

Algoritmi e strutture dati

Allievi della Laurea Specialistica in Ingegneria Informatica
Tema d'esame - 28 Giugno 2010 – ore 9.00-10.30

NOME: **COGNOME:**

MATRICOLA: **FIRMA:**

Il valore degli
esercizi è riportato
nel prospetto a lato.

Esercizio
Valore
Valutazione

1	2a	2b	3	4	5a	5b	6a	6b	7a	7b
3	1	2	3	4	2	2	3	3	3	4

Esercizi

- Quali sono gli algoritmi di ordinamento *per confronto* presentati durante il corso e perché si chiamano così? Cosa si intende quando si afferma che un algoritmo di ordinamento per confronto è *asintoticamente ottimo*? Quali algoritmi di ordinamento per confronto sono asintoticamente ottimi?
- Definire la proprietà di stabilità di un algoritmo di ordinamento.
 - Indicare se l'algoritmo HEAP-SORT sia stabile o meno, giustificando quanto affermato.
- Indicare le possibili tecniche di scansione di una tabella hash a indirizzamento aperto e, per ciascuna di esse, la forma corrispondente della funzione hash.
- Scrivere e risolvere l'equazione di ricorrenza relativa al tempo di esecuzione del seguente algoritmo.

```
BOH(A, n)
1  k ← 0
2  for i ← 1 to n
3    do k ← k + A[i] / n
4    if (k > 44)
5      then return 3 * BOH(A, n - 2) + k
6    else return k
```
- Scrivere lo pseudocodice di un algoritmo che inverta il contenuto di una pila usando una coda.
 - Analizzare la complessità spaziale (cioè il limite asintotico dell'occupazione di memoria) dell'algoritmo proposto.
- Si consideri il problema definito come segue.
Input: un intero x e un vettore A , contenente una sequenza di n ($n > 0$) valori interi;
Output: due interi distinti h e k , $1 \leq h \leq n$, $1 \leq k \leq n$, tali che $A[h] + A[k] = x$;
due valori interi identici se non esiste alcuna coppia di indici distinti h e k del vettore A , $1 \leq h \leq n$, $1 \leq k \leq n$, tale che $A[h] + A[k] = x$.
 - Scrivere lo pseudocodice di un algoritmo che risolva tale problema.
 - Analizzare la complessità temporale dell'algoritmo proposto.

- 7) Sia dato il seguente algoritmo di ordinamento in ordine non decrescente dei valori contenuti nell'array A.

```
BUBBLESORT(A)
1  n ← length[A]
2  while n > 0
3    do i ← 1
4      while i < n
5        do if A[i] > A[i+1]
6          then tmp ← A[i]
7             A[i] ← A[i+1]
8             A[i+1] ← tmp
9          i ← i + 1
10     n ← n - 1
```

- a) Analizzarne la complessità temporale.
- b) Proporne una versione ricorsiva.