Introduzione agli Algoritmi
Recupero Prima Prova intermedia a canali unificati con spunti per la soluzione

docenti: T. Calamoneri, A. Monti
Sapienza Università di Roma

4 Giugno 2025

ATTENZIONE: I compiti in cui l'esercizio 1 sia valutato meno di 7/10 non saranno ritenuti sufficienti, indipendentemente dalla qualità delle soluzioni degli esercizi 2 e 3.

Esercizio 1 (10 punti):
Si risponda alle seguenti domande, dando una giustificazione:

1. a) $n^2 \log n - 10n \log^2 n = \Theta(\ldots)$?
   b) $2^{\log_4 n} = \Theta(\ldots)$?

2. Dimostrare o confutare le seguenti affermazioni:
   a) se $f(n)$ e $g(n)$ sono in $\Theta(n)$ allora $f(n) - g(n) = \Theta(1)$;
   b) se $f(n)$ e $g(n)$ sono in $\Omega(n)$ allora $f(n) + g(n) = \Omega(1)$.

3. Se un algoritmo ha tempo di esecuzione $O(n \log n)$ è possibile che in qualche caso il suo tempo di esecuzione sia $\Theta(n)$?

4. Se un algoritmo ha tempo di esecuzione $\Theta(n^2)$, è possibile che il suo tempo di esecuzione nel caso migliore sia $\Theta(n)$?

5. Qual è l'equazione di ricorrenza relativa alla versione ricorsiva dell'algoritmo della ricerca sequenziale? La si risolva con uno dei metodi studiati.

Le seguenti risposte vogliono solo dare un'idea della soluzione e non vanno considerate come esaustive:

1. **a)** $n^2 \log n - 10n \log^2 n = \Theta(n^2 \log n)$ **in quanto il termine negativo è di un ordine inferiore rispetto all'altro. Si può usare la definizione di** Θ **per dimostrarlo.**
   b) $2^{\log_4 n} = \Theta(\sqrt{n})$ **in quanto** $\log_4 n = 1/2 \log_2 n$**.**

2. **a) è falso che se** $f(n)$ **e** $g(n)$ **sono in** $\Theta(n)$ **allora** $f(n) - g(n) = \Theta(1)$**; come controesempio, si prenda** $f(n) = 2n$ **e** $g(n) = n$**, la cui differenza è in** $\Theta(n)$**;**
   b) è vero che se $f(n)$ **e** $g(n)$ **sono in** $\Omega(n)$ **allora** $f(n) + g(n) = \Omega(1)$**; infatti, informalmente, se sia** f **che** g **sono almeno lineari, la loro somma sarà almeno lineare e quindi, a maggior ragione, almeno costante. E' richiesta una dimostrazione formale.**

3. **Si: tempo di esecuzione** $O(n \log n)$ **significa** $\Theta(n \log n)$ **o meno, pertanto può esistere un'istanza che richieda tempo** $\Theta(n)$**.**

4. **La risposta è no. Se nel caso migliore si è dimostrato che il costo è in** $\Omega(n^2)$**, ogni altro caso ha questo stesso limite inferiore e quindi non può avere un costo in** $O(n)$**.**

5. **L'equazione di ricorrenza per l'algoritmo di ricerca sequenziale ricorsiva è, a seconda di come si decide di spezzare l'array:** $T(n) = T(n-1) + \Theta(1)$ **oppure** $T(n) = 2T(n/2) + \Theta(1)$**; per entrambe il caso base è** $T(1) = \Theta(1)$**.**

Entrambe le equazioni hanno come soluzione $T(n) = \Theta(n)$, che va giustificata risolvendo con uno dei metodi studiati.

Esercizio 2 (10 punti):

Si consideri la seguente funzione:

```
def esercizio_iter(n):
   a=b=1
   sum=0
   while a<=n:
      b=b*a
      a+=1
   for a in range(b):
      sum+=c
   return sum
```

Se ne determini il costo computazionale, motivando bene la risposta.

In modo molto stringato, possiamo dire che il primo while richiede $\Theta(n)$ tempo ed, alla fine, b risulta essere pari ad $n! = \Theta(n^n)$. Di conseguenza il for viene iterato $\Theta(n^n)$ volte e, per ogni iterazione, viene eseguita un'operazione di costo $\Theta(1)$. Quindi il costo computazionale complessivo è $\Theta(n^n)$.
E' necessario dettagliare i calcoli utili ad arrivare a questo risultato.

Esercizio 3 (10 punti): Si consideri la seguente funzione:

```
def fun (n):
   i=0
   for a in range(n+1):
      for b in range(n+1):
         i=i+1
   if n<2 return i
   return 2*fun(n/4)
```

Si scriva l'equazione di ricorrenza che ne definisce il costo computazionale, la si risolva con il metodo iterativo, e si discuta se sia possibile applicarvi il teorema principale o no, motivando bene la risposta.

C'è una sola chiamata su un quarto degli elementi e il tempo di esecuzione delle istruzioni al di fuori delle chiamate è $\Theta(n^2)$. Quindi la relazione di ricorrenza, nella sua forma generale, è $T(n) = T(n/4) + \Theta(n^2)$; il caso base rispetta $T(0) = T(1) = \Theta(1)$. Risolvendo con il metodo iterativo, si ottiene che $T(n) = \Theta(n^2)$.
Questa soluzione è confermata dal teorema principale, di cui sfruttiamo il caso 3 e ricordiamo di dimostrare l'ipotesi aggiuntiva.