeat with some period P , which is either 1, 3, 4 or 6, so there are P ways to place that pattern on the drum.

Each overall drum assignment also has a period, which is the least common multiplier of the periods of all of the patterns on it.

An extra complication is that drum assignments that are rotations of each other should only be counted once. So our dynamic programming state should also include the period of the current

drum assignment. When counting the final number of solutions, we divide the number of solutions in each state by the state's period. For example, if we have 18 ways to produce a solution of period 3, this only corresponds to 6 solutions.

Another way to handle equivalent patterns is to use [Burnside's lemma](#). We leave the details as an exercise to the reader!