Titolo unità didattica: Strutture dati dinamiche gerarchiche [P2_09]

Titolo unità didattica: struttura dati heap [6-T]

Definizioni ed algoritmi di gestione

Argomenti trattati:

- ✓ Definizione e proprietà di un heap
- ✓ Rappresentazione di un heap mediante array
- ✓ Algoritmo di ripristino della “proprietà heap” su un nodo

Prerequisiti richiesti: alberi binari

Struttura dati HEAP

Un heap è un albero binario quasi completo i cui nodi sono etichettati tramite chiavi (da un insieme ordinato).

Proprietà heap:

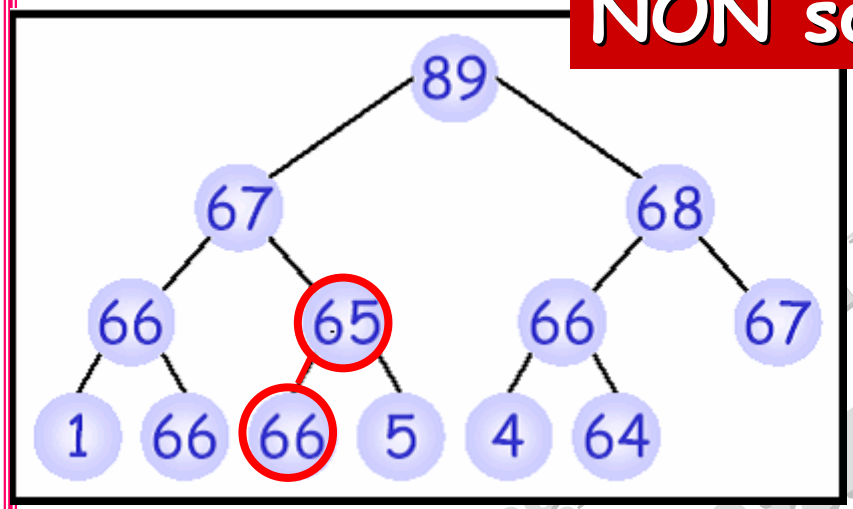
Se x è un qualsiasi nodo dell'heap (ad esclusione della radice) si ha

$$key(x) \leq key(padre(x))$$

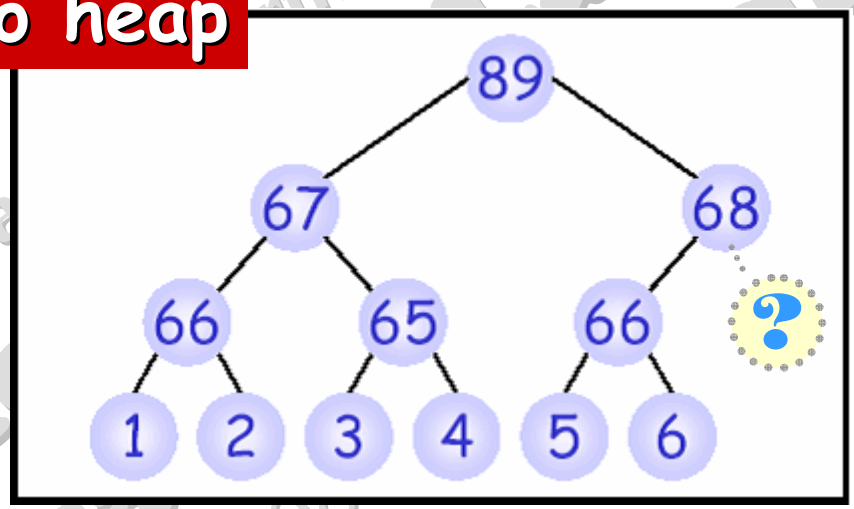
Ne consegue che in un heap l'elemento con valore massimo è memorizzato nella radice.

Esempi

NON sono heap

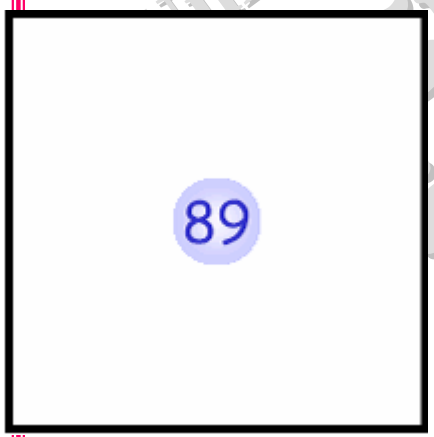


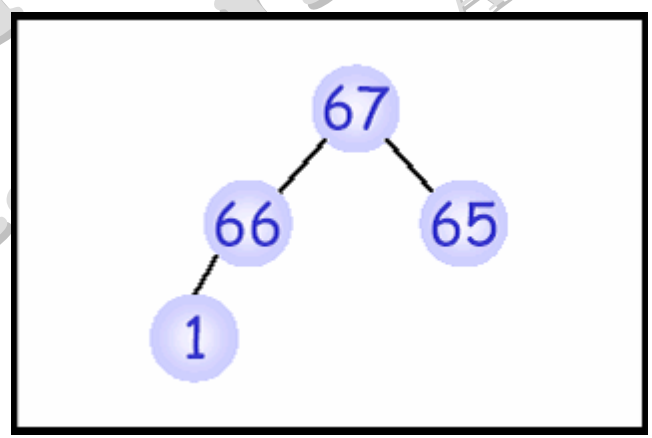
non verifica la proprietà heap

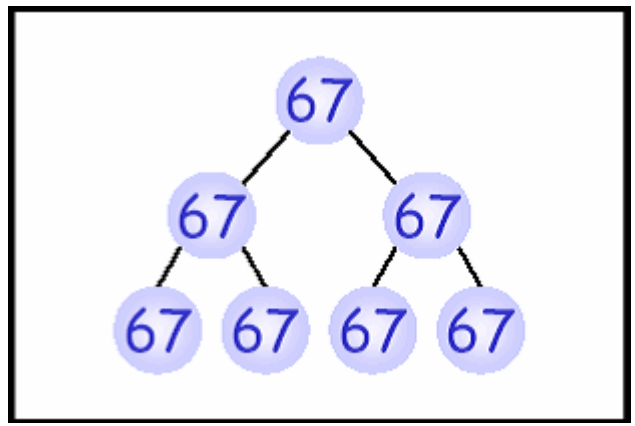


non è un albero quasi completo

sono heap







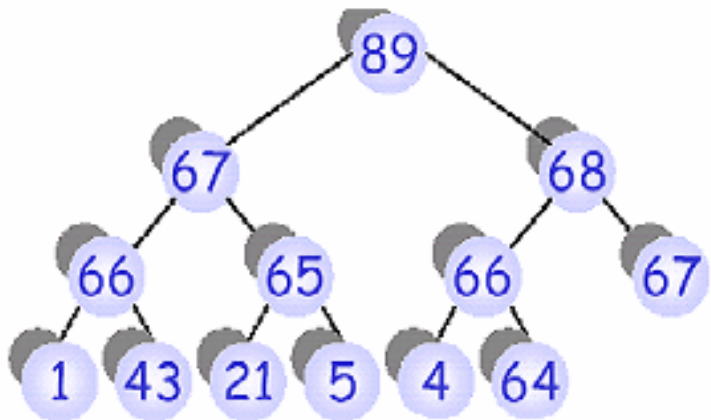
HEAP: rappresentazione tramite array

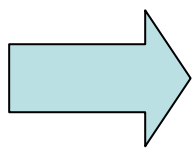
Essendo un albero binario, un heap può essere memorizzato in un array con solite le relazioni:

$\forall\ i \neq 1$ $\quad padre(a_i) = a_{\lfloor i/2 \rfloor}$

$\forall\ i \leq \lfloor n/2 \rfloor$ $\quad figlio\_sx(a_i) = a_{2i}$

$\forall\ i \leq \lfloor (n-1)/2 \rfloor$ $\quad figlio\_dx(a_i) = a_{2i+1}$





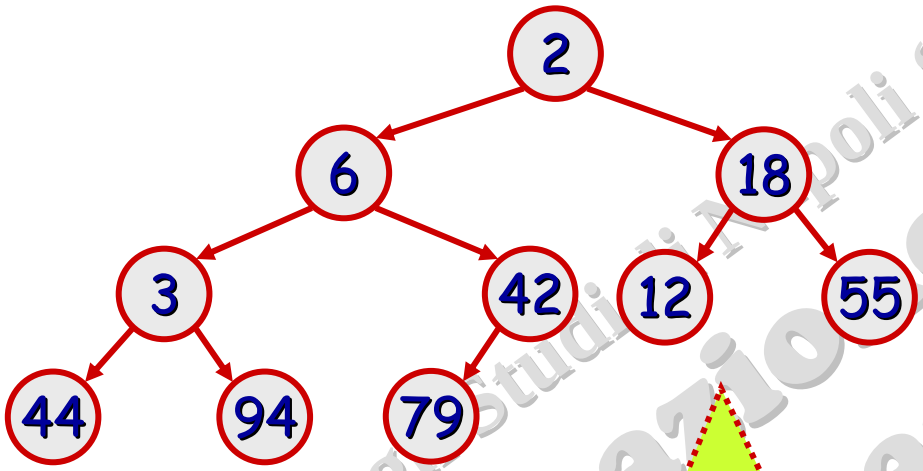
array a

1	89
2	67
3	68
4	66
5	65
6	66
7	67
8	1
9	43
10	21
11	5
12	4
13	64

Come trasformare un array in un heap?

albero binario quasi completo rappresentato su array

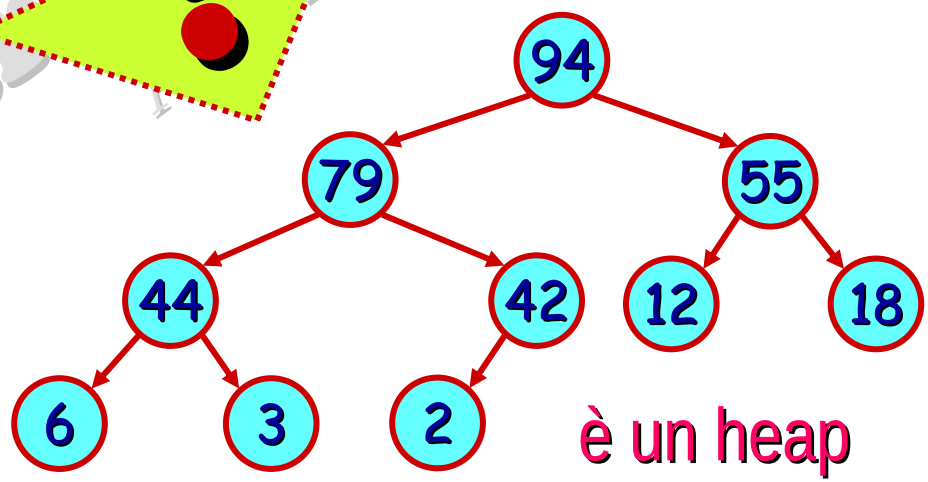
2
6
18
3
42
12
55
44
94
79



non è un heap

heap rappresentato su array

94
79
55
44
42
12
18
6
3
2

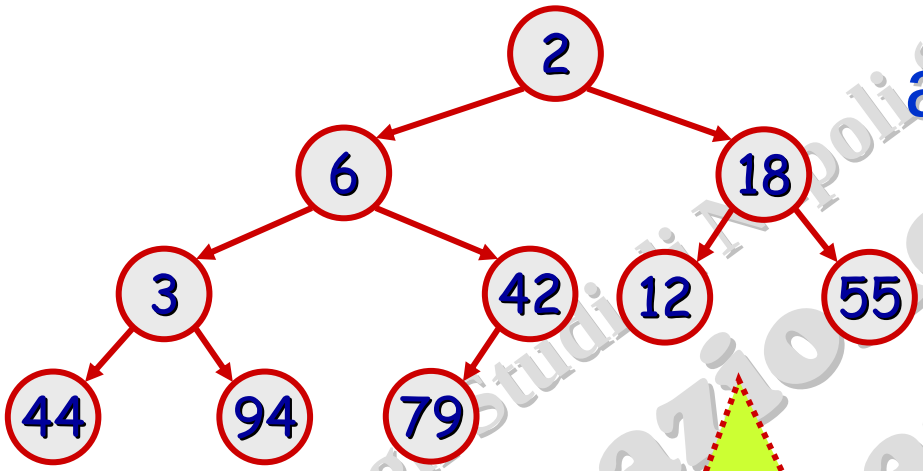


è un heap

L'heap non è unico!

a albero binario rappresentato su array

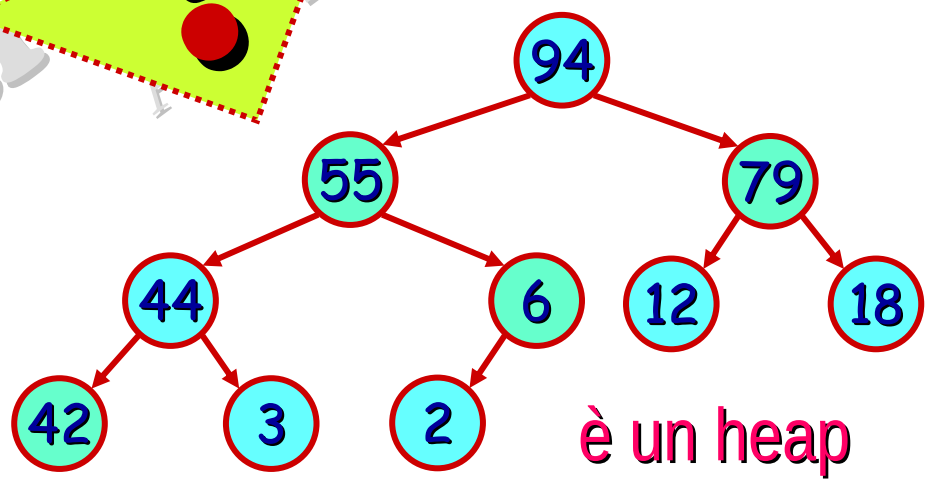
2
6
18
3
42
12
55
44
94
79



non è un heap

a heap rappresentato su array

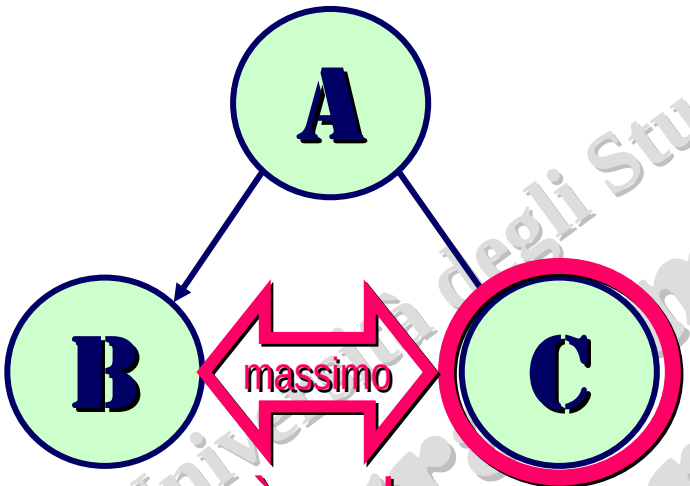
94
55
79
44
6
12
18
42
3
2



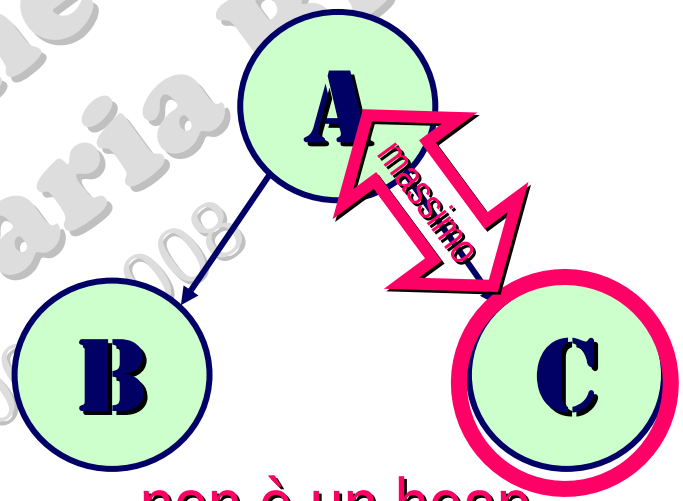
è un heap

Operazione base: procedura Heapify

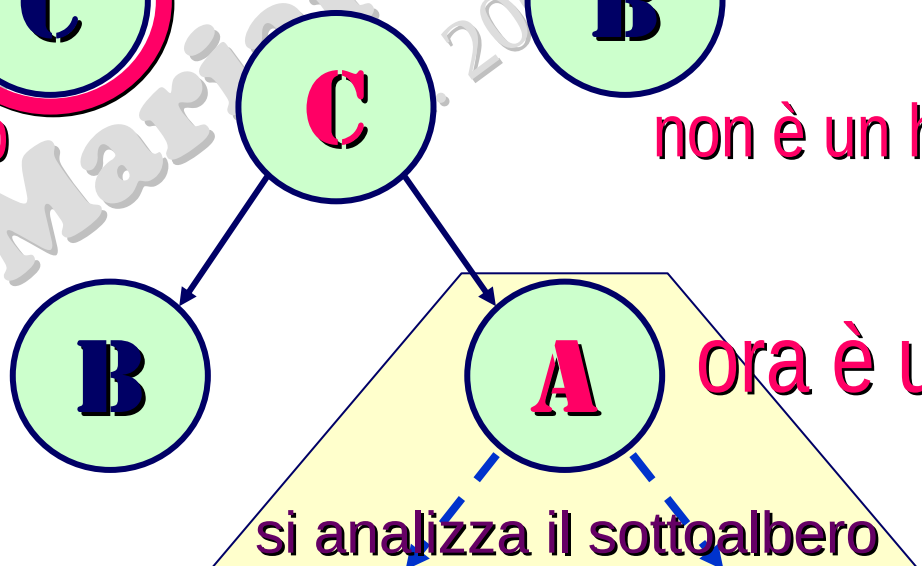
Per ripristinare la proprietà heap si considera un nodo ed i suoi figli: si determina il figlio con chiave massima e si scambia il valore della chiave fra padre e tale figlio se non verificano la proprietà heap. Se è avvenuto lo scambio, si scende poi nel relativo sottoalbero per ripristinare la proprietà heap.



non è un heap



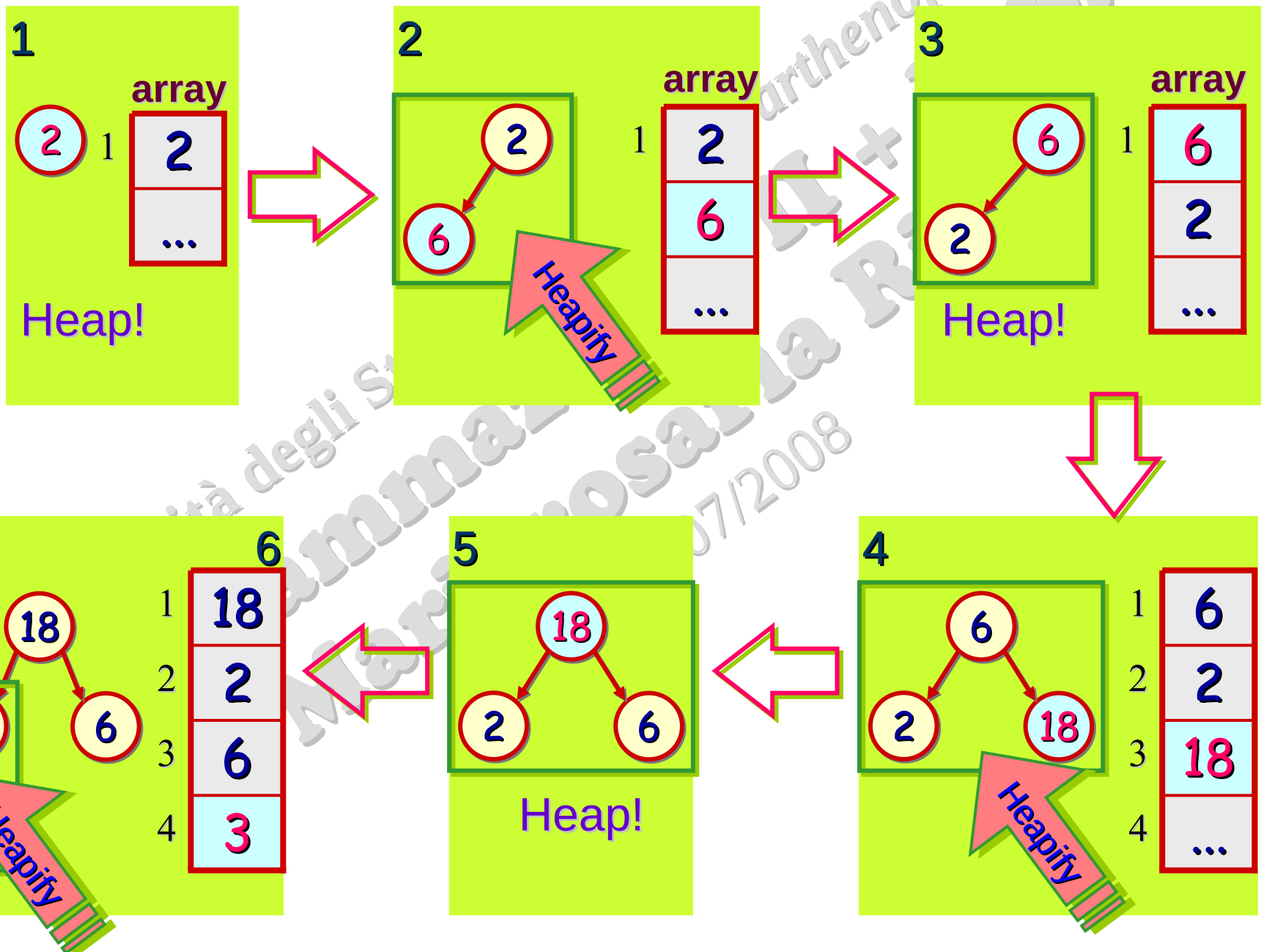
non è un heap



ora è un heap!

Costruzione di un heap: idea dell'algoritmo

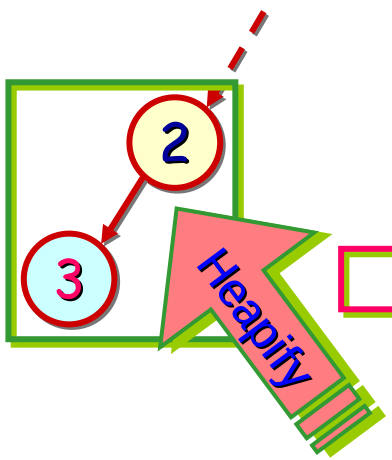
ordine di arrivo dei dati
2
6
18
3
42
12
55
44
94
79

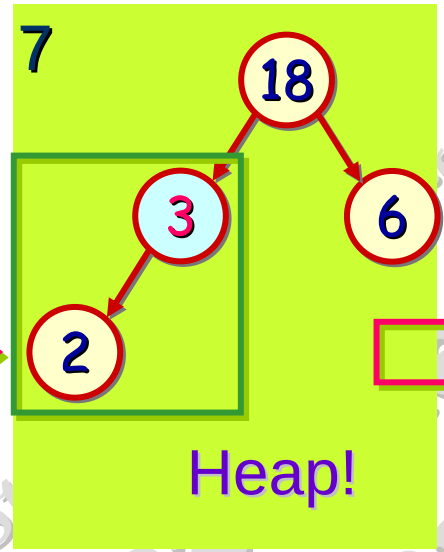


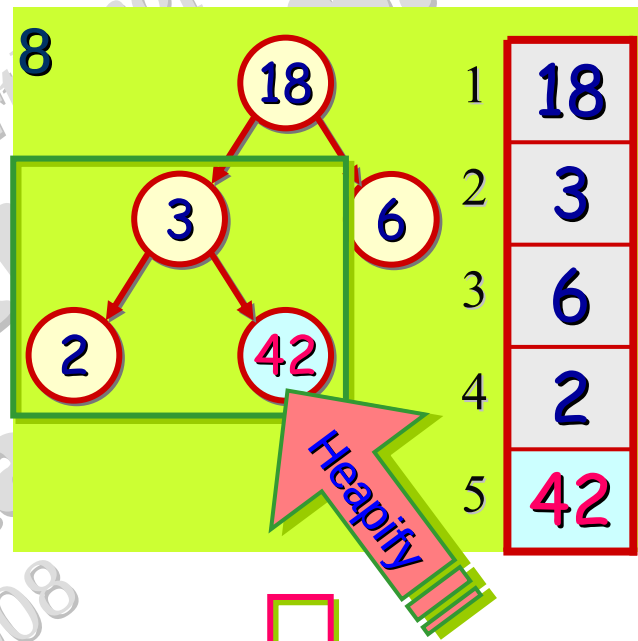
Costruzione di un heap

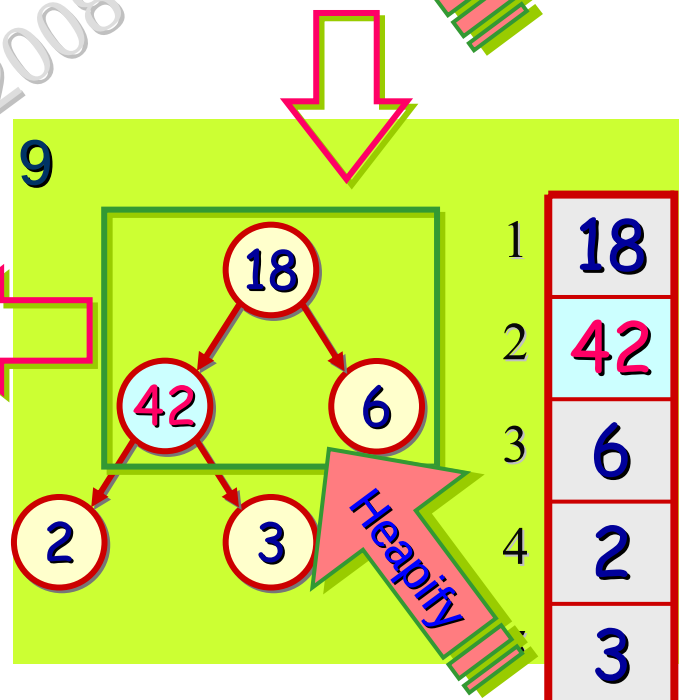
1	2
2	6
3	18
4	3
5	42
6	12
7	55
8	44
9	94
10	79

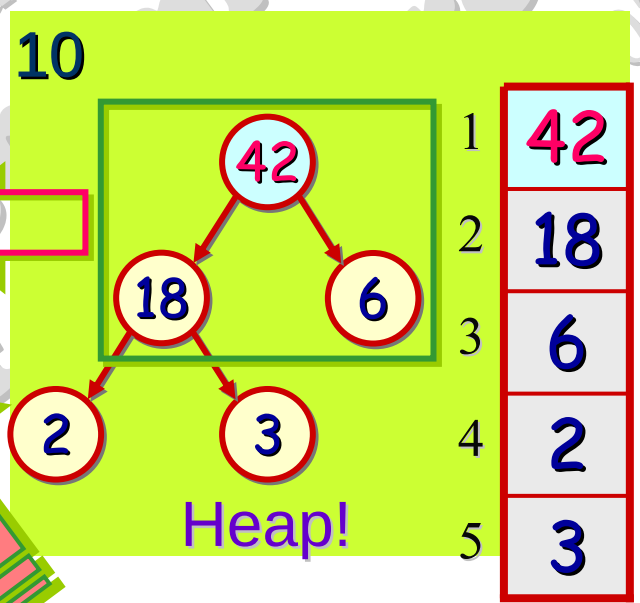
Ordine di arrivo dei dati

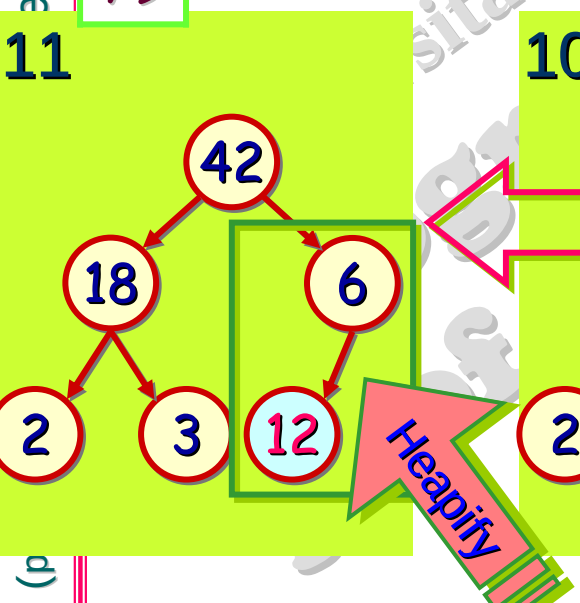










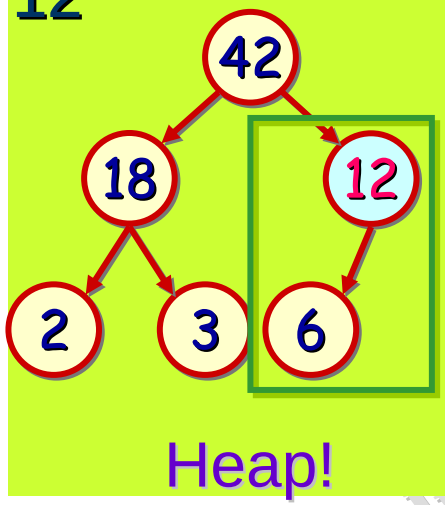


Costruzione di un heap

2
6
18
3
42
12
55
44
94
79

ordine di arrivo dei dati

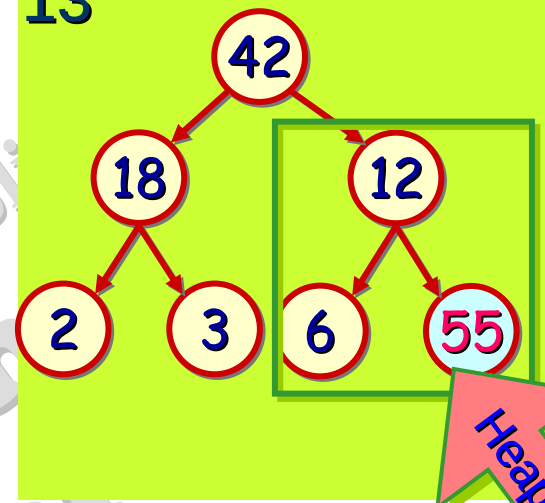
12



Heap!

1	42
2	18
3	12
4	2
5	3
6	6

13



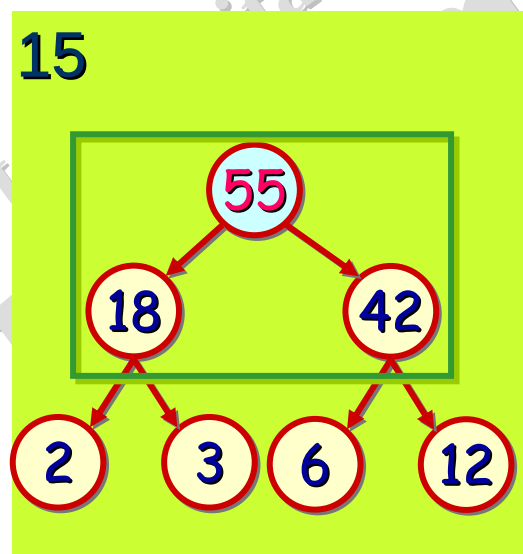
Heapify

1	42
2	18
3	12
4	2
5	3
6	6
7	55

14

1	42
2	18
3	55
4	2
5	3
6	6
7	12

15



1	55
2	18
3	42
4	2
5	3
6	6
7	12

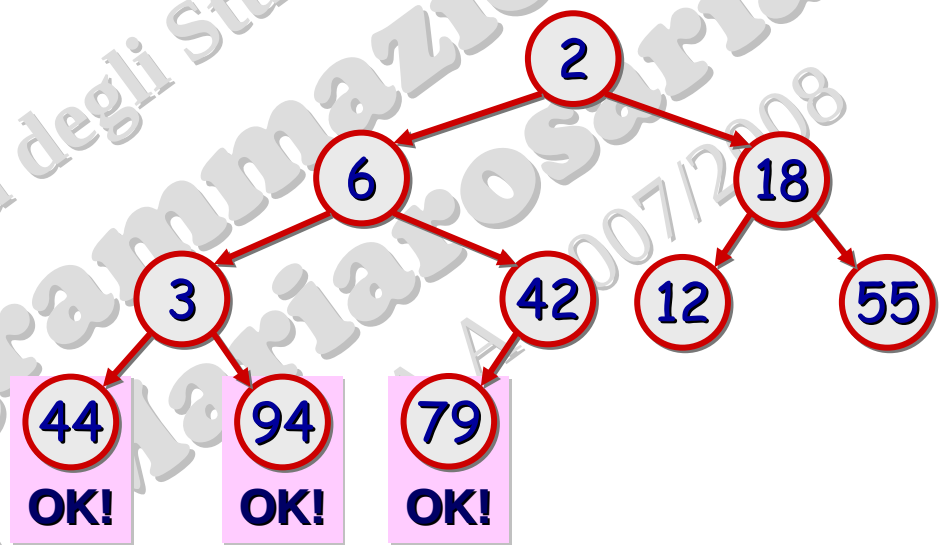
...

Come trasformare un albero binario in un heap?

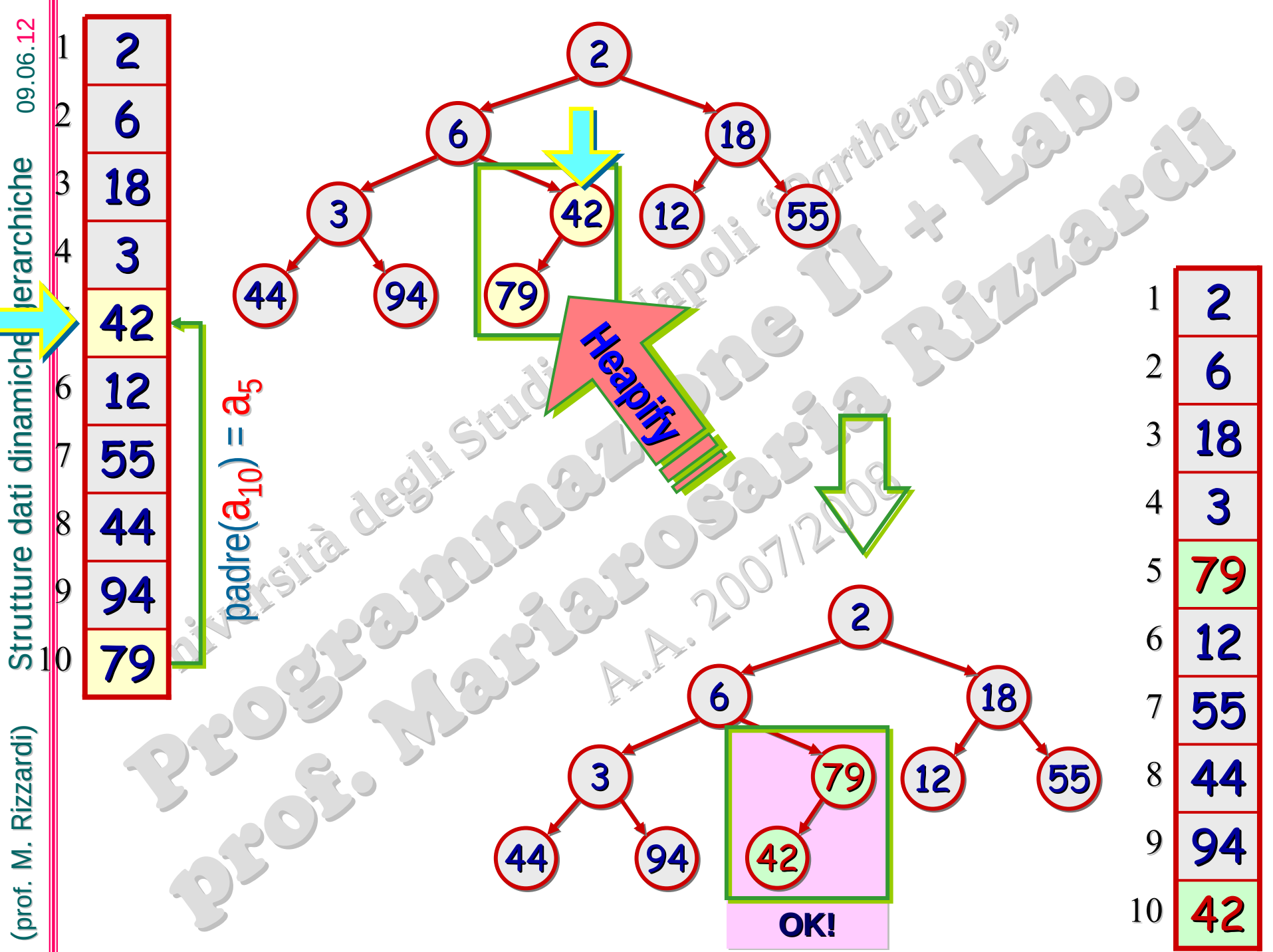
2
6
18
3
42
12
55
44
94
79

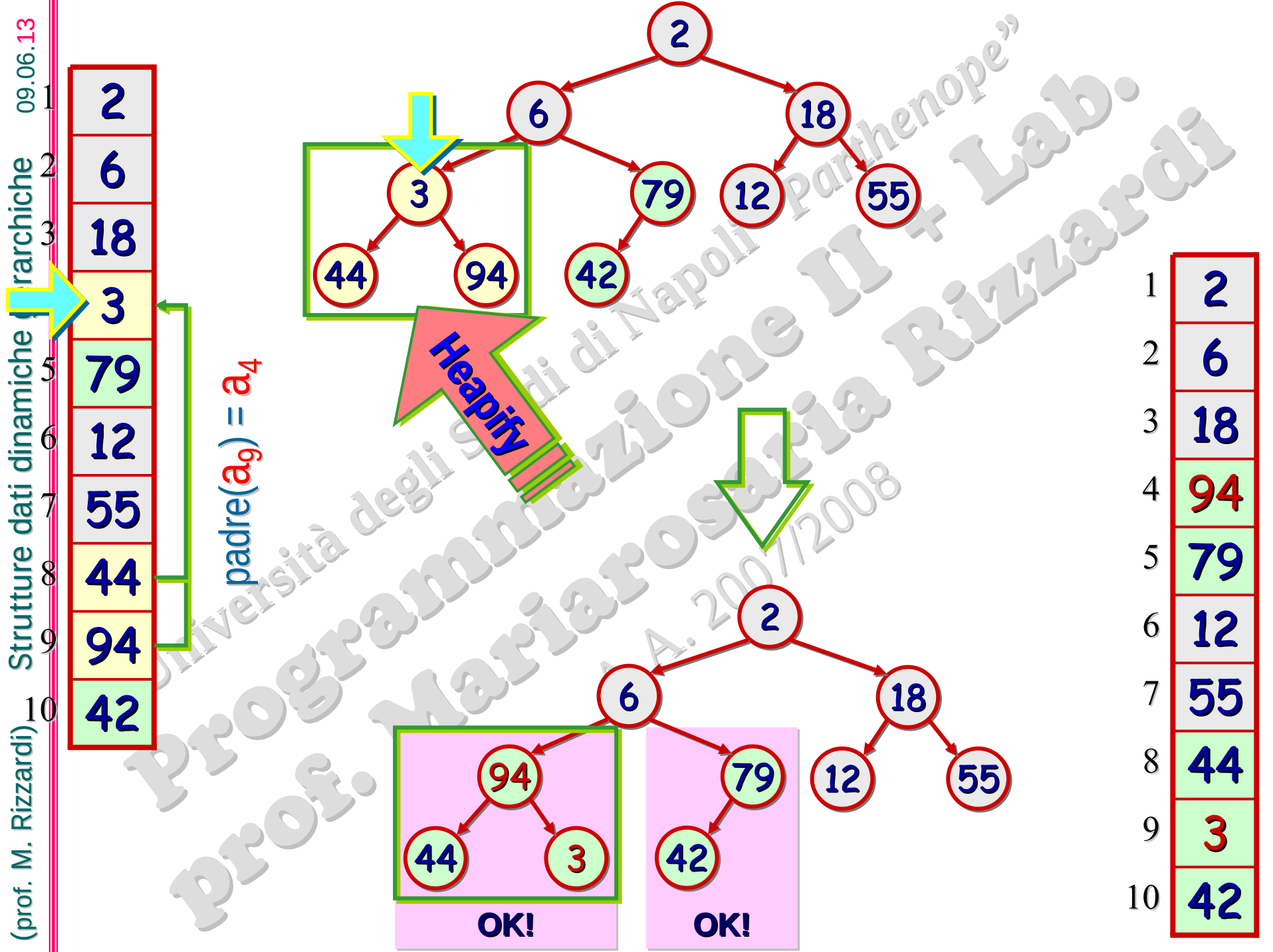
Si usa la procedura **Heapify*** in modo **bottom-up**, dai **livelli più bassi** dell'albero in su, per convertire l'array in un heap.

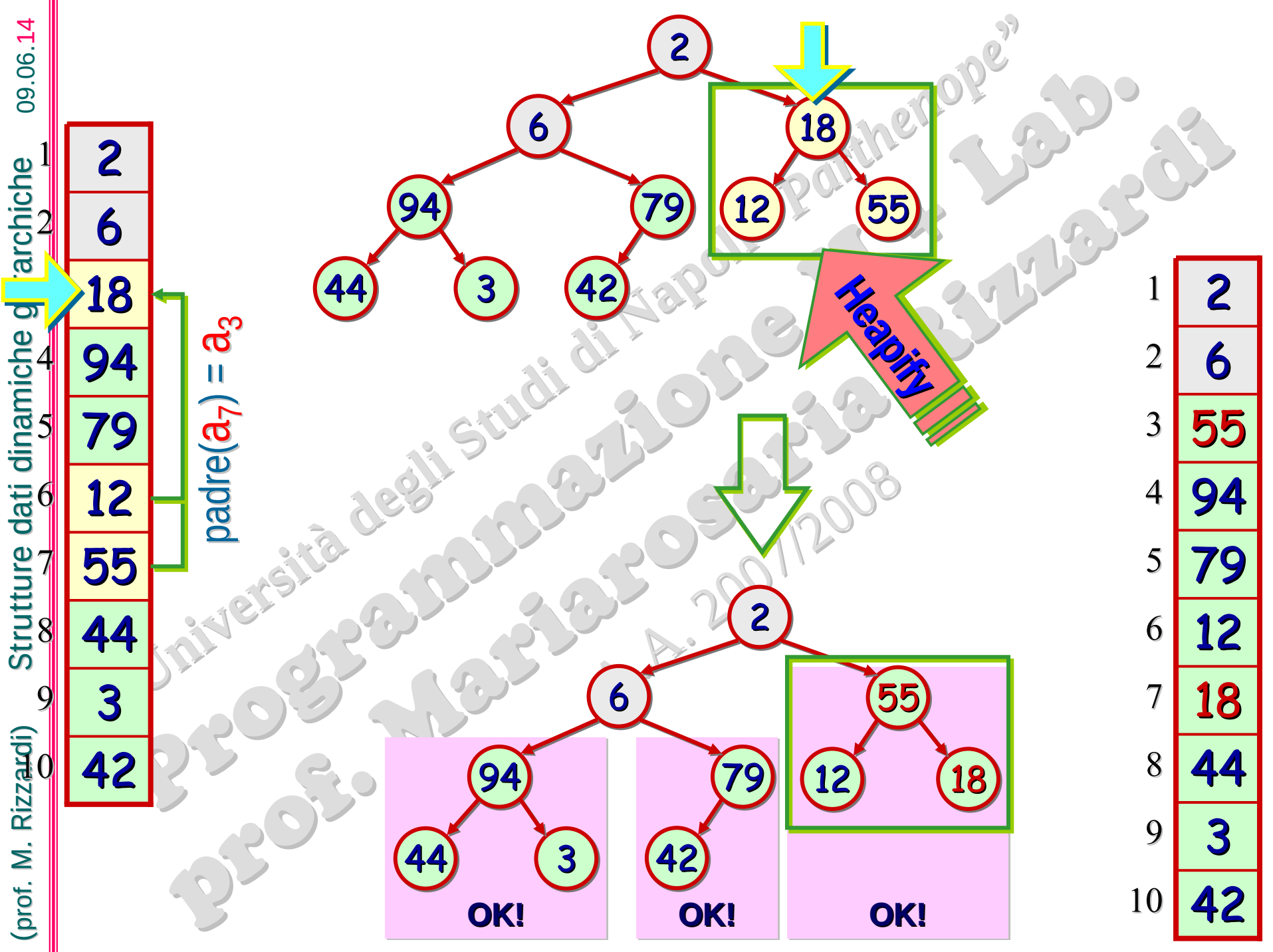
* Heapify ripristina la proprietà heap sui nodi

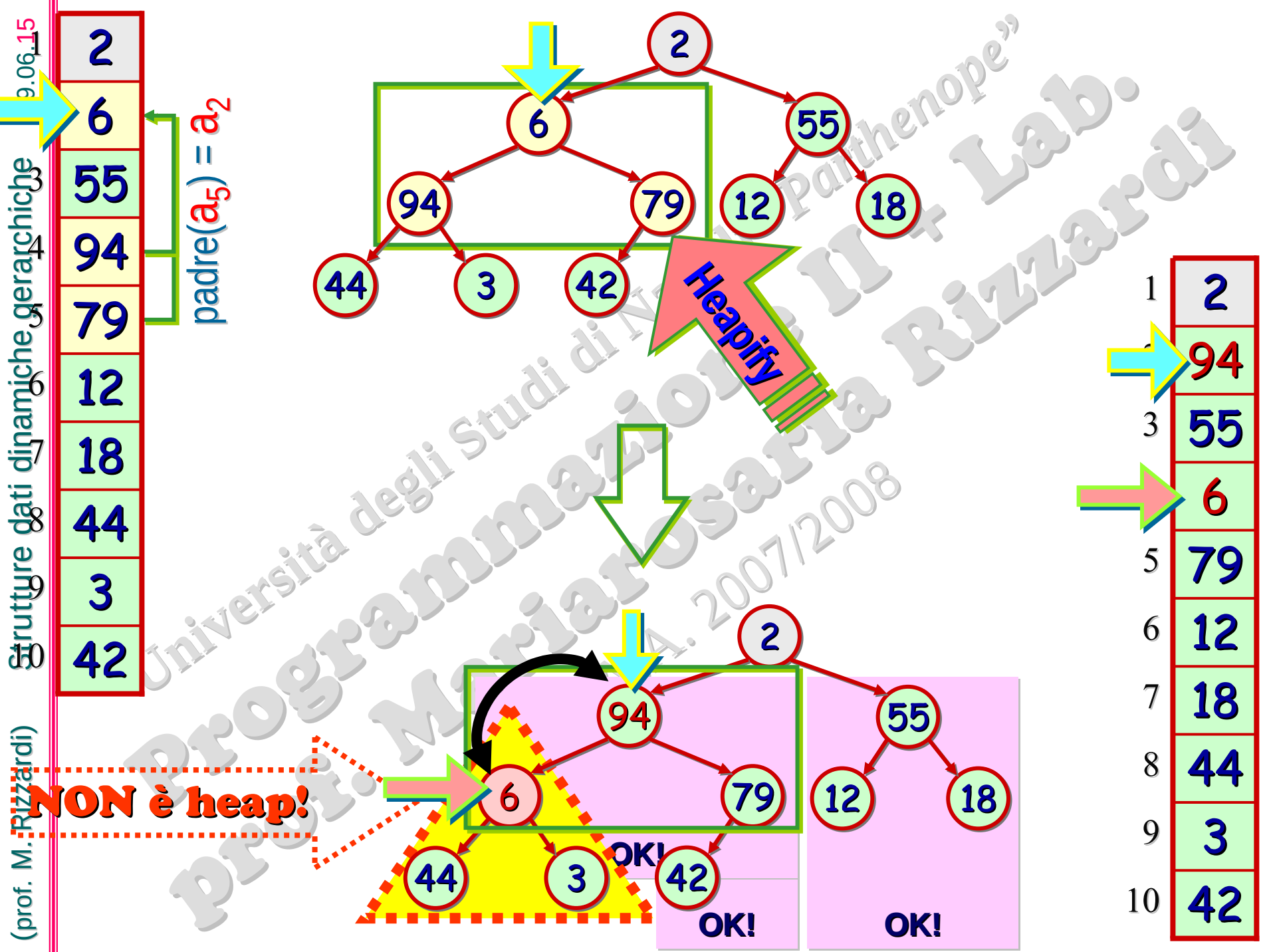


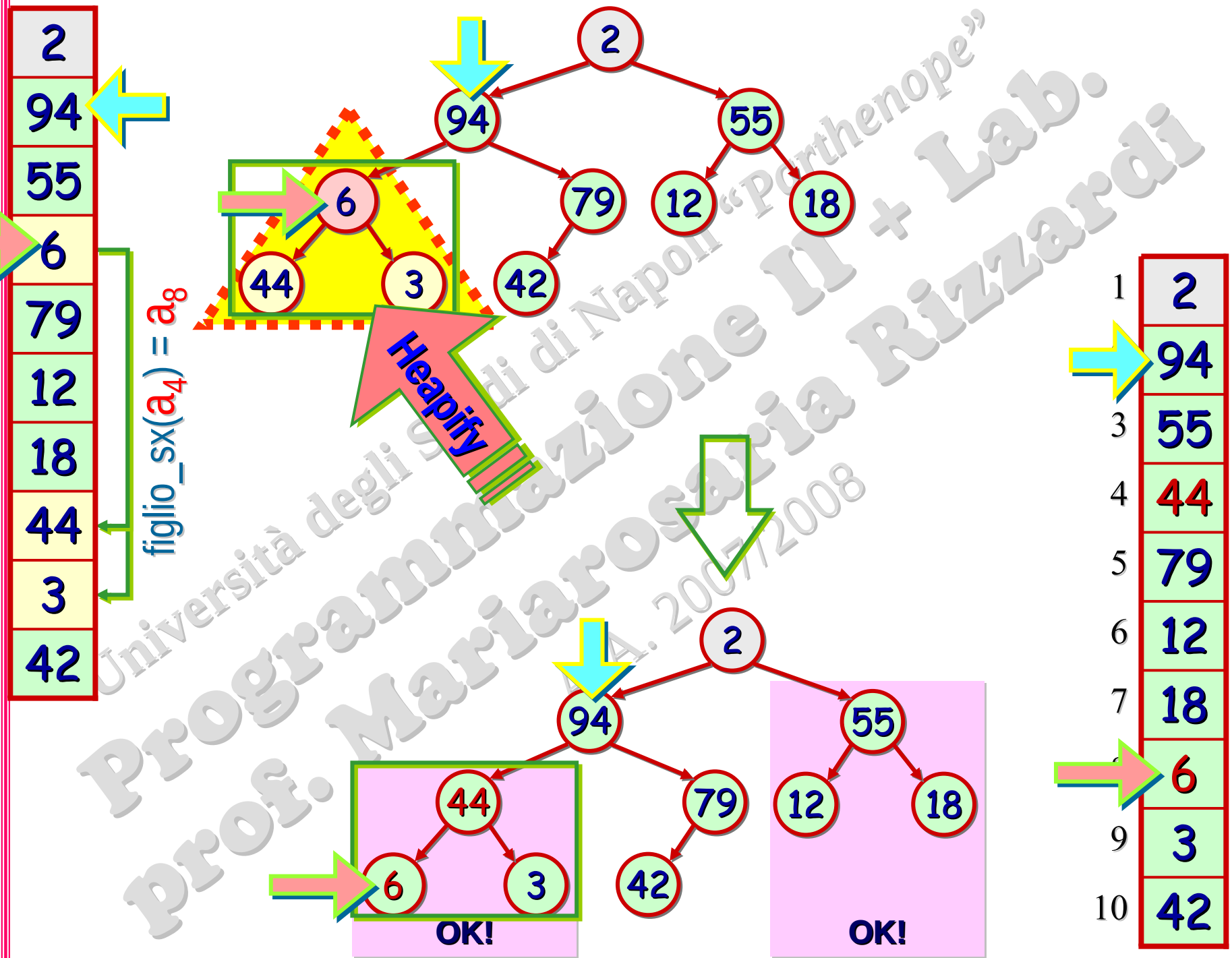
1) Poiché le singole foglie costituiscono ognuna un heap, si inizia ad applicare la procedura Heapify dal penultimo livello (dal padre dell'ultima foglia).

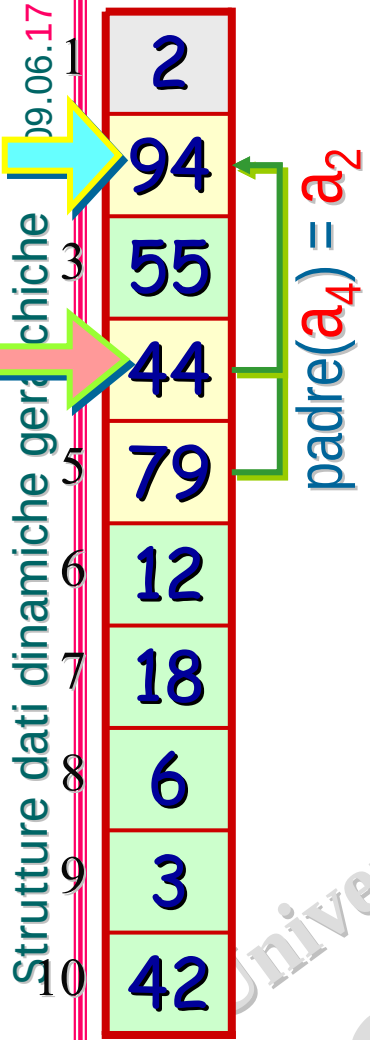




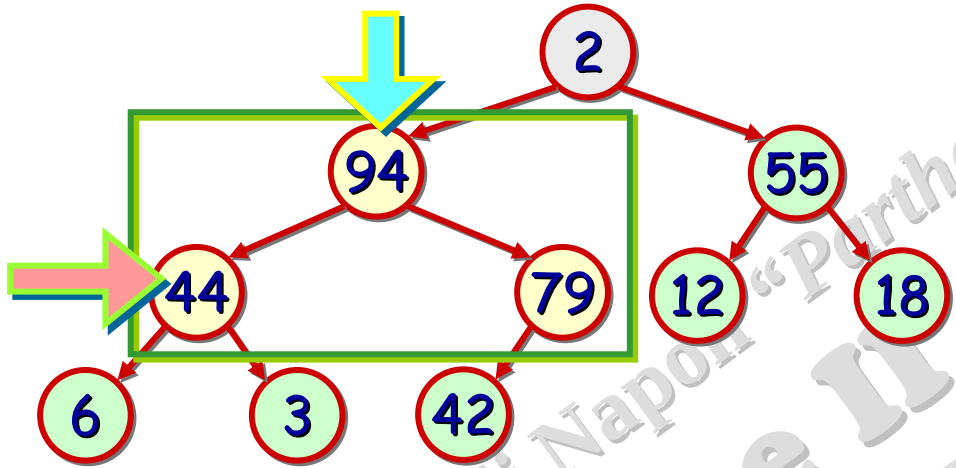




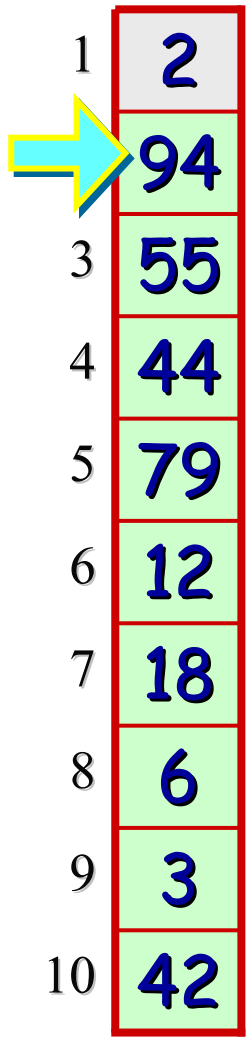


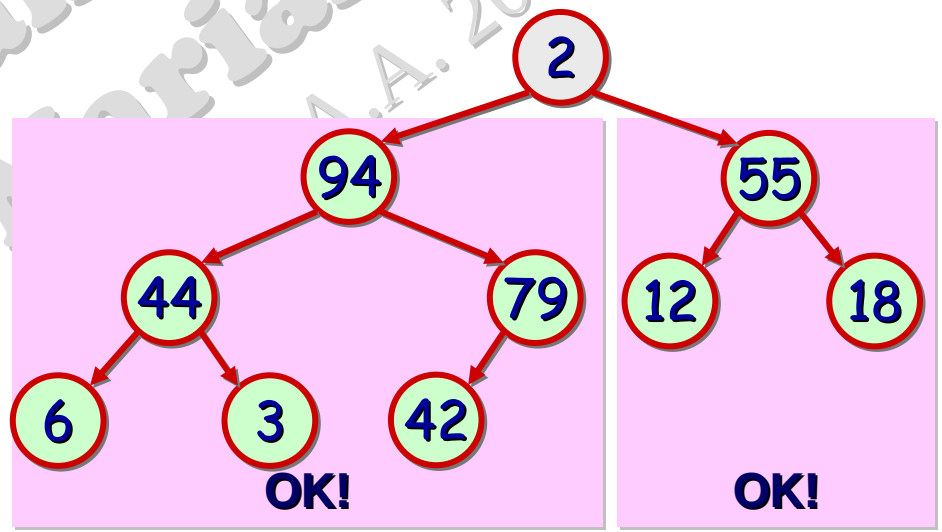


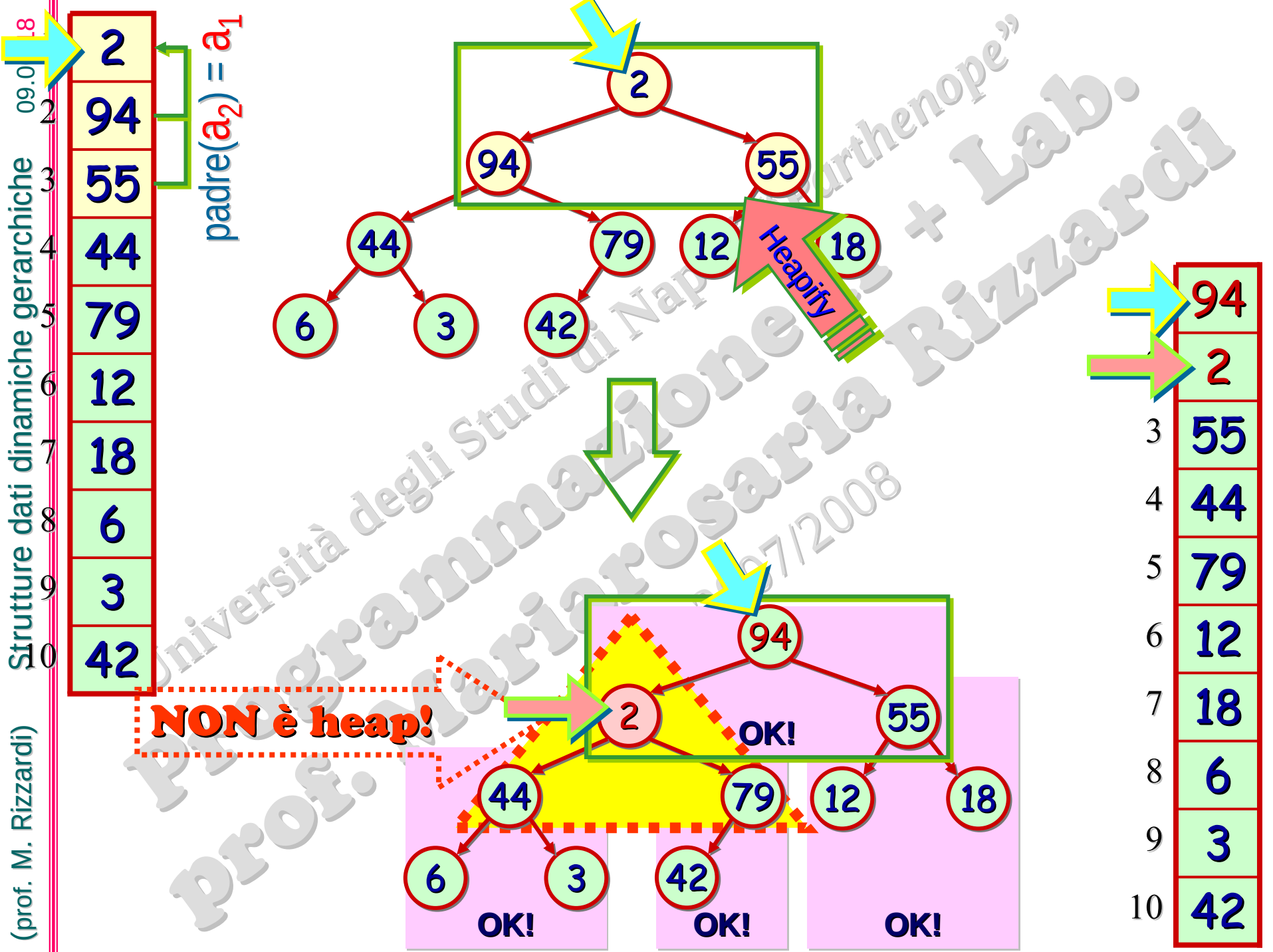
$padre(a_4) = a_2$

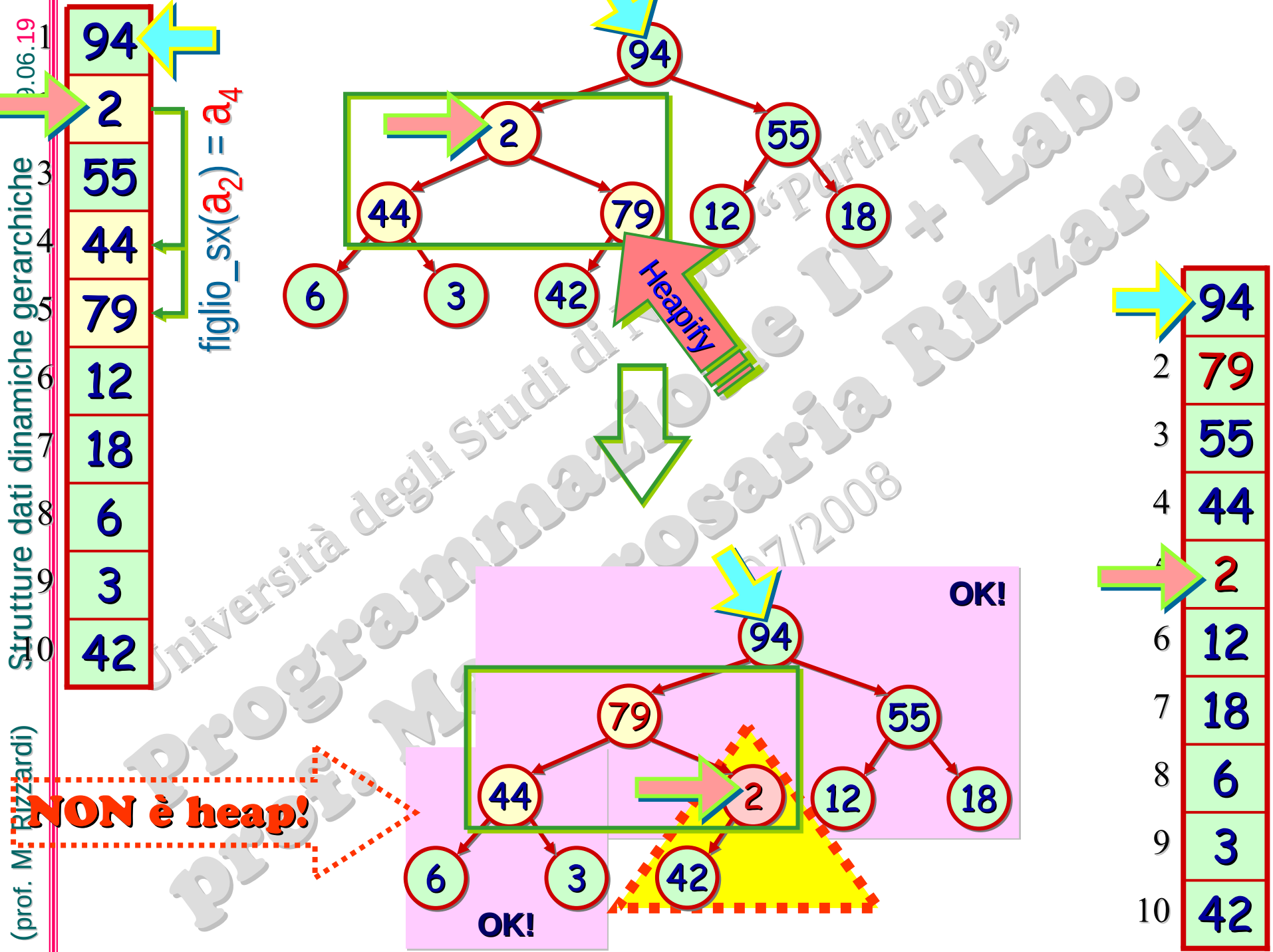


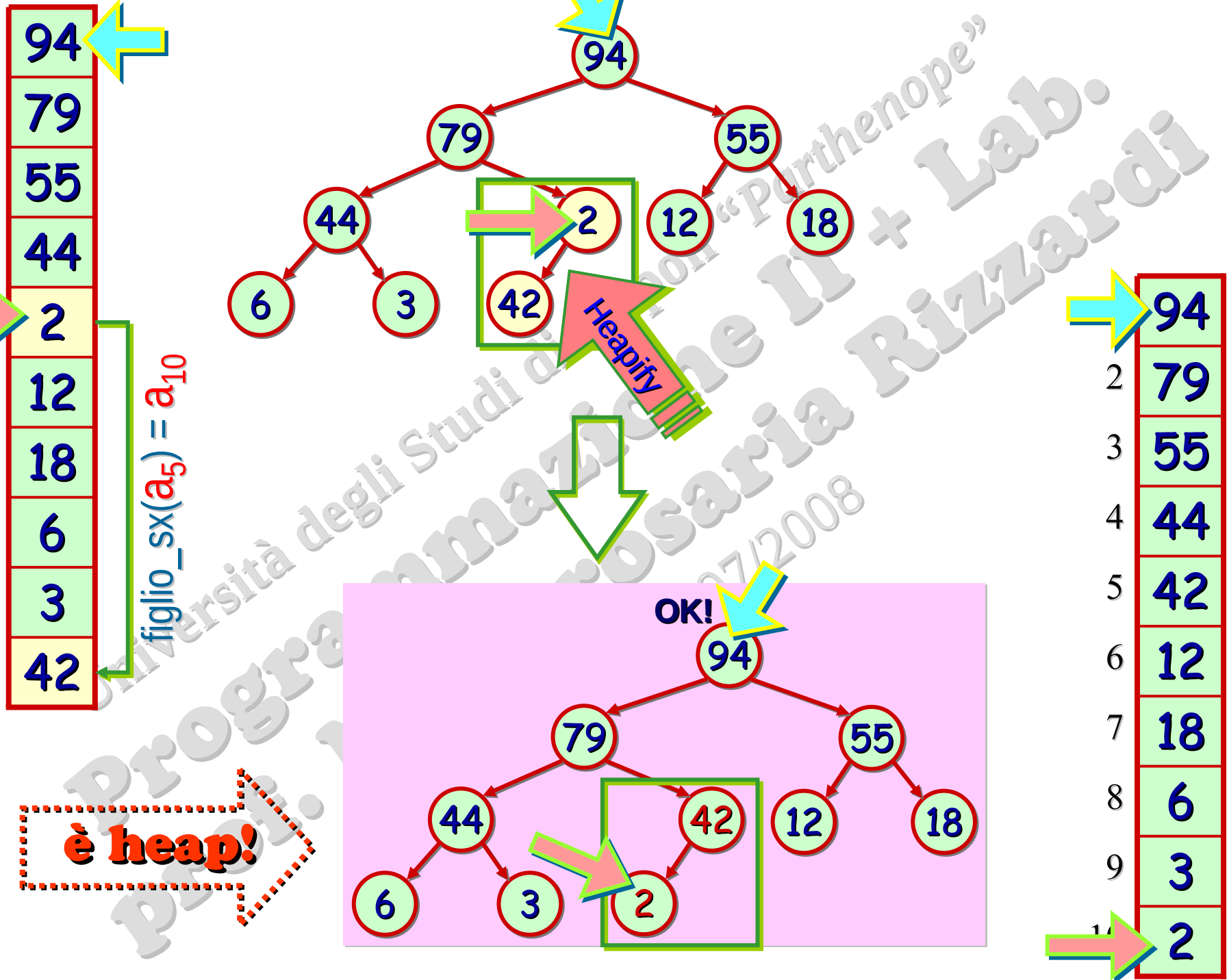
è heap!











Esercizi

1 Scrivere *function* C iterativa per la costruzione di un *heap* rappresentato mediante array.

[liv. 2]

2 Scrivere *function C* iterativa per la trasformazione in un *heap* di un albero binario rappresentato mediante array.

[liv. 3]