Это тот редкий случай, когда решение проще записать в виде псевдокода.

FindBug(Node* root):

Если граф — дерево, то бракованное реле находится перед первой (от корня) не горящей лампочкой.
Если такого нет, отключаем все ребра графа. **Return**.
Отключаем все ребра, выходящие из **root**.
for (ребра **j**, выходящие из корня):
  Включаем **j**
  Его другой конец обозначим через **new_root**.
  FindBug(new_root). Если успешно — **return**.
  Выключаем **j**.

Важное замечание. После выхода из **FindBug** все ребра в обработанном подграфе оказываются отключены и не мешают в дальнейшем.

Оценим время работы алгоритма. Пусть S_n — максимальная сложность **FindBug** для графа с n ребрами. Пусть, далее, $j_1 \dots j_t$ — все ребра, выходящие из источника. В худшем случае нам пришлось включить и выключить каждое из ребер $j_1 \dots j_t$ и запустить процедуру **FindBug** для каждого из них.
Тогда

$$S_n \leqslant 2t + S_{k_1} + S_{k_2} + \dots + S_{k_t},$$

где $k_1 + k_2 + \dots + k_t = n - t$.
Нетрудно видеть, что $S_1 = S_2 = 0$, $S_3 = 1$. В любом случае, $S_m \leqslant 2m$ при $m = 1, 2, 3$.
Примем это за рабочую гипотезу. По индукции тогда можно заключить, что

$$\begin{aligned} S_n &\leqslant 2t + S_{k_1} + S_{k_2} + \dots + S_{k_t} \leqslant 2t + 2k_1 + 2k_2 + \dots + 2k_t \\ &= 2t + 2(k_1 + k_2 + \dots + k_t) = 2t + 2(n - t) = 2n, \end{aligned}$$

что и требовалось доказать.