**Control de lectura 06-AYED**

Jefer Alexis González Romero. AYED-2. 13/11/2021

**1.** Ilustre el paso a paso de heapsort sobre el arreglo:

A = [5, 13, 2, 25, 7, 17, 20, 8, 4]

Primero se crea un max-heap con el arreglo dado:

[25, 13, 20, 8, 7, 17, 2, 5, 4]

A.heap\_size = 9

**8**

**2**

**20**

**17**

**25**

**13**

**7**

**5**

**4**

Se intercambian en el arreglo el primer elemento y el último.

[4, 13, 20, 8, 7, 17, 2, 5, 25]

Ahora para mantener la propiedad de max-heap se usa max-heapify con los primeros 8 elementos del arreglo y con índice 0.

[20, 13, 17, 8, 7, 4, 2, 5, 25]

A.heap\_size = 8

**8**

**2**

**27**

**4**

**20**

**13**

**7**

**5**

Ahora se vuelve hace un intercambio, pero del primero con el penúltimo:

[5, 13, 17, 8, 7, 4, 2, 20, 25]

Se usa max-heapify con los primeros 7 elementos y con índice 0:

[17, 13, 5, 8, 7, 4, 2, 20, 25]

A.heap\_size = 7

**8**

**2**

**5**

**4**

**17**

**13**

**7**

En este caso, este proceso se hace 9 veces, el cual es el número de elementos que tiene el heap.

Intercambio:

[2, 13, 5, 8, 7, 4, 17, 20, 25]

Max-heapify:

[13, 8, 5, 2, 7, 4, 17, 20, 25]

A.heap\_size = 6

**2**

**5**

**4**

**13**

**8**

**7**

Intercambio:

[4, 8, 5, 2, 7, 13, 17, 20, 25]

Max-heapify:

[8, 7, 5, 2, 4, 13, 17, 20, 25]

A.heap\_size = 5

**2**

**5**

**8**

**7**

4

Intercambio:

[4, 7, 5, 2, 8, 13, 17, 20, 25]

Max-heapify:

[7, 4, 5, 2, 8, 13, 17, 20, 25]

A.heap\_size = 4

**2**

**5**

**7**

**4**

Intercambio:

[2, 4, 5, 7, 8, 13, 17, 20, 25]

Max-heapify:

[5, 4, 2, 7, 8, 13, 17, 20, 25]

A.heap\_size = 3

**2**

**5**

**4**

Intercambio:

[2, 4, 5, 7, 8, 13, 17, 20, 25]

Max-heapify:

[4, 2, 5, 7, 8, 13, 17, 20, 25]

A.heap\_size = 2

**4**

**2**

Intercambio:

[2, 4, 5, 7, 8, 13, 17, 20, 25]

Max-heapify:

[2, 4, 5, 7, 8, 13, 17, 20, 25]

A.heap\_size = 1

**2**

**2.** Ilustre el paso a paso de heap\_extract\_max sobre el heap:

A = [15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1]

**8**

**7**

**9**

**6**

**2**

**4**

**0**

**5**

**13**

**15**

**12**

**1**

Primero se toma el primer valor en el arreglo y a este se le asigna el valor del último elemento en el arreglo.

A = [1, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1]

Ahora se hace un max-heapify con los primeros 11 elementos y con índice 0:

**8**

**7**

**9**

**6**

**2**

**4**

**0**

**5**

**13**

**1**

**12**

Max-heapify con índice 1.

**8**

**7**

**9**

**6**

**2**

**4**

**0**

**5**

**1**

**13**

**12**

Max-heapify con índice 5.

**8**

**7**

**9**

**6**

**2**

**4**

**0**

**5**

**12**

**13**

**1**

Max-heapify con índice 10.

**8**

**7**

**9**

**1**

**2**

**4**

**0**

**5**

**12**

**13**

**6**

**3.** Ilustre el paso a paso de max\_heap\_insert(10) sobre el heap:

A = [15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1]

**8**

**7**

**9**

**6**

**2**

**4**

**0**

**5**

**13**

**15**

**12**

**1**

Se inserta al final del arreglo el nuevo elemento en este caso el 10:

A = [15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1, 10]

**8**

**7**

**9**

**6**

**2**

**4**

**0**

**5**

**13**

**15**

**12**

**1**

**10**

Se revisa si el padre el padre es mayor, en este caso no, así que se intercambian:

**7**

**9**

**6**

**2**

**4**

**0**

**5**

**13**

**15**

**12**

**1**

**8**

**10**

Ahora se hace lo mismo con el 10:

**7**

**6**

**2**

**4**

**0**

**5**

**13**

**15**

**12**

**1**

**8**

**9**

**10**

Como 15 es mayor que 10 ya se deja así el heap.

**4.** Implemente el código para las siguientes operaciones sobre min-heap:

Adjunto código con nombre “4. MinHeap”

**5.** Desarrolle un algoritmo para determinar si un árbol binario es un max-heap.

Adjunto código con nombre “5”