Code Jam 2014 - Round 1A

Analysis: Charging Chaos

This problem is looking for the minimum number of switches that need to be flipped so that all devices can be charged at the same time. All devices can be charged at the same time if each of the outlets can be paired with exactly one device and vice versa. An outlet can be paired with a device if both have the same electric flow after flipping some switches.

We observe that flipping the same switch two times (or an even number of times) is equivalent to not flipping the switch at all. Also, flipping the same switch an odd number of times is equivalent to flipping the switch exactly once. Thus, we only need to consider flipping the *i-th* switch once or not at all.

Letâ€™s define a **flip-string** of 0s and 1s of length **L** as a way to flip the switches. If the *i-th* character of the string is 1, it means we flip the *i-th* switch (otherwise we do not flip the *i-th* switch). A flip-string is **good** if the resulting outletsâ€™ electric flows (after the flips) allow all the devices to be charged at the same time. We can check whether a flip-string is good by flipping the bits in the outletsâ€™ electric flow according to the flip-string and then checking whether each deviceâ€™s electric flow can be matched exactly to one outlet with the same electric flow and vice versa. We can use hashing to perform the check and thus the complexity to check whether a flip-string is good is O(**LN**). Note that we can (optionally) encode the flip-string into a 64-bit number and reduce the complexity of the check to O(N).

Brute force algorithm:

A naive brute force solution is to try all the $\mathbf{2^L}$ possible flip-strings and check whether it is a good flip-string and keep the one that has the least number of 1s in the string (i.e., it has the minimum number of flips). The time complexity of this algorithm is O($\mathbf{2^L}$ * **LN**). For the small dataset where the maximum for **L** is 10, this brute force algorithm is sufficient. However, it is really slow for the large dataset where **L** can be as large as 40.

Better algorithm:

To improve the naive brute force algorithm we need two more observations. First is that there are only a few good flip-strings. If we can efficiently generate only the good flip-strings, we can improve the algorithm complexity significantly. The second observation is that given a deviceâ€™s electric flow (of **L** bits) and an outletâ€™s electric flow (of **L** bits), we can generate a flip-string that will flip the outletâ€™s electric flow such that it matches the deviceâ€™s electric flow. There are only $\mathbf{N^2}$ possible pairs of devices and outlets. Thus, only $\mathbf{N^2}$ flip-strings need to be generated. Note that the generated flip-strings **may or may not be** a good flip-string. However, all other flip-strings that are not generated by any pair is guaranteed to **not** be a good flip-string. With these observations, we can reduce the complexity of the naive algorithm down to O($\mathbf{N^2}$ * **LN**), which is fast enough for the large input.

The last (optional) observation we can make is that since a device must be plugged in to exactly one outlet, we can further reduce the number of possible flip-strings from $\mathbf{N^2}$ down to **N**. The **N** possible flip-strings can be generated by pairing any one device with the **N** outlets.