algoritmia-PARCIAL-1-explicado.pdf

gemma_o2

Algoritmia y Estructuras de Datos Avanzadas

2º Grado en Ingeniería Informática

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

ENGINEERING THE FUTURE
En Indra sentimos #OrgulloIngeniero
indra
orgullo-ingeniero.indra.es
Comparte los motivos por los que sientes #OrgulloIngeniero y ayúdanos a poner en valor a la ingeniería.

ALGORITMIA

Ejercicios de exámenes explicados paso a paso.

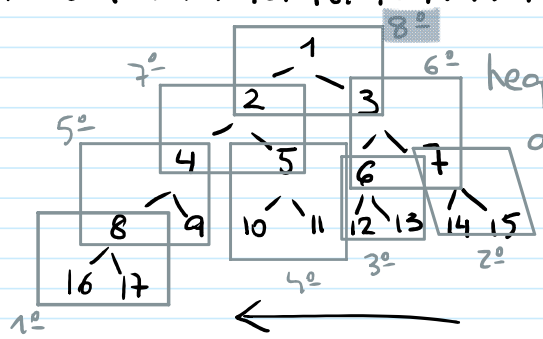
PARCIAL 1

PARTE 1:

1. (2 points)
   i. We have a list with 17 elements that we have to convert, in place, to a min heap. How many times will we call heapify?
   ii. Compute $\lg^* 179$. You can compute the logarithms approximately, if necessary (for example, $\lg 42 = 5.x$, with $x > 5$)

i) IN PLACE significa colocar un array dado y aplicar sobre este el algoritmo:

17 elem: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]:



Heapify comienza de derecha a izda, y de abajo arriba

Los rectángulos indican el nº de veces (mínimo) que se aplica.

Se aplica 8 veces.

ii). $\lg^* 179$.

Para calcular cualquier $\lg^*$ debemos aplicar $\lg_2$ al resultado hasta obtener un número menor que 1.

El número de veces que hayamos hecho la operación, será el resultado de $\lg^*$:

$\lg^* 179 \simeq 4$

① $\lg 179 \Rightarrow$ ② $\lg 7,\text{algo} \Rightarrow$ ③ $\lg 2,\text{algo} \Rightarrow$ ④ $\lg 1,\text{algo} = 0, \ldots$

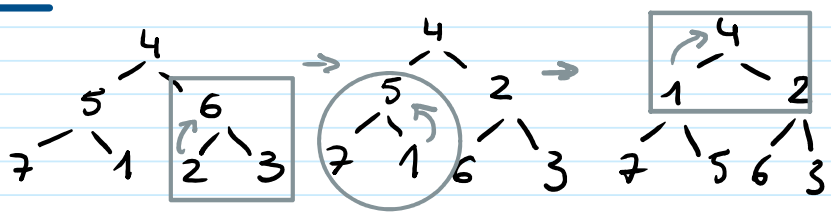
WUOLAH

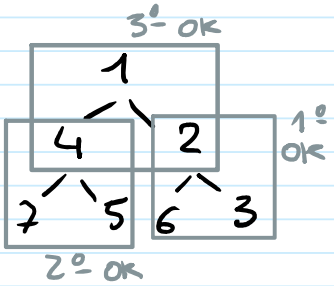
Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

→ SI NO PONE IN PLACE, HAY QUE INSERTAR LOS ELEMENTOS 1 A 1. (PQ)

2. (3 points). We want to create a min heap in place from the list [4, 5, 6, 7, 1, 2, 3]. Indicate how such a creation is carried out and represent the state of the heap (draw it as a tree) after each step. Give the final representation of the heap as a list.

Colocamos el array:





Cada padre es menor que sus hijos ✓

Array final: [1, 4, 2, 7, 5, 6, 3].

Si no lo poneis quitan puntos :(

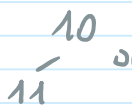
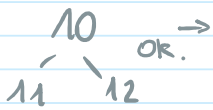
3. (5 points). We are creating a priority queue (PQ) based on the list $l = [10, 11, 12, 7, 6, 5, 4]$ using a min heap as a data structure.

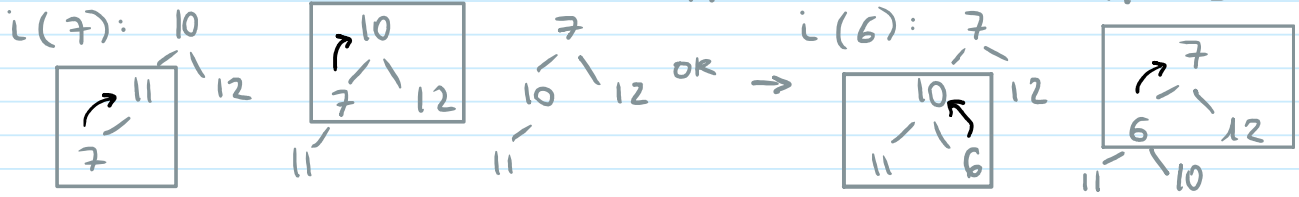
i. Create the PQ starting with an empty heap and inserting the elements one by one. Indicate the state of the heap after each insertion.

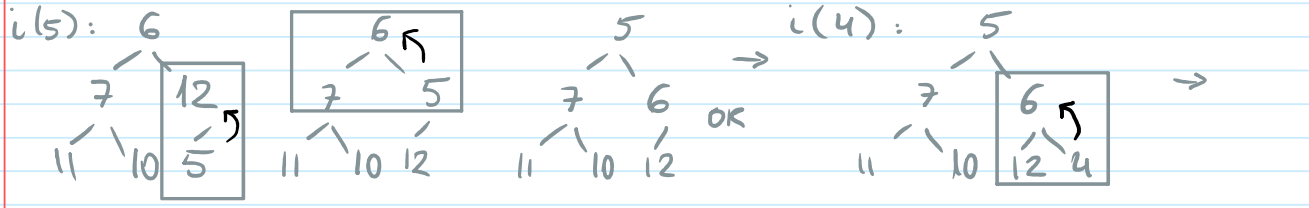
ii. Extract the first element of the PQ and then insert element 9. Indicate the state of the heap after the extraction and after the insertion.

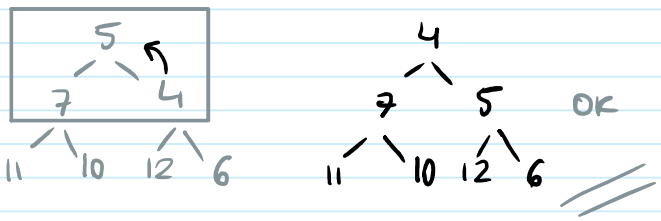
iii. From the PQ resulting in ii, extract the first element and then insert element 2. Indicate the state of the heap after the extraction and after the insertion.

Una cola de prioridad consiste en hacer un min heap NO IN PLACE, insertando los elementos uno a uno, y aplicar heapify:

i) PQ: i(10): 10 → i(11):  → i(12): 







Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

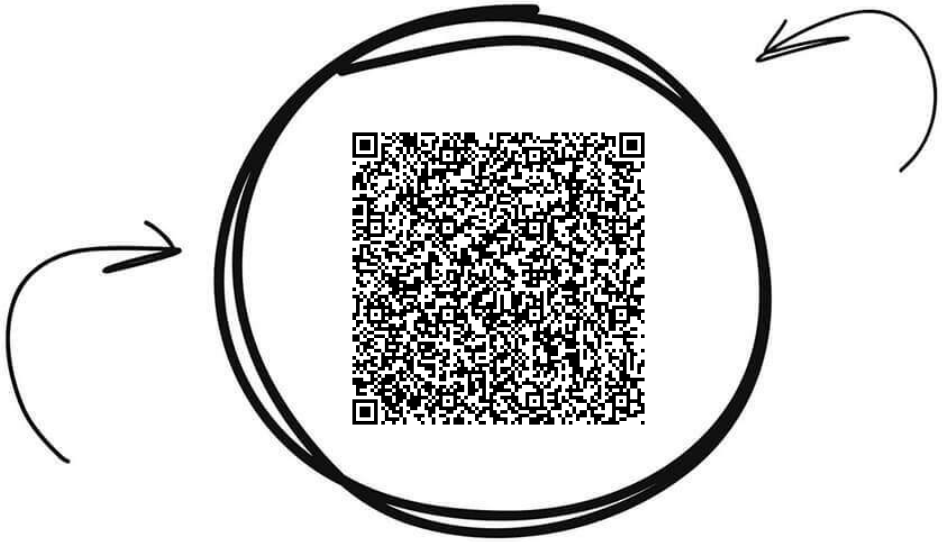
WUOLAH

Prueba el Menú Burrito San Diego por solo 4,95€ en Taco Bell ¡Clic aquí!

QUE LOS EXÁMENES NO TE ASUSTEN

ESCAQUÉATE CON CHEETOS

Algoritmia y Estructuras de...



Banco de apuntes de la

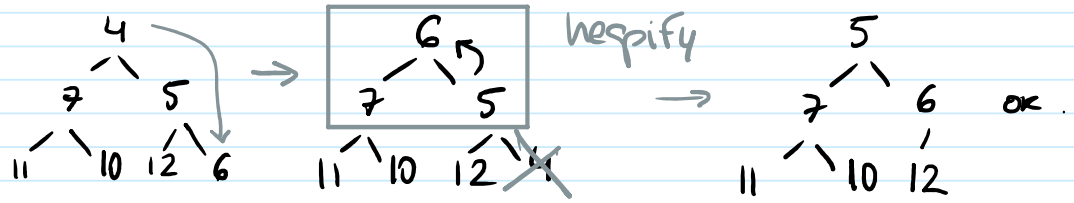
WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

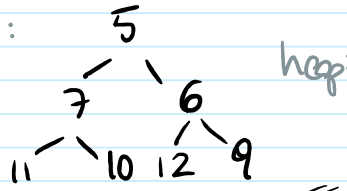
1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

Para extraer el 1er elemento, lo cambiamos por el último y aplicamos heapify:

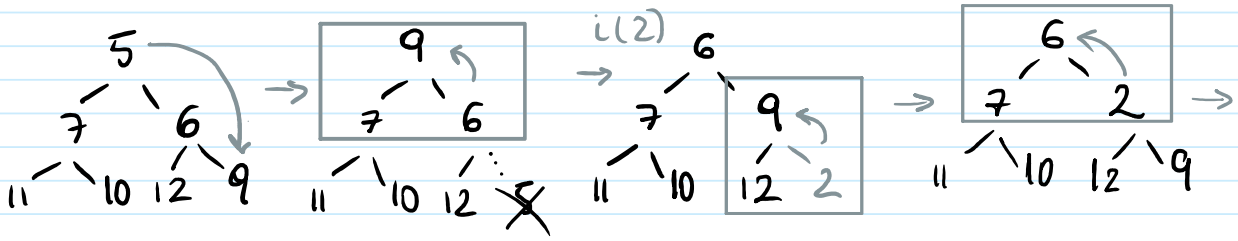
ii) Extraer 1er elem e insertar 9.

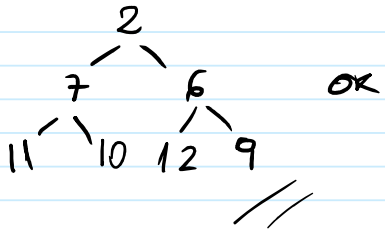


i(9):



iii) Extraer 1er elemento e insertar 2:





PARTE 2:

b. (4 points) The list $[5, 6, 5, 8, -3, -2, -3, -2]$ represents a disjoint set (DS) with indices between 1 and 8. On this list we are going to execute `find` with path compression.

a) Represent the DS as a family of trees (forest).

b) On this family of trees, execute `find(2)`, then `find(8)` and finally apply `union` to the result of the two finds. Show the results.

c) On the result of b), execute `find(7)`, then `find(4)` and finally apply `union` to the result of the two finds. Show the results.

d) Sow the final result in the form of a list.

Show your work in enough detail to be able to follow the execution.

Los conjuntos disjuntos se representan en una lista, donde los números negativos son raíces, que indican la "profundidad" de su árbol.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Prueba el Menú Burrito San Diego por solo 4,95€ en Taco Bell ¡Clic aquí!

→ 5, 6 y 8 son raíces.

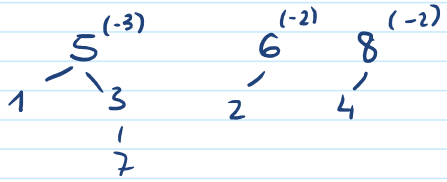
a) lista: [5, 6, 5, 8, -3, -2, 3, -2]
1 2 3 4 5 6 7 8

↳ 5 es padre de 1 y de 3.

→ 6 es padre de 2.

→ 8 es padre de 4. etc

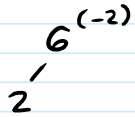
ÁRBOL:



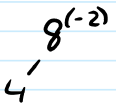
b) Find(2) y Find(8). Luego union(find(2), find(8)).

Find de un número devuelve su representante (raíz)

Find(2): 6 →



Find(8): 8 →



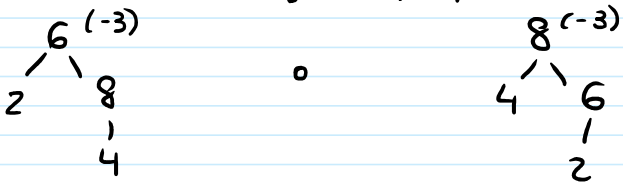
Hay que volver a mostrar los árboles, por si cambian.

Al hacer find con compresión de caminos, cada nodo que encontremos de camino hasta la raíz, será hijo directo. [EJEMPLO EN EL SIGUIENTE APARTADO] ⊛

union(6, 8): DE LAS RAÍCES (que son resultado de Find)

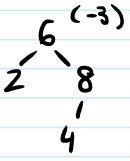
Para hacer la unión de 2 conjuntos, buscaremos obtener siempre la profundidad mínima.

En este caso nos da igual porque obtenemos la misma profund.

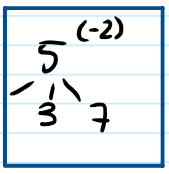


c) Al resultado anterior,

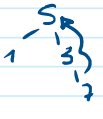
find(7): 5 →



(queda igual)



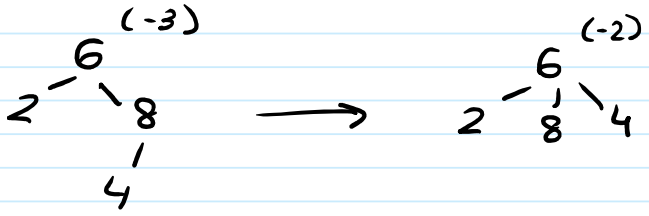
Compresión de caminos ⊛



WUOLAH

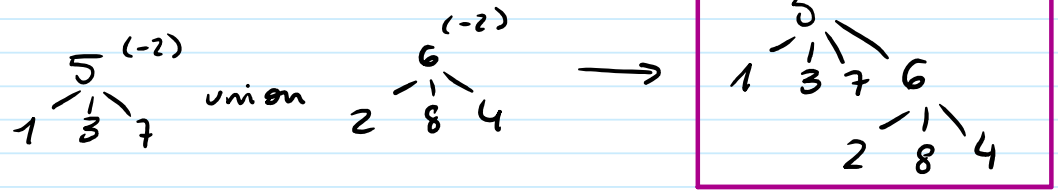
Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

c) Find(4) : 6



union(Find(7), Find(4)) =

union(5, 6) :

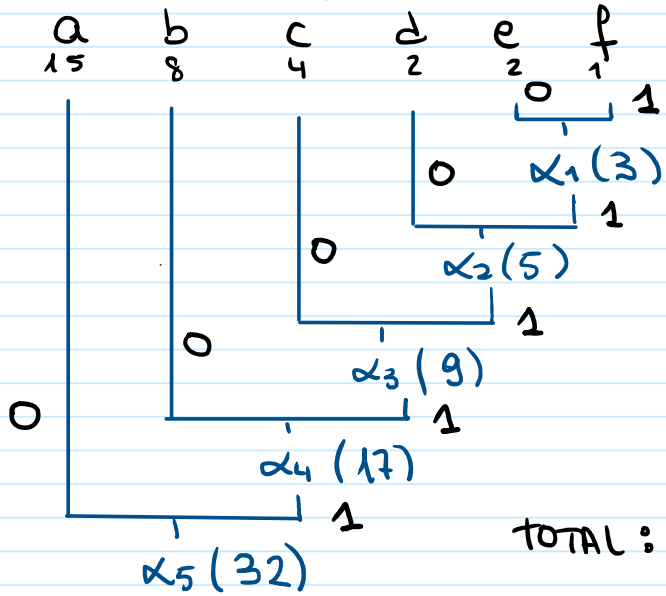


d) Dar resultado sobre una lista:

1	2	3	4	5	6	7	→ elementos
[5	6	5	6	-3	5	5]	→ sus padres (raíces en negativo)

✍ (4 points) A file contains the charachters a, b, c, d, e, f with frequencies $15, 8, 4, 2, 2, 1$. Give the Huffman and Shannon codes for this file. Give the size, in bit, of the resulting archives. Show your work in enough detail to be able to follow what you are doing.

Huffman: el algoritmo consiste en escoger cada vez las dos frecuencias más bajas y sumarlas:



a: 0 (1 bit)
b: 10 (2 bits)
c: 110 (3 bits)
d: 1110 (4 bits)
e: 11110
f: 11111
} 5 bits

TOTAL: $\sum$ frecuencia * nº bits:

$15 \cdot 1 + 8 \cdot 2 + 4 \cdot 3 + 2 \cdot 4 + 2 \cdot 5 + 1 \cdot 5 = 66$ bits.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Prueba el Menú Burrito San Diego por solo 4,95€ en Taco Bell ¡Clic aquí!

Shannon:

Tenemos que separar los números de tal forma que haya la menor diferencia entre cada una de las separaciones:
El 1º número es 15. Si sumamos todos los demás $8+4+2+2+1 = 17$. Se acercan.
Si cogiésemos $15+8 = 23$ por un lado, y los demás $4+2+2+1 = 9$. Hay mayor diferencia, por lo que separaremos 15 de los demás frecuencias. Y le asignamos 0, mientras que a las demás no elegidas, 1.

a	15	0					
b	8	1	10				
c	4	1	11	110			elegimos 4 y 5
d	2	1	11	111	1110		ahora 2 y 3
e	2	1	11	111	1111	11110	2 por un lado
f	1	1	11	111	1111	11111	1 por el otro.

de antes

ahora, elegimos 8 por un lado y $4+2+2+1 = 9$

En este caso obtenemos la misma codificación que con Huffman, pero no tiene por qué ser así.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

Prueba el Menú Burrito San Diego por solo 4,95€ en Taco Bell ¡Clic aquí!