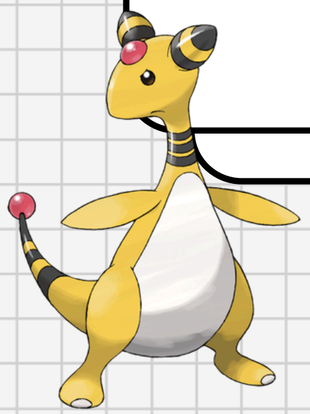
Ayudantía 10

Funciones y tablas de hash





Definición

Diccionarios

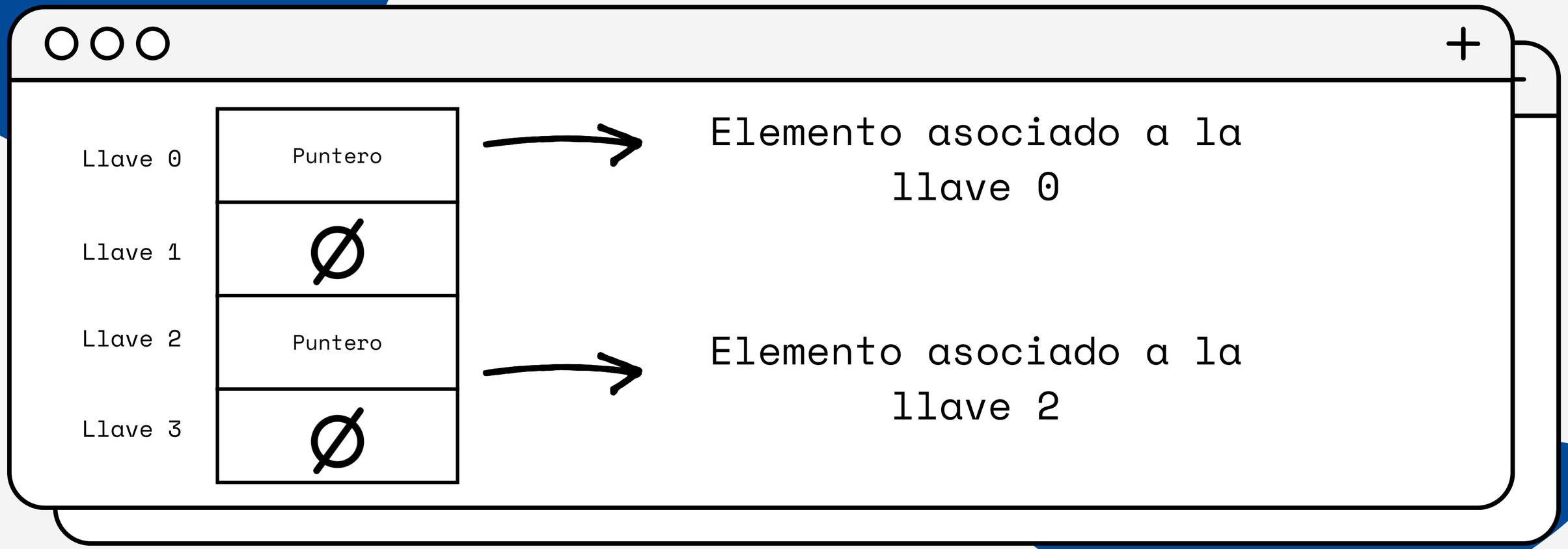
EDD que permite las siguientes operaciones:

- Asociar un valor (v) a una llave (k)
- Actualizar el valor asociado a una llave
- Obtener el valor asociado a una llave
- Eliminar una llave, y por tanto su valor asociado

Su gracia principal es la **búsqueda eficiente de un par (llave,valor)**

Ejemplo

Diccionarios



Definición

Funciones de hash

Es una función h que cumple con tomar un espacio de tamaño K y llevarlo a un espacio de tamaño M, donde idealmente M<K.

Como M no es necesariamente menor a K, la funció **no es necesariamente inyectiva.**

Y vamos a buscar que M << K.

Ejemplos

Funciones de hash

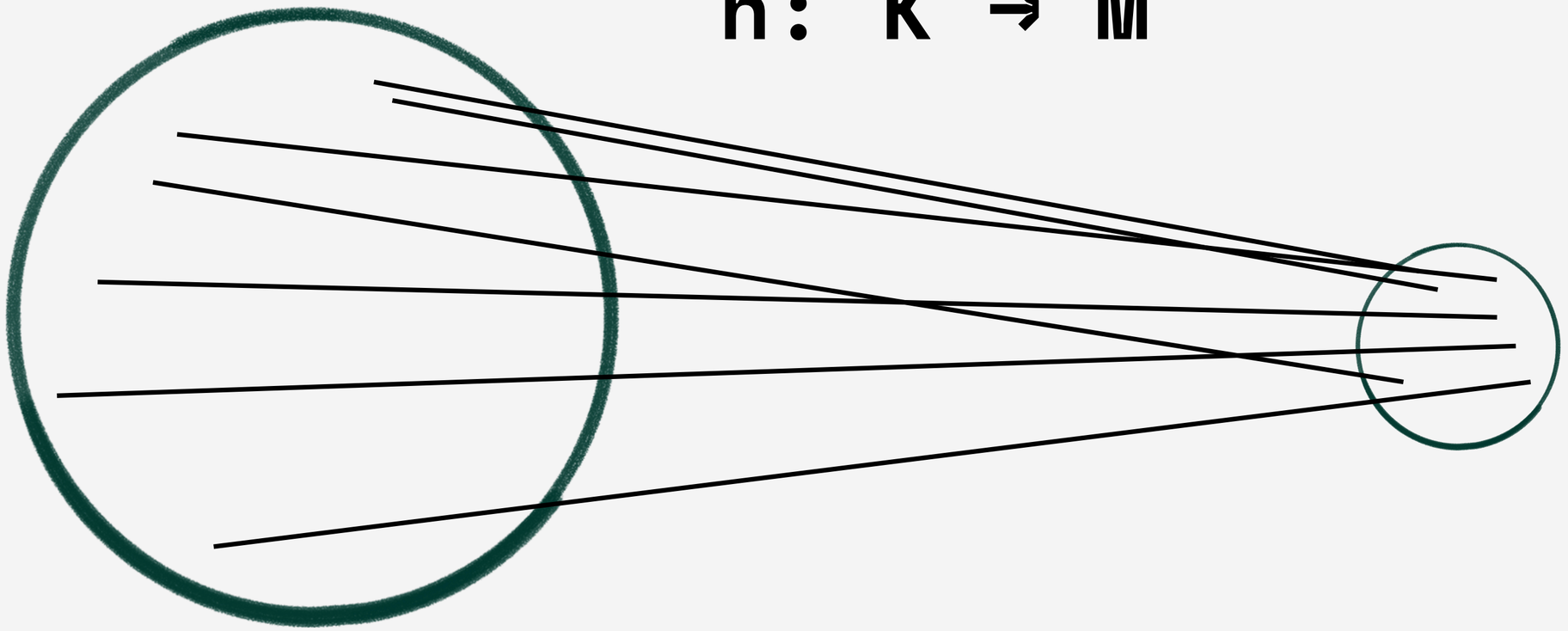
- f(x)= x mod n
- f(x)= x // 10
- f(x)= x

Estas 3 funciones toman un espacio y lo llevan a otro espacio que no es necesariamente menor.

Rep.Gráfica

Funciones de hash

$$h: K \rightarrow \mathbb{M}$$



Las funciones de hash sirven para convertir espacios muy grandes en espacios más manejables.

Definición

Tablas de hash

Dado un espacio de llaves K, vamos a tener la tabla de hash A, de tamaño M (con m elementos), donde vamos a pedir la existencia de una función h: K → [0...m-1].

Esto es equivalente a pensar en que quiero guardar todos los elementos de K en A. Para eso le asigno un lugar a cada k, y para que tenga sentido en A sólo puede ser de 0 a m-1.

Ejemplo

Tablas de hash

Digamos que tenemos el espacio de llaves del 1 al 1M. Las cuales están asociadas a nombres.

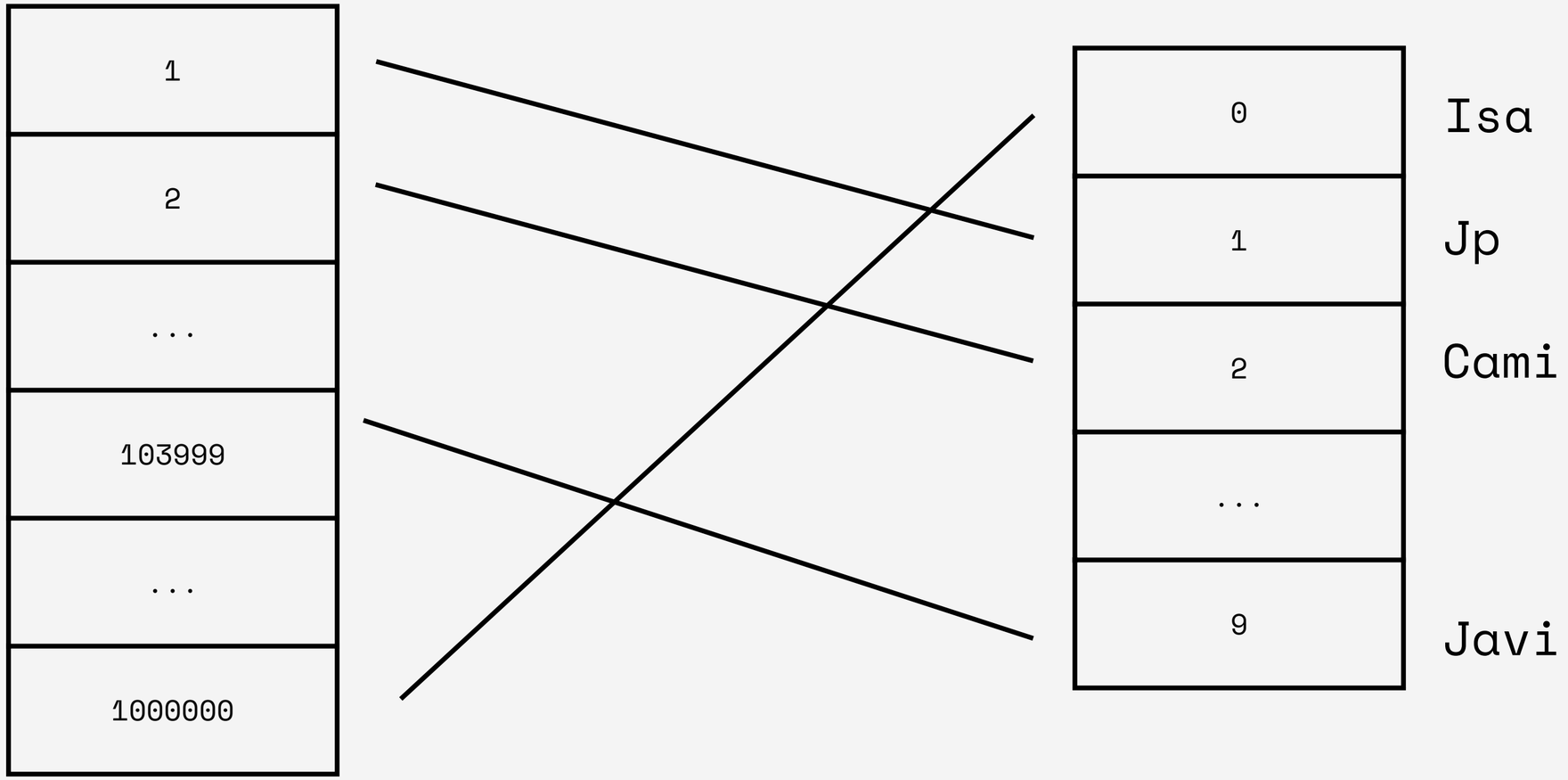
Vamos a definir A como un arreglo de sólo 10 posiciones.

Entonces h tiene que ser una función cualquiera que sea capaz de tomar un número del 1 al 1M y llevarlo a ser un número del 0 al 9.

Rep.Gráfica

Tablas de hash

h(x)=x mod 10



Definición

Tablas de hash-Colisiones

Como se pudieron dar cuenta, con 1M de llaves y 10 espacios en la tabla no se puede guardar todo...

Si usamos h(x) = x mod n, vamos a ver que los valores se repiten cada n, por lo que, en el caso anterior 10 y 20 resultan ambos en 0.

A esto se le conoce como **colisión**, si dos llaves llevan a la misma cajita de la tabla de hash existe una colisión.

Definición

Tablas de hash-Colisiones

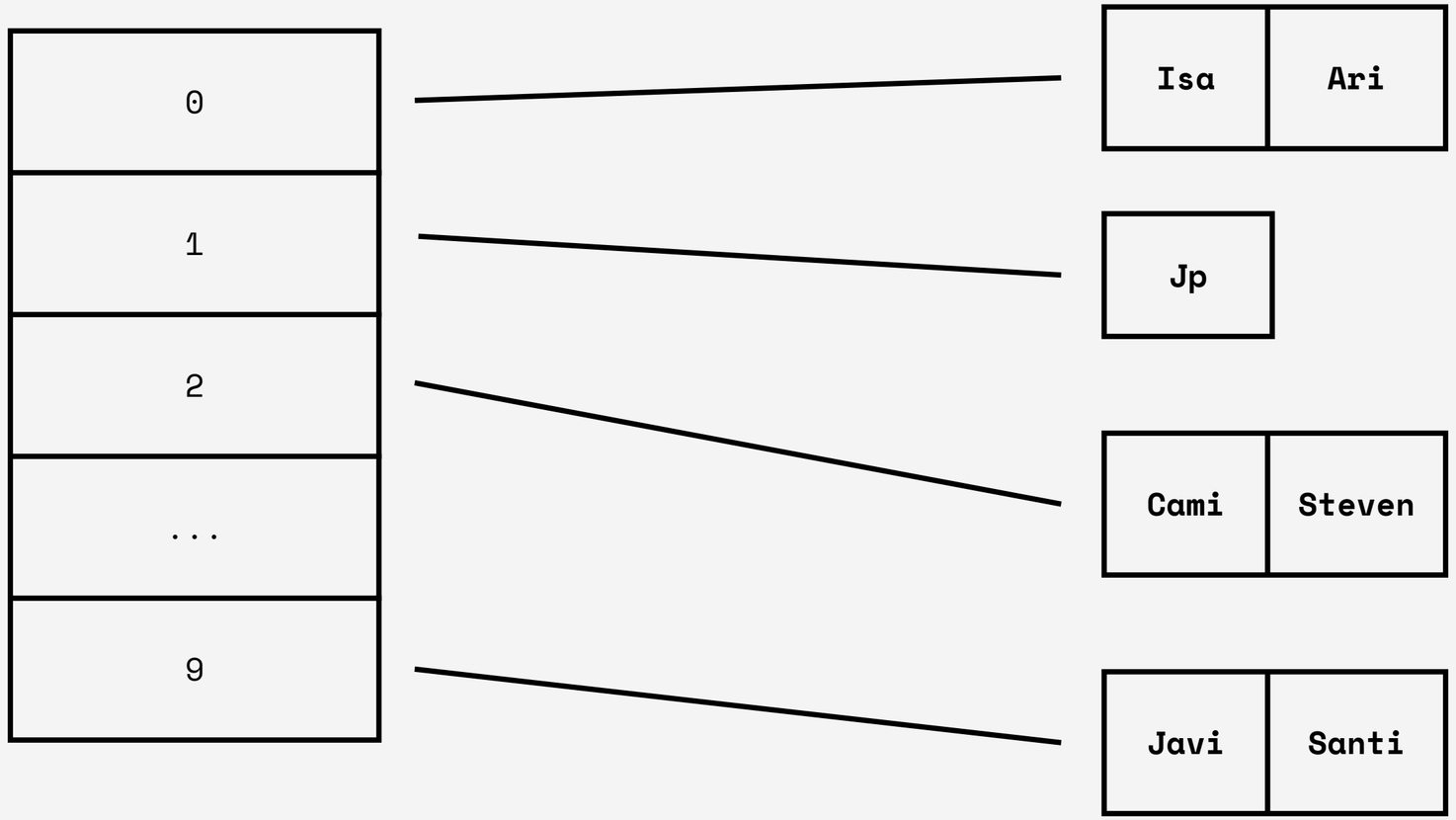
Para solucionar esto tenemos dos opciones:

- Encadenamiento:
  - Cada cajita tiene una **lista ligada**.
- Direccionamiento abierto:
  - Se busca **sistemáticamente** una posición vacía

Rep.Gráfica

Tablas de hash-Colisiones

Encadenamiento



Rep.Gráfica

Tablas de hash-Colisiones

Direccionamiento abierto (Sondeo lineal)

0	Isa
1	Jp
2	
...	
9	Javi

Queremos insertar:
(1, Cami)
1 está ocupado
Pero 2 (el siguiente)
está libre

0	Isa
1	Jp
2	Cami
...	
9	Javi

Definición

Factor de carga

Si buscamos tomar un espacio grande y reducirlo a un espacio más pequeño, queremos asegurarnos de que lo estamos haciendo de una manera eficiente.

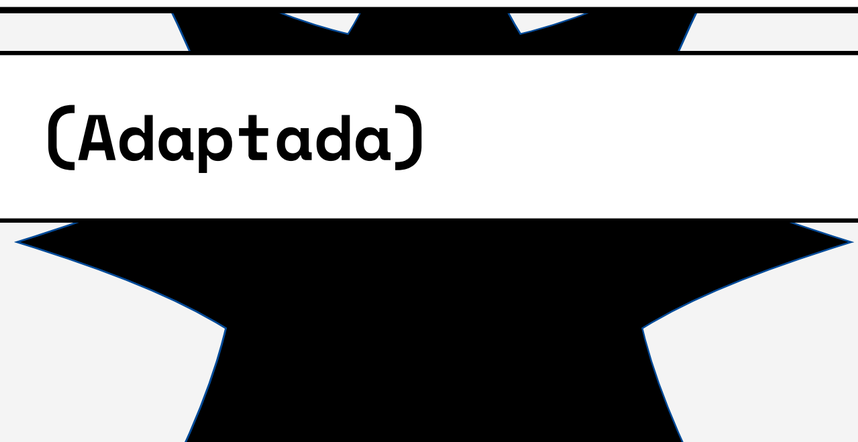
El factor de carga sirve para medir qué tan llena está la tabla. Esto va a ser n/m, donde n es la cantidad de elementos guardados y m el tamaño de la tabla.

Ejercicio 1

La DCCafetería sirve 2133 tipos de cafes y necesita ayuda para hacer más eficiente la atención de sus clientes. Se saben que llegan miles de pedidos por segundo, los cuales funcionan de la siguiente manera:

- Una persona pide un DCCafé y da su nombre. El pedido se agrega al sistema
- Cuando un DCCafé está listo, se llama la persona que haya pedido ese tipo de DCCafé hace más tiempo y se borra del sistema.

Pregunta 1. 2) - I2-2019-2 (Adaptada)

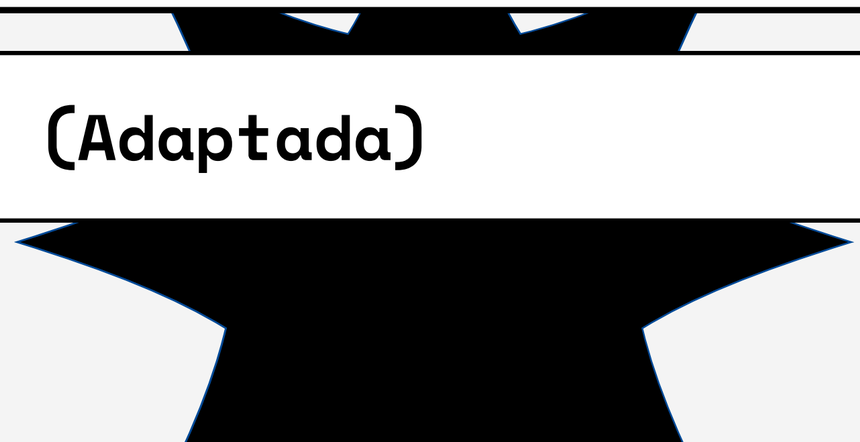


Explica cómo usar tablas de hash para resolver esto, para eso debes justificar:

- Elección de esquema de resolución de colisiones
- Qué se guarda en la tabla?

Ten en cuenta que el sistema necesita la mejor eficiencia, por lo que saber a quién le toca el DCCafé tiene que ser una operación O(1).

Pregunta 1. 2) - I2-2019-2 (Adaptada)



Pregunta 1. 2) - I2-2019-2 (Adaptada)

Para resolver esto podemos plantear una tabla de hash con tamaño 2133. Donde para un pedido (DCCafé, nombre), usamos h:DCCafés → [0,...,2132], donde se numeran los cafés con un índice arbitrario.

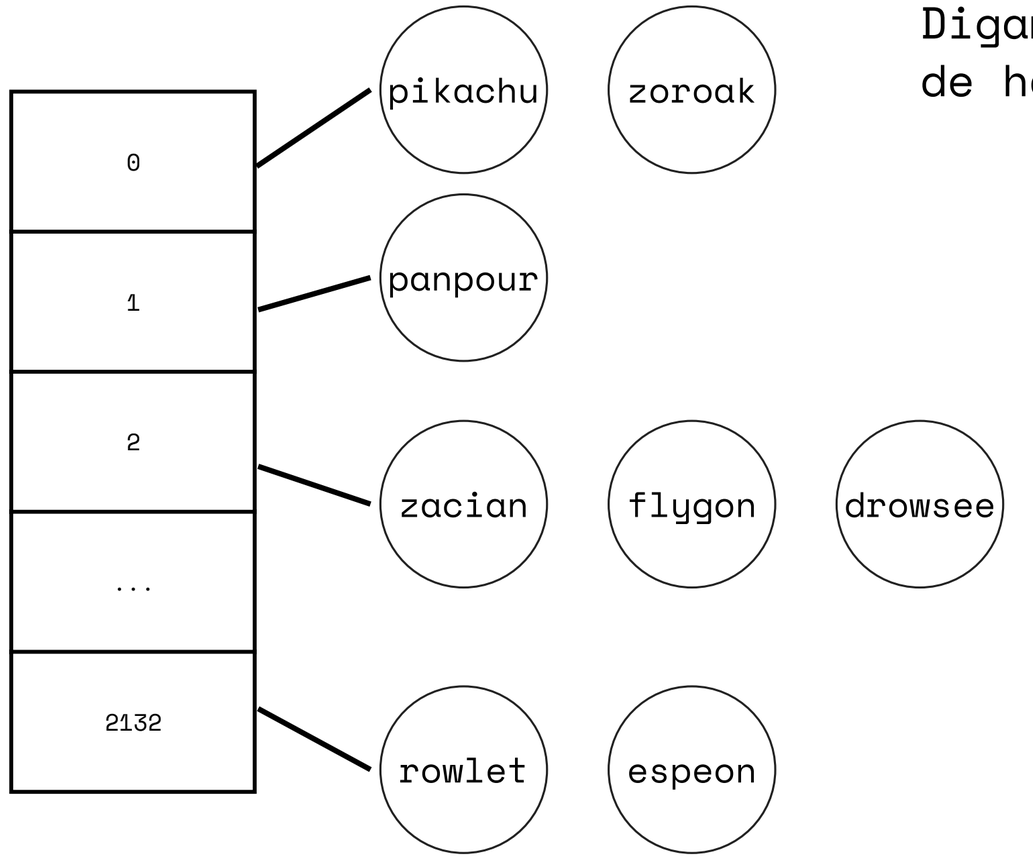
Si llegan miles de pedidos por segundo, y no queremos perder información vamos a tener que usar **encadenamiento**, ya que así podemos almacenar infinitamente los pedidos que lleguen.

Pregunta 1. 2) - I2-2019-2 (Adaptada)

Si queremos lograr que la inserción y la extracción de los pedidos sean O(1) tenemos que usar una cola FIFO, de manera que todos los datos se insertan al final, y cuando sale un pedido de tipo DCCafé, voy a su índice y cambio el puntero de la cabeza hacia el segundo elemento.

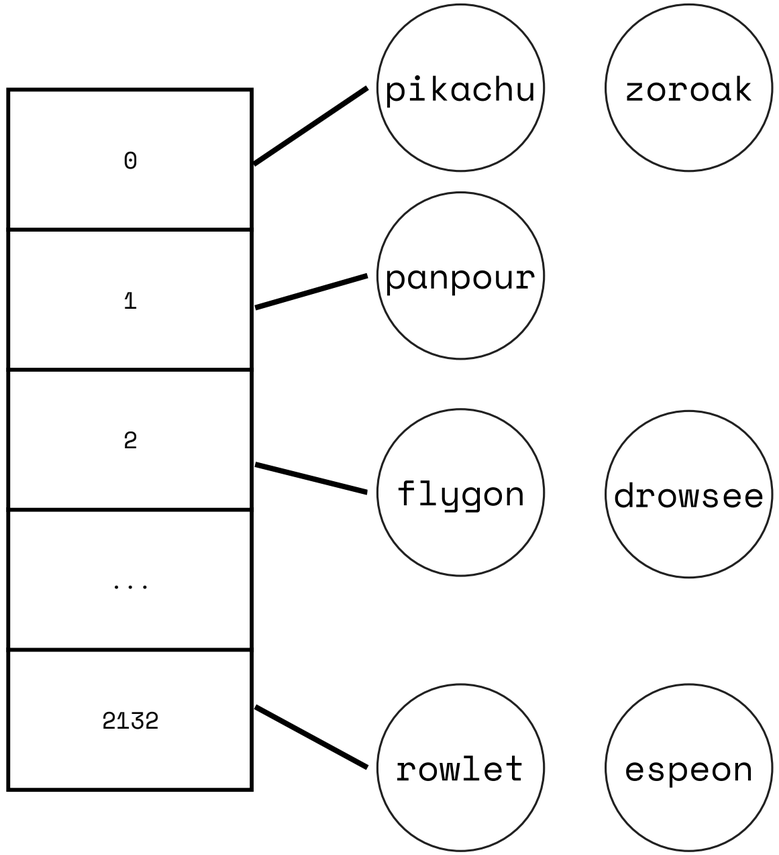
Los nodos de esta cola van a almacenar únicamente los nombres de los clientes.

Pregunta 1. 2) - I2-2019-2 (Adaptada)



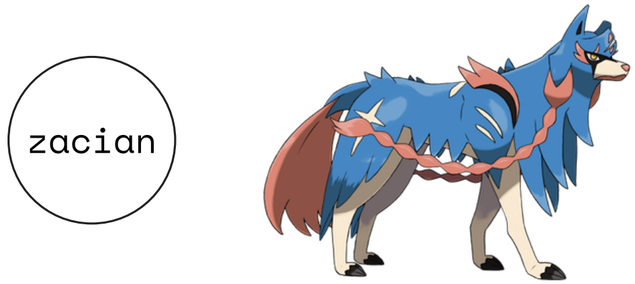
Digamos que está listo un pedido de hash 2

Pregunta 1. 2) - I2-2019-2 (Adaptada)

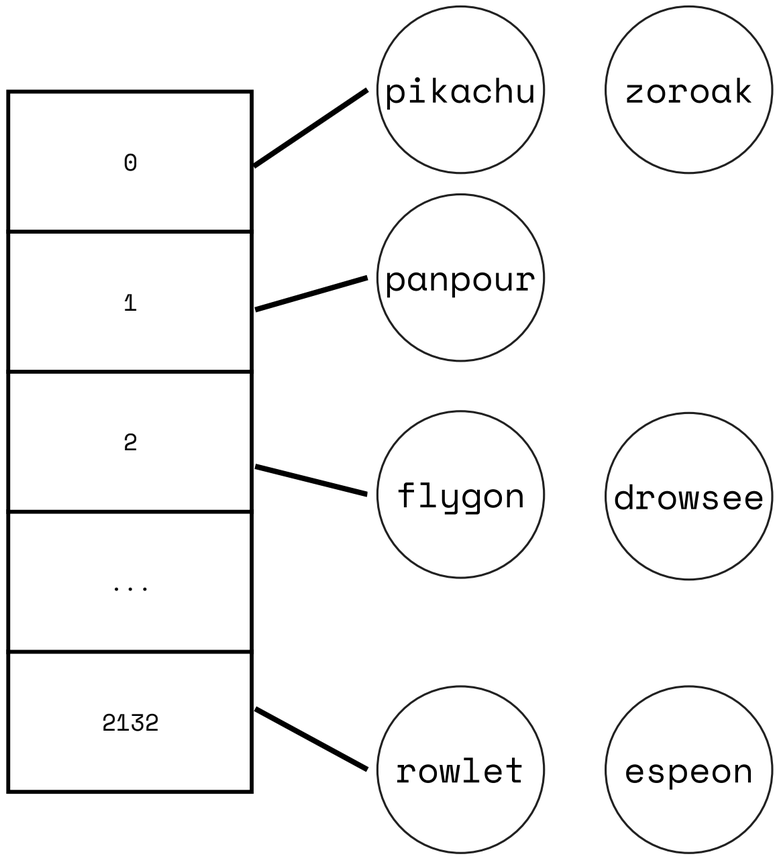


1° Quitamos el nodo
2° Cambiamos la cabeza de la cola

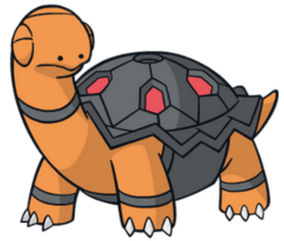
Llamamos a Zacian por el DDCafé con hash 2



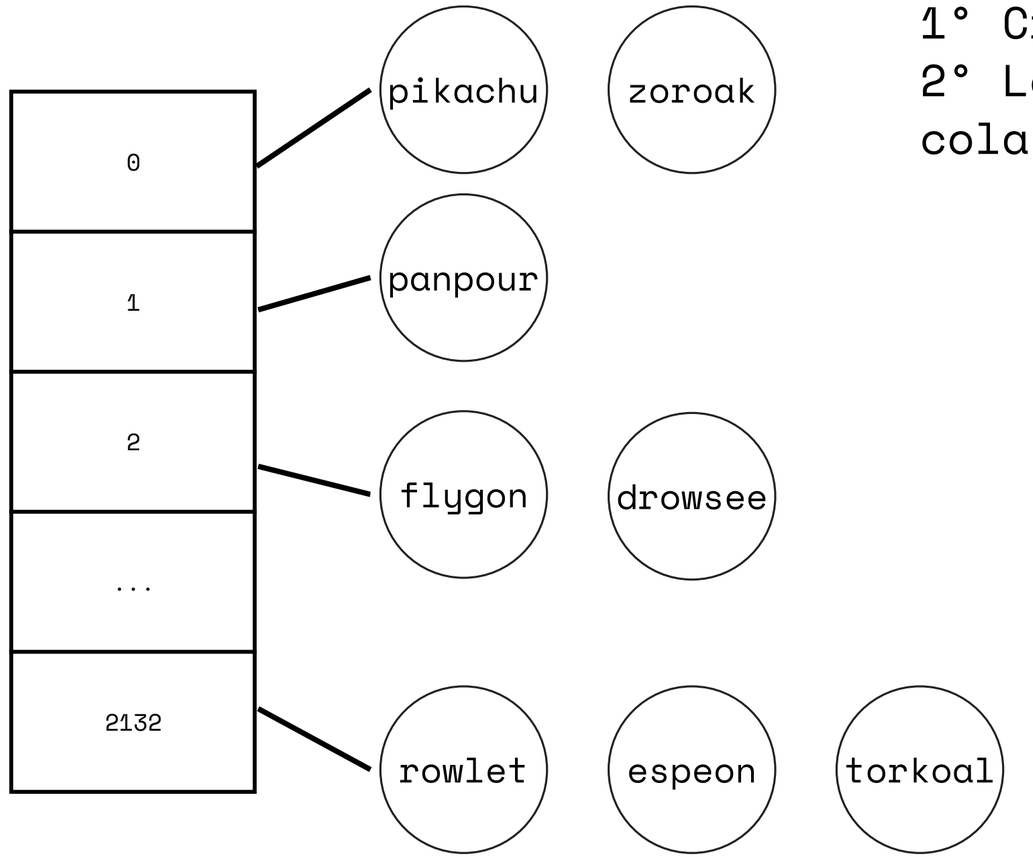
Pregunta 1. 2) - I2-2019-2 (Adaptada)



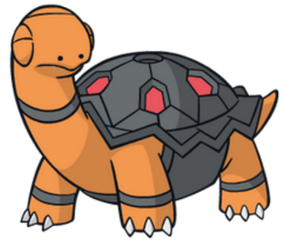
Digamos que llega Torkoal y hace un pedido por el DCCafé de hash 2132.



Pregunta 1. 2) - I2-2019-2 (Adaptada)



1° Creamos un nodo con el nombre
2° Lo añadimos al final de la cola, ya que es FIFO.



Ejercicio 2

Se está proponiendo un sistema que permita entra al DCC usando sólo el color de ojos y pelo. Dado que hay mucha gente que quiere entrar al DCC, saber si es que una persona puede pasar o no debe ser una operación O(1).
Proponga una EDD que permita, sabiendo el color de pelo y de ojos de una persona, abrir o mantener cerrada la puerta dependiendo de si esta persona tiene acceso.
Es importante notar que un máximo de K personas habilitadas a entrar tienen el mismo color de pelo y de ojos.



Pregunta 2.a

Pregunta 2.a

Si decimos que hay una cantidad de colores de pelo finita m_1, y una cantidad de colores de ojos finita m_2, podemos pensar en las funciones h_1 y h_2 que numeran estos colores desde el 0 hasta m_1-1 y m_2 correspondientemente.

Sobre estas funciones podemos pensar en una tabla anidada de hash (o visualmente una matriz). Donde el primer índice va a estar dado por h_1(color_de_pelo) y el segundo índice va a estar dado por h_2(color_de_ojos).

Pregunta 2.a

Se puede justificar que saber si una persona puede entrar o no al DCC es O(1) porque sabemos que hay un máximo de K personas que coinciden en color de pelo y ojos, por lo que si sugerimos un esquema de encadenamiento en la tabla interior (entiéndase la que nace de h_2), las listas ligadas como máximo tendran K elementos, lo que hace que encontrar un elemento en esa lista sea una operación constante, y por tanto, O(1).

Pregunta 2.a

Digamos que h_1 lleva los colores de pelo a número y h_2 lleva los colores de ojos a número.

x	Ojos rojos	Ojos Negros
Pelo café	Patrat	Bidoof
Pelo negro	Umbreon	
Pelo rubio	Mienfoo Fennekin	Ampharos



(2, 1)



(2, 0)



(0, 1)



(1, 0)





(0, 0)

Pregunta 2.a

OJO, acá pikachu tiene los mismos valores que Ampharos, pero como el no está en la tabla de hash, no puede entrar al DCC.

x	Ojos rojos	Ojos Negros
Pelo café	Patrat	Bidoof
Pelo negro	Umbreon	
Pelo rubio	Mienfoo Fennekin	Ampharos



(2, 1)



(2, 0)



(0, 1)



(1, 0)



(2,1)



(0, 0)

Ahora, el DCC, lleno de dinero, fama y poder, decide hacer un nuevo edificio (sin dejar de lado el DCC del legado), pero a este nuevo edificio no pueden entrar exactamente las mismas personas que al DCC antiguo.

Cómo cambia esta condición a la estructura planteada? Hay alguna forma de corregirlo, de forma que, sabiendo la instalación a la que se quiere ir, el color de ojos y el color de pelo, se pueda saber si es que una persona puede entrar o no?



Pregunta 2.b

Pregunta 2.b

Como la idea de las tablas de hash es almacenar valores, podemos pensar en que dadas las tablas anidadas anteriores, se puede tener en cada nodo un campo que diga para qué instalación se está permitiendo el acceso.

Otra forma de solucionar esto es crear un tercer nivel de tabla de hash (algo así como un cubo), donde una función h_3 lleva las instalaciones a un espacio numérico del 0 al 1 en este caso (extensible con más centro simplemente ampliando el rango), y rellenando con las personas que estén habilitadas.

Feedback