Collaborator: yc2454 (Yalu Cai)

The intuition behind this is sorting the inputs (putting all 1's before all 0's), and seeing if the $\left\lceil \frac{n}{2} \right\rceil$-th input after sorting is 1. If it is 1, then there is a majority of 1's in the input, otherwise no. The type of sorting we will use is bubble sort, which we know to be $O(n^2)$ time, which we will translate into a $O(n^2)$ sized circuit.

There are n^2 "levels" the circuit (details later). We will implement bubble sort on the inputs as such: for the first level, we compare x_1 with x_2, and if x_1 is 0 and x_2 is 1, then in the "next level", x_1 will become 1 and x_2 will become 0; otherwise, they stay the same in the next level. x_3 through x_n will stay the same in the next level. In the next level, we compare x_2 with x_3 and so on. After we've reached the comparison between x_{n-1} and x_n, we will have done $n-1$ comparisons, which is $O(n)$. Let's call each sequence of $n-1$ comparisons from x_1 to x_n a "line comparison". Within n line comparisons, the inputs will have been completely sorted (I hope I don't have to prove this…). After the inputs are sorted, the answer to this MAJORITY problem is whether the $\left\lceil \frac{n}{2} \right\rceil$-th bit is a 1. This whole thing is $O(n^2)$ time.

The first detail we will implement is switching a "0 ,1" to a "1, 0" (not switching them otherwise). As CS 3410 taught me, we implement this by writing out the truth table for it first:

x_1-old	x_2-old	x_1-new	x_2-new
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

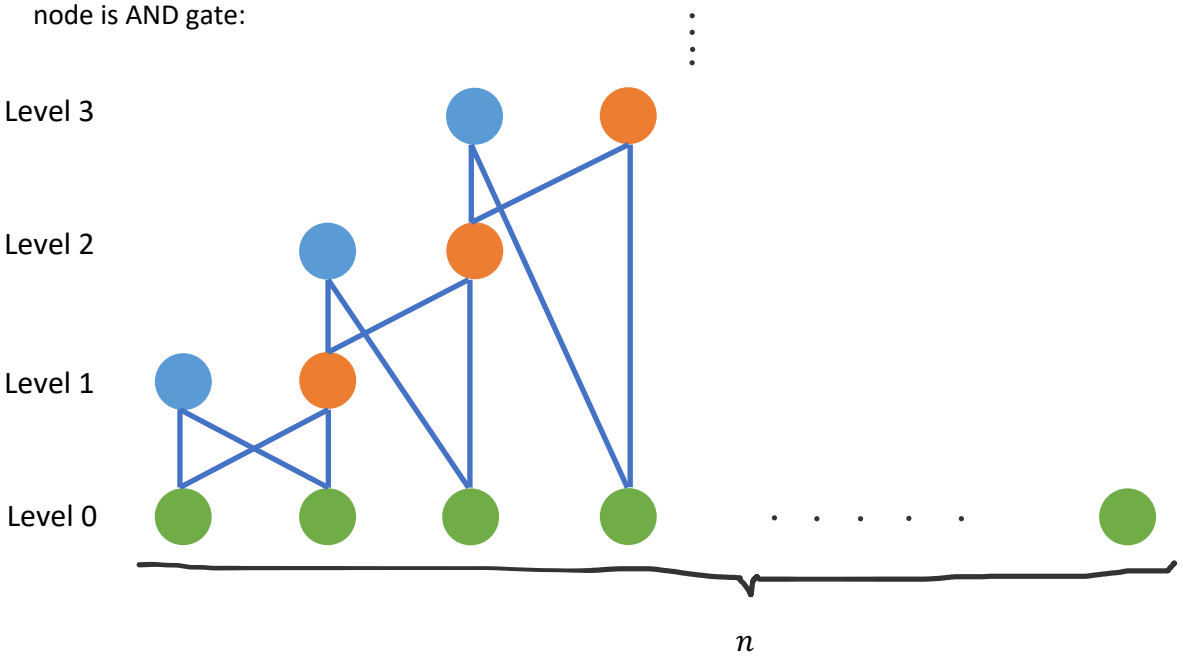
We see that x_1-new is only 1 when EITHER of the old values is 1, so

$$x_1\text{-new} = (x_1\text{-old}) \vee (x_2\text{-old})$$

And that x_2-new is only 1 when BOTH of the old values is 1, so

$$x_2\text{-new} = (x_1\text{-old}) \wedge (x_2\text{-old})$$

So, the lowest few levels look like this, where green node is input, blue node is OR gate, orange node is AND gate:



For each “line comparison”, there are $n-1$ levels (each level is comprised of a blue and orange node). In total (excluding the green level), there are $n(n-1)$ levels to sort the inputs. The output is the $\left\lceil \frac{n}{2} \right\rceil$-th bit of the sorted output, i.e. the $\left\lceil \frac{n}{2} \right\rceil$-th blue node from the left of the last n layers.

This circuit is $O(n^2)$ in size because each level adds four wires/edges, and there's $O(n^2)$ levels, $O(4n^2) = O(n^2)$. QED.