UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Corso di Laurea Magistrale in Ingegneria Informatica

Dipartimento di Ingegneria Elettrica e delle tecnologie dell'informazione

PRIMO HOMEWORK DEL CORSO DI

ALGORITMI E STRUTTURE DATI

Professore:
Roberto Pietrantuono

Candidati:
del Gaudio Raffaele M63001389
Vitrano Arianna M63001171

ANNO ACCADEMICO 2022/2023
Primo Semestre

Indice

Capitolo 1

Esercizi

1.1 Notazione asintotica e crescita delle funzioni

Per ogni gruppo di funzioni, ordina le funzioni in ordine crescente di complessità asintotica (big-O):

1)

$f1(n) = n^{0.999999} logn$

$f2(n) = 10000000n$

$f3(n) = 1.000001^n$

$f4(n) = n^2$

2)

$f1(n) = 2^{2^{1000000}}$

$f2(n) = 2^{10000n}$

$f3(n) = \binom{n}{2}$

$f4(n) = n\sqrt{n}$

3)

$f1(n) = n^{\sqrt{n}}$

$f2(n) = 2^n$

$f3(n) = n^{10} \cdot 2^{n/2}$

$f4(n) = \sum_{i=1}^{n}(i + 1)$

1.1.1 Soluzione

Partiamo dalla definizione di complessità asintotica (big-O):

O(g(n))={f(n): esistono c e n_0 positive tali che $0 \le f(n) \le cg(n)$ per ogni $n \ge n_0$}

1) L'ordine corretto in termini di complessità asintotica (big-O) è il seguente:

$f1(n) < f2(n) < f4(n) < f3(n)$

- $f1(n) = O(f2(n))$ perché $\forall c > 0$:

  $\lim_{n\to\infty} \frac{f2(n)}{f1(n)} = \frac{c \cdot 10000000n}{n^{1-0.000001} logn} = \frac{c \cdot 10000000n^{0.000001}}{logn} = +\infty$

- la funzione $f4$ è quadratica mentre la funzione f2 è lineare quindi: $f2(n) = O(f4(n))$;

- la funzione $f3$ è esponenziale quindi: $f4(n) = O(f3(n))$;

2) L'ordine corretto in termini di complessità asintotica (big-0) è il seguente:

$f1(n) < f4(n) < f3(n) < f2(n)$

- $f1(n) = 2^{2^{1000000}} = O(1)$

- $f4(n) = n\sqrt{n} = n^{\frac{3}{2}} = O(n^{\frac{3}{2}})$

- $f3(n) = \binom{n}{2} = \frac{n(n-1)(n-2)!}{2!(n-2)!} = \frac{n(n-1)}{2!} = O(n^2)$

- $f2(n) = 2^{100000n} = (2^{100000})^n = O(\alpha^n)$

3) L'ordine corretto in termini di complessità asintotica (big-0) è il seguente:

$f4(n) < f1(n) < f3(n) < f2(n)$

- $f4(n) = \sum_{i=1}^{n}(i+1) = \sum_{i=1}^{n} i + n = \frac{n(n+1)}{2} + n = \frac{n^2+n}{2} + n = \frac{n^2+3n}{2} = O(n^2)$

- $f1(n) = n^{\sqrt{n}} = 2^{\sqrt{n} logn}$

- $f2(n) = 2^n$

- $f3(n) = n^{10} \cdot 2^{\frac{n}{2}} = 2^{log(n^{10})} \cdot 2^{\frac{n}{2}} = 2^{\frac{n}{2}+10logn}$

$f4$ è l'unica funzione non esponenziale e quindi asintoticamente è la più piccola. Per l'analisi delle altre tre, basta osservare che:

$\lim_{n\to\infty} \frac{f2(n)}{f1(n)} = \infty$

$\lim_{n\to\infty} \frac{f2(n)}{f3(n)} = \infty$

$\lim_{n\to\infty} \frac{f3(n)}{f1(n)} = \infty$

1.2 Notazione asintotica e crescita delle funzioni

Per ognuna delle seguenti funzioni si determini se f(n) = O(g(n)), g(n) = O(f(n)) o entrambe.

$-f(n) = (n^2 - n)/2 \qquad g(n) = 6n$

$-f(n) = n + 2\sqrt{n} \qquad g(n) = n^2$

$-f(n) = nlogn \qquad g(n) = n\sqrt{n}/2$

$-f(n) = n + logn \qquad g(n) = \sqrt{n}$

$-f(n) = 2(logn)^2 \qquad g(n) = logn + 1$

$-f(n) = 4nlogn + n \qquad g(n) = (n^2 - n)/2$

1.2.1 Soluzione

- $f(n) = (n^2 - n)/2 = O(n^2) \qquad g(n) = 6n = O(n)$

  g(n) = O(f(n))

  In quanto n^2 è asintoticamente maggiore di n.

- $f(n) = n + 2\sqrt{n} = O(n) \qquad g(n) = n^2 = O(n^2)$

  f(n) = O(g(n))

  In quanto n^2 è asintoticamente maggiore di n.

- $f(n) = nlogn = O(nlogn) \qquad g(n) = n\sqrt{n}/2 = O(n\sqrt{n})$

  f(n) = O(g(n))

  In quanto il fattore $\sqrt{n}$ è asintoticamente maggiore di $logn$.

- $f(n) = n + logn = O(n) \qquad g(n) = \sqrt{n} = O(\sqrt{n})$

  g(n) = O(f(n))

  In quanto n è asintoticamente maggiore di $\sqrt{n}$.

- $f(n) = 2log^2n = O(log^2n) \qquad g(n) = logn + 1 = O(logn)$

  g(n) = O(f(n))

  In quanto log^2n è asintoticamente maggiore di $logn$.

- $f(n) = 4nlogn + n = O(nlogn) \qquad g(n) = (n^2 - n)/2 = O(n^2)$

  f(n) = O(g(n))

  In quanto il fattore n è asintoticamente maggiore di $logn$.

1.3 Notazione asintotica e crescita delle funzioni

Si indichi se le seguenti affermazioni sono vere o false

- $2^{n+1} = O(2^n)$
- $2^{2n} = O(2^n)$

1.3.1 Soluzione

1) $2^{n+1} = O(2^n)$ è **vero**:

$2^{n+1} = 2 \cdot (2^n) = O(2^n)$

La costante 2 non influisce sull'ordine di grandezza.

$2^{n+1} = 2 \cdot 2^n \Rightarrow \lim_{n\to\infty} \frac{2^n}{2 \cdot 2^n} = \frac{1}{2} \Rightarrow 2^{n+1} = \Theta(2^n) \Rightarrow 2^{n+1} = O(2^n)$

2) $2^{2n} = O(2^n)$ è **falso**:

$2^{2n} = 2^n \cdot 2^n = 4^n = O(4^n) \neq O(2^n)$

A conferma di quanto detto:

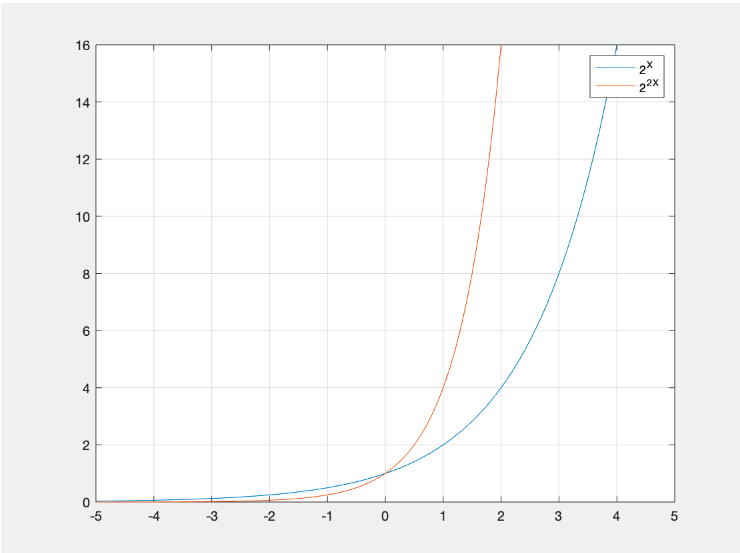


Figura 1.1: La funzione 2^{2n} in rosso è asintoticamente maggiore di 2^n

1.4 Ricorrenze

Fornire il limite inferiore e superiore per T(n) nella seguente ricorrenza, usando il metodo dell'albero delle ricorrenze ed il teorema dell'esperto se applicabile. Si fornisca il limite più stretto possibile giustificando la risposta.

$T(n) = 2T(n/2) + O(\sqrt{n})$
$T(n) = T(\sqrt{n}) + \Theta(loglogn)$ (suggerimento: si operi una sostituzione di variabili)
$T(n) = 10T(n/3) + 17n^{1.2}$

Utilizzando l'albero di ricorsione, dimostrate che la soluzione della ricorrenza:
T(n) = T(n/3) + T(2n/3) + cn, dove c è una costante, è $\Omega(nlogn)$

1.4.1 Soluzione

- $T(n) = 2T(n/2) + O(\sqrt{n})$

  Metodo dell'esperto

  T(n)=aT(n/b)+f(n)

  Avremo che: a=2, b=2 e f(n)=$\sqrt{n}$

  Calcoliamo: $\log_b a = \log_2 2 = 1$

  $n^{\frac{1}{2}} = f(n) = O(n^{1-\epsilon})$ con $0 < \epsilon \leq \frac{1}{2}$, si applica il caso 1° del teorema e quindi avremo che T(n) = $\Theta(n)$

  Metodo dell'albero di ricorrenza

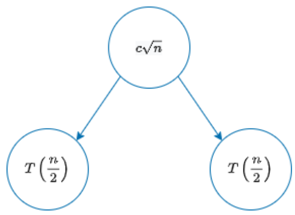


Figura 1.2: Albero di ricorsione

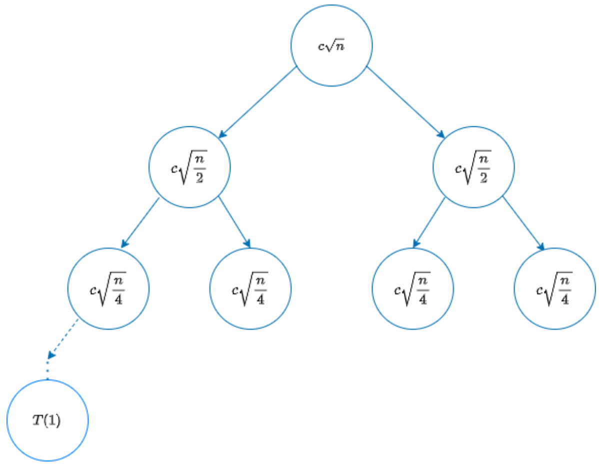


Figura 1.3: Albero di ricorsione

$T(n) = \sum_{i=0}^{\log_2 n-1} 2^i \cdot c\sqrt{\frac{n}{2^i}} + T(1) \cdot 2^{\log_2 n} =$

$= c\sqrt{n} \cdot \sum_{i=0}^{\log_2 n-1} \sqrt{2}^i + T(1) \cdot n =$

$= c\sqrt{n} \cdot \frac{1-\sqrt{2}^{log_2 n}}{1-\sqrt{2}} + T(1) \cdot n =$

$= c\sqrt{n} \cdot \frac{1-\sqrt{n}}{1-\sqrt{2}} + T(1) \cdot n =$

$= c\frac{\sqrt{n}-n}{1-\sqrt{2}} + T(1) \cdot n = \Theta(n)$

Dato che $T(n) = \Theta(n)$ il limite superiore ed inferiore più stretto è n.

- $T(n) = T(\sqrt{n}) + \Theta(loglogn)$

  Effettuiamo la seguente sostituzione di variabili:

  $logn = m \Rightarrow n = 2^m$

  $T(2^m) = T(2^{\frac{m}{2}}) + \Theta(logm)$

  Si consideri la nuova ricorrenza $S(m) = T(2^m)$:

  $S(m) = S(\frac{m}{2}) + \Theta(logm)$

  Metodo dell'esperto

  Si tenta di applicare il metodo dell'esperto alla ricorrenza $S(m) = S(\frac{m}{2}) + \Theta(logm)$

  Tuttavia si nota che f(m) = $logm$ non è né polinomialmente più grande né polinomialmente più piccolo di $m^{\log_2 1}$ né tantomeno è $\Theta(m^{\log_2 1})$.

  Per tale ragione il metodo dell'esperto non è applicabile ad S(m) e quindi nemmeno a T(n).

Metodo dell'albero di ricorrenza

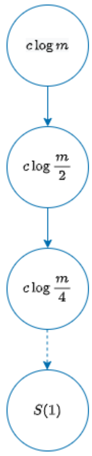


Figura 1.4: Albero di ricorsione

$S(m) = \sum_{i=0}^{logm-1} clog(\frac{m}{2^i}) + S(1) \cdot n^{log1} =$

$= c\sum_{i=0}^{logm-1} logm - c\sum_{i=0}^{logm-1} i + S(1) =$

$= clog^2m - c\frac{(logm-1)\cdot(logm)}{2} =$

$= c\frac{1}{2}log^2m$ - $c\frac{1}{2}logm + S(1) = \Theta(log^2m)$

Quindi $S(m) = \Theta(log^2m)$ e sostituendo di nuovo la m otteniamo $T(n) = \Theta(log^2logn)$

Dato che $T(n) = \Theta(log^2logn)$ il limite superiore ed inferiore più stretto è log^2logn.

- $T(n) = 10T(n/3) + 17n^{1.2}$

  Metodo dell'esperto

  T(n)=aT(n/b)+f(n)

  Avremo che: a=10, b=3 e f(n)=$17n^{1.2}$

  Calcoliamo: $\log_b a = \log_3 10 \approx 2.1$

  $17n^{1.2} = f(n) = O(n^{2.1-\epsilon})$ $\forall\epsilon < 0.9$. Si applica il 1° caso del teorema e quindi avremo che $T(n) = \Theta(n^{\log_3 10})$

  Metodo dell'albero di ricorrenza

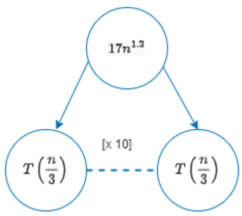


Figura 1.5: Albero di ricorsione

L'altezza dell'albero h è $\log_3 n$ perché $\frac{n}{3^i} = 1$ quando $i = \log_3 n$

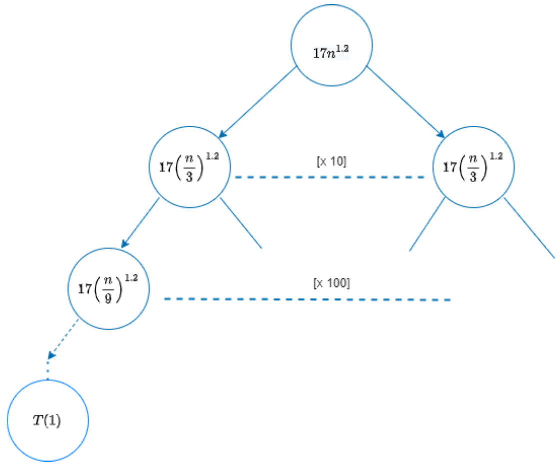


Figura 1.6: Albero di ricorsione

T(n)= $10 \cdot (17(\frac{n}{3})^{1.2}) + 100 \cdot (17(\frac{n}{9})^{1.2}) + ... + n^{\log_3 10} \cdot T(1)$

T(n)= $10 \cdot (17(\frac{n}{3})^{1.2}) + 100 \cdot (17(\frac{n}{9})^{1.2}) + ... + \Theta(n^{\log_3 10})$

Quindi:

T(n) $= \sum_{i=0}^{log_3 n-1} 17n^{1.2} \cdot (\frac{10}{3^{1.2}})^i + \Theta(n^{\log_3 10}) =$

$= 17n^{1.2} \cdot (\frac{1-(\frac{10}{3^{1.2}})^{\log_3 n}}{1-\frac{10}{3^{1.2}}}) + \Theta(n^{\log_3 10}) =$

$= \frac{17 \cdot n^{1.2}}{1-\frac{10}{3^{1.2}}} \cdot (1 - n^{\log_3 \frac{10}{3^{1.2}}}) + \Theta(n^{\log_3 10}) = \Theta(n^{\log_3 10})$

Dato che $T(n) = \Theta(n^{\log_3 10})$ il limite superiore ed inferiore più stretto è $n^{\log_3 10}$.

- Utilizzando l'albero di ricorsione, dimostrate che la soluzione della ricorrenza

  T(n) = T(n/3) + T(2n/3) + cn, dove c è una costante, è $\Omega(nlogn)$

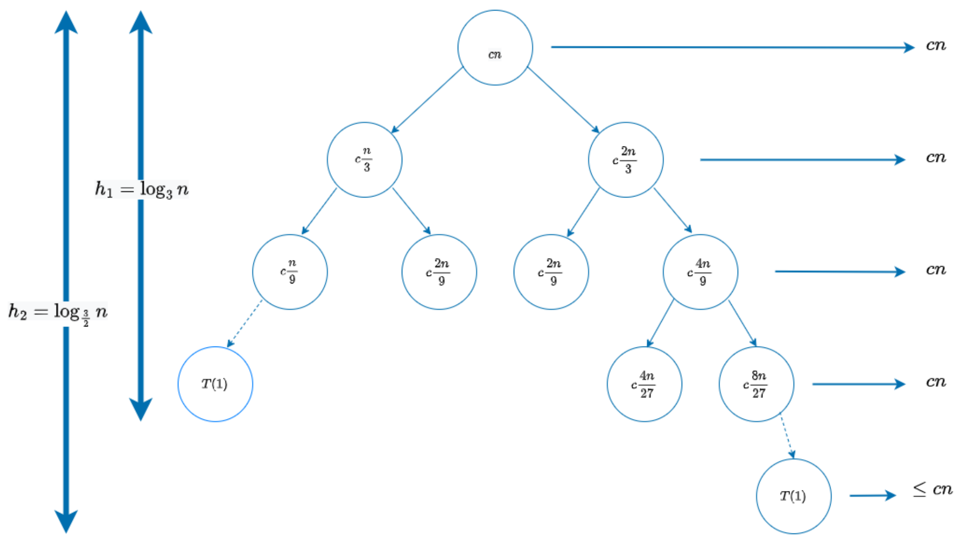


Figura 1.7: Albero di ricorsione

T(n) $= \sum_{i=0}^{log_3 n} cn + \sum_{i=log_3 n+1}^{\log_{\frac{3}{2}} n} \gamma(i) \cdot n$

Considerando $\gamma(i)$ come il costo generico al livello i-esimo dell'albero per i livelli compresi dal $(log_3 n + 1)$-esimo al $(log_{\frac{3}{2}} n)$-esimo. La prima sommatoria rappresenta il costo fino al livello $log_3 n$, mentre la seconda sommatoria dal livello $log_3 n + 1$ fino alla fine.

T(n)= $cn(log_3 n) + \Gamma n(\log_{\frac{3}{2}} n - log_3 n)$

con $\Gamma = \frac{\sum_{i=log_3 n+1}^{\log_{\frac{3}{2}} n} \gamma(i)}{(\log_{\frac{3}{2}} - log_3)}$ per il teorema della media integrale.

Si consideri ora la definizione di complessità asintotica (Notazione Ω):

$\Omega(g(n))$=$\{f(n)$: esistono k e n_0 positive tali che $0 \leq kg(n) \leq f(n)$ per ogni $n \geq n_0\}$

Si dimostra che T(n) = $\Omega(nlog_3 n)$:

$0 \leq knlog_3 n \leq cn \log_3 n + \Gamma n \log_{\frac{3}{2}} n - \Gamma nlog_3 n$

$0 \leq k \leq c + \frac{\Gamma}{\log_3 \frac{3}{2}} - \Gamma \leq c + \Gamma(\frac{1}{\log_3 \frac{3}{2}} - 1)$

Valido $\forall n > 1$ con $k \leq c + \alpha$, con $\alpha = \Gamma(\frac{1}{\log_3 \frac{3}{2}} - 1)$

Capitolo 2

Problemi

2.1 Problema 1.1 - Soluzione

Il codice della soluzione del problema 1.1 è nella cartella ”Problema 1.1”. I casi di test sono generati automaticamente e casualmente ad ogni lancio e il programma fornisce sullo stdout l'input dell'esecuzione e il risultato, in modo che ne sia facilmente verificabile la correttezza.

L'analisi della complessità temporale dell'algoritmo è svolta nel codice mediante brevi commenti durante e alla fine della funzione che lo implementa.

2.2 Problema 1.2 - Soluzione

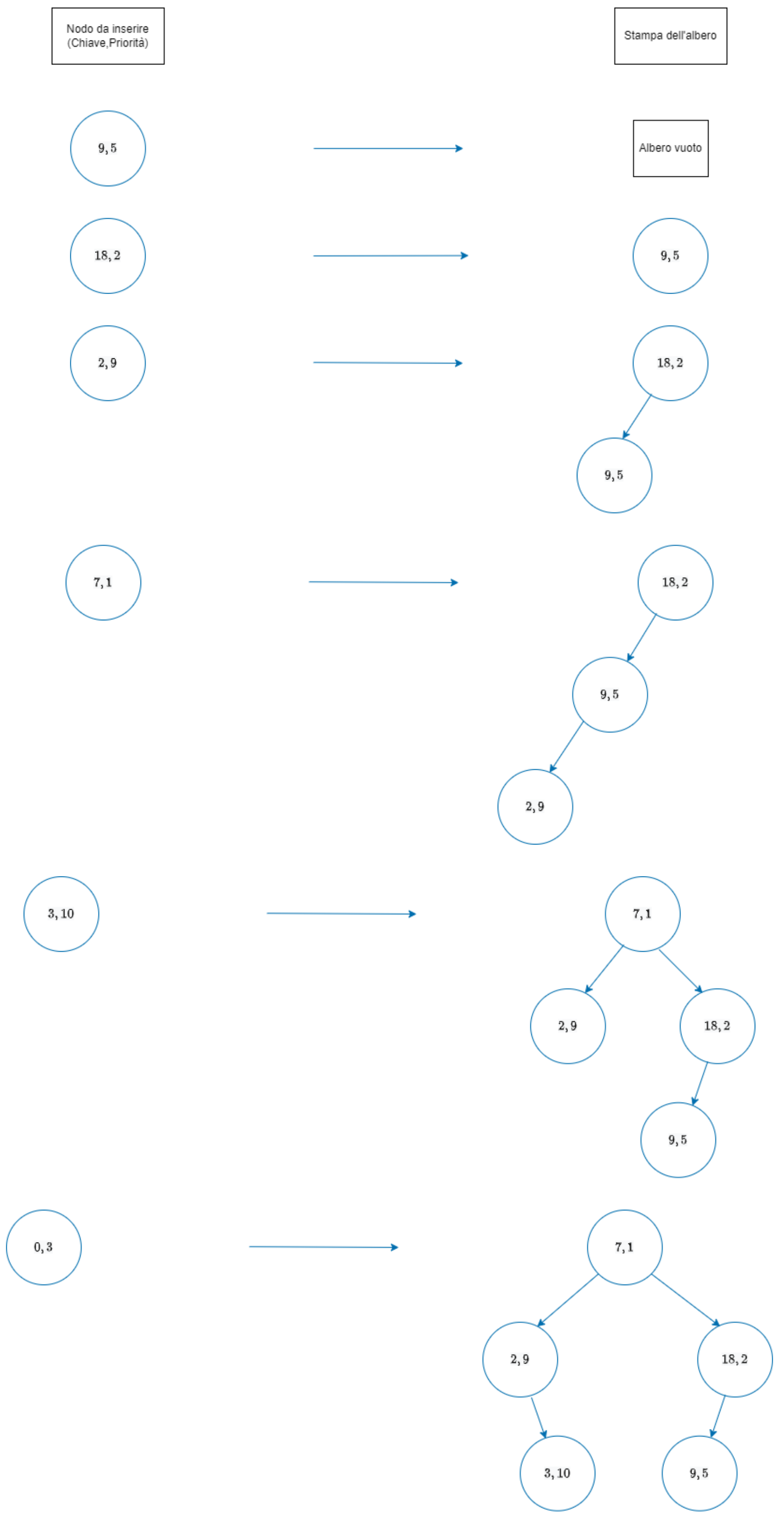
Il codice della soluzione del problema 1.2 è nella cartella ”Problema 1.2”. I casi di test sono generati automaticamente e casualmente ad ogni lancio e il programma fornisce sullo stdout l'input dell'esecuzione e il risultato, in modo che ne sia facilmente verificabile la correttezza.

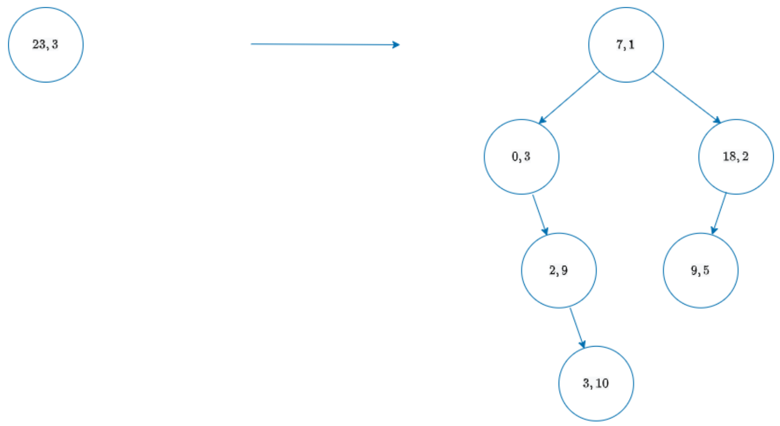
L'analisi della complessità temporale dell'algoritmo è svolta nel codice mediante brevi commenti durante e alla fine della funzione che lo implementa.

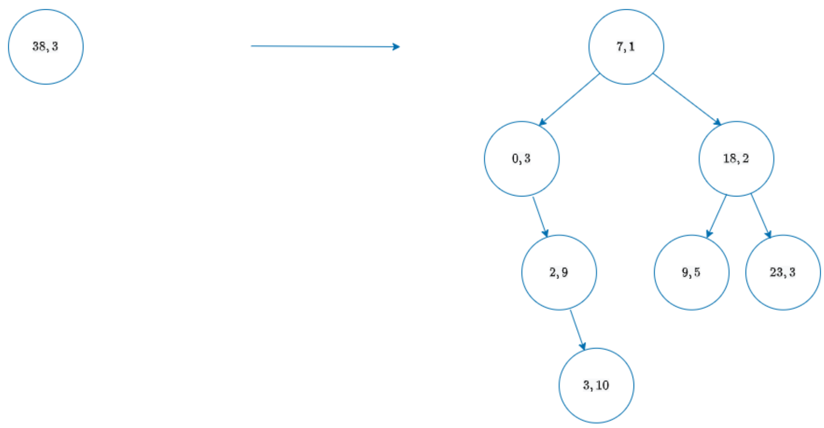
2.3 Problema 1.3 - Soluzione

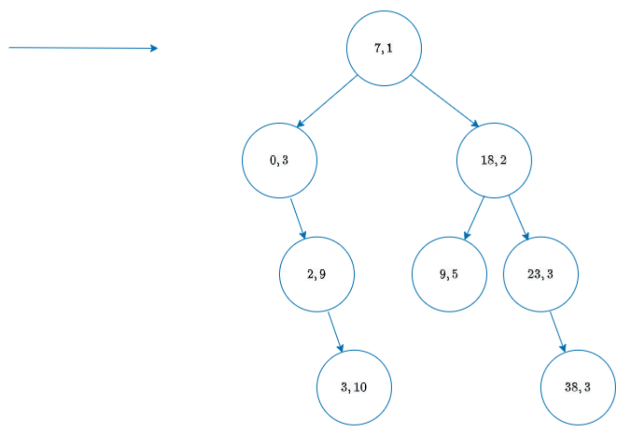
Il codice della soluzione del problema 1.3 è nella cartella ”Problema 1.3”.

Nel codice viene eseguito un test di inserimento in sequenza di 8 nodi in un Treap e viene stampato l'albero dopo ogni inserimento. Di seguito, invence, è possibile osservare lo stesso processo eseguito manualmente, in modo da verificare il funzionamento della funzione implementata.


Nodo da inserire (Chiave,Priorità)
Stampa dell'albero
9, 5
Albero vuoto
18, 2
9, 5
2, 9
18, 2
9, 5
7, 1
18, 2
9, 5
2, 9
3, 10
7, 1
2, 9
18, 2
9, 5
0, 3
7, 1
2, 9
18, 2
3, 10
9, 5


23, 3
7, 1
0, 3
18, 2
2, 9
9, 5
3, 10


38, 3
7, 1
0, 3
18, 2
2, 9
9, 5
23, 3
3, 10


7, 1
0, 3
18, 2
2, 9
9, 5
23, 3
3, 10
38, 3