Nicolás Ramírez Calderón
257084

**EJERCICIOS**

**6.5-3**
Para soportar operaciones en un MIN-HEAP se cambian todas las comparaciones entre los elementos claves del *heap* en la implementación del MAX-HEAP.

**6.5-8**
Dadas *k* listas ordenadas con un total de *n* elementos mostrar como combinarlas en tiempo $\mathrm{O}(n \lg k)$.
Insertar todos los *k* elementos en la posición 1 de cada lista dentro del *heap*. Usar EXTRACT-MAX para obtener el primer elemento de la lista combinada. Insertar el elemento en la posición 2 de a lista de donde vino el elemento más grande dentro del *heap*. Continuando con este patrón se llega al algoritmo deseado. Claramente el tiempo de ejecución es $\mathrm{O}(n \lg k)$.

**PROBLEMA**

**6.1**

**a)** Los procedimientos BUILD-MAX-HEAP y BUILD-MAX-HEAP’ no siempre vana crear un mismo *heap* cuando tienen un mismo arreglo.
**R//**
Por ejemplo para el arreglo [2, 3, 4] BUILD-MAX-HEAP produce [4, 3, 2], mientras que para el mismo arreglo BUILD-MAX-HEAP’ produce el arreglo [4, 2, 3].

**b)** Una cota superior de tiempo $\mathrm{O}(n \lg n)$ se obtiene inmediatamente al hacer n-1 llamadas MAZ-HEAP-INSERT, cada una de ellas toma tiempo $\mathrm{O}(\lg n)$.
Para una cota inferior de $\mathrm{O}(n \lg n)$, se considera el caso en el cual el arreglo esta dado en orden incremental. Cada llamada al algoritmo causa que HEAP-INCREASE-KEY recorra todo el camino hasta la raíz. Por esto la profundidad del nodo *i* es $\lfloor \lg i \rfloor$, el tiempo total es:

$$\sum_{i=1}^{n} \Theta(\lfloor \lg i \rfloor) \geq \sum_{i=\lceil (n/2) \rceil}^{n} \Theta\left(\left\lfloor \lg\left(\left\lceil n/2 \right\rceil\right)\right\rfloor\right)$$

$$\sum_{i=1}^{n} \Theta(\lfloor \lg i \rfloor) \geq \sum_{i=\lceil (n/2) \rceil}^{n} \Theta(\lfloor \lg(n/2) \rfloor)$$

$$\sum_{i=1}^{n} \Theta(\lfloor \lg i \rfloor) \geq \sum_{i=\lceil (n/2) \rceil}^{n} \Theta(\lfloor \lg n - 1 \rfloor)$$

$$\sum_{i=1}^{n} \Theta(\lfloor \lg i \rfloor) = n/2 * \Theta(\lg n)$$

$$\sum_{i=1}^{n} \Theta(\lfloor \lg i \rfloor) = \Omega(n \lg n)$$

Dado esto se ve que el BUILD-MAX-HEAP’ necesita un tiempo de $\Theta(n \lg n)$ para construir un *heap* de *n* elementos.