Nombre y Apellido:

# Examen Final

## Tablas hash híbridas

Una *tabla hash híbrida* es una variante de la tabla de hash con área de rebalse en la cual la estructura interna de cada *bucket* puede adaptarse dinámicamente según su tamaño. La tabla posee una cantidad *fija* de *buckets* y utiliza una función de hash para asignar claves a cada una de ellos, como en una tabla hash tradicional.

La diferencia principal es que cada *bucket* puede representarse de dos formas: como una lista enlazada cuando contiene pocos elementos, o como un árbol balanceado (por ejemplo, un AVL) cuando la cantidad de elementos supera un cierto umbral. Este cambio de representación es local al *bucket* y no implica redimensionar ni rehacer la tabla completa.

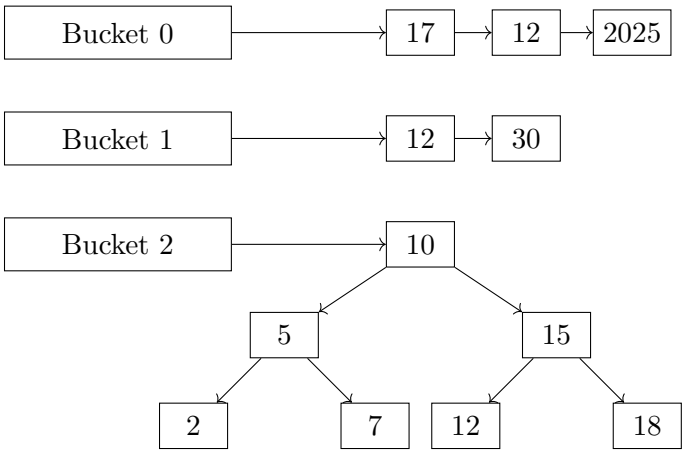


Figura 1: Tabla hash híbrida de enteros con listas y árboles balanceados

Este enfoque aprovecha el bajo costo de inserción de las listas enlazadas cuando las áreas de rebalse contienen pocos elementos, y al mismo tiempo limita el costo de búsqueda en casos de muchas colisiones. Notemos que, al mutar las listas enlazadas en árboles balanceados, reducimos la complejidad de la búsqueda y la eliminación de un costo lineal a un costo logarítmico cuando realmente es significativo, es decir, cuando la cantidad de datos es alta.

Cabe destacar que, en esta implementación, la tabla hash posee una cantidad *fija* de *buckets*. No se requiere implementar mecanismos de redimensionamiento ni de *rehashing*.

## Ejercicios

**1.** Defina una estructura de datos `HybridHash` para representar **tablas hash híbridas generales**. Para esto, tenga en cuenta las operaciones sobre la estructura que se piden a continuación.

Puede utilizar a modo de referencia la implementación provista de `TablaHash`. Recuerde que no es necesario implementar redimensionamiento ni *rehashing*: la capacidad de la tabla se fija al momento de crearla.

**2.** Implemente la función `hybrid_hash_crear()`, que dada la capacidad y las funciones para datos genéricos, retorne una tabla hash híbrida adecuada.

**3.** Implemente la función

```
void hybrid_hash_insertar(HybridHash tabla, void* dato)
```

que inserta una nueva clave en la tabla.

Para ello, defina un umbral $T$ tal que, cuando una clave deba insertarse en un área de rebalse y la cantidad de elementos almacenados en dicha área supere $T$, esta se convierta en un `AVL`, en caso de no haber sido convertida previamente.

**4.** Implemente la función

```
void* hybrid_hash_buscar(HybridHash tabla, void* dato)
```

que busca en la tabla un dato que coincida con el dato dado, y retorna el dato encontrado o `NULL` en caso de no hallarse en la tabla.

**5.** Implemente la función

```
void hybrid_hash_eliminar(HybridHash tabla, void* dato)
```

que elimina de la tabla el dato que coincida con el proporcionado.

Considere que, si luego de eliminar el dato de un área de rebalse la cantidad de elementos almacenados en dicha área desciende por debajo de $\frac{T}{2}$ (donde $T$ es el valor definido en su implementación de inserción), esta debe reconvertirse en una lista enlazada, siempre que previamente hubiera sido convertida en un `AVL`.