DOCUMENTO TÉCNICO

# Requisitos

## Especificación

#RESUMEN DEL PROBLEMA (opcional)

Implementar hash con linked list y hash de segundo nivel

# Diseño

## Estrategia

### Para implementar el punto a, simplemente se cambia el segundo nivel de las tablas hash de una lista normal a una linked list

### La estrategia a realizar será igual a la de los laboratorios pasados de linked list, será implementar los métodos search, insert, delete, update los cuales serán exactamente los mismos pero ahora en una clase diferente.

### Para implementar el punto b, se tiene que cambiar el segundo nivel de las tablas de hash de una lista normal a una segunda tabla hash

### La estrategia para realizar será implementar dos clases; la primera que contenga una tabla hash común y corriente y la segunda que contenga la tabla hash de hash.

### Los métodos search, insert, delete, update de la tabla hash de hash llamaran a estos mismos métodos, pero de cada hash por separado

#ALGORITMO

Programa adjunto

## Casos de prueba

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Justificación** | **Salida** |
| Búsqueda de elemento (linked list) | Caso genérico | El valor del elemento a buscar |
| Búsqueda de elemento (hash) | Caso genérico | El valor del elemento a buscar |
|  |  |  |
|  |  |  |

# 

## Temporal

La complejidad del código depende del contexto en el que se esté manejando; si se está usando una linked list, la complejidad depende de la función que se vaya a usar (un insert costara 1 o por otro lado un search costará n) sin embargo si usa una hash de hash el costo será de 1 ya que las operaciones en una tabla de hash son constantes.

# 

# Código

*Al igual que el punto anterior pueden guiarse con este ejemplo*

## Documentación

Dentro del código.

## Fuentes

*Las fuentes son el código sobre el cual escribieron el documento*

/arena\_1\_a