Robotzi

2021 Baraj 1 Seniori

Non-scalable solutions include anything with explicit counting with states:
1. Just count with all the states -- robots_dummy.txt.
2. Count with 2 counters (next to each other) and use CRT -- robots_crt.txt.

A few actually generic but suboptimal approaches are also implemented,
but will not be discussed here. The main ideas are to "slide" around
the Ys and Ns and either sort/partition them or cancel them out.

The full solutions starts with a fairly obvious idea:
Use a binary counter to count number seen Ys - number of seen Ns.
The interesting part is how to implement it:
0. The Y and N robots when not next to the counter do nothing.
   I.e. they keep their state the same.
1. Keep a binary counter on the "tape" of robots.
   Actually, we allow 2 states to represent carry.
   This is needed since we need to propagate carries.
   We also allow -1 and -2 since we can't access
   a sign bit from far away, so we store negative digits.
   Still the interpretation of the counter as a nubmer is
   Sum_i 2^i d_i (just allowing -2 <= d_i <= 2)
2. Move the counter to the right on each iteration.
   The least senior bit (the rightmost one) "eats"
   the leftmost Y/N (incrementing or decrementing).
   All 2/-2 become 0 (barring any incoming carry)
   and the bit before them "eats" their carry.
   Since we are moving the counter to the right,
   all of this state is actually transfered to the "next" robot.
   E.g. for -2 <= d <= 2
   d -1/0/1 ? -> d
   d 2/Y ? -> d + 1
   d -2/N ? -> d - 1
3. On the left end of the counter we move a fake X, X'.
   It is equivalent to 0, except it means that all bits to
   its left are also X' (0 in value).
   This is useful for the next part so we know where the
   counter ends on its left part.
4. When the counter reaches the end we check its sign.
   We do this by still shifting the counter to the right.
   We can't stop it anyway. The rightmost robot (the one
   that sees the border) recors the sign of the shifted out bits.
   That is dominated by its current value if it is non-zero.

However, if the bit's current value is zero, it matters.
So we also allow a special -0 bit value there.
It means the shifted out part was negative,
but is otherwise equivalent to 0.
When we are finally left with only one bit with an X'
to its left it just "returns" its sign.
It announces Minor if it is negative and Major otherwise.
This solution takes N + log_2(N) steps.
See robots_counter_binary.txt.

We can nearly double the speed with an easy improvement.
Instead of keeping the Y/N still, we move them to the left.
This means that on each iteration the counter "eats" two votes.
We also drag an X' from the right, to mark the right end of the tape.
Note that naively, we may need to allow 3 and -3 states
(for when 1 eats YY or -1 eats NN).
However, we can notice that the rightmost bit is always even,
so we don't need these states there.
The only other non-trivial part is how to handle the end,
which now depends on the parity of the length of the tape.
For convenience we introduce +X' and -X' to store the sign
on the actual end of tape marker and we just need a few rules
for all the cases like d Y/N X' and d X' X'.
This solution takes N / 2 + log_2(N) steps.
See robots_counter_fast_binary.txt.

The final improvement is to reduce the log term.
It is determined from the length of the counter.
This can be done by using a higher base in an analogous way.
We have enough states for base 12, which is what the full solution is.
Instead of writing it by hand, since there are many transitions,
we can write a program to print the instructions.
See robots_counter_fast_base_K_generator.txt.