Práctica I

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y T.I.

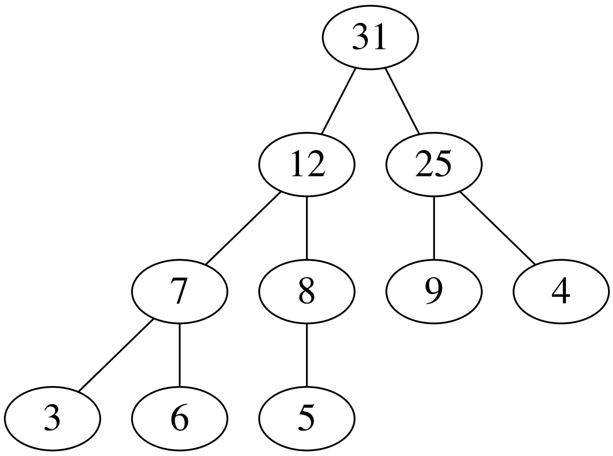
CI-2612: Algoritmos y Estructuras de Datos II

Pregunta 1

Dado el arreglo $A = [31, 12, 25, 7, 8, 9, 4, 3, 6, 5]$. Indique si A es o no un max-heap.

Respuesta 1

Si es un Max-heap, porque el mayor valor está en la raíz y para todos los nodos x, excepto la raíz, se cumple que $Parent(x) \geq x$.



Pregunta 2

Sea $A[1..n]$ un arreglo en donde los elementos son de tipo entero, son distintos y están ordenados de manera ascendente. Diseñe un algoritmo que retorne cualquier índice i del arreglo, es decir $1 \leq i \leq n$, que cumpla que $A[i] = i$. En caso de que tal índice no exista, el algoritmo retorna -1. El algoritmo debe tener un tiempo de ejecución de $O(\log n)$ en el peor caso. Debe demostrar el tiempo del peor caso.

Respuesta 2

Función BusquedaIndice(T, i, j)

inicio
 si $i = j$ **entonces**
  si $T[i] = i$ **entonces** **retornar** i;
  en otro caso retornar-1;
 en otro caso
  $k \leftarrow (i + j) \div 2$
  si $T[k] = k$ **entonces**
   retornar k
  si no, si $k \leq T[k]$ **entonces**
   retornar `BusquedaIndice(`$T, i, k - 1$`)`
  en otro caso
   retornar `BusquedaIndice(`$T, k + 1, j$`)`

Llamada: `BusquedaIndice`(T, 1, n).

Continuación Respuesta 2

- El tiempo de ejecución viene dado por $T(n) = T(n/2) + O(1)$
- Usando el teorema, visto en clase, de Brassard [1] se tiene que $l = 1, b = 2, k = 0$
- Se tiene que $1 = 2^0$
- $T(n) = \Theta(n^0 \log n)$
- $T(n) = \Theta(\log n)$

Pregunta 3

Dado el siguiente algoritmo:

Procedimiento ALGOX(A)

```
1 inicio
2   para i ← 1 a A.length − 1 hacer
3     para j ← A.length decrementar hasta i + 1 hacer
4       si A[j] < A[j − 1] entonces
5         SWAP (A[j], A[j − 1])
```

Determine el tiempo de ejecución del peor caso del procedimiento `ALGOX`.

Respuesta 3

- Se usa la técnica del barómetro de Brassard [1]
- Se debe encontrar la instrucción que más se repite en el algoritmo
- Se tiene que la instrucción que más se repite es el ciclo **para** de la línea 3
- La cantidad de veces que se repite la línea 3 es $\sum_{i=1}^{n-1} n - (i+1)$
- $\sum_{i=1}^{n-1} n - (i+1) = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i - \sum_{i=1}^{n-1} 1 =$ $(n-1)n - n(n-1)/2 - (n-1) = O(n^2)$
- El tiempo de ejecución del peor caso es $O(n^2)$

Pregunta 4

RQX es una variante del algoritmo RANDOMIZED-QUICKSORT. En RQX el pivote es escogido al azar repetidamente hasta obtener un **buen pivote**. Se define un **buen pivote**, como aquel que permite una partición de un arreglo de n elementos, en dos subarreglos conteniendo al menos $n/3$ elementos. Para escoger un **buen pivote**, el algoritmo RQX trabaja de la siguiente manera. RQX escoge un pivote al azar, luego RQX usa el procedimiento PARTITION para obtener la partición del arreglo. En caso de que alguna de las particiones no tenga al menos $n/3$ elementos, entonces RQX deshace la partición realizada, y se escoge un nuevo pivote. Estos pasos se repiten hasta obtener un **buen pivote**. Es caso de obtenerse un **buen pivote** el algoritmo continua. La operación de deshacer la partición es orden lineal. Se quiere que determine la profundidad máxima de recursión de RQX.

Respuesta 4

- La recurrencia de RQX es $T(n) = T(n/3) + T(2n/3) + O(n)$
- Usamos un árbol de recursión para estudiar el algoritmo
- El tamaño del problema a nivel i del árbol es $\left(\frac{2}{3}\right)^i n$
- El tamaño del problema es 1 cuando $\left(\frac{2}{3}\right)^i n = 1$
- La altura h de la rama más larga se obtiene cuando $\left(\frac{2}{3}\right)^h n = 1$
- Entonces, se tiene que la máxima altura es $h = \log_{3/2} n$
- Por lo tanto, la profundidad máxima de recursión de RQX es $O(\log_{3/2} n)$

Referencias

G. Brassard and P. Bratley.
Fundamentals of Algorithmics.
Prentice Hall, 1996.