Codeforces Problem 1909E

> **Problem.** Suppose we have n lamps labelled 1 to n which are all initially turned off. We may then pick a set of indices S of size at least 1, with a set of m constraints for which if $u_i \in S$ then $v_i \in S$. For each index i in the chosen S, all lamps which have indices that are multiples of i are toggled.
>
> Determine any choice (if there exists one) of S such that, after all lamps have been toggled according to the indices, there are at most $\lfloor n/5 \rfloor$ lamps that are on.

Solution. We shall first get some simple cases out of the way that fall out from direct observation or a bit of playing around with the problem.

Claim. For $n < 5$, it is always impossible.

Proof. This follows trivially from the fact that, under these conditions $\lfloor n/5 \rfloor = 0$. This is impossible to achieve when we are picking indices from the set. (Actually I'm not sure the best way to prove this though). ■

Claim. For $n \geq 20$, it is always possible to achieve such a configuration, and we may simply choose all indices.

Proof. For this, we shall show that the number of lamps that are on after choosing all indices is strictly less than or equal to $\lfloor n/5 \rfloor$ for all $n \geq 5$.

Let $f(n)$ denote the number of lamps on after choosing the set $S = \{1, 2, \ldots, n\}$. Observe that

$$\begin{aligned} f(n) &= f(n-1) + \sum_{d \mid n} 1 \bmod 2 \\ &= f(n-1) + d(n) \bmod 2, \end{aligned}$$

where $d(n)$ denotes the number of divisors of n. If one recalls that for some number $k = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$,

$$d(k) = (e_1 + 1)(e_2 + 1) \cdots (e_n + 1),$$

we can see that $d(n)$ is only odd when all exponents are even. This means that $d(n) \bmod 2$ is the same as the indicator function for whether n is a perfect square. Since $f(1) = 1$, we can solve this recurrence to see that

$$f(n) = \lfloor \sqrt{n} \rfloor,$$

which means that this problem is trolling, but also since $\lfloor\sqrt{n}\rfloor < \lfloor n/5\rfloor$ for $n \geq 20$, we can essentially ignore this case when problem solving. ■

This tells us that the only bounds for which we have to actually solve this problem are when $5 \leq n < 20$. This means that we can actually use an $O(2^n)$ solution or even perhaps $O(n2^n)$, which is funny if not a little stupid.

We shall try for an $O(n2^n)$ solution when $n < 20$. For each index i, precompute a bitmask, denoted $\mathsf{bitmask}(i)$, that corresponds to the lights toggled after including i in the set. One may see that for some set of indices $S = \{i_1, i_2, \ldots, i_k\}$, the resulting lamps chosen are given by the following bitmask:

$$\mathsf{bitmask}(S) := \bigoplus_{j=1}^{k} \mathsf{bitmask}(i_k),$$

where $\oplus$ denotes bitwise XOR. Thus, the number of lamps toggled are the number of ones in the resulting value of $\mathsf{bitmask}(S)$. Using the fact that having a running bitwise XOR and counting the number of ones bits in a number are theoretically both $O(1)$ operations, this solution part runs in $O(2^n)$ time, as we simply just iterate through all subsets Since the problem also gives us the additional condition for elements to include, though, we must handle this and there are at worst $n - 1$ pairs for each element, so the final running time is $O(n2^n)$.

Actually, the above solution isn't quite what I have coded (there are some complications in the chaining of conditions which required directed graphs, dfs, and dp, but I'll write about this later).