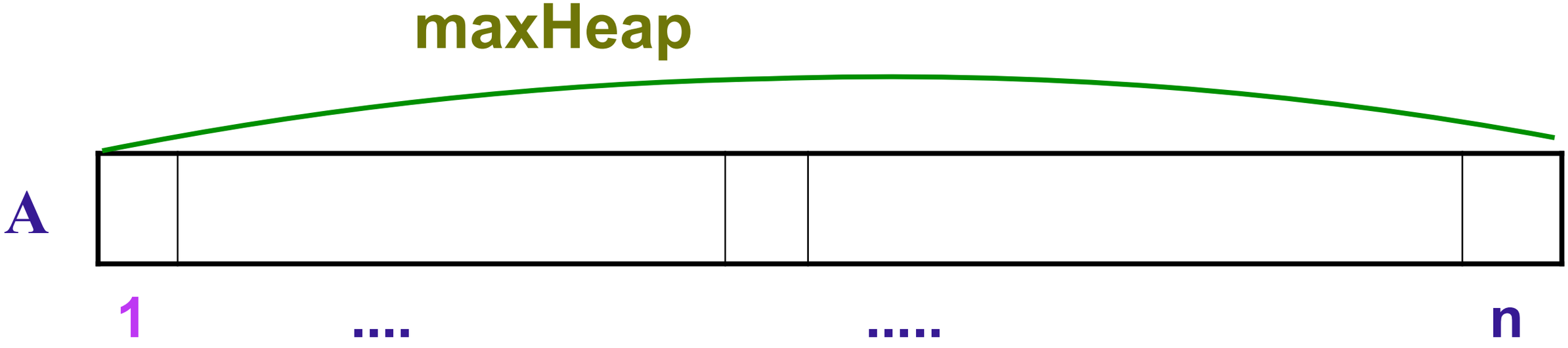
heapsort

[CLRS10] cap. 6, par. 6.1 - 6.4

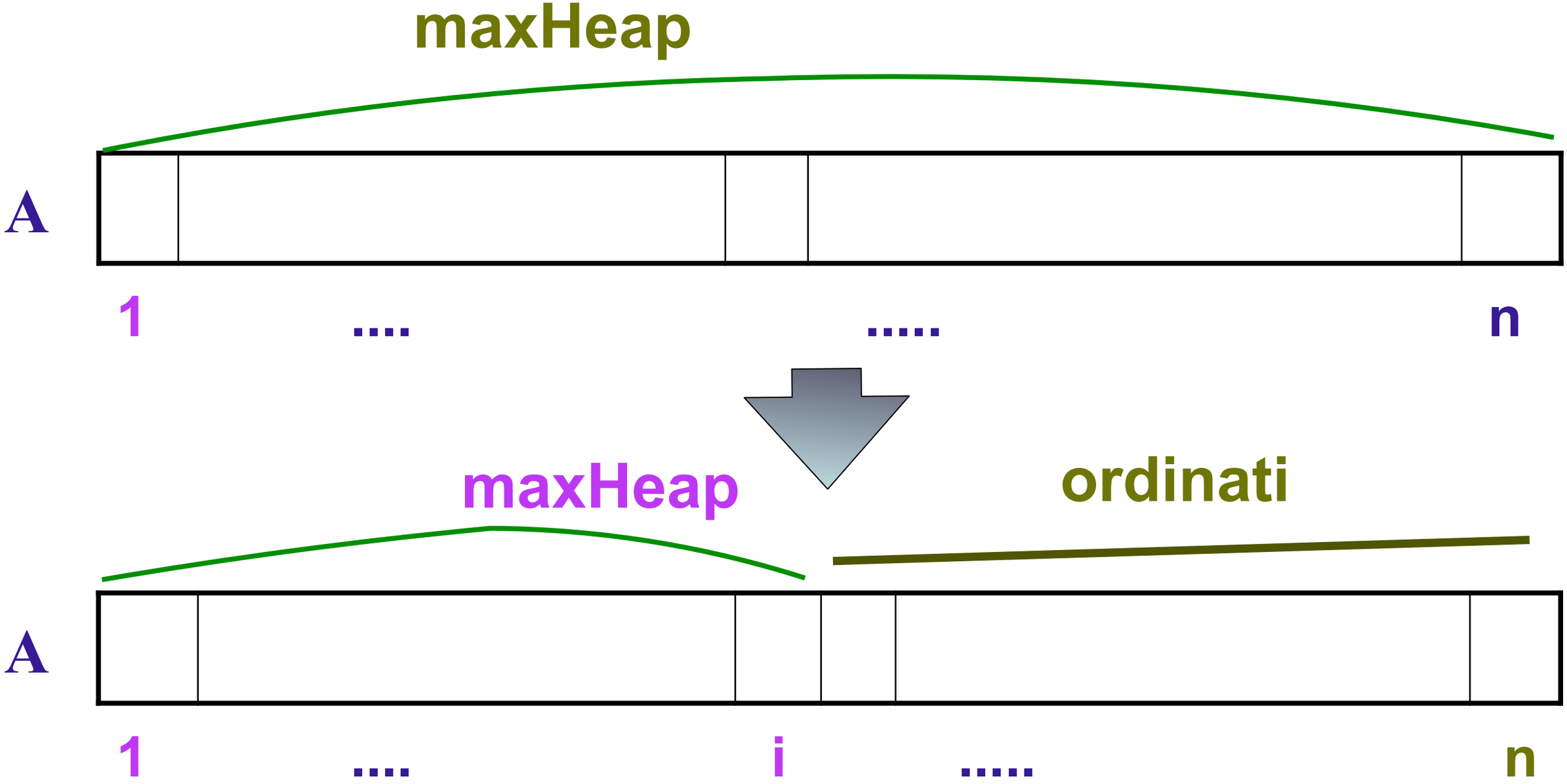
Heapsort

Heapsort: l'idea



In un maxHeap, il massimo si trova nella prima posizione mentre in un array ordinato in ordine crescente dovrebbe stare nell'ultima.

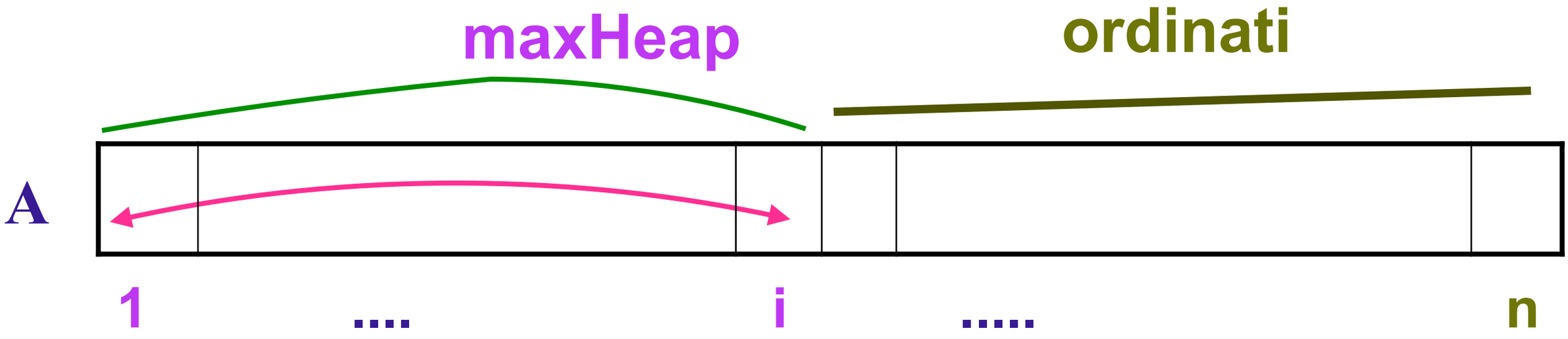
Se si scambiassero i due elementi avrei "sistemato" un elemento, e considerando come maxheap gli elementi di A da 1 a n-1, otterrei una situazione che la maxheapify può risolvere: solo il primo elemento in A[1:n-1] viola la proprietà del max-heap.



In un passo intermedio gli elementi da 1 a i sono un maxHeap, mentre i rimanenti da i+1 a n sono ordinati e sono più grandi di quelli nel maxHeap.

Come posso espandere la zona degli ordinati?

In un passo intermedio gli elementi da 1 a i sono un maxHeap, mentre i rimanenti da i+1 a n sono ordinati e sono più grandi di quelli nel maxHeap.



Per espandere la zona degli ordinati, basta osservare che mettendo il primo elemento nel maxheap nella posizione i, abbiamo esteso di uno la porzione degli ordinati, visto che è il massimo tra i più piccoli di quelli già sistemati. L'elemento di indice i va quindi al primo posto. Ma dopo questo scambio, occorre ripristinare la proprietà di essere MaxHeap sui primi i-1 elementi, perché può essere violata nella prima posizione.

L’algoritmo heapsort

1. **trasforma l’array dato in un max-heap**
2. **scambia il primo (è il massimo!) e l’ultimo elemento del maxheap**
3. **riduci la dimensione del maxHeap**
4. **ripristina la proprietà del maxHeap sugli elementi iniziali,**
5. **ripeti i passi 2-4 fino a che tutto l’array è ordinato**

Heapsort: più dettaglio

A.length= n

Heapsort(A)

Trasforma A in un MaxHeap

for i = A.length downto 2 do

In ogni passo gli elementi A[1.. i] costituiscono un maxheap su i elementi di A, gli elementi A[i+1.. n] sono più grandi di quelli in A[1.. i] e sono ordinati in ordine crescente

scambia A[1] e A[i]

A.heap-size = A.heap-size - 1

Max-Heapify (A,1)

Heapsort: esempio

4	1	3	2	16	9	10	14	8	7
1	2	3	4	5	6	7	8	9	10

trasformato in un maxHeap

16	14	10	8	7	9	3	2	4	1
1	2	3	4	5	6	7	8	9	10

scambio

1	14	10	8	7	9	3	2	4	16
1	2	3	4	5	6	7	8	9	10

Max-Heapify

Heapsort: esempio

1	14	10	8	7	9	3	2	4	16
1	2	3	4	5	6	7	8	9	10

Max-Heapify

14	8	10	4	7	9	3	2	1	16
1	2	3	4	5	6	7	8	9	10

scambio

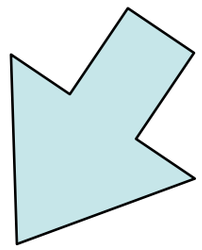
1	8	10	4	7	9	3	2	14	16
1	2	3	4	5	6	7	8	9	10

Max-Heapify

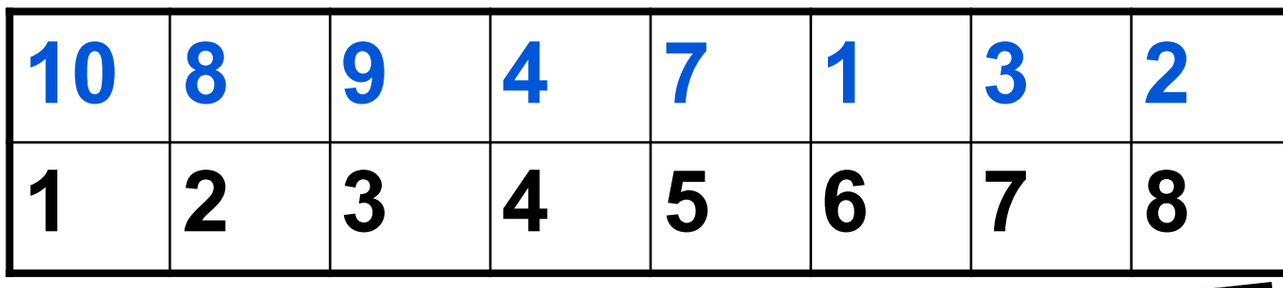
Heapsort: esempio

Max-Heapify

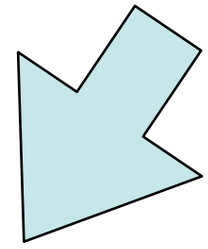
1	8	10	4	7	9	3	2	14	16
1	2	3	4	5	6	7	8	9	10



10	8	9	4	7	1	3	2	14	16
1	2	3	4	5	6	7	8	9	10



scambio

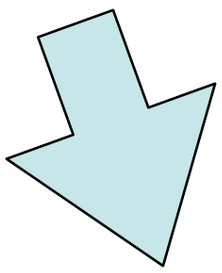


2	8	9	4	7	1	3	10	14	16
1	2	3	4	5	6	7	8	9	10

Max-Heapify

Heapsort: esempio

Max-Heapify



2	8	9	4	7	1	3	10	14	16
1	2	3	4	5	6	7	8	9	10

1	2	3	4	7	8	9	10	14	16
1	2	3	4	5	6	7	8	9	10

Heapsort: analisi

n = A.length

```
Heapsort(A)
n = A.length
trasforma A in un Maxheap                    O(n)
for i = n downto 2  do
    scambia A[1] e A[i]                      Θ(1)
    A.heap-size   = A.heap-size  - 1         Θ(1)
    Max-Heapify (A,1)                        O(log n)
                                                      } O(n log n)
```

In tutti i casi, per l'Heapsort T(n) = O(n log n)

Inoltre l'Heapsort ordina in loco, cioè non usa memoria aggiuntiva più che costante.
Faremo vedere in seguito come trasformare un array qualunque in un maxHeap in tempo lineare.