3

Ricorrenze, analisi e notazione asintotica

Esercizio 3.1.

1. Si consideri il seguente frammento di pseudocodice:

```
for i := 0 to n
    P1(n)
```

dove P1(n) è definito nel seguente modo:

```
P1(n)
if n < 1
    STAMPA(n)
for j := 1 to 3
    P1(n/2)
for j := 1 to n
    for k := 1 to n
        STAMPA(j, k)
```

Dire quale è la complessità temporale del frammento di codice in funzione del parametro n.

2. Si consideri il seguente frammento di pseudocodice:

```
i := n
j := 1
while (i > 0)
    P2(j)
    i := i/3
    j := j + 1
```

dove P2(n) è definito nel seguente modo:

```
P2(n)
if n < 1
    STAMPA(n)
for l := 1 to 4
    P2(n/2)
for k := 1 to n
    for j := k + 1 to n
        STAMPA(k, j)
```

Dire quale è la complessità temporale del frammento di codice in funzione del parametro n.

3. Sia dia la complessità del seguente algoritmo:

```
P3(n)
if n < 1
    STAMPA(n)
for i := 1 to 9
    P3(n/3)
P2(n)
```

dove P2(n) è definito come al punto 2.

NB: Tutte le variabili nei frammenti di codice precedenti sono da intendersi di tipo intero, quindi anche le operazioni su di esse sono da intendersi come operazioni sugli interi.

Soluzione

1. P1 è un algoritmo ricorsivo la cui ricorrenza è $T(n) = 3T(\frac{n}{2}) + n^2$, la cui soluzione è, applicando il teorema dell'esperto, $T(n) = \Theta(n^2)$. Infatti, $f(n) = n^2$ è un polinomio, per cui è sufficiente confrontare il grado del polinomio, cioè 2, con $\log_b a = \log_2 3$; siccome vale $2 > \log_2 3$, siamo nel caso 3 del teorema dell'esperto, e la condizione aggiuntiva $af(\frac{n}{b} \leq cf(c)$ (per qualche $c < 1$ e n grande a sufficienza) è automaticamente verificata, per cui $T(n)$ è dell'ordine di grandezza di $f(n)$.

   Di conseguenza la complessità globale del frammento di pseudocodice è $\Theta(n \cdot n^2) = \Theta(n^3)$, in quanto P1 viene invocata n volte con parametro n.

2. In questo secondo caso la ricorrenza di P2 è $T(n) = 4T(\frac{n}{2}) + n^2$, che ha soluzione (sempre applicando il teorema dell'esperto) $T(n) = \Theta(n^2 \log(n))$ (come al punto precedente, $f(n)$ è un polinomio di grado 2, ma in questo caso abbiamo $\log_b a = \log_2 4 = 2$, per cui siamo nel caso 2 del teorema dell'esperto).

   P2 viene invocata un numero $\log_3(n)$ di volte con valori crescenti del parametro, quindi il costo totale del frammento di pseudocodice è $\sum_{j=1}^{\log(n)} j^2 \log(j)$, che è $\Theta(\log^3(n)\log(\log(n)))$.
   Per vedere che vale quest'ultima relazione, è sufficiente mostrare che, da un lato, vale $\sum_{j=1}^{\log(n)} j^2 \log(j) \leq \log^3(n)\log(\log(n))$ (quindi $\sum_{j=1}^{\log(n)} j^2 \log(j) = \mathcal{O}(\log^3(n)\log(\log(n)))$), e, dall'altro, che vale $\sum_{j=1}^{\log(n)} j^2 \log(j) \geq (\frac{\log(n)}{2})^3 \log(\frac{\log(n)}{2})$ (quindi abbiamo anche che $\sum_{j=1}^{\log(n)} j^2 \log(j) = \Omega(\log^3(n)\log(\log(n)))$). Si noti che $\sum_{j=1}^{\log(n)} j^2 \log(j) \leq \log(n)^3 \log(\log(n))$ è banalmente vera, in quanto vale $j \leq \log(n)$. Per mostrare invece che vale $\sum_{j=1}^{\log(n)} j^2 \log(j) \geq (\frac{\log(n)}{2})^3 \log(\frac{\log(n)}{2})$ si può notare che $\sum_{j=1}^{\log(n)} j^2 \log(j) \geq \sum_{j=\frac{\log(n)}{2}+1}^{\log(n)} j^2 \log(j) \geq (\frac{\log(n)}{2})^3 \log(\frac{\log(n)}{2})$ è vera, in quanto nell'ultima sommatoria vale sempre $j > \frac{\log(n)}{2}$.

3. In questo caso la ricorrenza di P3 è $T(n) = 9T(\frac{n}{3}) + n^2 \log_3(n)$ (si noti che, per semplificare i calcoli che seguono, usiamo $\log_3(n)$ invece che $\log(n)$, che è legittimo, in quanto il cambio di base non cambia l'ordine di grandezza del logaritmo). Tale ricorrenza non può essere risolta con il teorema dell'esperto, in quanto $n^2 \log_3(n)$ non è polinomialmente più grande di n^2. Tuttavia, tramite il metodo di sostituzione, è possibile mostrare che in questo caso vale $T(n) = \Theta(n^2 \log_3^2(n))$.
   Per ottenere ciò, mostriamo innanzi tutto che vale $T(n) \leq cn^2 \log_3^2(n)$. Per l'ipotesi induttiva, abbiamo

$$
\begin{aligned}
T(n) &= 9T(\frac{n}{3}) + n^2 \log_3(n) \\
&\leq 9c(\frac{n}{3})^2 \log_3^2(\frac{n}{3}) + n^2 \log_3(n) \\
&= cn^2(\log_3^2(n) - 2\log_3(n) + \log_3^2(3)) + n^2 \log_3(n) \\
&= cn^2 \log_3^2(n) - 2cn^2 \log_3(n) + cn^2 + n^2 \log_3(n) \\
&\leq cn^2 \log_3^2(n)
\end{aligned}
$$

   se è $-2cn^2 \log_3(n) + cn^2 + n^2 \log_3(n) \leq 0$, cioè se $c \geq \frac{\log_3(n)}{2\log_3(n)-1}$, che vale per $n \geq 3$ e $c \geq 1$. Inoltre, assumendo $T(1) = T(2) = 1$ come casi base, abbiamo che vale $T(3) = 9 + 9\log_3(3) = 9 \cdot 2$ e $T(9) = 81 \cdot 2 + 81 \cdot 2 \leq c81 \cdot 4$, che vale per $c \geq 1$.
   Inoltre, si mostra, con calcoli analoghi a quelli sopra (con $\geq$ al posto di $\leq$), che vale $T(n) \geq cn^2 \log_3^2(n)$, se è $-2c\log_3(n) + c + \log_3(n) \geq 0$, che a sua volta è vero per $n \geq 1$ e $c \leq \frac{1}{2}$. In questo caso, abbiamo anche che vale $T(3) = 18 \geq \frac{1}{2} \cdot 9 \log_3(3)$.

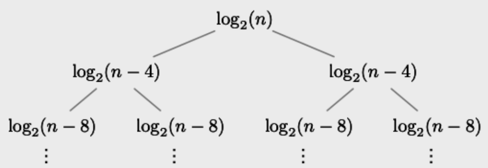
Esercizio 3.2. Trovare un limite asintotico superiore per la soluzione della seguente ricorrenza:

$$T(n) = 2T(n-4) + \Theta(\log(n))$$

Soluzione

In questo caso la ricorrenza non ha una forma che permette di applicare il teorema dell'esperto (a destra, il termine in T è della forma $T(n-b)$ invece che $T(\frac{n}{b})$). Possiamo risolvere la ricorrenza formulando un'ipotesi tramite il metodo dell'albero di ricorsione, e poi verificarla con il metodo della sostituzione. Si noti che, per semplificare i conti, nel seguito usiamo $\log_2(n)$ invece che $\log(n)$, cosa che comunque è possibile in quanto vale $\log_2(n) = \Theta(\log(n))$.

L'albero di ricorsione è il seguente:



Mediante il metodo dell'albero di ricorsione si ottiene l'ipotesi $\mathcal{O}(2^{\frac{n}{4}} \log_2(n))$; infatti, il costo di ogni livello è $\Theta(2^k \log_2(n-4k))$, e la profondità dell'albero è $k = \frac{n}{4}$. Poi si verifica che, con opportune costanti, vale $T(n) \leq c2^{\frac{n}{4}} \log_2(n)$.

In effetti, per arrivare a concludere che abbiamo $T(n) \leq c2^{\frac{n}{4}} \log_2(n)$, è utile mostrare un vincolo leggermente più stretto, e cioè che vale $T(n) \leq c2^{\frac{n}{4}} \log_2(n) - bn^2$. Infatti si ha:

$$
\begin{aligned}
T(n) = 2T(n-4) + \log_2(n) &\leq 2c2^{\frac{n-4}{4}} \log_2(n-4) - 2b(n-4)^2 + \log_2(n) \\
&= c2^{\frac{n}{4}} \log_2(n-4) - bn^2 - bn^2 + 16bn - 32 + \log_2(n) \\
&\leq (c2^{\frac{n}{4}} \log_2(n) - bn^2) - bn^2 + 16bn - 32 + \log_2(n) \leq c2^{\frac{n}{4}} \log_2(n) - bn^2
\end{aligned}
$$

per b ed n tali che vale $-bn^2 + 16bn - 32 + \log_2(n) \leq 0$. Questo si ha, per esempio, per $n \geq 16$ e $b \geq 1$. Inoltre, prendendo come caso base $T(1) = T(2) = T(3) = T(4) = 1$, si ha che $T(8) = 2+3$, $T(12) = 10 + \log_2(12) = 10 + 2 + \log_2(3)$, $T(16) = 2(12 + \log_2(3)) + 4 \leq c16 \cdot 4 - 256b$, che è vero, per esempio, se $c \geq \frac{8b+1}{2}$ (perché in questo caso vale $c16 \cdot 4 - 256b \geq 32 \geq 2(12 + \log_2(3)) + 4$).

Esercizio 3.3. Si consideri la seguente —inutile!— funzione, definita mediante pseudocodice, e che riceve in ingresso un numero intero n.

```
USELESS(n)
  for i := 1 to n
      AUX(i)
```

dove a sua volta la funzione AUX(m), che prende in ingresso un numero intero m, è definita nel modo seguente (ODD(M) restituisce `true` se m è un numero dispari, altrimenti restituisce `false`):

```
AUX(m)
  if (m = 1)
      return
  else
      if ODD(m)
          AUX(m − 1)
      else AUX(m/2)
```

Si valuti la complessità asintotica della funzione USELESS mediante la notazione $\mathcal{O}$, o, preferibilmente, mediante la notazione Θ.

SOLUZIONE

La funzione AUX ha una complessità che è soluzione della seguente equazione alle ricorrenze:

$$\begin{aligned} T(1) &= 1; \\ T(n) &= \text{if } \texttt{odd(n)} \text{ then } 1 + T(n-1) \\ &\qquad \text{else } 1 + T\left(\frac{n}{2}\right) \end{aligned}$$

Consideriamo due casi estremi:

a) $n = 2^k$ per qualche $k \in \mathbb{N}$; in tal caso, abbiamo

$$T(n) = 1 + T\left(\frac{n}{2}\right) = 1 + 1 + T\left(\frac{n}{4}\right) = 1 + 1 + 1 \ldots 1 + T(1) = 1 \cdot k$$

quindi $T(n)$ è $\mathcal{O}(\log(n))$.

b) n è dispari, $\frac{n-1}{2}$ è pure dispari e così via finché la ricorsione si chiude; in tal caso vale

$$\begin{aligned} T(n) &= 1 + T(n-1) = 1 + 1 + T\left(\frac{n-1}{2}\right) = 1 + 1 + 1 + T\left(\frac{n-3}{2}\right) \\ &= 1 + 1 + 1 + 1 + T\left(\frac{n-3}{4}\right) \ldots \end{aligned}$$

che è $\mathcal{O}(2\log(n)) = \mathcal{O}(\log(n))$. Quindi AUX ha una complessità $\mathcal{O}(\log(n))$ (e anche $\Theta(\log(n))$). USELESS chiama n volte AUX con parametro i che va da 1 a n; quindi per USELESS vale $T(n) \leq \sum_{i=1}^{n} c\log(i)$ che è $\mathcal{O}(n\log(n))$. Vale anche $T(n) \geq \sum_{i=\frac{n}{2}}^{n} c\log(i) \geq \frac{n}{2}\log(\frac{n}{2})$, che è $\Omega(n\log(n))$.

Quindi $T(n)$ è $\Theta(n\log(n))$.

Esercizio 3.4. Si risolva la seguente equazione alle ricorrenze:

$$T(n) = T(n-1) + n^2$$

SOLUZIONE

È facile sviluppare $T(n)$ in $n^2 + (n-1)^2 + (n-2)^2 + (n-3)^2 \ldots + c$ ossia $T(n) = \sum_{i=1}^{n} i^2$, che è $\Theta(n^3)$. Infatti, valgono le seguenti relazioni: $\sum_{i=1}^{n} i^2 \leq n^3$ e $\sum_{i=1}^{n} i^2 \geq (\frac{n}{2})^2(\frac{n}{2}) = \frac{n^3}{8}$.

Esercizio 3.5. Si calcoli la complessità del seguente frammento di codice:

```
PROC(n)
  if (n < 5)
      return 0
  s := PROC(2 * n/5)
  s := s + PROC(3 * n/5)
  i := 1
  while (i <= 3 * n)
      s := s + 1
      i := i + 2
  return s
```

SOLUZIONE

Lo pseudocodice ricorsivo dell'esercizio ha una funzione di complessità definita dalla seguente ricorrenza:

$$T(n) = T\left(\frac{2}{5}n\right) + T\left(\frac{3}{5}n\right) + \Theta(n)$$

che si risolve mediante il metodo di sostituzione, dopo avere fatto l'ipotesi che la soluzione sia $\Theta(n\log(n))$ (tale ipotesi può essere ottenuta mediante l'albero di ricorsione, in modo analogo a quanto descritto in [2, capitolo 4] per la ricorrenza $T(n) = T(\frac{n}{3}) + T(\frac{2}{3}n) + \Theta(n)$).

Più precisamente, abbiamo quanto segue:

$$\begin{aligned} T(n) &= T\left(\frac{2}{5}n\right) + T\left(\frac{3}{5}n\right) + dn \leq \frac{2}{5}cn\log\left(\frac{2}{5}n\right) + \frac{3}{5}cn\log\left(\frac{3}{5}n\right) + dn \\ &= \frac{2}{5}cn\left(\log(n) - \log\left(\frac{5}{2}\right)\right) + \frac{3}{5}cn\left(\log(n) - \log\left(\frac{5}{3}\right)\right) + dn \\ &= cn\log(n) - cn\left(\frac{2}{5}\log\left(\frac{5}{2}\right) + \frac{3}{5}\log\left(\frac{5}{3}\right)\right) + dn \\ &= cn\log(n) - cn\left(\log(5) - \frac{2}{5}\log(2) - \frac{3}{5}\log(3)\right) + dn \\ &\leq cn\log(n) \end{aligned}$$

se vale $-cn\left(\log(5) - \frac{2}{5}\log(2) - \frac{3}{5}\log(3)\right) + dn \leq 0$, che è vero se è $c \geq \frac{d}{(\log(5) - \frac{2}{5}\log(2) - \frac{3}{5}\log(3))}$ (cioè, siccome abbiamo $\frac{1}{2} < \log(5) - \frac{2}{5}\log(2) - \frac{3}{5}\log(3) < 1$, $c \geq 2$ se è $d = 1$). Inoltre, ponendo come casi base $T(1) = T(2) = 1$, si ha che vale (usando $c = 2$ e $d = 1$) $T(3) = 2 + 3 \leq 2 \cdot 3\log(3)$.

In maniera analoga (con conti analoghi a quelli sopra, scambiando $\leq$ con $\geq$) si mostra che vale $T(n) \geq cn\log(n)$ se è $c \leq \frac{d}{(\log(5) - \frac{2}{5}\log(2) - \frac{3}{5}\log(3))}$ (che è vero se vale $c \leq 1$ e $d = 1$).

Esercizio 3.6. Risolvere la seguente ricorrenza:

$$T(n) = 9T\left(\frac{n}{10}\right) + n\log(n)$$

SOLUZIONE

La ricorrenza si risolve applicando il teorema dell'esperto. Siamo nel caso 3 del teorema, per cui occorre verificare la condizione aggiuntiva, $af(\frac{n}{b}) \leq cf(n)$, con $f(n) = n\log(n)$. La condizione è verificata, in quanto diventa $\frac{9n}{10}\log(\frac{n}{10}) \leq cn\log(n)$, cioè $\frac{9n}{10}\log(n) - \frac{9n}{10}\log(10) \leq cn\log(n)$, che è banalmente verificata prendendo, per esempio, $c = \frac{99}{100}$. Di conseguenza, la soluzione è $T(n) = \Theta(n\log(n))$.

Esercizio 3.7. Si risolva la seguente ricorrenza:

$$T(n) = 8T(\frac{n}{6}) + n^{\frac{3}{2}}\log^2(n)$$

SOLUZIONE

La ricorrenza si risolve applicando semplicemente il teorema dell'esperto. Si ha che $\log_b a = \log_6 8$, quindi $n^{\log_6 8}$, che vale circa $n^{1.16}$, è polinomialmente più piccolo di $n^{\frac{3}{2}}\log^2(n)$. Siamo quindi nel caso 3 del teorema, e si ha che la condizione aggiuntiva $af(\frac{n}{b}) \leq cf(n)$ (con $f(n) = n^{\frac{3}{2}}\log^2(n)$) è verificata già per $c = 1$. Quindi, si ha anche che $T(n) = \Theta(n^{\frac{3}{2}}\log^2(n))$.

Esercizio 3.8. Si calcoli la complessità del seguente frammento di codice:

```
FUN(A)
  n := A.length
  res := 0
  if (n > 5)
      res := res + FUN(A[1..n/3]) + FUN(A[n/3 + 1..2 * n/3])
                 + FUN(A[2 * n/3 + 1..n])
      res := res + FUN(A[n/6 + 1..n/2]) + FUN(A[n/2 + 1..5 * n/6])
  for i := 1 to n
      res := res + A[i]
  return res
```

SOLUZIONE

La ricorrenza corrispondente all'algoritmo è $T(n) = 5T(\frac{n}{3}) + n$, che si risolve con il teorema dell'esperto (caso 1), e la cui soluzione è $\Theta(n^{\log_3 5})$. Infatti, in questo caso si ha che $f(n) = n$ è un polinomio di grado 1 per cui, per capire in quale caso del teorema dell'esperto ci troviamo, è sufficiente confrontare il grado di $f(n)$ con $\log_b a = \log_3 5 > 1$. Sono quindi soddisfatte le condizioni del caso 1 del teorema dell'esperto.

Esercizio 3.9. Si consideri la seguente ricorrenza.

$$T(n) = T(n-1) + T(n-2) + n^2$$

Indicare una funzione $g(n)$ per cui $T(n) = \mathcal{O}(g(n))$, spiegando come si è arrivati ad individuare $g(n)$.

SOLUZIONE

Applicando il Teorema 1 di [1], che copre il calcolo della complessità di ricorrenze lineari di ordine costante, cioè della forma $\sum_{i=1}^{h} a_i T(n-i) + cn^k$, dove abbiamo $h = 2$, $k = 2$ e $a = \sum_{i=1}^{h} a_i = 2$, si ha direttamente che vale $T(n) = O(n^2 2^n)$.

Esercizio 3.10. Si fornisca un limite asintotico superiore per l'espressione $T(n)$ definita dalla seguente equazione di ricorrenza:

$$T(n) = T\left(\frac{n}{2}\right) + n(2 + \sin(n)).$$

SOLUZIONE

Nonostante $f(n)$ (che in questo caso è $n(2 + \sin(n))$) sia polinomialmente più grande di $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$, il teorema dell'esperto non si può applicare poiché è violata la condizione (di "regolarità") $af\left(\frac{n}{b}\right) \leq cf(n)$, che dovrebbe applicarsi in questo caso (saremmo nel caso 3 del teorema):

$$\frac{n}{2}\left(2+\sin\left(\frac{n}{2}\right)\right) \le cn(2+\sin(n))$$

per n grande e $c<1$. Infatti, le curve $\frac{n}{2}(2+sin\left(\frac{n}{2}\right))$ e $n(2+sin(n))$ oscillano e si intersecano continuamente, quindi a maggior ragione la relazione non vale moltiplicando $f(n)=n(2+\sin(n))$ per $c<1$.

Tuttavia, possiamo notare come l'espressione $f(n)=n(2+sin(n))$ sia maggiorata da $3n$, quindi vale $T(n)\le T'(n)$, dove $T'(n)$ è definita dalla ricorrenza $T'(n)=T'(\frac{n}{2})+3n$. Possiamo risolvere la ricorrenza di $T'(n)$ con il teorema dell'esperto, ricadendo nel caso 3 e ottenendo $T'(n)=\Theta(n)$. Ipotizziamo allora che valga $T(n)=\mathcal{O}(n)$, ossia $T(n)\le dn$, e procediamo per sostituzione:

$$T(n)=T\left(\frac{n}{2}\right)+n(2+\sin(n)) \le d\frac{n}{2}+n(2+\sin(n)) \le d\frac{n}{2}+3n$$

La relazione $d\frac{n}{2}+3n \le dn$ è verificata per $d\ge 6$ (e $n>0$). Occorre poi verificare la condizione iniziale, che è soddisfatta, ad esempio, con $T(1)=1$.

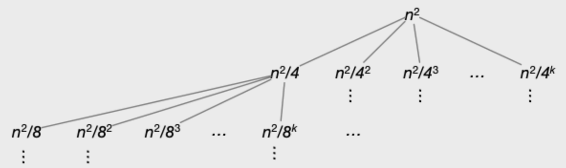
Esercizio 3.11. Trovare un limite superiore per la seguente ricorrenza:

$$T(n)=T\left(\lfloor\frac{n}{2}\rfloor\right)+T\left(\lfloor\frac{n}{4}\rfloor\right)+T\left(\lfloor\frac{n}{8}\rfloor\right)+\ldots+T\left(\lfloor\frac{n}{2^k}\rfloor\right)+n^2$$

dove k è una costante intera maggiore di 1 e n è una potenza di 2.

SOLUZIONE

Possiamo risolvere la ricorrenza usando il metodo della sostituzione. Per formulare l'ipotesi da verificare, possiamo disegnare il seguente albero di ricorsione.



Si noti che vale $\sum_{i=1}^{k}\frac{1}{4^k} < \sum_{i=1}^{\infty}\frac{1}{4^k} = \frac{1}{1-\frac{1}{4}}-1=\frac{1}{3}$, in quanto è noto (si veda [2, Appendice A]) che abbiamo $\sum_{i=0}^{\infty}x^k=\frac{1}{1-x}$ se vale $|x|<1$, e abbiamo anche che $\sum_{i=1}^{\infty}x^k=\frac{1}{1-x}-1=\frac{x}{1-x}$. Analogamente, abbiamo $\sum_{i=1}^{k}\frac{1}{8^k}<\frac{1}{7}$, $\sum_{i=1}^{k}\frac{1}{16^k}<\frac{1}{15}$, e così via.

Si noti che $\frac{1}{3}+\frac{1}{7}+\frac{1}{15}\ldots$ è maggiorata da $\sum_{i=1}^{\infty}\frac{1}{2^i}=1$. Possiamo quindi intuire che la somma di tutti i termini dell'albero di ricorsione è maggiorata da cn^2, di conseguenza facciamo l'ipotesi che valga $T(n)\le cn^2$, per una qualche costante positiva c, e procediamo a verificare l'ipotesi. Otteniamo:

$$\begin{aligned}
T(n) &= n^2+T\left(\lfloor\frac{n}{2}\rfloor\right)+T\left(\lfloor\frac{n}{4}\rfloor\right)+T\left(\lfloor\frac{n}{8}\rfloor\right)+\ldots+T\left(\lfloor\frac{n}{2^k}\rfloor\right)\\
&= n^2+\sum_{i=1}^{k}T\left(\lfloor\frac{n}{2^i}\rfloor\right)\\
&\le n^2+\sum_{i=1}^{k}c\frac{n^2}{4^i}\\
&= n^2\left(1+\sum_{i=1}^{k}\frac{c}{4^i}\right)\\
&= n^2\left(1+c\sum_{i=1}^{k}\frac{1}{4^i}\right)\\
&< n^2\left(1+\frac{c}{3}\right)
\end{aligned}$$

L'espressione ottenuta soddisfa l'ipotesi se vale $n^2(1+\frac{c}{3})\le cn^2$, ossia se vale (considerando $n\ge 1$) $c\ge\frac{3}{2}$.

Per verificare il caso base, poniamo $T(n)=1$ per tutti gli $n<2^k$. Notiamo che vale $T(1)=1\le c\cdot 1^2$ per $c\ge\frac{3}{2}$, e, analogamente, $T(2)=1\le c\cdot 2^2,\ldots,$ $T(2^{k-1})=1\le c\cdot 2^{2k-2}$.

In definitiva, vale $T(n)=O(n^2)$.

Esercizio 3.12. Si consideri il seguente algoritmo, descritto in pseudocodice, che prende in ingresso un array A di numeri interi:

```
FUN(A)
if A.length < 5
    return FUN1(A)
a := A.length − 1
FUN(A[0 . . a − 1])
FUN(A[1 . . a])
FUN2(A)
```

Si supponga che FUN1 abbia complessità $\Theta(n)$ e FUN2 abbia complessità $\Theta(n^2)$, dove per entrambe le funzioni n è la lunghezza dell'array passato come argomento.

1. Scrivere la ricorrenza che definisce la complessità temporale $T_{\text{FUN}}(n)$ di FUN.
2. Dare un limite superiore asintotico per la funzione $T_{\text{FUN}}(n)$, giustificando opportunamente la risposta.

SOLUZIONE

Si ottiene una ricorrenza $T(n)=\Theta(1)$, per $n<5$, altrimenti $T(n)=2T(n-1)+\Theta(n^2)$. Da [1, Teorema 1] abbiamo la seguente maggiorazione: $T(n)=\mathcal{O}(n^2 2^n)$.

In aggiunta, è possibile dimostrare un limite asintotico più stretto, ossia $T(n)=O(2^n)$. Si può facilmente verificare che dimostrare $T(n)\le c2^n$ risulta complicato, a causa del termine $\Theta(n^2)$. Proviamo quindi a dimostrare che vale $T(n)\le c2^n-bn^2$, che ovviamente implica che $T(n)\le c2^n$ vale.

La nostra ipotesi induttiva è dunque $T(n-1)\le c2^{n-1}-b(n-1)^2$. Sostituendola nella ricorrenza, otteniamo:

$$\begin{aligned}
T(n) &= 2T(n-1)+dn^2\\
&\le 2(c2^{n-1}-b(n-1)^2)+dn^2\\
&= c2^n-2bn^2-2b+4bn+dn^2\\
&\le c2^n-2bn^2+4bn+dn^2\\
&\le c2^n-2bn^2+\frac{b}{2}n^2+dn^2\\
&= c2^n-\frac{3}{2}bn^2+dn^2\le c2^n-bn^2
\end{aligned}$$

L'ultima maggiorazione è valida se vale $-\frac{3}{2}b+d\le -b$, cioè se vale $d\le\frac{b}{2}$, quindi se $b\ge 2d$. Consideriamo la penultima maggiorazione, in cui si sfrutta la condizione $4bn\le\frac{b}{2}n^2$. Innanzi tutto, notiamo che si potevano sfruttare anche altre condizioni, per esempio $4bn\le\frac{b}{5}n^2$, in quel punto; la scelta di usare l'espressione $\frac{b}{2}n^2$ deriva dalla volontà di fare comparire un termine della forma pbn^2 che sia più piccolo di $2bn^2$ (in modo che $-2bn^2+pbn^2$ sia negativo), ma che sia anche tale che $-2bn^2+pbn^2$ sia più piccolo di $-bn^2$, per poter imporre opportuni vincoli su b e d in modo da poter soddisfare la condizione finale di essere $\le c2^n-bn^2$. La condizione $4bn\le\frac{b}{2}n^2$ (cioè $4\le\frac{n}{2}$) è valida per un valore di n sufficiente grande, ed in particolar modo per $n\ge 8$.

Di conseguenza, consideriamo come caso base $n=8$. Notiamo che per la validità della dimostrazione non è necessario imporre vincoli sulla costante c, quindi possiamo sceglierla grande a sufficienza per verificare la tesi per $n=8$, ovvero che $T(8)\le c2^8-b8^2=256c-64b$.

Si poteva anche direttamente notare, senza effettuare ulteriori maggiorazioni, che vale

$$c2^n-2bn^2+4bn+dn^2\le c2^n-bn^2$$

se, e solo se, vale

$$-bn+4b+dn\le 0$$

e quest'ultima disuguaglianza è verificata per $b>d$ e $n\ge\frac{4b}{b-d}$.

Esercizio 3.13.

1. Indicare la complessità asintotica della seguente funzione F(n) (si rammenta che in questo caso si fa riferimento al criterio di costo costante).

```
F(n)
sum := 0
for i := 1 to n
    for j := 1 to i^2
        if j mod i = 0
            for k := 1 to j
                sum := sum + 1
return sum
```

2. È possibile riscrivere la funzione F(n) in modo che, a parità di risultato calcolato, la sua complessità asintotica sia inferiore?

NB: L'operazione $a \bmod b$ restituisce il resto della divisione tra gli interi a e b.

SOLUZIONE

1. La complessità è $\Theta(n^4)$. Infatti, il ciclo più esterno esegue $\Theta(n)$ iterazioni. Il secondo ciclo ne esegue $\Theta(i^2)$ (che sono quindi dell'ordine di grandezza di $\Theta(n^2)$), per ogni iterazione i del primo ciclo. Il terzo ne esegue $\Theta(j)$, ossia $\Theta(i^2)$, ma viene eseguito solo ogni i iterazioni del secondo ciclo, quindi complessivamente esegue $\Theta(i)$ (ossia $\Theta(n)$) iterazioni per ogni iterazione del secondo ciclo. L'iterazione interna ha una complessità di $\Theta(1)$. Complessivamente abbiamo $\Theta(n)\cdot\Theta(n^2)\cdot\Theta(n)=\Theta(n^4)$.

   Un calcolo più preciso (che aiuta anche a svolgere i passi successivi) è il seguente. Come detto sopra, il ciclo più interno viene eseguito j volte, ma solo quando j è un multiplo di i. Quindi, esso viene eseguito quando $j=i,2i,3i,\ldots,i\cdot i$. Possiamo quindi scrivere la seguente sommatoria, per calcolare quante volte viene eseguita l'istruzione di incremento di sum:

   $$\sum_{i=1}^{n}\sum_{t=1}^{i}\sum_{k=1}^{t\cdot i}1.$$

   Ora, la sommatoria più interna chiaramente vale $t\cdot i$ e si ha che

   $$\sum_{t=1}^{i}t\cdot i=i\frac{i(i+1)}{2}=\frac{i^2(i+1)}{2} \qquad (3.1)$$

sfruttando la nota formula di Gauss, secondo cui vale la seguente uguaglianza:

$$\sum_{t=1}^{i} t = \frac{i(i+1)}{2}.$$

Infine, la sommatoria completa ha la forma

$$\sum_{i=1}^{n} \frac{i^2(i+1)}{2}$$

che è dell'ordine di grandezza di $\Theta(n^4)$. Questo si può vedere anche ricordando che sono noti i seguenti risultati (si veda anche [2, Appendice A]):

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \qquad (3.2)$$

$$\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4} \qquad (3.3)$$

2. La parte ciclica della funzione si può riscrivere come segue, abbassando la complessità a $\Theta(n^2)$.

```
for i := 1 to n
    j := i
    while j ≤ i * i
        sum := sum + j
        j := j + 1
```

Una riscrittura equivalente, ottenuta raccogliendo il fattore i dal secondo ciclo, è la seguente:

```
for i := 1 to n
    for k := 1 to i
        sum := sum + k * i
```

che ha complessità $\sum_{k=1}^{i} k \cdot i = i\frac{i(i+1)}{2}$ (come già calcolato in (3.1)). Questo consente di sostituire il ciclo interno con una formula esplicita. La riscrittura seguente porta a una complessità di $\Theta(n)$.

```
for i := 1 to n
    sum := sum + i * (i * (i + 1)/2)
```

Si può ricavare una semplificazione estrema mediante le formule per la somma di quadrati e per la somma di cubi riportate in (3.2) e (3.3), rispettivamente. Abbiamo quindi che vale la formula seguente

$$\sum_{i=1}^{n} \frac{i(i+1)}{2} i = \frac{1}{2}\sum_{i=1}^{n}(i^3+i^2) = \frac{n(n+1)(4n+2+3n^2+3n)}{2\cdot 12}$$
$$= \frac{n(n+1)(3n^2+7n+2)}{24}$$

ottenendo così un'unica formula (polinomio di quarto grado in n) che dà direttamente il risultato.

```
return n * (n + 1) * (3 * n * n + 7 * n + 2)/24
```

La complessità risultante è chiaramente $\Theta(1)$.

Esercizio 3.14. Sia $T[1..n][1..n]$ una matrice quadrata di dimensione $n \times n$. Si consideri la seguente procedura $\text{F}(T, n)$.

```
F(T, n)
v := ⌊n/2⌋
if v ≤ 1
    return T
w := ⌈n/2⌉
if w = v
    w := w + 1
A := T[1..v][1..v]
A1 := F(A, v)
B := T[w..n][w..n]
B1 := F(B, n − w + 1)
for i := 1 to n
    for j := 1 to n
        if i ≤ v and j ≤ v
            T[i][j] := A1[i][j]
        elseif i ≥ w and j ≥ w
            T[i][j] := B1[i − w + 1][j − w + 1]
        else T[i][j] := T[i][j] * T[i][j]
return T
```

1. Scrivere la ricorrenza associata alla funzione $\text{F}(T, n)$ definita sopra.
2. Si valuti la complessità temporale della funzione $\text{F}(T, n)$ risolvendo la ricorrenza definita al punto 1.

Soluzione

Si denoti con $N = n^2$ il numero di elementi della matrice T. La complessità della procedura è espressa dalla seguente ricorrenza (per evitare ambiguità con il nome della matrice T, nella ricorrenza indichiamo con T_{F} il tempo necessario per eseguire la funzione F):

$$T_{\text{F}}(N) = 2T_{\text{F}}\left(\frac{N}{4}\right) + \Theta(N)$$

Per risolvere la ricorrenza è possibile applicare il terzo caso del teorema dell'esperto, ottenendo quindi $T_{\text{F}}(N) = \Theta(N) = \Theta(n^2)$.

Esercizio 3.15. Si considerino le seguenti funzioni:

```
FUNC1(n)
k := 0
i := 0
while i ≤ n
    k := k + 1
    i := i + k
    (*)
return k
```

```
FUNC2(m)
if m ≤ 1
    return m
j := 1
while j ≤ m
    j := j * 3
return FUNC2(m/3) + FUNC2(m/3)
```

Si calcoli la complessità di $\text{FUNC1}(n)$ quando al posto di $(*)$ si trovano le seguenti istruzioni:

1. $\text{FUNC2}(10^5)$
2. $\text{FUNC2}(n)$

Soluzione

Il valore di i all'inizio della j-esima iterazione del ciclo in FUNC1 non è mai influenzato dall'esecuzione della funzione FUNC2. Esso è pari a $1+2+3+4+5+\ldots+(j-1)$, e quindi è dell'ordine di j^2. Quindi, il ciclo viene eseguito un numero di volte che è $\Theta(\sqrt{n})$.

Calcoliamo innanzi tutto la complessità di $\text{FUNC2}(m)$. Essa è data dalla seguente ricorsione: $T(m) = 2T\left(\frac{m}{3}\right) + \log_3(m)$. La ricorsione si risolve tramite il teorema dell'esperto, in quanto $\log_3(m)$ è polinomialmente più piccola di $m^{\log_3 2}$. Siamo quindi nel caso 1 del teorema dell'esperto, e la complessità di $\text{FUNC2}(m)$ è $\Theta(m^{\log_3 2})$.

Nel caso 1 l'istruzione $(*)$ ha costo costante, indipendente da n. Quindi, in questo caso la complessità del codice è data dal numero di iterazioni del ciclo, ed è $\Theta(\sqrt{n})$.

Nel caso 2 a ogni iterazione del ciclo il costo dell'istruzione $(*)$ è $\Theta(n^{\log_3 2})$, quindi il costo totale è $\Theta\left(n^{\frac{1}{2}+\log_3 2}\right)$.

Esercizio 3.16. Si considerino le seguenti ricorrenze:

1. $T(n) = 16T\left(\frac{n}{2}\right) + n^5\log(n)$
2. $T(n) = 4T\left(\frac{n}{64}\right) + \sqrt[4]{n}$
3. $T(n) = 2T\left(\frac{n}{2}\right) + (n\cos(n)\log(n))^2$

Per ogni ricorrenza, indicare un *limite asintotico superiore* per la funzione $T(n)$.

Soluzione

Le prime due ricorrenze si risolvono con il teorema dell'esperto.

Più precisamente, nella prima ricorrenza si ha che $\log_b a = \log_2 16 = 4$, e n^4 è polinomialmente più piccolo di $n^5\log(n)$, per cui siamo nel caso 3. In questo caso occorre anche verificare la condizione aggiuntiva, cioè che $af(\frac{n}{b}) \le cf(n)$ vale per qualche $c < 1$ e n grande a sufficienza. Questo corrisponde a verificare che valga $16(\frac{n}{2})^5\log(\frac{n}{2}) \le cn^5\log(n)$, cioè che valga $\frac{n^5}{2}(\log(n) - \log(2)) \le cn^5\log(n)$, che è banalmente verificata, ad esempio, per $c = \frac{3}{4}$ e $n \ge 1$. Quindi, vale $T(n) = \Theta(f(n)) = \Theta(n^5\log(n))$ (quindi ovviamente anche $T(n) = \mathcal{O}(n^5\log(n))$).

Nel caso della seconda ricorrenza, invece, $f(n)$ è della forma n^k, per cui basta confrontare k con $\log_b a$. Si ha che $\log_b a = \log_{64} 4 = \frac{1}{3} > \frac{1}{4} = k$, per cui siamo nel caso 1, e vale $T(n) = \Theta(n^{\log_b a}) = \Theta(\sqrt[3]{n})$ (e quindi $T(n) = \mathcal{O}(\sqrt[3]{n})$).

Nel caso della terza ricorrenza, vale $f(n) = (n\cos(n)\log(n))^2$, quindi $f(n)$ oscilla tra 0 e $(n\log(n))^2$, e quindi non è né una maggiorazione, né una minorazione di $n^{\log_b a} = n^{\log_2 2} = n$. Non si può quindi applicare il teorema dell'esperto. D'altro canto, è facile vedere che $T(n)$ è maggiorata da $(n\log(n))^2$. Infatti si ha che, per ogni n, vale $T(n) \le T'(n) = 2T'(\frac{n}{2}) + (n\log(n))^2$; questo può facilmente essere visto per induzione, notando che valgono sia $T(\frac{n}{2}) \le T'(\frac{n}{2})$, sia $f(n) \le (n\log(n))^2$. Applicando il teorema dell'esperto (caso 3), si ottiene che $T'(n) = \mathcal{O}((n\log(n))^2)$; infatti anche in questo caso la condizione aggiuntiva $af(\frac{n}{b}) \le cf(n)$, cioè $2(\frac{n}{2})^2\log^2(\frac{n}{2}) \le cn^2\log^2(n)$ (o, equivalentemente, $\frac{n^2}{2}(\log^2(n) - 2\log(n) + \log^2(2)) \le cn^2\log^2(n)$), vale per $c = \frac{3}{4}$ e $n > 1$. Di conseguenza, vale anche $T(n) = \mathcal{O}((n\log(n))^2)$.

Esercizio 3.17. Si consideri la seguente funzione G, che riceve un array di interi A e restituisce un intero:

```
G(A)
if A.length ≤ 1
    return 1
k := G(A[1..n/2])
j := G(A[n/2 + 1..n])
for h := 1 to A.length
    i := 1
    while i ≤ A.length
        i := i * 2
        k := k + i * A[h]
        j := j − i * A[h]
return (k + j)/2
```

Detta n la lunghezza dell'array A,

1. si scriva la ricorrenza associata al codice della funzione;
2. si fornisca un limite asintotico superiore (possibilmente stretto) per la complessità temporale di $\mathrm{G}(A)$ in funzione di n.

SOLUZIONE

La ricorrenza associata al codice della funzione è:

$$T(n) = \begin{cases} \Theta(1) \text{ se } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n\log(n)) \text{ se } n > 1 \end{cases}$$

o anche

$$T(n) = \begin{cases} 1 \text{ se } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n\log_2(n) \text{ se } n > 1 \end{cases}$$

La funzione $f(n) = n\log_2(n)$ cresce più velocemente di $\Theta(n^{\log_2 2}) = \Theta(n)$, ma non polinomialmente più velocemente. Pertanto non si può applicare il teorema dell'esperto.

Osserviamo che $T(n)$ cresce più velocemente di $T'(n) = 2T'\left(\frac{n}{2}\right) + n$ ma meno di $T''(n) = 2T''\left(\frac{n}{2}\right) + n^{1+\epsilon}$, per ogni $\epsilon > 0$ (le relazioni $T(n) \geq T'(n)$ e $T(n) \leq T''(n)$ si mostrano per banale induzione). Pertanto valgono sia $T(n) \in \Omega(n\log n)$ che $T(n) \in O(n^{1+\epsilon})$. Intuitivamente, ha quindi senso ipotizzare una complessità che sia più alta di $\Theta(n\log n)$, ma non polinomialmente più alta.

Formuliamo allora l'ipotesi che valga $T(n) \leq cn(\log_2 n)^2$ e procediamo per sostituzione.

$$\begin{aligned}
T\left(\frac{n}{2}\right) &\leq c\frac{n}{2}\left(\log_2\left(\frac{n}{2}\right)\right)^2 && \text{Ipotesi induttiva} \\
T(n) &\leq cn\left(\log_2\left(\frac{n}{2}\right)\right)^2 + n\log_2(n) && \text{Sostituzione} \\
cn\left(\log_2\left(\frac{n}{2}\right)\right)^2 + n\log_2(n) &\leq cn(\log_2 n)^2 && \text{Da verificare} \\
c\left(\log_2\left(\frac{n}{2}\right)\right)^2 + \log_2(n) &\leq c(\log_2 n)^2 && (n > 0) \\
c((\log_2 n)^2 - \left(\log_2\left(\frac{n}{2}\right)\right)^2) &\geq \log_2(n) \\
c((\log_2 n)^2 - (\log_2 n - 1)^2) &\geq \log_2(n) \\
c(2\log_2 n - 1) &\geq \log_2(n)
\end{aligned}$$

Quest'ultima relazione è sempre soddisfatta per $n > 1$ e $c \geq 1$.

Nel caso base, assumendo che valga $T(1) = 1$, notiamo che si ha che $c1(\log_2(1))^2 = 0 \not\geq T(1) = 1$. Tuttavia, è sufficiente mostrare che la disuguaglianza vale da un certo n in poi, per cui tramite la ricorrenza otteniamo $T(2) = 2T(1) + 2\log_2(2) = 4$ e $T(3) = 2T(1) + 3\log_2(3) = 2 + 3\log_2(3)$; inoltre $4 \leq c2(\log_2 2)^2$ e $2 + 3\log_2(3) \leq c3(\log_2 3)^2$ valgono entrambe per $c \geq 2$, e per $n > 3$ la ricorrenza non dipende più da $T(1)$. La relazione è soddisfatta anche nel caso base e quindi si ha che vale $T(n) = O(n(\log n)^2)$.

Si dimostra analogamente che $T(n) \geq dn(\log_2 n)^2$. Questa relazione è soddisfatta se

$$d(2\log_2 n - 1) \leq \log_2(n)$$

il che vale, ad esempio, per $d \leq 1/2$ e $n \geq 1$. Nel caso base vale $T(1) = 1 \geq d1(\log_2(1))^2 = 0$. Quindi vale $T(n) = \Omega(n(\log n)^2)$.

Complessivamente, abbiamo che vale $T(n) = \Theta(n(\log n)^2)$.

Un altro modo per risolvere la ricorrenza consiste nell'effettuare un cambio di variabile, ponendo $m = \log_2 n$:

$$\begin{aligned}
T(2^m) &= 2T(2^{m-1}) + 2^m m && \text{Pongo } m = \log_2 n, \text{ quindi } n = 2^m \\
T(2^m)/2^m &= 2T(2^{m-1})/2^m + m && \text{Divido per } 2^m \\
S(m) &= 2T(2^{m-1})/2^m + m && \text{Definisco } S(m) = T(2^m)/2^m \\
S(m) &= S(m-1) + m && \text{Quindi } S(m-1) = 2T(2^{m-1})/2^m \\
S(m) &= S(m-2) + m - 1 + m && \text{Sostituisco} \\
&\vdots \\
S(m) &= S(0) + 1 + 2 + \ldots + m && \text{Con } S(0) = T(2^0)/2^0 = 1 \\
S(m) &= \Theta(m^2) \\
T(2^m)/2^m &= \Theta(m^2) \\
T(n)/n &= \Theta((\log n)^2) \\
T(n) &= \Theta(n(\log n)^2)
\end{aligned}$$

Esercizio 3.18. Con la notazione $f(\mathcal{O}(n))$ indichiamo l'insieme delle funzioni $\{f(g(n)) \mid g(n) \in \mathcal{O}(n)\}$, cioè che sono date dalla composizione di una funzione f e di una funzione g, laddove la funzione g appartiene all'insieme $\mathcal{O}(n)$. Per esempio, la funzione $\frac{1}{4n^3+n}$ appartiene all'insieme $\frac{1}{\mathcal{O}(n^3)}$.

Si dica, giustificando opportunamente le risposte, se valgono le seguenti uguaglianze:

a) $\log(\mathcal{O}(n)) = \mathcal{O}(\log(n))$

b) $2^{\mathcal{O}(n)} = \mathcal{O}(2^n)$

c) $(\mathcal{O}(n))^k = \mathcal{O}(n^k)$

SOLUZIONE

a) Falsa. Si consideri ad esempio $f(n) = 2\log(n)$. Abbiamo chiaramente che vale $f(n) \in \mathcal{O}(\log(n))$. Tuttavia, vale anche $f(n) \notin \log(\mathcal{O}(n))$, cioè non è vero che $f(n)$ è una funzione della forma $\log(g(n))$, con $g(n) \in \mathcal{O}(n)$; infatti si ha che $f(n) = \log(n^2)$ e $n^2 \notin \mathcal{O}(n)$.

b) Falsa. Si prenda $f(n) = 2n$: certamente vale $f(n) \in \mathcal{O}(n)$, quindi $2^{f(n)} \in 2^{\mathcal{O}(n)}$, ma si ha anche che $2^{f(n)} = 2^{2n} \notin \mathcal{O}(2^n)$.

c) Vera. L'insieme $\mathcal{O}(n)$ è dato dalle funzioni g per cui esistono $c > 0, n_0 > 0$ tali che $g(n) \leq cn$ per $n > n_0$. Affinché valga $(g(n))^k \in \mathcal{O}(n^k)$ devono allora esistere $c' > 0, n_0' > 0$ tali che $(g(n))^k \leq c'n^k$ per $n > n_0'$, che è senz'altro verificata per c' opportuno e $n_0' = n_0$. Infatti, per $n > n_0$, vale $g(n) \leq cn$ e quindi $(g(n))^k \leq (cn)^k \leq c'n^k$ se $c' > c^k$.

L'altra direzione è analoga: sia h una funzione in $\mathcal{O}(n^k)$, cioè tale per cui esistono $c > 0, n_0 > 0$ tali che $h(n) \leq cn^k$ per $n > n_0$. Affinché valga $h(n) \in \mathcal{O}(n)^k$ si deve mostrare che esiste una funzione $g(n) \in \mathcal{O}(n)$ tale che vale $h(n) = g(n)^k$. Tale funzione è $\sqrt[k]{h(n)}$, basta mostrare che vale $\sqrt[k]{h(n)} \in \mathcal{O}(n)$. Ciò è vero, infatti vale $\sqrt[k]{h(n)} \leq (\sqrt[k]{c})n$ per $n > n_0$.