

Tutoraggio

Linguaggi regolari

$L \in \text{Regolare} \iff \text{DFA}$

1.

$\Sigma = \{0, 1\}$ $L = \{\sigma \in \Sigma^* \mid \text{val}(\sigma) \equiv 0 \pmod{3}\}$ \uparrow $\text{val}: \Sigma^* \rightarrow \mathbb{N}$
(numeri divisibili per 3)
Esempi $11 = 3 \in L$

$n \div 3$

$0 \rightarrow 0$	$3 \rightarrow 0$	$6 \rightarrow 0$
$1 \rightarrow 1$	$4 \rightarrow 1$	$7 \rightarrow 1$
$2 \rightarrow 2$	$5 \rightarrow 2$	$8 \rightarrow 2$

\hookrightarrow classi di equivalenza

Aggiungere 0 \rightarrow $\begin{array}{r} 01010 \\ \hline 5 \\ \hline 10 \end{array} \rightarrow \cdot 2$

Aggiungere 1 \rightarrow $\begin{array}{r} 01011 \\ \hline 5 \\ \hline 11 \end{array} \rightarrow \cdot 2 + 1$

Classi di equivalenza

$n = 3K + 0$ Resto 0

$n = 3K + 1$ Resto 1

$n = 3K + 2$ Resto 2

$n = 3K + 0$

Aggiungere 0

$$n = 3K \rightarrow 2n = 3K$$

$$n = 3 \frac{K}{2} + \underline{0}$$

Aggiungere 1

$$n = 3K \rightarrow 2n + 1 = 3K$$

$$2n = 3K - 1 \quad \begin{array}{l} -1 \rightarrow 2 \\ 0 \rightarrow 0 \\ 1 \rightarrow 1 \\ 2 \rightarrow 2 \\ 3 \rightarrow 0 \end{array}$$

$$2n = 3K + 2$$

$$n = 3 \frac{K}{2} + \underline{1}$$

$n = 3K + 1$

Aggiungere 0

$$2n = 3K + 1$$

$$2n = 3K - 2$$

Aggiungere 1

$$2n + 1 = 3K + 1$$

$$n = 3 \frac{K}{2} + \underline{0}$$

$$n = 3k + 2$$

Aggiungere 0

$$2n = 3k + 2$$

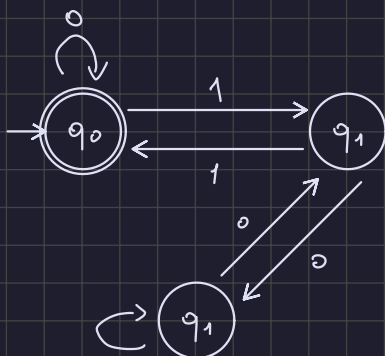
$$n = 3 \frac{k}{2} + 1$$

Aggiungere 1

$$2n + 1 = 3k + 2$$

$$2n = 3k + 1$$

$$n = 3 \frac{k}{2} + 1$$



Dimostrazione

$$x \in L \iff \hat{\delta}^1(q_0, x) \in F$$

Separando il se e solo se bisogna dimostrare le seguenti implicazioni:

$$1) x \in L \Rightarrow \hat{\delta}^1(q_0, x) \in F$$

$$2) x \in L \Leftarrow \hat{\delta}^1(q_0, x) \in F \quad \equiv \quad x \notin L \Rightarrow \hat{\delta}^1(q_0, x) \notin F$$

Dimostro per induzione sulla lunghezza della stringa: $|\sigma|$

Caso base

È una stringa lunga il minimo possibile per mostrare sia che appartiene sia che non appartiene

$$|\sigma| = 1$$

$$\sigma = 0 \Rightarrow \hat{\delta}^1(q_0, 0) = q_0 \in F$$

$$\sigma = 1 \Rightarrow \hat{\delta}^1(q_0, 1) = q_1 \notin F$$

Non bisogna necessariamente coprire tutti gli stati dell'automa nei casi base perchè basta una stringa che finisce in uno stato finale e una che non finisce in uno stato finale

Passo induttivo

Ipotesi induttiva

$$1) \forall \sigma. \exists n > 1. |\sigma| \leq n : \sigma \in L \Rightarrow \hat{\delta}^1(q_0, \sigma) = \{q_0\} \in F$$

$$2) \forall \sigma. \exists n > 1. |\sigma| \leq n : \sigma \notin L \Rightarrow \hat{\delta}^1(q_0, \sigma) = \{q_1, q_2\} \notin F$$

Prendo una stringa di lunghezza $n+1$: Σ è una stringa binaria multiplo di 3

$$\sigma = \sigma' \alpha \quad |\sigma'| = n, \alpha \in \Sigma$$

$$- \sigma \in L, |\sigma| = n+1$$

Pumping lemma linguaggi regolari

$L \in \text{Reg} \Rightarrow \text{Pumping lemma}$



$z \in L, |z| \geq n, \exists z = uvw, |uv| \leq n, |v| > 0, \forall i \in \mathbb{N}. uv^i w \in L$

$\neg \text{Pumping lemma} \Rightarrow L \notin \text{Reg}$

Definizione

Sia $L \in \text{REG}$, allora:

$$\exists n \in \mathbb{N}. \forall z \in L. |z| \geq n \Rightarrow \exists u, v, w \in \Sigma^*. z = uvw \wedge \underbrace{\begin{cases} |uv| \leq n \\ |v| > 0 \\ \forall i \geq 0. uv^i w \in L \end{cases}}_{\text{Pumping property}}$$

$a \rightarrow b \equiv \neg a \vee b$ $\downarrow \equiv$

$$\underbrace{(\neg(z = uvw) \vee \neg(|uv| \leq n) \vee \neg(|v| > 0) \vee \neg(\forall i \geq 0. uv^i w \in L))}_{\neg \alpha} \quad \beta$$

Negazione

$\forall n \in \mathbb{N}. \exists z \in L. |z| \geq n \wedge \forall u, v, w \in \Sigma^*.$

$(z = uvw \wedge |uv| \leq n \wedge |v| > 0) \Rightarrow \exists i \geq 0. uv^i w \notin L$

$$1. L = \{0^n 1^m \mid n \neq m\}$$

Condizioni di appartenenza

$$|0| \neq |1|$$

Consideriamo la stringa

$$z = 0^k 1^{k+1} \quad \begin{matrix} n=k \\ m=k+1 \end{matrix}$$

Pompriamo la stringa

$$z_i = 0^{k+(i-1)|v|} 1^{k+1}$$

$$- i=0$$

$$z_0 = 0^{k-|v|} 1^{k+1}$$

$$|0| \neq |1| \Rightarrow k-|v| \neq k+1 \Rightarrow |v| \neq -1 \quad \checkmark$$

$$- i=1$$

$$z_1 = 0^{k+|v|} 1^{k+1}$$

$$|0| \neq |1| \Rightarrow k+|v| \neq k+1 \Rightarrow |v| \neq 1 \quad \checkmark$$

Non si riesce a trovare un pompaggio che rompe le condizioni del pumping lemma utilizzando la stringa z . Bisogna quindi usare un altro approccio.

Per le proprietà di chiusura sappiamo che: $L \in \text{Reg} \Leftrightarrow \bar{L} \in \text{Reg}$

Quindi se dimostriamo che complemento di L non è regolare con il pumping lemma negato, allora anche L originale non è regolare

$$\bar{L} = \{0^n 1^m \mid n=m\}$$

Scegliamo una stringa

$$z = 0^k 1^k \quad n=m=k$$

Pompriamo la stringa

$$z_i = 0^{k+(i-1)|v|} 1^k$$

$$- i=2$$

$$z_2 = 0^{k+|v|} 1^k$$

$$|0| = |1| \Rightarrow k+|v| = k \Rightarrow |v| = 0 \text{ Viola i vincoli del pumping lemma}$$

Abbiamo quindi dimostrato che \bar{L} non è regolare e quindi non è regolare nemmeno L

$$2. L_m = \{ x 2 y 2 x^R \mid y \in \{0\}^*, |y| \leq m, x = 1^{n^m} \} \quad m \in \mathbb{N}$$

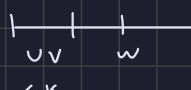
Siccome x è formata solo da 1 il reverse di x non ha alcun effetto

$$\begin{aligned} \cdot L_0 &= \{ x 2 y 2 x^R \mid y \in \{0\}^*, |y| \leq 0, x = 1^{n^0} = 1 \} \\ &= \{ 1 2 2 1 \} \quad \text{Qualunque linguaggio FINITO è regolare} \Rightarrow L_0 \in \text{Reg} \end{aligned}$$

$$\begin{aligned} \cdot L_1 &= \{ x 2 y 2 x^R \mid y \in \{0\}^*, |y| \leq 1, x = 1^n \} \\ &= \{ 1^n 2 \quad 2 1^n \} \cup \{ 1^n 2 0 2 1^n \} \end{aligned}$$

Sembra non sia regolare perchè ci sono due "gruppi" vincolati tra di loro (1^n) cioè il numero di uni deve essere uguale sia a sinistra che a destra. Dimostriamo quindi che il linguaggio non è regolare:

Prendiamo una stringa z

$$z = 1^k 2 2 1^k \in L \quad n=k$$


Pompriamo la stringa

$$z_i = 1^{k+(i-1)|v|} 2 2 1^k$$

$$- i=2 \quad z_2 = 1^{k+|v|} 2 2 1^k$$

$$|1|_{sx} = |1|_{dx} \Rightarrow k+|v| = k \Rightarrow |v| = 0 \quad \text{Viola i vincoli del pumping lemma}$$

Quindi il linguaggio L_1 non è regolare

$$\cdot L_2 = \{ x 2 y 2 x^R \mid y \in \{0\}^*, |y| \leq 2, x = 1^{n^2} \}$$

Questo linguaggio non è nè regolare e nè CF, quindi da $m \geq 2$ si ottengono soltanto linguaggi non CF (da dimostrare con il pumping lemma dei CF)

2.1 L'unione è regolare?

Verifichiamo l'unione di tutti i linguaggi è regolare

$$\begin{aligned}\bigcup_m L_m &= L' \\ &= \{x \geq 0^m \geq x^R \mid x = 1^{n^m}\} \quad (\text{In un'unione è come se la variabile "sparisse"}) \\ &= \{1^{n^m} \geq 0^m \geq 1^{n^m} \mid x = 1^{n^m}\}\end{aligned}$$

Questo linguaggio non è CF (da dimostrare con il pumping lemma dei CF)

L'intersezione è regolare?

Verifichiamo se l'intersezione di tutti i linguaggi è regolare

$$\begin{aligned}\bigcap_m L_m &= L' \\ &= L_0 \cap L_1 \cap L_2 \cap \dots \cap L_m \\ &= \{1221\} \cap \{1221, \dots\} \cap \dots \cap L_m \\ &= \{1221\}\end{aligned}$$

Siccome il linguaggio è finito, allora è regolare

$$\begin{aligned}3. L &= \{0^m 0^{2n} 0^{k+m} \mid m, n, k > 0\} \\ &= \{0^{2m+2n+k} \mid m, n, k > 0\}\end{aligned}$$

Siccome le variabili sono su un solo simbolo, e quindi non ci sono dipendenze tra simboli diversi, si può dire che il linguaggio è regolare.

$$4. L_p = \{a^{p \cdot m} b^m a^{p \cdot n} \mid m, n \geq 1\} \quad p > 0$$

$$\bullet L_1 = \{a^m b^m a^n \mid m, n \geq 1\} \quad \text{Sembra CF perchè } a \text{ e } b \text{ dipendono entrambi dalla stessa variabile } m$$

Dimostriamo che non è regolare con il pumping lemma

Prendiamo una stringa nel linguaggio

$$z = a^k b^k a \in L$$

Pompriamo la stringa

$$z_i = a^{k+(i-1)|v|} b^k a$$

$$\bullet i = 0$$

$$z_0 = a^{k-|v|} b^k a$$

$$|a| = |b| \Rightarrow k - |v| = k \Rightarrow |v| = 0 \quad \text{Viola i vincoli del pumping lemma} \quad z_i \notin L_1$$

$$\bullet L_2 = \{a^{2^m} b^m a^{2^n} \mid m, n \geq 2\} \quad \text{CF}$$

$$z_i = a^{2^k} b^k a^2 \in L$$

$$z_i = a^{2k + (i-1)|v|} b^k a^2$$

$$\bullet i=0$$

$$z_0 = a^{2k-|v|} b^k a^2$$

$$|a| = 2|b| \Rightarrow 2k - |v| = 2k \Rightarrow |v| = 0 \quad \text{Viola i vincoli del pumping lemma} \quad z_i \notin L$$

Osserviamo che con $p \geq 1$ i linguaggi sono tutti CF

4.1 L'unione è regolare?

Verifichiamo l'unione di tutti i linguaggi è regolare

$$\bigcup_p L_p = L' \\ = \{ a^{p^m} b^m a^{p^n} \mid m, n, p \geq 1 \}$$

Sicuramente non è regolare. Possiamo vedere che non è CF perchè a è legato a b per la variabile m , e la a a sinistra è legata a quella a destra per la variabile p .

$$= \{ a^{p^m} b^m c^{p^n} \mid m, n, p \geq 1, c = a \}$$

Le condizioni di appartenenza sono:

$$1) \begin{cases} |a| = p|b| \\ |a|n = |c|m \end{cases}$$

Pumping lemma:

Prendiamo una stringa nel linguaggio

$$z = a^{k^2} b^k c^{k^2} \in L \quad m=n=p=k$$

Pompriamo la stringa

$$z_i = a^{k^2 + (i-1)|v|} b^k c^{k^2}$$

$$\bullet i=2$$

$$z_2 = a^{k^2 + |v|} b^k c^{k^2}$$

$$1) |a| = p|b| \Rightarrow k^2 + |v| = k \cdot k \Rightarrow |v| = 0 \quad \text{Viola i vincoli del pumping lemma}$$

$$2) |a|n = |c|m \Rightarrow (k^2 + |v|)k = k^2 \cdot k \Rightarrow |v| = 0 \quad \text{Viola i vincoli del pumping lemma}$$

$$\Downarrow \\ z_i \notin L$$

L'unione dei linguaggi non è regolare

L'intersezione è regolare?

Verifichiamo l'intersezione di tutti i linguaggi è regolare

$$\begin{aligned}\bigcap_p L_p &= L' \\ &= L_1 \cap L_2 \cap \dots \cap L_p \\ &= \{a^m b^m a^n, m, n \geq 1\} \cap \{a^{2^m} b^m a^{2^n}, m, n \geq 1\} \cap \dots \cap L_p \\ &= \emptyset \in \text{Reg}\end{aligned}$$

Linguaggi Context Free

$$x \in L \iff x \in L(G), \quad G = \langle V, T, P, S \rangle$$

$$S \in V$$

$$P: X \in V \rightarrow \alpha \in (V \cup T)^*$$

$$\text{Se } P = \emptyset \rightarrow G = \langle \{S, X, Y, \dots\}, \{a, b, c, \dots\}, \emptyset, S \rangle \rightarrow L(G) = \emptyset$$

Un linguaggio può essere generato da più grammatiche

$$1. L = \{a^n b^n \mid n \neq m\} \notin \text{Reg}$$

Bisogna trovare una grammatica e dimostrarla induttivamente

$$G = \begin{cases} S \rightarrow aSb \mid A \mid B \\ A \rightarrow aA \mid a \\ B \rightarrow bB \mid b \end{cases}$$

oppure

$$n \neq m \equiv \underbrace{n > m}_{G_1} \vee \underbrace{n < m}_{G_2}$$
$$G' = G_1 \cup G_2$$

$$G_1 = A \rightarrow aAb \mid aA \mid a$$

$$G_2 = B \rightarrow aBb \mid b \mid Bb$$

$$G' = \begin{cases} S \rightarrow A \mid B \\ A \rightarrow aAb \mid aA \mid a \\ B \rightarrow aBb \mid b \mid Bb \end{cases} \quad (G_1 \vee G_2)$$

Dimostrazione di G'

Tesi:

$$1) x \in L \Rightarrow S \rightarrow^* x$$

$$2) S \rightarrow^* x \Rightarrow x \in L$$

$$1) x' \in L \Rightarrow S \rightarrow^* x'$$

$$|a| > |b|:$$

Caso base

$$|x| = 1$$

$$x = a \Rightarrow S \rightarrow A \rightarrow a$$

$$x = b \Rightarrow S \rightarrow B \rightarrow b$$

Passo induttivo

$$\text{Ipotesi induttiva: } x = a^{n+h} b^n, h > 0 \Rightarrow S \rightarrow^* x \equiv S \rightarrow A \rightarrow^* x$$

Consideriamo una stringa x' lunga $n+1$ $|x'| > |x|$

$$1. x' = a^{h+(n+1)} b^{n+1}$$

$$= a \underbrace{a^{n+h} b^n}_x b \Rightarrow S \rightarrow A \rightarrow a A b \xrightarrow{ii}^* a x b \equiv x'$$

$$2. x' = a^{n+(h+1)} b^n$$

$$= a a^{n+h} b^n \Rightarrow S \rightarrow A \rightarrow a A \xrightarrow{ii}^* a x \equiv x'$$

$|a| < |b|$ analogo

$$2) S \xrightarrow{n} x \Rightarrow x \in L$$

$|a| > |b|$:

Caso base

$n=2$ (numero minimo di passi)

$$S \rightarrow A \rightarrow a$$

$$S \rightarrow B \rightarrow b$$

Passo induttivo

$$\text{Ipotesi induttiva: } S \xrightarrow{n} x \equiv S \rightarrow A \xrightarrow{n-1} x$$

$$\cdot \underbrace{S \rightarrow A \rightarrow a A \xrightarrow{ii}^{n-1} a x}_{n+1 \text{ passi}} \in L$$

$$\cdot \underbrace{S \rightarrow A \rightarrow a A b \xrightarrow{n-1} a x b}_{n+1 \text{ passi}} \in L$$

$|a| < |b|$: analogo

$$2. L = \{ \sigma c^n d^m \mid \sigma \in \{a, b\}^*, |\sigma| = |c|, n, m \in \mathbb{N} \}$$

È intuitivamente context free perchè c'è un legame tra il gruppo sigma e il gruppo c, mentre invece il gruppo d è indipendente dagli altri

$$S \rightarrow S d \mid D$$

$$D \rightarrow a D c \mid b D c \mid \varepsilon$$

Questo linguaggio è formato da due gruppi indipendenti quindi si possono separare e unire due grammatiche

$$S \rightarrow A D$$

$$D \rightarrow d D \mid \varepsilon$$

$$A \rightarrow a A c \mid b A c \mid \varepsilon$$

Pumping lemma context free

Sia $L \in CF$, allora:

$$\exists n \in \mathbb{N}. \forall z \in L. |z| \geq n \Rightarrow \exists u, v, w, x, y \in T^*. z = uvxy \wedge \begin{cases} |vwx| \leq n \\ |vx| > 0 \\ \forall i \geq 0. uv^iwx^iy \in L \end{cases}$$

Negazione

$$\forall n \in \mathbb{N}. \exists z \in L. |z| \geq n \Rightarrow \forall u, v, w, x, y \in T^*.$$

$$.(z = uvwxy \wedge |vwx| \leq n \wedge |vx| > 0) \Rightarrow \exists i \geq 0. uv^iwx^iy \notin L$$

Esempio: $L = \{0^n 1^{2^n} \mid n \geq 0\}$

Non è intuitivamente regolare perchè non si riesce a creare un automa che lo riconosce.
Non è nemmeno intuitivamente context free perchè contiene una funzione non lineare.

$$z = 0^n 1^{2^n} \rightarrow |z| = n + 2^n \geq n$$

Esistono diverse combinazioni di suddivisioni e si devono verificare tutte:



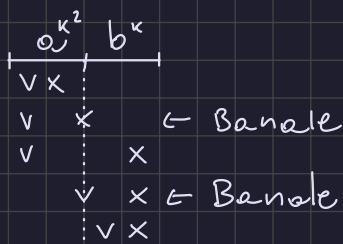
Questi due casi sono spesso banali perchè rompono lo schema del linguaggio

$$1. L = \{a^{m \cdot n} b^n \mid n, m \geq 1\}$$

Scegliamo una stringa z

$$\bullet k = m = n \geq 1$$

$$z = a^{k^2} b^k$$



$$1. (vx) \in a^{k^2}$$

$$z' = a^{k^2 + (i-1)|vx|} b^k \quad |a| = |b| \cdot l \quad l \geq 1$$

$$- i = 2$$

$$k^2 + |vx| = k \cdot l$$

$$k^2 - kL + |vx| = 0$$

$$k = \frac{L \pm \sqrt{L^2 - 4|vx|}}{2}$$

Se $|vx| = 1$ la stringa appartiene ancora al linguaggio, quindi con questo pompaggio non si riesce ad uscire dal linguaggio.

$$\bullet m = 1 \quad n = k \rightarrow |a| = |b| \cdot l \quad l \geq 1$$

$$z = a^k b^k$$

$$1) z' = a^{k + (i-1)|vx|} b^k$$

$$- i=2$$

$$z'_2 = a^{k+|vx|} b^k$$

$$k+|vx| = kL$$

$$k - kL = |vx|$$

$$k(L-1) = |vx|$$

$$L=1 \rightarrow |vx|=0$$

$$L>1 \rightarrow |vx|>k$$

} *Assurdo* ($|vx|>0$ e $|vx| \leq k$)

$$2) z' = a^{k+(i-1)|v|} b^{k+(i-1)|x|}$$

$$- i=2$$

$$z'_2 = a^{k+|v|} b^{k+|x|}$$

$$k+|v| = (k+|x|)L$$

$$k+|v| = kL+|x|L$$

$$k(L-1) = |v|-|x|L$$

$$L=1 \rightarrow |v|=|x| \quad \text{Non si può dire niente a riguardo}$$

Quindi questo pompaggio non fa uscire la stringa dal linguaggio, si deve trovare un'altra stringa

$$\bullet m=k \quad n=1 \rightarrow |a|=L \quad L \geq 1$$

$$z = a^m b$$

$$1. z' = a^{m+(i-1)|vx|} b$$

$$- i=2$$

$$k+|vx| = L \quad \text{Non si può dire nulla a riguardo}$$

Questo pompaggio non porta la stringa fuori dal linguaggio, quindi bisogna trovarne un'altra

$$\bullet m=k \quad n=p \quad p \text{ è primo } > k \rightarrow |a|=|b|L$$

$$z = a^{kp} b^p$$

$$1. z' = a^{kp+(i-1)|vx|} b^p$$

$$- i=2$$

$$kp+|vx| = pL$$

$$p(L-k) = |vx|$$

$$\text{Se } l < k \rightarrow p = |vx|(k-1)$$

$$\text{Se } k = 1 \rightarrow |vx| = p \text{ Assurdo } \forall k$$

$$\text{Se } l > k \rightarrow -p = |vx|(k+1) \text{ Assurdo } \forall k$$

Abbiamo dimostrato che tutte le possibili suddivisioni portano all'assurdo, quindi il linguaggio non è context free.

Ulteriore soluzione: $m = k \quad n = k^2$

Insiemi ricorsivi

1.

$$A = \{x \mid \exists y \in \mathbb{N} . x = y^2 + 1\}$$

Definiamo la funzione caratteristica

$$f_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

Bisogna fornire lo pseudocodice che calcola la funzione caratteristica:

```
input(x)
if x ≤ 1 return x;
for y = 1 to x {
  if x == y^2 + 1 {
    return 1;
  }
}
return 0;
```

2.

$$A = \{x \mid \varphi_x(x^2) \text{ non converge in meno di } n \text{ passi}\}$$

Definiamo la funzione caratteristica

$$f_A(x) = \begin{cases} 1 & x \in A = \varphi_x(x)^2 \nmid \text{ in meno di } n \text{ passi} \\ 0 & x \notin A = \varphi_x(x)^2 \downarrow \text{ in meno di } n \text{ passi} \end{cases}$$

Bisogna fornire lo pseudocodice che calcola la funzione caratteristica:

```
input(x,n)
costruisco phi_x // Esiste una procedura algoritmica che la costruisce

y = x^2
for i = 1 to n {
  esegui uno step di phi_x(y)
  if phi_x(y) ha terminato {
    return 0
  }
}
return 1
```

3.

$$A = \{x \mid \exists y \in \mathbb{N} . x = y^3 + y^2 + y\}$$

Definiamo la funzione caratteristica

$$f_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

Bisogna fornire lo pseudocodice che calcola la funzione caratteristica:

```
input(x)
for y = 0 to x {
  z = y^3 + y^2 + y
  if x == z {
    return 1;
  }
}
return 0;
```

4.

$$A = \{ x^2 \mid \varphi_x(x^3 + 1) \text{ non converge in 2 passi} \}$$

Definiamo la funzione caratteristica

$$f_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

Bisogna fornire lo pseudocodice che calcola la funzione caratteristica:

```
input(x)
if x non è un quadrato return 0

y = sqrt(x)
costruisco phi_y # Esiste un procedimento algoritmico che la calcola

z = y^3 + 1
for i = 1 to 2 {
  eseguo il prossimo step di phi_y(z)
  if phi_y(z) converge then return 0
}

return 1
```