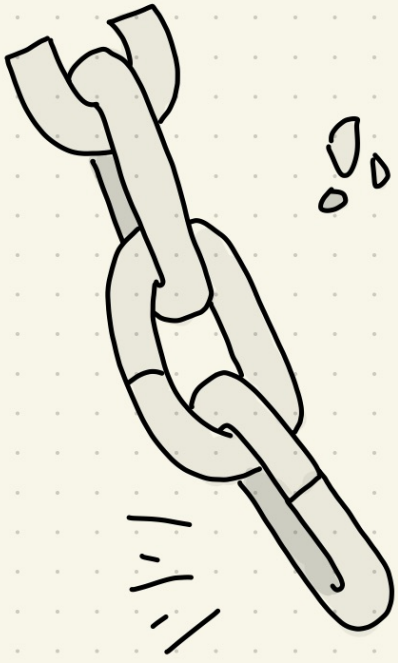
arrays & strings

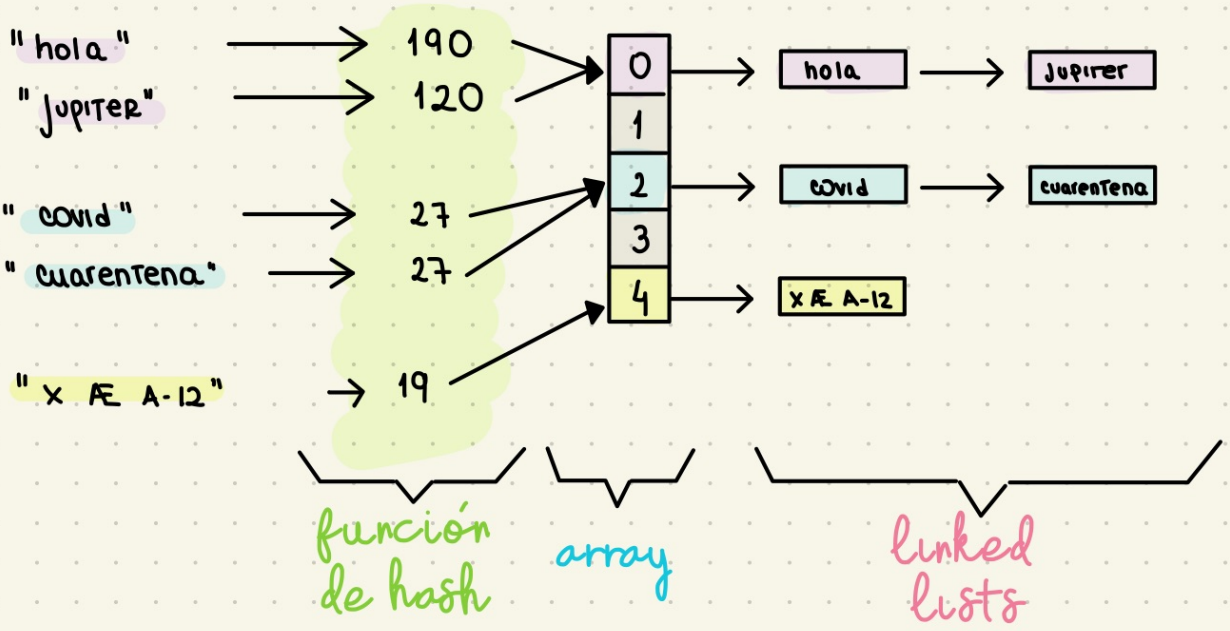


@heyimnowi

HASH TABLE

Hash table o una tabla de hash es un tipo de estructura de datos en la cual se mapean keys y values para que la búsqueda de un elemento sea más eficiente

Consiste en un array de Linked Lists y una función de hash



Con esta estructura no se mantiene contigüidad ni orden de las componentes

Se prioriza la busqueda, tratando de que el algoritmo tenga complejidad cercana a O(1)

agregar un elemento

- [x] Primero hay que calcular el valor de hash del elemento. Generalmente va a ser un **INT** o **long**.
Es normal que dos elementos tengan el mismo valor de hash ya que hay un número infinito de keys y un numero finito de **INTS**

- [x] El segundo paso es mapear el valor de hash con un índice del array.
La forma más facil es usar **hash(key) %. array.lenght**

- [x] En el índice del paso anterior hay una Linked List y se guarda el valor ahí.

buscar un elemento

El proceso es parecido al caso anterior. Se calcula el hash del elemento a buscar y se busca el elemento secuencialmente en la Linked List.

Si el número de colisiones es muy alto, es decir todas las keys caen en la misma lista, la complejidad es O(n) siendo n la cantidad de elementos

Se asume que con una buena implementación las colisiones son mínimas la búsqueda es O(1)

colisiones

Se usa una función de hash para asignarle un index a cada elemento

Cuando la función de hash computa el mismo index para distintos elementos se llama colisión

Existen dos formas de resolver dichas colisiones:

- Open hashing ó Chaining
- Open addressing ó Closed hashing

factor de carga

Es una medida que indica el porcentaje de la tabla de hash que debe ocuparse antes de cambiar de tamaño. Si se supera el umbral definido hay que agrandar la tabla y rehashear los elementos.

ARRAY LIST

En algunos lenguajes los arrays se resizean automáticamente, a medida que se van agregando elementos.

En Java los arrays son de un tamaño fijo y es definido cuando se crea el array.

Cuando se quiere una estructura de datos que crezca dinámicamente se usa, generalmente, un **Array List**

- La estructura se duplica cuando se llena
- O(1) para insertar y devolver un elemento
- Duplicar la capacidad del array es O(n) pero como generalmente no pasa, se dice que se amortiza
- El factor por el cual se agranda el array (para Java es 2) puede variar

STRING BUILDER

El objeto String Builder es como un String pero que puede modificarse. Internamente se tratan como un array de caracteres

Los Strings son inmutables y no permiten apendear caracteres al mismo.

El típico caso de uso es la concatenación de Strings.

```
String joinWords(String[] words){
    String sentence = "";
    for(String s: words){
        sentence = sentence + s;
    }
    return sentence;
}
```

La complejidad del algoritmo es $O(n^2)$ porque por cada loop se crea una copia del elemento y se copian todos los caracteres

Siendo i la i-nesima iteración, y n la cantidad de caracteres de cada word la complejidad se calcula como:

$$\sum i \, n \leq n^2$$