Ada Byron 2024
Regional de Andalucía - I Edición

Soluciones

Pablo Dávila Herrero
Pablo Reina Jiménez
Kenny Jesús Flores Huamán

Notas

- Los problemas aparecen ordenados por dificultad.
- Puedes encontrar los enunciados y las soluciones a los problemas en este repositorio del Club de Algoritmia de la Universidad de Sevilla.

PROBLEMA F

Problema F

- Equipos que lo han intentado: 29
- Submissions: 53
- Porcentaje de aciertos: 50,94%

Problema F

- Para cada número de entrada distinto de 0:
  - Separamos los dígitos del número
  - Elevamos cada dígito a la potencia total del número total de dígitos
  - Sumamos estos resultados
  - Si la suma de los dígitos elevados es igual al número original, entonces mostramos “Segura”. En caso contrario, mostramos por pantalla “Insegura”

PROBLEMA J

Problema J

- Equipos que lo han intentado: 23
- Submissions: 53
- Porcentaje de aciertos: 35,85%

Problema J

- Recorremos toda la lista y almacenamos en un diccionario como clave cada número y como valor el número de repeticiones.

- Ordenamos por número de repeticiones y en caso de empate por devolvemos el menor valor.

PROBLEMA K

Problema K

- Equipos que lo han intentado: 20
- Submissions: 53
- Porcentaje de aciertos: 28,30%

Problema K

- Creamos un diccionario con con clave cada una de las letras y valor su número de apariciones
- Para cada palabra vemos si para cada una de sus letras el valor existe en el diccionario y es menor o igual que su clave asociada
- Nos vamos quedando con la palabra más larga que cumpla la condición anterior
- En caso de empate nos fijamos en el orden lexicográfico

PROBLEMA D

Problema D

- Equipos que lo han intentado: 11
- Submissions: 27
- Porcentaje de aciertos: 29,63%

Problema D

- Comenzamos con array indicando si hemos comprado el cromo i-ésimo. Al comienzo todo Falso.
- Para cada sobre de cromos calculamos cuánto hubiera costado dejar de comprar sobres en ese momento y comprar cromos individuales. Podemos actualizar los cromos comprados actualizando el índice de la lista menos 1.
- Devolvemos el mínimo de todos los valores calculados
- OJO: Llevar la suma acumulada del precio restando el precio de los cromos que ya tienes, en lugar de sumar en cada paso lo que te costaría.

PROBLEMA I

Problema I

- Equipos que lo han intentado: 22
- Submissions: 71
- Porcentaje de aciertos: 8,45%

Problema I

- Si para cada día recorremos hacia atrás para contar los días anteriores nos encontramos con una complejidad 0(N^2) en el peor de los casos.
- Como esto nos va a dar TLE por parte del juez online, hay que buscar una forma de poder resolverlo en una menor complejidad.
- Una estrategia más eficiente sería utilizar una pila (stack) que me permita mantener un registro de los días anteriores con ventas menores o iguales al día actual.

Problema I

- Crearíamos una lista de longitud N (número de días) inicializada en 1. Empleamos un stack vacío para rastrear los índices de días anteriores relevantes.
- Para cada índice i en la lista de ventas:
  - Mientras la pila no esté vacía y la venta en el índice superior de la pila sea menor o igual a la venta del día actual, sacamos el elemento del stack.
  - Calculamos el ARE para el día actual i
    - Si la stack no está vacía, are[i] se calcula como i - stack[-1] + 1, representando días consecutivos anteriores (incluyendo el día actual) con ventas menores o iguales.
    - Si la stack está vacía, are[i] se establece como 1, indicando que es el primer día.
  - Agregamos el índice i al stack para futuros cálculos

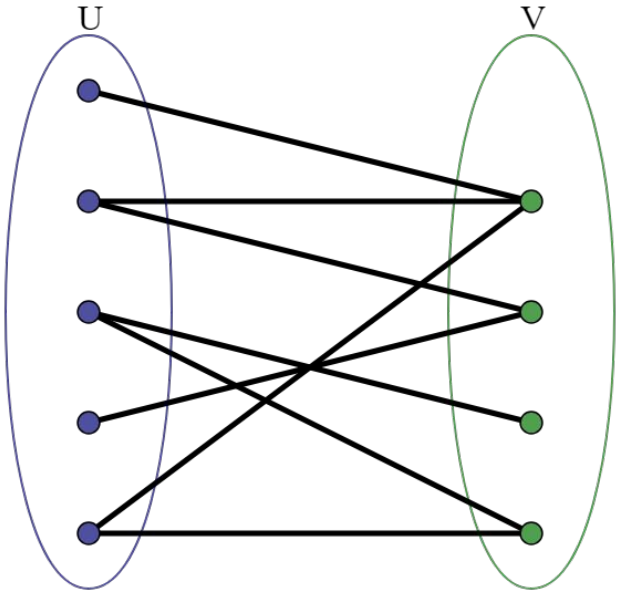
PROBLEMA G

Problema G

- Equipos que lo han intentado: 7
- Submissions: 16
- Porcentaje de aciertos: 18,75%

Problema G

Grafo bipartito



Problema G

- Creamos un grafo en el que cada nodo será un jugador y cada arista unirá a dos rivales. Trataremos de pintar el grafo con dos colores (0 y 1) de tal forma que los rivales estén en distintos equipos.
- Comenzamos con un nodo al azar y lo coloreamos de color 0 (o 1 da igual) y asignamos a sus adyacentes el opuesto. Mediante DFS vamos asignando a los nodos vecinos los colores opuestos. Una vez finalizamos eliminamos los nodos coloreados.
- Si hubiera más de una componente conexa (aún quedan nodos) seleccionamos otro al azar e iteramos
- Si en algún momento hay dos nodos adyacente del mismo color paramos.

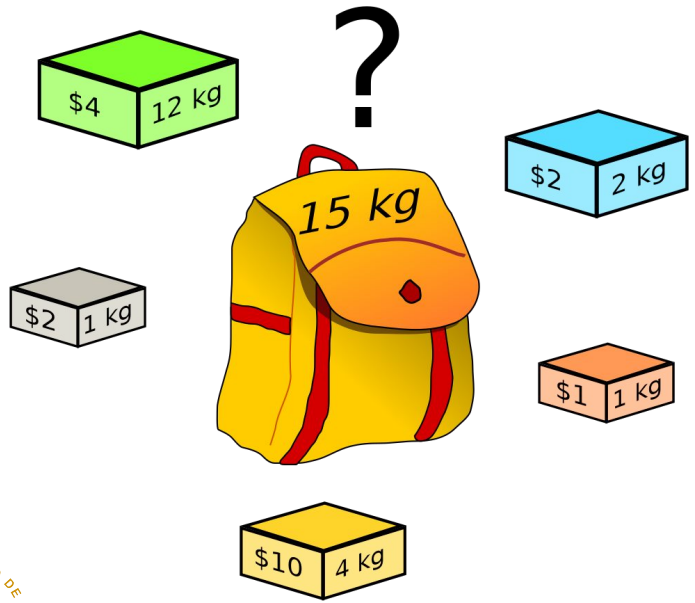
PROBLEMA A

Problema A

- Equipos que lo han intentado: 6
- Submissions: 16
- Porcentaje de aciertos: 6,25%

Problema A

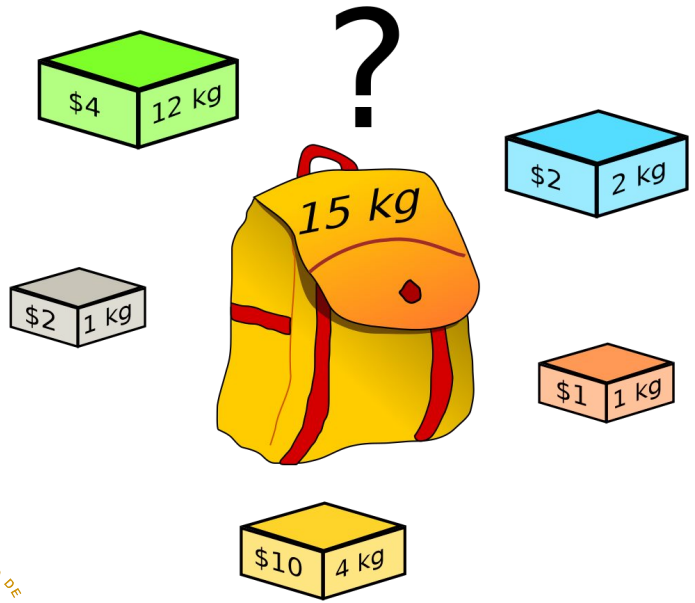
El problema de la mochila



- La mochila tiene una capacidad máxima (gramos de masa)
- Tenemos M tipos de objetos (M tipos de chorizos diferentes)
- Cada objeto tiene un peso (gramos de masa necesaria)
- Cada objeto tiene un valor (precio)

Problema A

El problema de la mochila



¿Cómo conseguimos que los objetos de la mochila sumen el máximo valor?

Programación dinámica

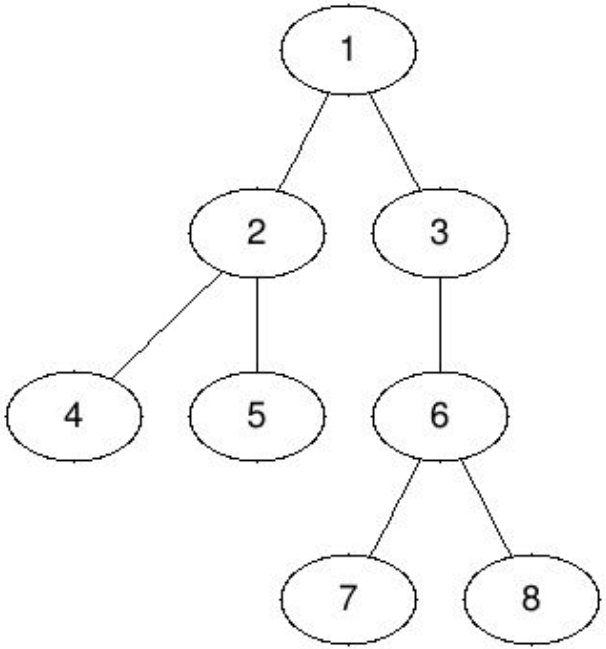
PROBLEMA C

Problema C

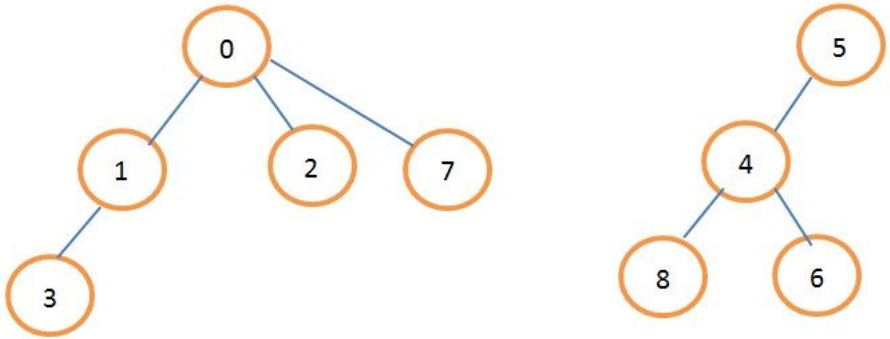
- Equipos que lo han intentado: 3
- Submissions: 7
- Porcentaje de aciertos: 28,57%

Problema C

Búsqueda DFS



Componentes conexas



Problema C

- Creamos un grafo en el que cada nodo será un campista y cada arista representa una relación de amistad.
- Comenzamos por un nodo cualquiera, recorremos mediante DFS toda su componente conexa y los añadimos a todos a la misma tienda. Eliminamos los nodos ya asignados.
- Realizamos el anterior paso para un nodo nuevo hasta quedarnos sin nodos.
- El número de tiendas necesarias será al final el número de componentes conexas.

Problema C

- Para calcular el número de posibles configuraciones de delegado basta con multiplicar el número de campistas en cada tienda.
- Ej: Si tenemos 2 4 5 campistas en cada tienda, en la primera tenemos 2 elecciones para delegado, en la segunda 4 y la última 5, haciendo uso de combinatoria vemos que hay 2*4*5=40 posibles configuraciones de delegado.

PROBLEMA E

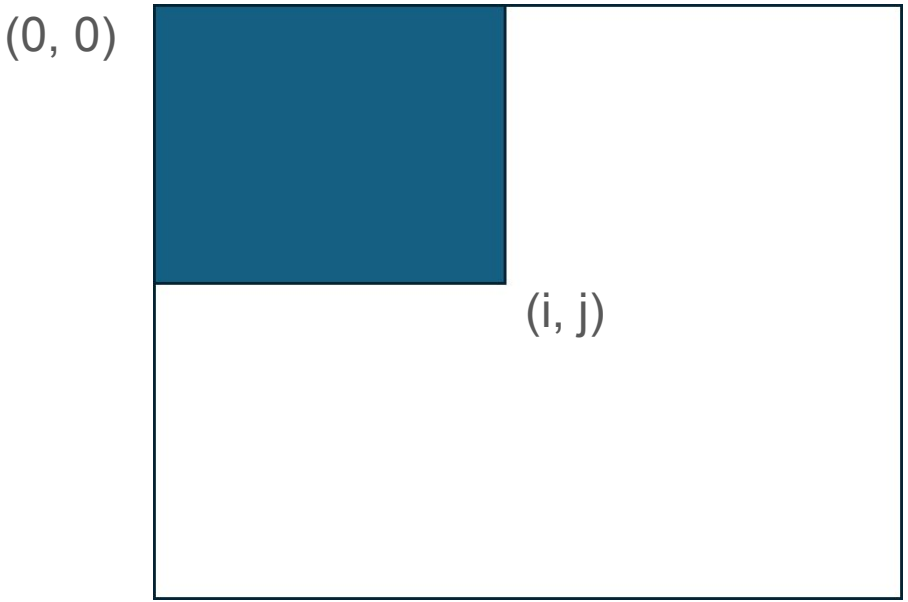
Problema E

- Equipos que lo han intentado: 16
- Submissions: 27
- Porcentaje de aciertos: 7,41%

Problema E

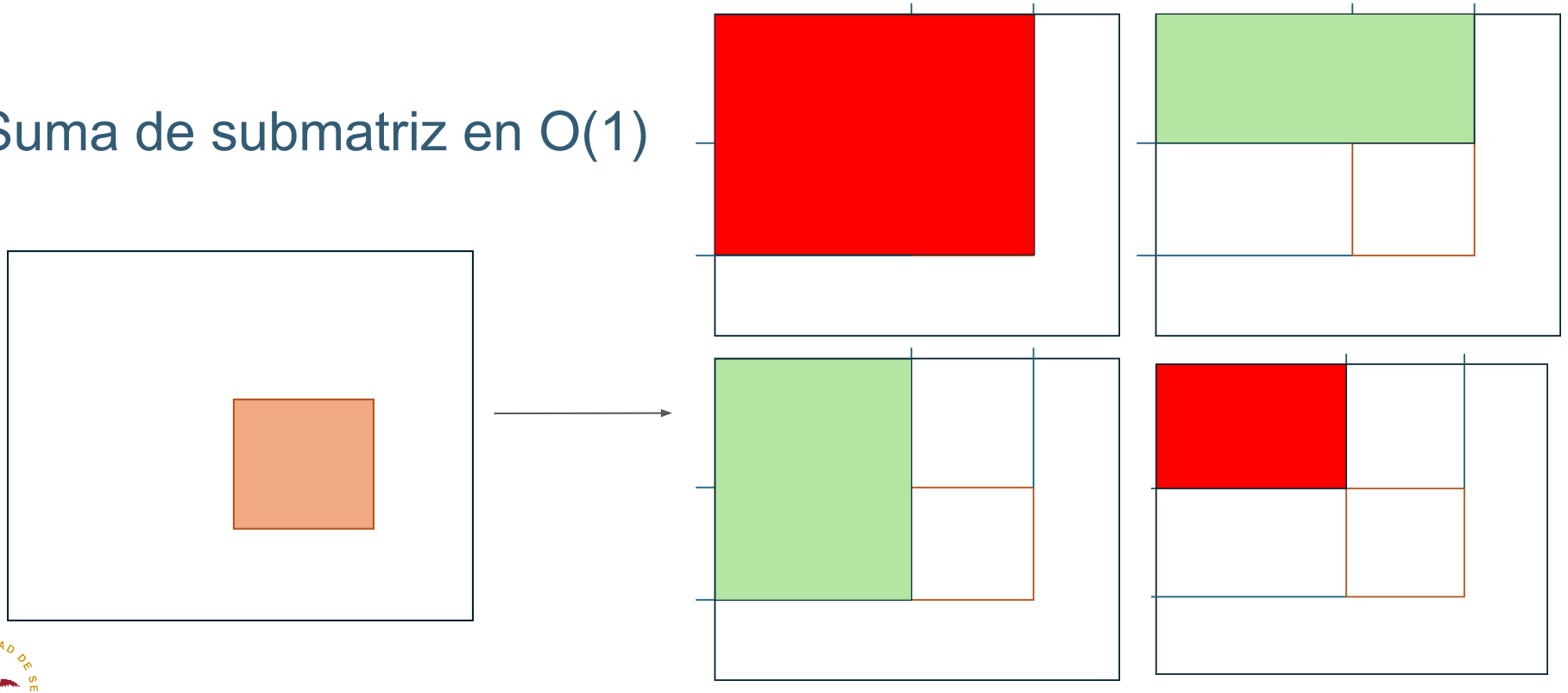
Número muy elevado de consultas ~ 50.000

Tamaño de matriz pequeño ~ 1000 x 1000



Problema E

Suma de submatriz en O(1)



Problema E

- En primer lugar calculamos todas las sumas de submatrices que comienzan en el (0, 0) y acaban en (i, j). Denotamos este cálculo como submatriz(i, j)
- Una vez con esto para calcular la matriz que comience en (a, b) y acabe en (c, d) calculamos submatriz(c, d) - submatriz(a-1, d) - submatriz(c, b-1) + submatriz(a-1, b-1)

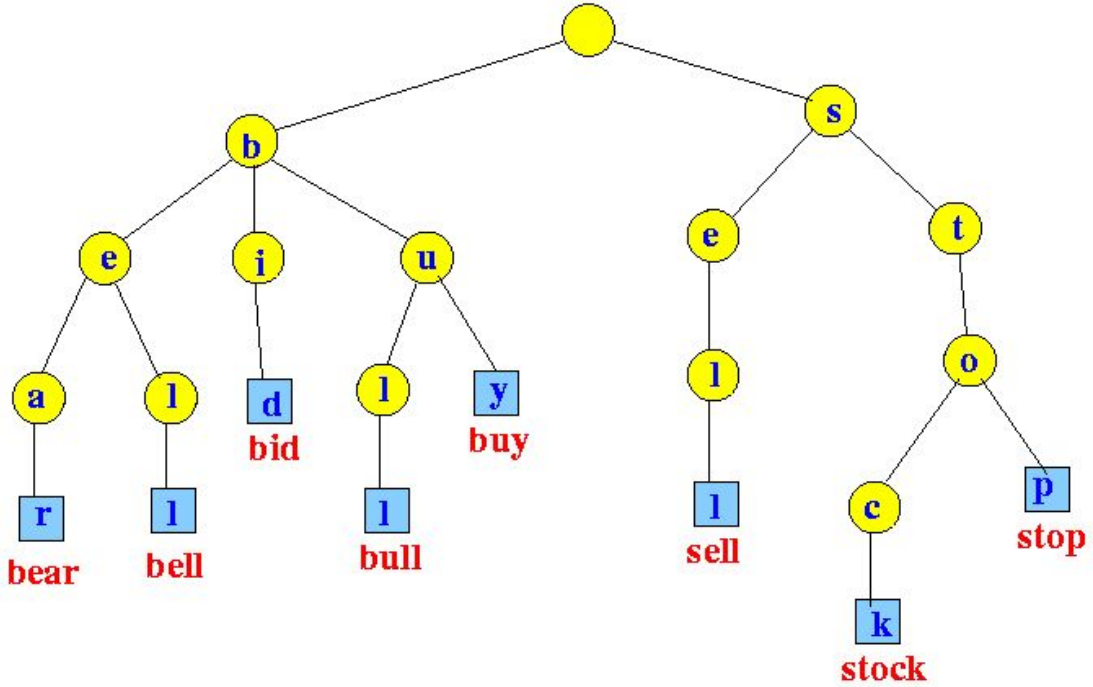
PROBLEMA B

Problema B

- Equipos que lo han intentado: 7
- Submissions: 11
- Porcentaje de aciertos: 9,09%

Problema B

Estructura trie



Problema B

- Crear un trie para almacenar los nombres de los buenos. Con esto podremos comprobar rápidamente si un prefijo es válido o no.
- Para cada malo, utilizamos el trie para calcular su mínimo prefijo que no es también prefijo de ningún bueno.
- Almacenamos los prefijos en un conjunto para evitar duplicados.
- Ordenamos los resultados por orden alfabético.

PROBLEMA H

Problema H

- Equipos que lo han intentado: 1
- Submissions: 3
- Porcentaje de aciertos: 0

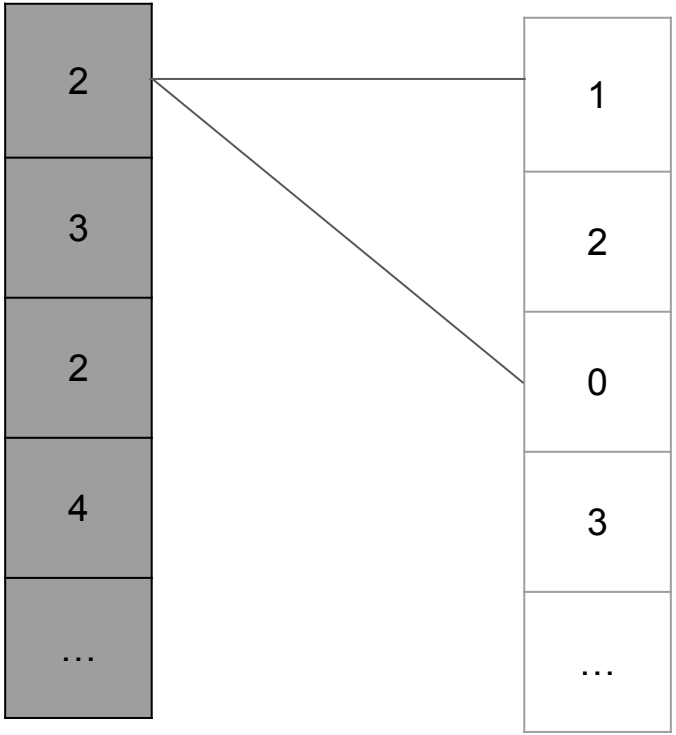
Problema H

2	1	3	2	2
0	4	3	2	3
0	4	0	0	2
3	1	4	0	0
1	1	4	4	4

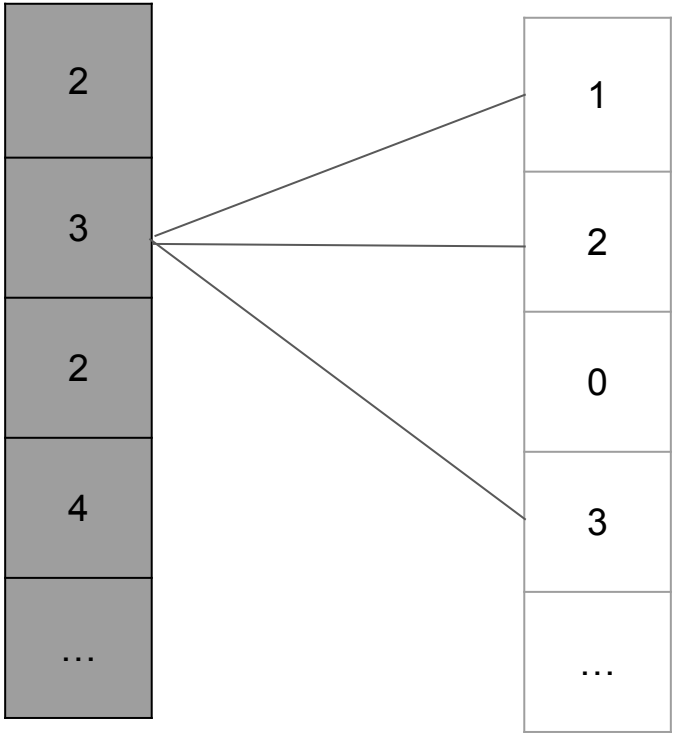
Problema H

2	1	3	2	2
0	4	3	2	3
0	4	0	0	2
3	1	4	0	0
1	1	4	4	4

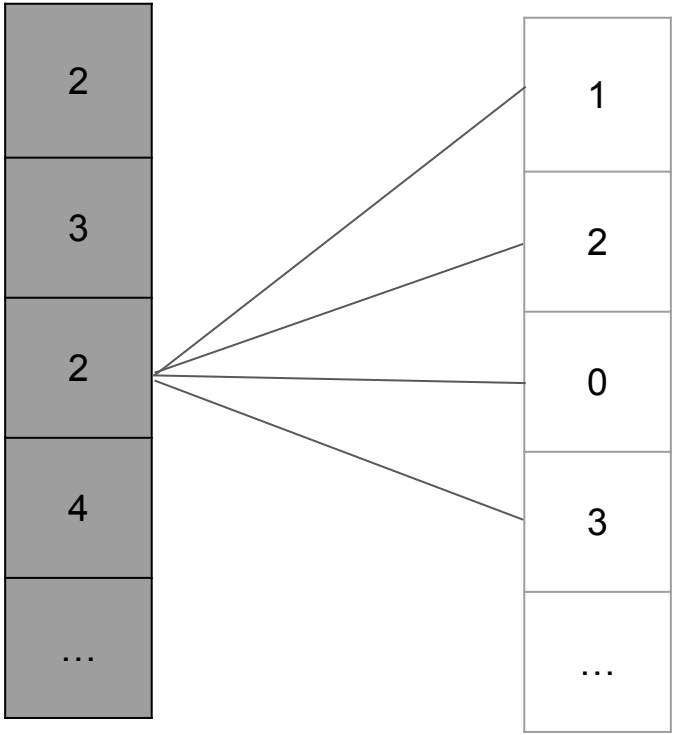
Problema H



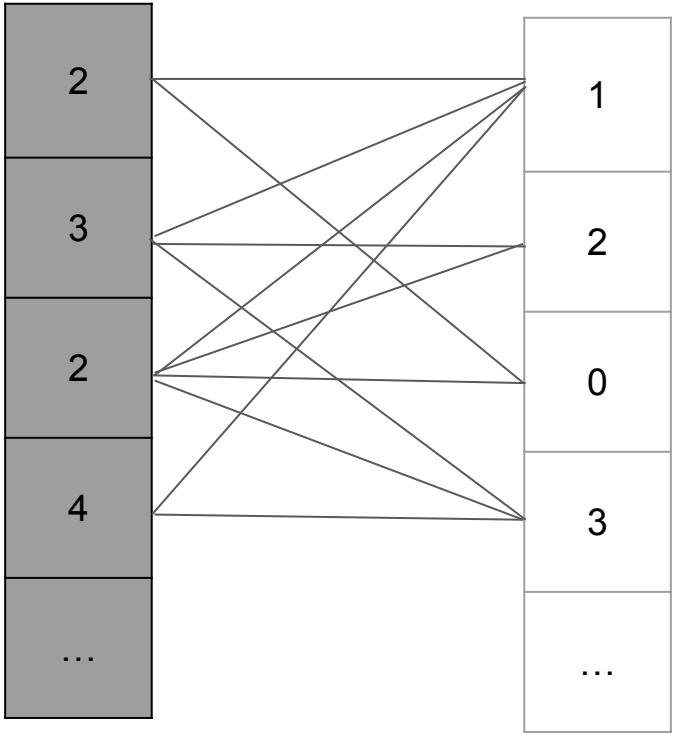
Problema H



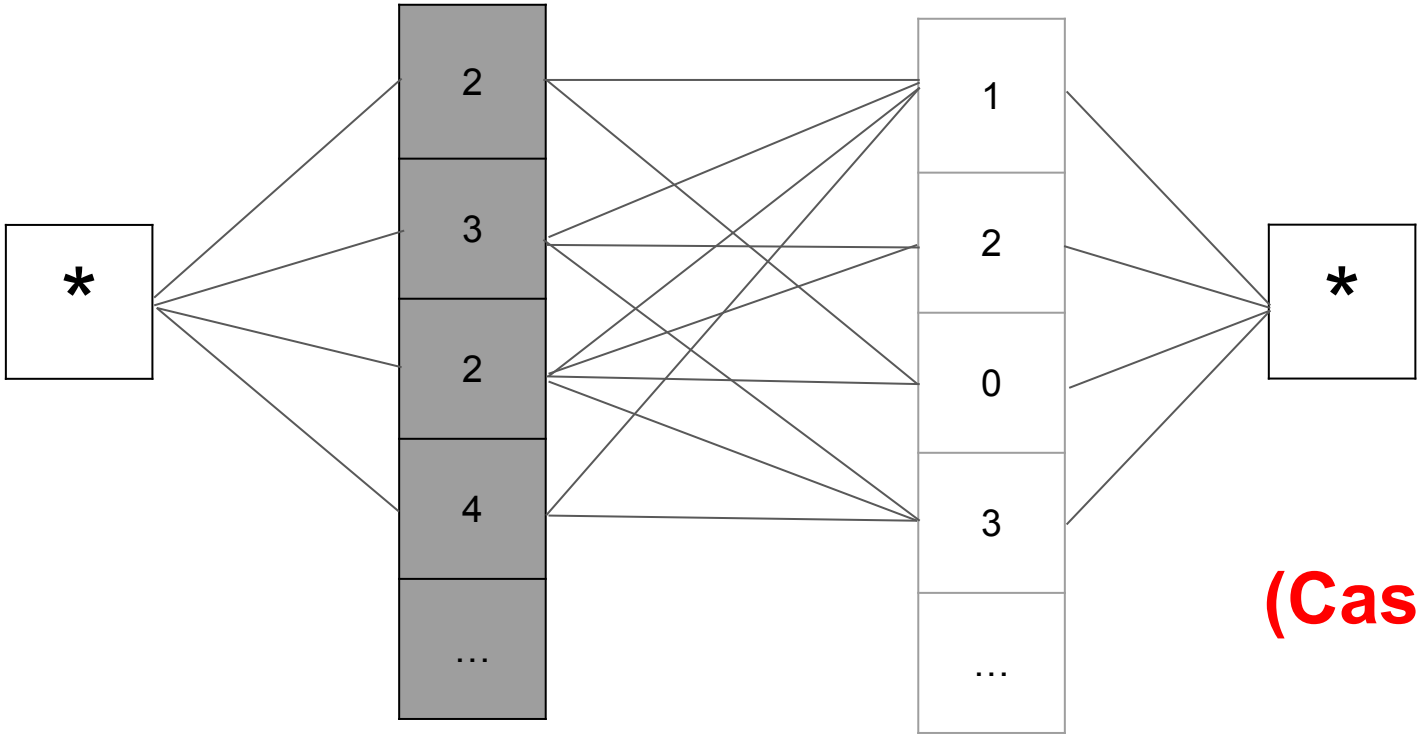
Problema H



Problema H

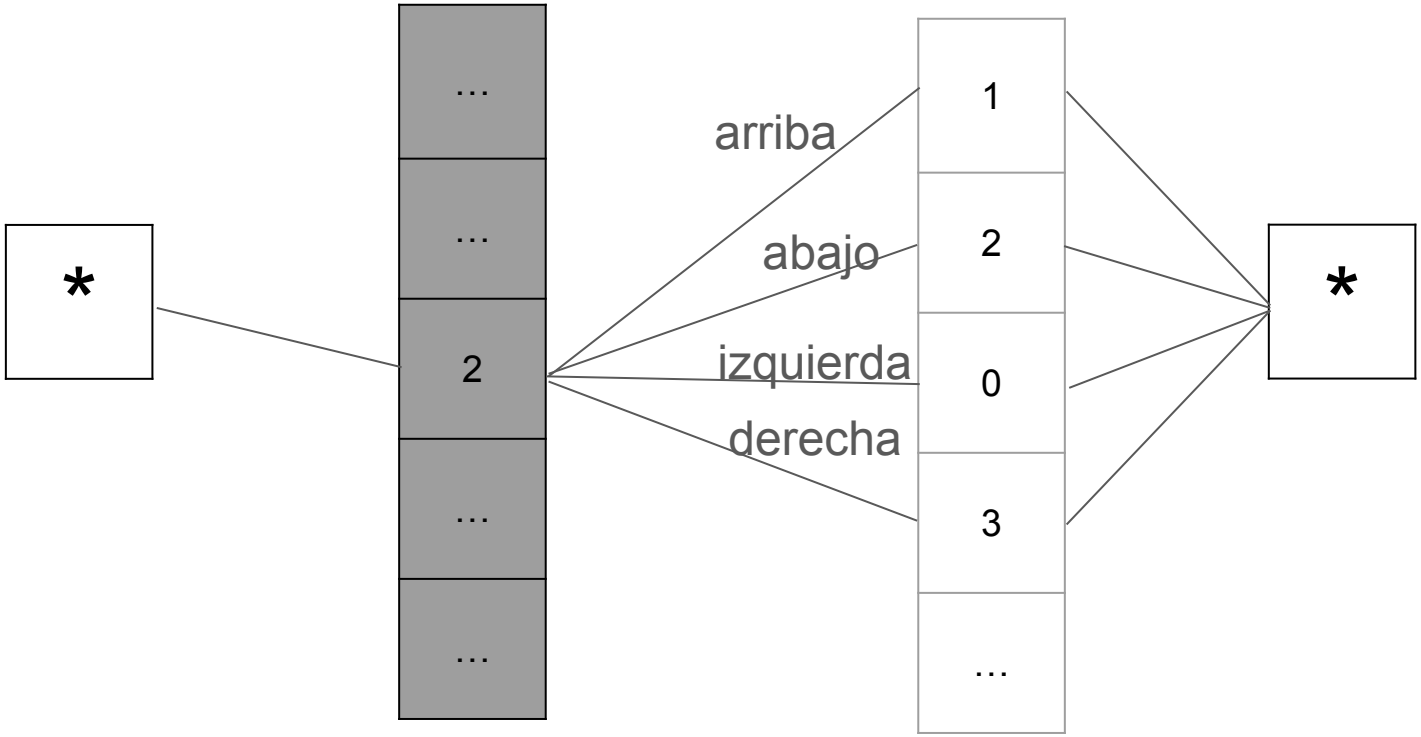


Problema H

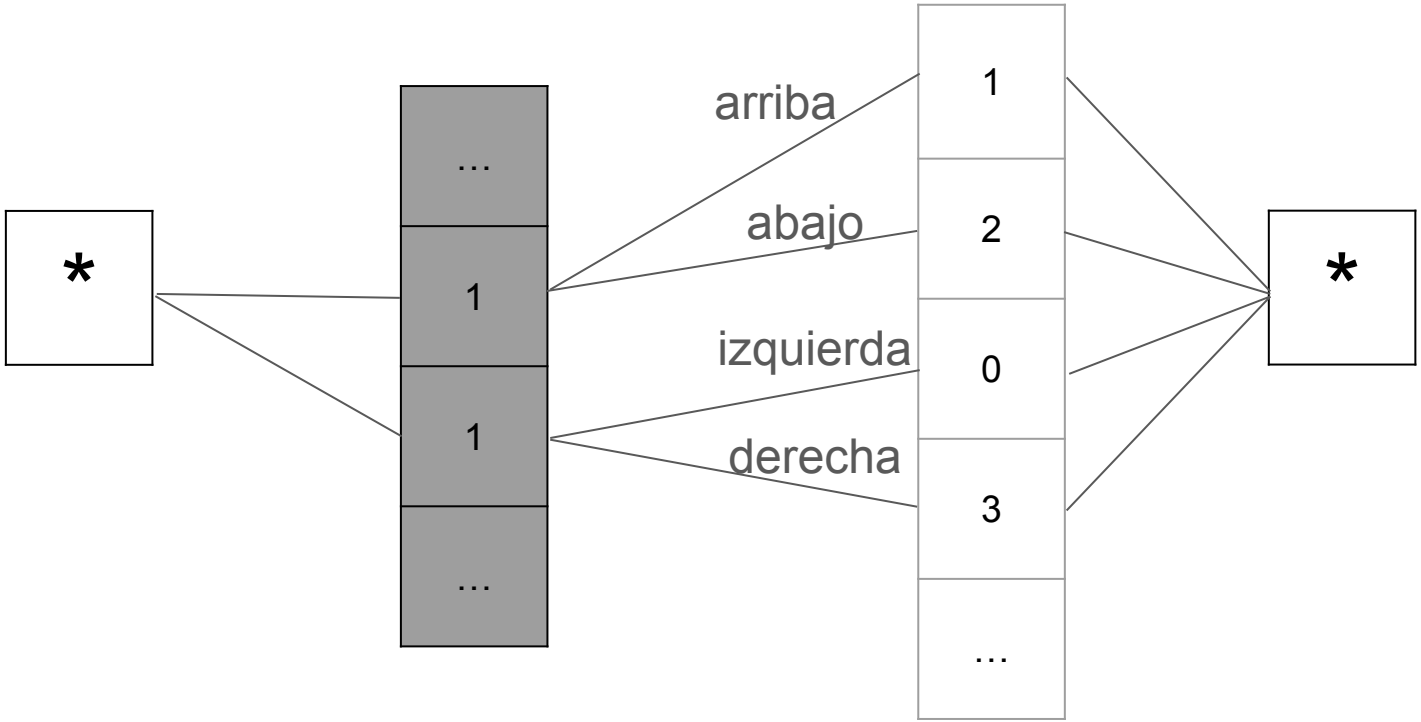


(Casi)

Problema H



Problema H



¡MUCHAS GRACIAS

POR PARTICIPAR!