OPERANDO CIRCUITOS

AUTOR: LAUTARO LASORSA

Solución Esperada

Lo primero que podemos hacer es convertir el problema en uno de conjuntos. Siendo cada **entrada** un conjunto (conformado por los tiempos en los que se envía corriente por esta entrada), luego cada una de las compuertas se convierte en una operación entre conjuntos.

Las compuertas **AND** se convierten en la **intersección de conjuntos**, las compuertas **OR** en la **unión de conjuntos** y las compuertas **XOR** en la **diferencia simétrica de conjuntos**. Finalmente, las **luces LED** simplemente indican responder el conjunto que devuelve alguna de las compuertas.

Para resolver el problema, lo que voy a querer es que cada una de estas operaciones pueda realizarse solamente iterando por los elementos del conjunto de menor tamaño. Notar que la mayor cantidad de operaciones se realiza si todas las compuertas son **OR**, ya que conservan todos los elementos de ambos conjuntos. Por tanto, acotar la cantidad de operaciones en este caso acota la cantidad de operaciones realizadas en el problema.

Para acotar este caso, noto que si itero un elemento solamente cuando pertenece al conjunto de menor tamaño, después de la operación el tamaño del conjunto al que pertenece se duplica (como mínimo). Como ningún conjunto puede tener más de los **M** elementos iniciales, entonces el tamaño se puede duplicar hasta $\log_2(M)$ veces. Por tanto, cada uno de los **M** elementos puede ser visto hasta $\log_2(M)$ veces. En total, la complejidad resulta en $O(M*\log_2(M)*O(T))$, donde **O(T)** es la complejidad asociada a iterar cada elemento.

Ahora vemos como realizar cada una de las 3 operaciones en una complejidad dependiente del tamaño del elemento más chico:

- **AND:** Debo crear un nuevo conjunto, inicialmente vacío. Luego, itero el conjunto más pequeño y para cada conjunto, si pertenece también al más grande lo incorporó al nuevo conjunto. La respuesta queda almacenada en el nuevo conjunto.

- **OR:** Itero los elementos del conjunto más chico y los agrego al conjunto más grande. La respuesta queda almacenada en el conjunto más grande.

- **XOR:** Itero los elementos del conjunto más chico y, si ya pertenecen al conjunto más grande, los elimino, y si no, los agrego. La respuesta queda almacenada en el conjunto más grande.

Notar que, a nivel implementativo, hay que evitar copiar los conjuntos para realizar las operaciones. Para esto pueden usarse índices a un array de conjuntos o bien punteros, según preferencia del concursante (la solución oficial utiliza punteros)

La **O(T)** va a depender de si utilizamos **set** o **unordered_set** (o sus análogos en otros lenguajes) para representar los conjuntos, ya que la **O(T)** es básicamente el costo de ver si un elemento pertenece a otro conjunto, agregarlo o eliminarlo. Igual, notar que por la implementación interna de estas estructuras de datos el tiempo de ejecución puede ser similar en ambos casos.

Finalmente, la complejidad resultante puede ser $\mathbf{O(M*\log^2(M))}$ o $\mathbf{O(M*\log(M))}$.

Soluciones Parciales

En todas las soluciones parciales se toma que estamos trabajando con un problema de conjuntos y operaciones sobre conjuntos, como se explicó para la solución esperada.

Para la subtarea **1<=N,M<=500** se espera que cada conjunto sea representado con vectores, y que la búsqueda de si un elemento pertenece o no a un conjunto se realice en tiempo lineal, pudiendo llegar a costar $\mathbf{O(M^2)}$ cada operación. También abarca soluciones que copien los conjuntos para utilizarlos, o que representen los conjuntos con **set** o **unordered_set**, pero no presten atención al tamaño de los conjuntos para optimizar la cuenta.

La complejidad de estas soluciones va en el orden de $\mathbf{O(M^3)}$ a $\mathbf{O(M^2)}$ (aunque con alta constante, comparable a $\mathbf{O(M^2*\log(M))}$)

Las 3 subtareas donde solo se toma una de las compuertas son para evaluar de forma independiente la resolución de las mismas, viendo que aunque son similares cada una es un problema independiente de las otras 2.

OPERANDO CIRCUITOS

AUTOR: LAUTARO LASORSA

La subtarea donde **1<=input$_{i,j}$<=5.000** está planteada para que los competidores implementen las operaciones utilizando **bitset** de C++ o máscara de bits. En este caso la complejidad queda **O(N*(5.000/64)),** siendo (5.000/64) el costo de las operaciones utilizando **bitset.** (puede variar ligeramente según la implementación y el compilador, pero en todos los casos es de un orden aceptable)