Introduzione agli algoritmi
Appello del 6/6/2017
E. Fachini - R. Petreschi

Le soluzioni degli esercizi scritte in modo illeggibile o in cui compaiano solo conti o pseudocodice senza commenti e risposte non motivate saranno valutati 0. Prima di descrivere un algoritmo in pseudocodice si deve delineare l'idea algoritmica. Inoltre deve essere precisato l'output atteso da eventuali singole funzioni utilizzate, oltre agli eventuali vincoli sul loro input (precondizioni).

Parte I

1. Si illustri l'algoritmo di trasformazione di un array qualunque in un Max-Heap e se ne analizzi il tempo asintotico di esecuzione nel caso peggiore.

2.
   a. Si consideri un algoritmo in cui un array ordinato di n elementi viene modificato in modo tale che gli elementi che sono potenze di 2 vengono divisi per 2.
   E' vero che nel caso peggiore l'algoritmo esegue un numero di divisioni pari a Θ(lg n)? Se sì, perché … Se no, perché …
   No perché nel caso peggiore sono tutti potenze di 2 e quindi si fanno n divisioni.
   E' vero che nel caso migliore l'algoritmo esegue un numero di divisioni pari a Θ(1)? Se sì, perché … Se no, perché …
   Sì, perché o non si fanno divisioni o se ne fa una.
   E' giusto dire che l'algoritmo esegue un numero di divisioni pari a O(n)? Se sì, perché … Se no, perché …
   Sì, perché il numero di divisioni nel caso peggiore dà un limite superiore a tutti gli altri casi
   E' vero che nel caso peggiore l'algoritmo ha un tempo di esecuzione asintotico Θ(n)? Se sì, perché … Se no, perché …
   Sì, perché in generale si scorrerà tutto l'array e nel caso peggiore si faranno anche tutte le divisioni.
   In effetti si potrebbe dire che questo algoritmo ha tempo di esecuzione asintotico pari a Θ(n) in tutti i casi, perché si deve sempre controllare ogni elemento dell'array.

3. Data una sequenza $a_1,\ldots,a_n$ di elementi distinti, supponiamo che siano memorizzati in un array in modo tale che sia ciclicamente ordinata.
Questo vuol dire che il minimo si trova in una posizione i non nota ma tale che la sequenza $a_i,\ldots,a_n,a_1,\ldots,a_{i-1}$ è ordinata in modo crescente. Il

Introduzione agli algoritmi
Appello del 6/6/2017
E. Fachini - R. Petreschi

problema della ricerca in una sequenza ciclicamente ordinata consiste nel determinare il minimo della sequenza.
Progettare un algoritmo che risolve il problema della ricerca in una sequenza in O(lg n).

Esempio:

15,16,20,30,8,10 qui il minimo è 8 nella posizione 5 (o 4 se si parte da 0)
20,30,3,5,7,13,15 qui il minimo è 3, nella posizione 3 (o 2 se si parte da 0)

Sol. Sia A[0],…A[n-1] l'array che contiene l'array ciclicamente ordinato. Se applichiamo un metodo ispirato alla ricerca binaria alla porzione A[i], …, A[j], per prima cosa confrontiamo l'elemento mediano A[m] con il precedente A[m-1], perché se A[m] < A[m-1] allora m è l'indice del minimo, e potremmo anche controllare il successivo A[m+1], perché se il minimo non è A[m], potrebbe essere A[m+1], verifichiamo quindi se A[m+1] < A[m], allora l'indice del minimo è m+1. Bisognerà naturalmente controllare che gli elementi in posizione precedente e successiva a m in A ci siano. Se non abbiamo trovato il minimo, dobbiamo andare a cercare il minimo nella metà giusta dell'array. Per semplicità mettiamoci nel caso iniziale: caso 1.

a_i	…	a_n	a_1	…	a_m	…	a_{i-1}
0	…	n-i		…	n/2	….	n-1

Se a_m = A[n/2] è nella seconda porzione crescente dell'array, allora deve essere minore di a_{i-1}, ovvero dell'elemento nell'ultima posizione. Questo ci dice che se A[n/2] < A[n-1] allora l'elemento minimo deve trovarsi nella metà sinistra di A.

Caso2.

a_i	…	a_m	…	a_n	a_1	…	a_{i-1}
0	…	n/2		n-i		….	n-1

Altrimenti, cioè a_m = A[n/2] è nella prima porzione crescente dell'array, allora deve essere maggiore di a_{i-1}, ovvero dell'elemento nell'ultima posizione.

Introduzione agli algoritmi
Appello del 6/6/2017
E. Fachini - R. Petreschi

In generale se A[m] è minore di A[j], allora se k l'indice del minimo, si ha A[i]<…<A[k-1]>A[k]<A[k+1] …<A[m] < … <A[j], e l'ordinamento sarebbe: A[k] <A[k+1] …<A[m] < … <A[j] < A[i]<…<A[k-1]. In altri termini l'indice m cade nella seconda parte crescente dell'array circolare e quindi il minimo va cercato nella prima parte della porzione di array.
Mentre se A[m] è maggiore di A[j], allora, detto k l'indice del minimo, si ha A[i] < …< A[m]< …<A[k-1] >A[k] <A[k+1] … … A[j], e l'ordinamento in questo caso è A[k] <A[k+1] … <A[j]< A[i] < …< A[m]< …<A[k-1]. In altri termini ora l'indice m cade nella prima porzione crescente dell'array.

Pseudocodice:

```
MinArrayCirc(A)
input: un array di numeri ordinato ciclicamente
precondizione: l'indice del minimo non è 0
output: l'indice del minimo
n = A.length
i=0 j = n-1
while i < j do
   m = (j + i )/2 # in m l'indice mediano
      if m==0 or m==n-1 or (m > 0 and  A[m] < A[m-1]) then  return m
      if m < n-1  and  A[m] >A[m+1] then  return m+1
      if A[m] < A[j] then  j = m else i = m+1
```