# Algoritmi e Principi dell'Informatica
# Soluzioni al Tema d'esame

**20 giugno 2024**

## Informatica teorica

### Esercizio 1 (8 punti)

Per ciascuno dei seguenti problemi, indicare, barrando coerentemente le opportune caselle e motivandolo, se sono decidibili o meno, semidecidibili o meno.

1. Stabilire se ogni macchina di Turing il cui indice è pari e minore di 100 si arresta ricevendo in ingresso il proprio indice.

2. Stabilire se ogni macchina di Turing il cui indice è pari si arresta ricevendo in ingresso il proprio indice.

3. Stabilire se una generica macchina di Turing ha un indice pari.

4. Stabilire se una generica macchina di Turing restituisce un numero pari per almeno un ingresso.

Soluzione

1. La domanda è chiusa, quindi il problema è decidibile (e quindi semidecidibile).

2. La domanda è ancora chiusa.

3. Tipicamente immaginiamo che una generica macchina di Turing sia fornita "passandone" l'indice e quindi il problema equivale a stabilire se un generico numero sia pari, il che è decidibile (e semidecidibile). Se anche immaginassimo che la macchina fosse fornita specificandone la descrizione formale (tupla), potremmo comunque utilizzare l'enumerazione di Gödel per ricostruire tale descrizione per ogni indice possibile, con la certezza che prima o poi troveremmo l'indice corrispondente alla macchina descritta, per poi stabilirne, come richiesto, la parità.

4. Il problema è indecidibile poiché la proprietà descritta caratterizza un insieme non banale di funzioni a cui applicare il teorema di Rice (c'è almeno una funzione computabile che restituisce un numero pari per almeno un ingresso, ad esempio la funzione $f_1(x) = 2$ e almeno una che non lo fa, per esempio $f_2(x) = 1$). Tuttavia il problema è semidecidibile poiché è possibile enumerare diagonalmente tutte le coppie $\langle x, n\rangle$, con $x$ ingresso e $n$ numero di passi di esecuzione, e stabilire così, mediante la macchina di Turing universale, se qualche coppia $\langle x, n\rangle$ ha causato l'arresto con la scrittura in uscita di un numero pari (se esiste, prima o poi viene individuata).

### Esercizio 2 (8 punti)

1. Si definisca una grammatica a potenza minima che generi il linguaggio $L_1$ su $\{a, b\}$ delle stringhe in cui il numero di $a$ sia pari al numero di $b$ più 2 (cioè, $\forall x(x \in L_1 \leftrightarrow \#(x, a) = \#(x, b) + 2)$).

2. Cambierebbe il tipo di grammatiche a potenza minima per generare il linguaggio $L_2$ delle stringhe in cui il numero di $a$ è doppio rispetto al numero di $b$ (cioè, $\forall x(x \in L_2 \leftrightarrow \#(x, a) = 2 \cdot \#(x, b))$)? Giustificare la risposta. Non serve specificare la grammatica che genera il linguaggio, ma solo il suo tipo.

Soluzione

1. Una possibile grammatica è la seguente:

$$S \to TaTaT$$
$$T \to aTbT \mid bTaT \mid \epsilon$$

   Il nonterminale $T$ corrisponde a stringhe con un egual numero di $a$ e di $b$. In particolare, ogni stringa siffatta ricade in uno dei casi descritti: se inizia con una $a$, ci sarà senz'altro un primo punto nella stringa in cui l'eccedenza di $a$ viene bilanciata da una $b$ e tra questi due caratteri ci potrà quindi solo essere una stringa bilanciata (così come dopo la $b$); simmetricamente se la stringa inizia con una $b$; la stringa vuota è pure bilanciata.

   La prima regola evidenzia come, in una stringa del linguaggio, ci dev'essere una prima $a$ in eccesso (preceduta quindi da una stringa bilanciata) e una seconda $a$ in eccesso (preceduta dalla prima $a$ in eccesso e da stringhe bilanciate), eventualmente seguita da una stringa bilanciata.

2. Non cambierebbe, in quanto il linguaggio sarebbe chiaramente riconoscibile mediante un automa a pila (deterministico), che dovrebbe solo utilizzare un contatore unario sulla pila togliendo/mettendo un simbolo per ogni $a$ letta e due per ogni $b$ e verificando che a fine stringa la pila sia vuota.

# Algoritmi e Principi dell'Informatica
# Soluzioni al Tema d'esame

**20 giugno 2024**

## Algoritmi e strutture dati

### Esercizio 3 (8 punti)

Si forniscano limiti asintotici superiori per le seguenti equazioni alle ricorrenze, giustificando brevemente la risposta:

1. $T(n) = T(n-1) + T(n-3) + n,$ con $T(1) = 2$ e $T(2) = 4$ e $T(3) = 6$
2. $T(n) = 2T\left(\frac{n}{2}\right) + n^2 \log(n)$
3. $T(n) = 4T\left(\frac{n}{3}\right) + \frac{n}{\log(n)}$
4. $T(n) = 2T\left(\frac{n}{2}\right) + 10n$

Soluzione

1. Si tratta di una ricorrenza lineare di ordine costante in cui la somma dei coefficienti dei termini in $T$ vale 2 e il termine esterno ha grado 1. Si ha allora $T(n) = \mathcal{O}(2^n n)$.
2. Caso 3 del Master Theorem: $f(n) = n^2 \log(n)$ e, poiché $f(n)$ è polinomialmente più grande di $n^{\log_2(2)}$ e soddisfa la condizione di regolarità, segue che $T(n) = \Theta(n^2 \log(n))$.
3. Caso 1 del Master Theorem: $f(n) = \frac{n}{\log(n)}$, con $f(n)$ asintoticamente minore di $n^{\log_3(4)}$, quindi $T(n) = \Theta(n^{\log_3(4)})$.
4. Caso 2 del Master Theorem: $f(n) = 10n$, che è proporzionale a $n^{\log_2(2)}$, quindi $T(n) = \Theta(n \log(n))$.

### Esercizio 4 (8 punti)

Un "multi-insieme di interi" è una collezione che si comporta come un insieme di interi, ma ammette duplicati e tiene traccia del numero di occorrenze di ciascun intero. Si richiede di descrivere in pseudocodice le seguenti procedure, dove $M$ è un multi-insieme e $n$ è un intero, fornendone anche le complessità asintotiche nel caso medio:

1. INSERT$(M, n)$, che inserisce $n$ in $M$ e restituisce il numero di occorrenze di $n$ in $M$ dopo l'inserimento;
2. DELETE$(M, n)$, che cancella $n$ da $M$ e restituisce il numero di occorrenze di $n$ in $M$ dopo la cancellazione (restituisce $-1$ se $n$ è assente);
3. COUNT$(M, n)$, che restituisce il numero di occorrenze di $n$ in $M$.

Soluzione

Utilizziamo una tabella hash (con chaining) in cui gli elementi siano dotati anche di un campo *contatore*, che tiene traccia del numero di occorrenze (quindi $M$ è la tabella hash e $n$ viene usato come chiave). Se la funzione hash $h$ soddisfa l'ipotesi di hashing uniforme semplice, la complessità di tutte e tre le procedure è $\mathcal{O}(1 + \alpha)$, con $\alpha$ che indica il fattore di carico (rapporto tra elementi inseriti e dimensione della tabella) nel caso medio.

1. L'impianto operativo di INSERISCI$(M, n)$ è il seguente: data una chiave $n$, viene cercato se l'elemento corrispondente è presente nella tabella. Se questo è il caso, viene incrementato il contatore delle occorrenze, altrimenti viene aggiunto un nuovo elemento, impostandone il contatore a 1. Di seguito, lo pseudocodice:

```
INSERISCI(M, n)
 1  i := h(n) // opportuna funzione hash
 2  x := M[i]
 3  while x ≠ NIL ∧ x.key ≠ n
 4        x := x.next
 5  if x ≠ NIL
 6        x.contatore := x.contatore + 1
 7        return x.contatore
 8  alloca nuovo nodo y
 9  y.key := n
10  y.contatore := 1
11  y.next := M[i]
12  M[i] := y
13  return 1
```

2. L'impianto operativo di CANCELLA$(M, n)$ è il seguente: data una chiave $n$, viene cercato se l'elemento corrispondente è presente nella tabella. Se questo è il caso, se l'elemento ha più di una occorrenza, si decrementa il contatore, altrimenti viene eliminato. Nel seguito lo pseudocodice:

```
CANCELLA(M, n)
 1  i := h(n)
 2  x := M[i]
 3  prev := NIL
 4  while x ≠ NIL ∧ x.key ≠ n
 5        prev := x
 6        x := x.next
 7  if x ≠ NIL
 8        if x.contatore > 1
 9              x.contatore := x.contatore − 1
10              return x.contatore
11        else if prev = NIL
12                    M[i] := x.next
13              else prev.next := x.next
14              dealloca x
15        return 0
16  return -1
```

3. La procedura COUNT$(M, n)$ necessita di effettuare una ricerca dell'elemento nella tabella, e restituire il contatore, se presente, 0 altrimenti.

```
COUNT(M, n)
1  x = M[h(n)]
2  while x ≠ NIL ∧ x.key ≠ n
3        x = x.next
4  if x = NIL
5        return 0
6  return x.contatore
```