Resumen de costos de operaciones de las estructuras vistas

A continuación se hace un resumen de los costos de ciertas operaciones básicas con las estructuras vistas en el curso, que pueden servir para la representación de distintas estructuras de más alto nivel (e.g., Diccionario o Conjunto).

La siguiente tabla sólo condensa de lo visto en las clases teóricas y prácticas, en las cuales se encuentra su justificación, además de en la bibliografía del curso (e.g., en Cormen, 2009):

	Insertar	Buscar	Eliminar
ABB	O(n)	O(n)	O(n)
AVL	O(log n)	O(log n)	O(log n)
Trie	O(|k|)	O(|k|)	O(|k|)
Heap Binario	O(log n)	O(1|n)	O(log n)
Heap Fibonacci	O(1)	O(1|n)	O(log n)
Array	O(1)	O(n)	O(1)
Lista	O(1|n)	O(n)	O(1|n)

Algunas observaciones

Trie:

|k| es la longitud de la clave utilizada. Se supone que el alfabeto usado es finito, ya que, de otra forma, el costo sería O(|k|*|a|), donde |a| es el tamaño del alfabeto.

Heap:

Si lo que se busca es el mínimo/máximo para un min/maxHeap el costo de buscar es O(1), mientras que si es cualquier otro elemento es O(n).

La eliminación está acotada por O(log n) siempre y cuando se sepa la ubicación del elemento a borrar, ya que sea que es el mínimo/máximo si es min/maxHeap, o porque primero se realizó una búsqueda del mismo (que puede tomar O(n)).

Se puede suponer que existen las operaciones increase-key/decrease-key para max/minHeaps que permiten definir una clave menor/mayor en O(log n), manteniendo el invariante de la estructura.

Ojo con el Fibonacci Heap, que el costo de O(1) de insertar en ese caso es el costo amortizado. Luego, no se puede mezclar con el resto de las cosas que tenemos.

Array:

Los costos de las operaciones indicadas implican cambiar o borrar un valor de una posición del array, si se sabe el índice de la posición a modificar. Si hubiera que mover el resto de los elementos, se agrega un costo O(n) en el peor caso.

Lista:

El insertar/borrar es O(1) si se agrega/borra al comienzo o al final. Si tengo que agregar/borrar en una posición determinada es O(n), ya que se requiere ubicarse en la posición deseada. Esto es, salvo que ya tuviera un iterador apuntando al elemento, ya que en ese caso se puede agregar/borrar antes o después en O(1).