Universidad del Istmo de Guatemala
Facultad de Ingenieria
Ing. en Sistemas
Informatica 2
Prof. Ernesto Rodriguez - erodriguez@unis.edu.gt

Hoja de trabajo #9

Fecha de entrega: 03 de Mayo, 2018 - 11:59pm

Instrucciones: Realizar cada uno de los ejercicios siguiendo sus respectivas instrucciones. El trabajo debe ser entregado a traves de Github, en su repositorio del curso, colocado en una carpeta llamada "Hoja de trabajo 9". Al menos que la pregunta indique diferente, todas las respuestas a preguntas escritas deben presentarse en un documento formato pdf, el cual haya sido generado mediante Latex. Los ejercicios de programación deben ser colocados en una carpeta llamada "Programas", la cual debe colocarse dentro de la carpeta correspondiente a esta hoja de trabajo.

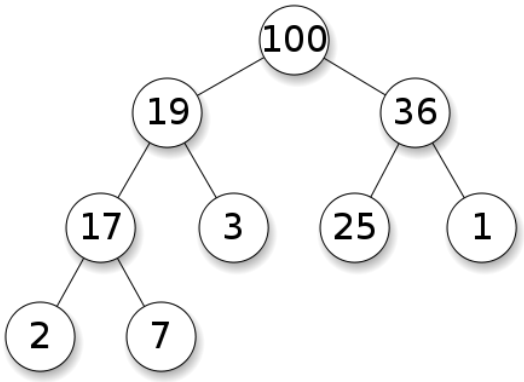
Iniciación

Crear una solución llamada *Heap*. Dentro de esta solución crear:

- un proyecto llamado *Heap* de tipo console
- un proyecto llamado *HeapTests* de tipo xunit

Heap

Un *heap* es un tipo especial de arbol binario que tiene la peculiaridad que todo hijo de cada nodo es menor o igual que su padre. La siguiente imagen ilustra un heap:



Sin embargo, un *heap* puede representarse como un arreglo en donde dado un indice i:

- El padre de i tiene el indice `floor`$((i-1)/2)$
- El hijo derecho de i tiene el indice $2*i+1$
- El hijo izquierdo de i tiene el indice $2*i+2$

Procedimiento *ShiftDown* (40%)

Dado un arreglo cualquiera, es simple convertirlo a un *heap*. El proceso consiste en recorrer cada uno de sus indices (empezando desde el final) y convertir dicho indice en un heap. Para ello se define el metodo `ShiftDown`[1] de tipo `ShiftDown : int[ ] ⊗ int → void`. El primer parametro es un arreglo y el segundo parametro es la posición siendo considerada. Este metodo debe hacer lo siguiente:

1. Se define el indice actual (i), el cual se inicializa con el segundo parametro del metodo.
2. Dado el valor ubicado en i y su hijo derecho e izquierdo, seleccionar el valor más grande de los tres. Ignorar valores que esten fuera del arreglo.
3. Si el valor seleccionado en el paso anterior es diferente del valor ubicado en i, intercambiar dicho valor con el valor ubicado en i.
4. Si los valores fueron intercambiados, se le asigna a i el indice del valor que fue intercambiado y se repite el procedimiento. De lo contrario, el metodo finaliza.

Procedimiento *Heapify* (30%)

Definir un procedimiento llamado `Heapify`[1] de tipo `Heapify : int[ ] → void` el cual recibe un arreglo y lo convierte a su representación como *heap*. El procedimiento debe hacer lo siguiente:

1. Comenzar seleccionando el indice (i), el padre del ultimo valor en el arreglo (`Length-1`)
2. Llamar al metodo `ShiftDown` con el arreglo y el indice i
3. Decrementar el indice i una unidad
4. Repetir hasta que $i = 0$

Unit test para *Heapify* (30%)

Crear un unit test para el procedimiento `Heapify`. Este test debe empezar con un arreglo desordenado, utilizar el metodo `Heapify` para convertirlo en un *heap* y por ultimo verifiar que dicho arreglo es un *heap*.

References

[1] Wikipedia. Heapsort. `https://en.wikipedia.org/wiki/Heapsort#Pseudocode`.