Corso di laurea in Informatica
Introduzione agli Algoritmi
Didattica blended

Esercizi

Angelo Monti

SAPIENZA
UNIVERSITÀ DI ROMA

Sulla base delle slides a cura di T. Calamoneri e G. Bongiovanni per il corso di informatica generale AA 2019/2020

Esercizio 1.

Progettare un algoritmo che, dato un vettore *A* con *n* interi ed un intero x, determina se nel vettore A esistono due interi la cui somma è *x*. L'algoritmo deve avere costo $O(n \log n)$.

Esempio:
$A = [0, -1, 2, -3, 1]$ e $x = -2$ l'algoritmo restituisce $True$ (grazie agli elementi -3 e 1)

Esercizio 2.

Siano dati due array A e B, composti da $n \geq 1$ ed $m \geq 1$ numeri interi, rispettivamente. Gli array sono entrambi ordinati in senso crescente. A e B non contengono valori duplicati; tuttavia, uno stesso valore potrebbe essere presente una volta in A e una volta in B. Progettare un algoritmo iterativo efficiente che stampi i valori che appartengono all'unione di A e di B; l'unione va intesa in senso insiemistico, quindi gli eventuali valori presenti in entrambi i vettori devono essere stampati solo una volta.

Ad esempio, se $A = [1,3,4,6]$ e $B = [2,3,4,7]$, l'algoritmo deve stampare $1, 2, 3, 4, 6, 7$.

Di tale algoritmo:
·Si dia una descrizione a parole e si scriva lo pseudocodice;
·Si determini il costo computazionale, in funzione di *n* ed *m*;

Esercizio 3.

Dato un vettore che contiene solo numeri negativi e positivi (nessun valore pari a zero), riorganizzarlo in modo che tutti i numeri negativi stiano a sinistra di quelli positivi.

Esercizio 4.

Sia data una funzione f che prende un intero e restituisce un intero, la funzione è strettamente crescente (vale a dire *f(x)<f(x+1)*) e vogliamo trovare il primo intero n non negativo per cui la funzione assume un valore non negativo.

Ad esempio, per $f(x) = -100 + 3 \cdot x$ il valore da trovare è 34.

Assumendo che il calcolo di un valore di f costi $\Theta(1)$ *p*rogettare un algoritmo che trova questo valore in tempo $O(\log n)$.

Esercizio 5.

Dato un vettore A che contiene n interi distinti e ordinati in modo crescente e due interi x e k, con $k < n$, progettare un algoritmo che stampi l'insieme dei k interi più vicini ad x presenti in A.

Nota: se l'elemento x appartiene all'insieme non deve essere conteggiato tra quelli da stampare.

La complessità dell'algoritmo deve essere $O(k + \log n)$.

Ad esempio per
$A = [12, 16, 22, 30, 35, 39, 42, 45, 48, 50, 53, 55, 56]$,
$x = 35$ e $k = 4$ l'algoritmo deve stampare $30, 39, 42$ e 45.

Esercizio 6.

Progettare un algoritmo che, dato in input un vettore che rappresenta un heap massimo di n elementi, restituisca il valore minimo.

L'algoritmo deve avere complessità $\Theta(n)$ ed il numero di elementi dell'heap esaminati deve essere al più $\left\lceil \frac{n}{2} \right\rceil$.

Esercizio 7.

Diciamo che un vettore di interi distinti è $k-ordinato$ se ogni elemento del vettore si trova a distanza al più k dalla sua posizione corretta (vale a dire quella che assumerebbe se il vettore venisse ordinato).

Ad esempio:
il vettore A= [10, 9, 8, 7, 4, 70, 60, 50] è 4-ordinato (basta ordinarlo per rendersene conto) mentre il vettore B=[6, 5, 3, 2, 8, 10, 9] è 3-ordinato

Progettare un algoritmo di ordinamento che dato un vettore A $k-ordinato$ di n interi e l'intero k ordina A.

La complessità dell'algoritmo deve essere $O(n \log k)$.

Esercizio 8.

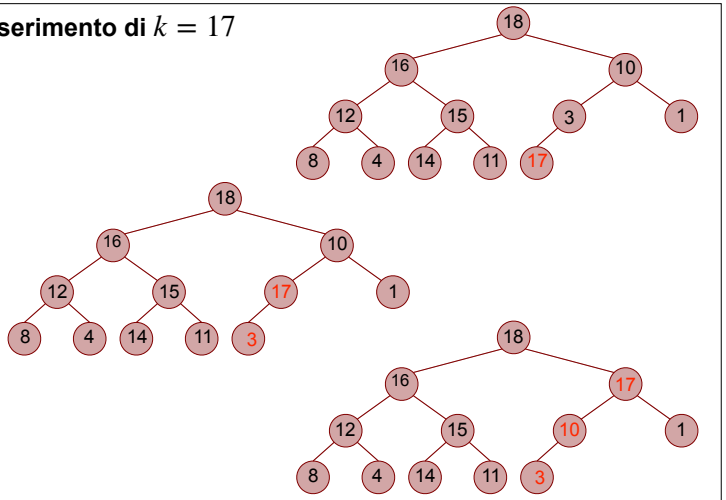
Scrivere una funzione che dati un Heap massimo, il suo size_heap n e un valore k inserisce k nell' heap.

IDEE

1. Inserire k nella prima posizione libera (in $A[n]$);
2. Incrementare n di 1
3. riaggiustare l'heap (il nodo deve risalire nell'albero fino a che non trova la sua posizione giusta).

Costo computazionale: $O(\log n)$

inserimento di $k = 17$



Esercizio 9.

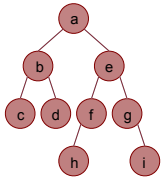
Scrivere una funzione che dati un Max Heap, ed il suo size_heap n restituisce l'elemento massimo dell'heap dopo averlo cancellato dall'heap.

Esercizio 10.

Scrivere una funzione che dati un Heap massimo A, il suo size_heap n ed una sua posizione i cancella dall'heap l'elemento $A[i]$

Esercizio 11.

Implementazione della stampa in preorder su un albero binario memorizzato con la notazione posizionale.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	e	c	d	f	g	-	-	-	-	h	-	-	i

stampaPreorder: a b c d e f h g i