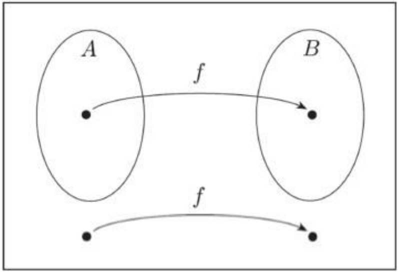
ELEMENTI DI TEORIA DELLA COMPUTAZIONE

M. Anselmo

a.a. 2022/23

RIDUZIONI E PROBLEMI INDECIDIBILI



11 maggio 2023

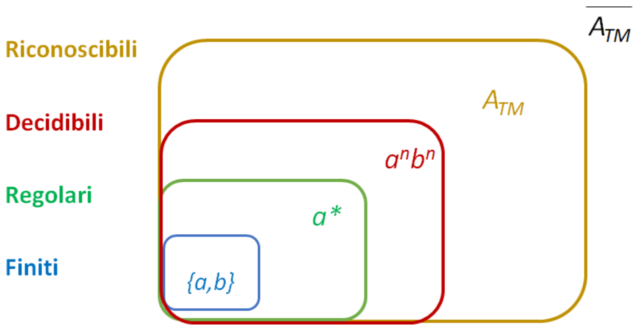
Obiettivo: analizzare i limiti della risoluzione dei ”problemi” mediante ”algoritmi”.

Problemi = Linguaggi di stringhe

Algoritmi = Macchine di Turing

Proveremo che esistono problemi che possono essere risolti mediante algoritmi e altri no.

Risultati



Aggiungeremo altri elementi a questo schema.

Risultati

Sia $A_{TM} = \{\langle M, w \rangle \mid M$ `e una MdT che accetta la parola $w\}$

- A_{TM} non `e decidibile
- A_{TM} `e riconoscibile
- $\overline{A_{TM}}$ non `e riconoscibile.

Per dimostrare che A_{TM} `e riconoscibile abbiamo mostrato che `e riconosciuta dalla Macchina di Turing Universale U. U ha un interesse che va oltre questa dimostrazione.

Una propriet`a dei linguaggi decidibili

Definizione

Diciamo che un linguaggio L `e co-Turing riconoscibile se il complemento di L è T uring riconoscibile.

Teorema

Un linguaggio L `e decidibile se e solo se L `e Turing riconoscibile **e** *co-Turing riconoscibile.*

Chiusura per complemento

La classe dei linguaggi **decidibili** è **chiusa** rispetto al complemento.

La classe dei linguaggi **riconoscibili** non è chiusa rispetto al complemento.

Provare indecidibilità

E' importante riconoscere che un problema P è indecidibile.

Come? Abbiamo 3 possibilità:

- Per assurdo, supporre l'esistenza di una MdT che decide P e provare che questo conduce a una contraddizione.

- Considerare un problema P_{ind} di cui sia nota l'indecidibilit\`a e dimostrare che P è "più difficile" di $P_{ind}$, ovvero che *P* "non è piu\` facile" di P_{ind}.

- Teorema di Rice (per alcuni casi).

Introduzione alle riduzioni

Come formalizzare il concetto che un problema è “più difficile” di un altro?

Lezione 1 di PA: Scheduling di attività

Problema computazionale:

Input / Istanza:

S = { 1, 2, … , n } insieme delle richieste

Per ogni richiesta i :

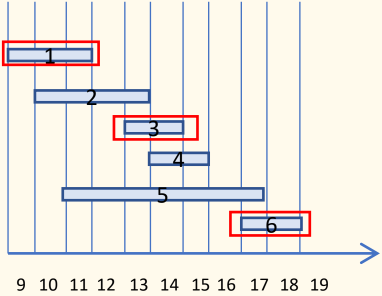
- s_i tempo di inizio
- f_i tempo di fine

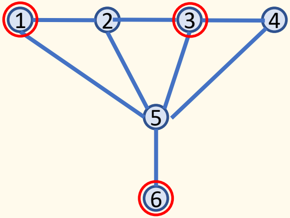
- **Output / Soluzione:**

S’ sottoinsieme di S di **attività compatibili** (= orari non si accavallano) tale che **Card(S’) massima**

Scheduling di attività: una soluzione

Grafo della compatibilità: ogni nodo è un intervallo; due nodi collegati se si accavallano



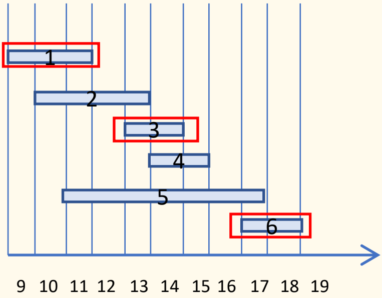


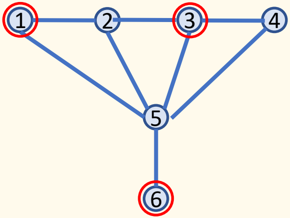
Input:
S = {1, 2, 3, 4, 5, 6} intervalli

$\Rightarrow$

Input:
Grafo $G_S = (V, E)$

Scheduling di attività & Insieme indipendente



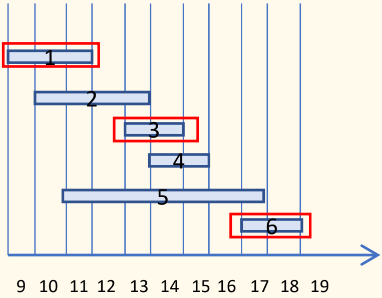


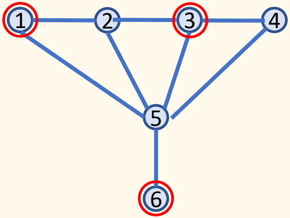
S = {1, 3, 6}
Insieme di intervalli compatibili

⇔

S = {1, 3, 6}
Insieme di nodi indipendenti
(tale che ogni coppia di nodi NON è collegata)

Scheduling di attività & Insieme indipendente





S = {1, 3, 6}
Insieme di intervalli compatibili di cardinalità massima

⇔

S = {1, 3, 6}
Insieme di nodi indipendenti di cardinalità massima

Scheduling di attività & Insieme indipendente

Scheduling di attività

Istanza:

S = { 1, 2, … , n } e ∀ i in S:

- s_i tempo di inizio
- f_i tempo di fine

k intero

Soluzione:

SI se esiste S' sottoinsieme di S di attività compatibili di cardinalità k

NO, altrimenti

Independent Set

Istanza:

G = (V, E) grafo

k intero

Soluzione:

SI se esiste V' sottinsieme di V indipendente di cardinalità k

NO, altrimenti

Scheduling di attività & Insieme indipendente

Scheduling di attività

Istanza:

$S = \{ 1, 2, \dots, n \}$ e $\forall$ i in S:

- s_i tempo di inizio
- f_i tempo di fine

k intero

Soluzione:

SI (esiste sottoinsieme di S di attività compatibili di cardinalità k)

NO

$\xrightarrow{f}$

Independent Set

Istanza (particolare):

$G_S = (V, E)$ grafo

k intero

Soluzione:

SI (esiste sottinsieme di V indipendente di cardinalità k)

NO

Scheduling di attività $\leq$ Insieme indipendente

Scheduling di attività		Insieme indipendente
Istanza	$\xrightarrow{f}$	**Istanza (particolare)**
Soluzione		**Soluzione**:
SI	$\longleftrightarrow$	**SI**
NO	$\longleftrightarrow$	**NO**

Se avessi un algoritmo per risolvere il problema dell'**insieme indipendente** $\Rightarrow$ avrei un algoritmo per risolvere il problema dello **scheduling di attività**. Viceversa, non è detto.

Quindi il problema dello scheduling di attività ha **difficoltà ≤** del problema dell'insieme indipendente.

Scheduling di attività $\leq_m$ Insieme indipendente

Scheduling di attività si riduce mediante funzione (f) a Insieme indipendente

Allineamento di sequenze

Allineamento di sequenza (problema di ricerca)

Istanza: Due stringhe $X = x_1\, x_2 \, . \, . \, . \, x_m$ e $Y = y_1\, y_2 \, . \, . \, . \, y_n$

Soluzione: un allineamento di costo minimo

m	e	a	n	-	
n	-	a	m	e	
1	2		1	2	= 6

Allineamento di sequenza (problema decisionale)

Istanza: Due stringhe $X = x_1\, x_2 \, . \, . \, . \, x_m$ e $Y = y_1\, y_2 \, . \, . \, . \, y_n$ e un intero k

Soluzione:

SI, se esiste un allineamento di costo k

NO, altrimenti

Allineamento di sequenze & Cammini minimi

Ad $X = x_1 \dots x_m$ e $Y = y_1 \dots y_n$ associamo il grafo G_{XY} seguente:

nodo (i,j) in corrispondenza di x_i e y_j

costi archi orizzontali e verticali = δ

costo arco diagonale verso (i,j) = $\alpha_{xi,\, yj}$

$X = x_1\, x_2\, x_3$

$Y = y_1\, y_2\, y_3\, y_4$

k intero

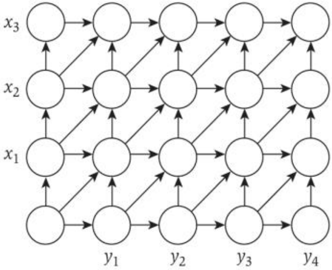


Figure 6.17 A graph-based picture of sequence alignment.

Allineamento di sequenze & Cammini minimi

$X = x_1\, x_2\, x_3$

$Y = y_1\, y_2\, y_3\, y_4$

k intero

$\xrightarrow{f}$

G_{XY}, k intero

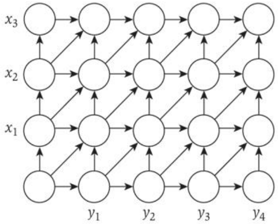


Figure 6.17 A graph-based picture of sequence alignment.

SI, se esiste un allineamento di costo k $\longleftrightarrow$ SI, se esiste un cammino di costo k da (0,0) a (m,n)

NO $\longleftrightarrow$ NO

Allineamento di sequenze ≤ Cammini minimi

Se avessi un algoritmo per risolvere il problema dei **cammini minimi** $\Rightarrow$

avrei un algoritmo per risolvere il problema dello **allineamento di sequenze**. Viceversa, non è detto.

Quindi:

il problema dell'**allineamento di sequenze** ha **difficoltà ≤** del problema dei **cammini minimi**.

Allineamento di sequenze $\leq_m$ **Cammini minimi**

Allineamento di sequenze si riduce mediante funzione (f) a **Cammini minimi**

Riducibilità: un esempio

Esempio $\Sigma = \{0, 1\}$.

$EVEN = \{w \in \Sigma^* \mid w$ `e la rappresentazione binaria di $n \in$ N pari$\}$

$ODD = \{w \in \Sigma^* \mid w$ `e la rappresentazione binaria di $n \in$ N dispari$\}$

Riducibilità: un esempio

Esempio $\Sigma = \{0, 1\}$.

$EVEN = \{w \in \Sigma^* \mid w$ `e la rappresentazione binaria di $n \in$ N pari$\}$

$ODD = \{w \in \Sigma^* \mid w$ `e la rappresentazione binaria di $n \in$ N dispari$\}$

Sia $w \in \Sigma^*$ e sia n il corrispondente decimale di w . E`‘ facile costruire la MdT *INCR*:

$$w \rightarrow \boxed{INCR} \rightarrow w' \quad (= \text{rappresentazione binaria di } n+1)$$

Riducibilità: un esempio

- *EVEN* "non è più difficile" di *ODD*:

se esiste una MdT *R* che decide *ODD*, la MdT *S* decide *EVEN*.

$$S : w \to \boxed{INCR} \to w' \to \boxed{R}$$

Riducibilità: un esempio

- *EVEN* "non è più difficile" di *ODD*:

se esiste una MdT *R* che decide *ODD*, la MdT *S* decide *EVEN*.

$$S : w \rightarrow \boxed{INCR} \rightarrow w' \rightarrow \boxed{R}$$

- Viceversa se *EVEN* *è* indecidibile proviamo così che anche *ODD* lo è:

  se per assurdo esistesse una MdT *R* che decide *ODD*, la MdT *S* deciderebbe *EVEN*.

Riducibilità: definizione informale

- Idea: convertire le istanze di un problema P nelle istanze di un problema P' in modo che un algoritmo per P', se esiste, possa essere utilizzato per progettare un algoritmo per P: P non è più difficile di P'.
- Sia A il linguaggio associato a P, sia B il linguaggio associato a P'. Allora proveremo che:
  B decidibile $\Rightarrow$ A decidibile,
  A indecidibile $\Rightarrow$ B indecidibile.
- **Nota**: nulla è detto sulla decidibilità di A o B ma solo sulla decidibilità di A assumendo di disporre di un algoritmo per decidere di B.

Funzioni calcolabili

Definizione

Una funzione $f : \Sigma^* \to \Sigma^*$ *è calcolabile se esiste una TM M tale che su ogni input w, M si arresta con* $f(w)$, *e solo con* $f(w)$, *sul suo nastro (e la testina sulla prima cella del nastro).*

Funzioni calcolabili

Definizione

Una funzione $f : \Sigma^* \rightarrow \Sigma^*$ *è calcolabile se esiste una TM M tale che su ogni input w, M si arresta con* $f(w)$, *e solo con* $f(w)$, *sul suo nastro (e la testina sulla prima cella del nastro).*

- Nota: questa definizione sottolinea la differenza tra definire una funzione f, cioé definire i valori di f e calcolare tali valori di f.

Funzioni calcolabili

Le seguenti funzioni aritmetiche sono calcolabili (dove $n, m \in \mathbb{N}$):

- $incr(n) = n + 1$
- $dec(n) = \begin{cases} n - 1 & \text{se } n > 0; \\ 0 & \text{se } n = 0 \end{cases}$
- $(m, n) \to m + n;$
- $(m, n) \to m - n;$
- $(m, n) \to m \cdot n$

Esempio di una funzione non calcolabile

Consideriamo A_{TM} e $B = \{ab\}$.

Consideriamo la funzione $f : \Sigma^* \to \Sigma^*$, dove $a, b \in \Sigma$, così definita.

$$f(y) = \begin{cases} ab & \text{se } y = \langle M, w \rangle \in A_{TM}; \\ a & \text{altrimenti} \end{cases}$$

Quindi f è una funzione tale che $f(y) = a$ se y non è della forma $\langle M, w \rangle$, oppure se $y = \langle M, w \rangle$ con $\langle M, w \rangle \notin A_{TM}$.

Invece $f(y) = ab$ se $y = \langle M, w \rangle$ con $\langle M, w \rangle \in A_{TM}$.

Quindi per ogni $y \in \Sigma^*$,

$$y \in A_{TM} \Leftrightarrow f(y) \in \{ab\}$$

Concludere che f non \`e calcolabile.

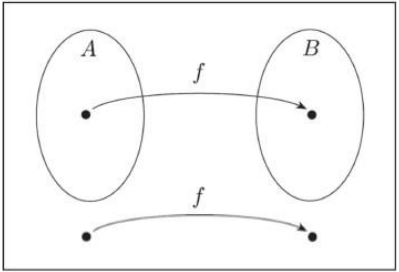
Riducibilità mediante funzione

Definizione

Un linguaggio $A \subseteq \Sigma^*$ *`e riducibile mediante funzione a un linguaggio* $B \subseteq \Sigma^*$*, e scriveremo* $A \leq_m B$*, se esiste una funzione calcolabile* $f : \Sigma^* \rightarrow \Sigma^*$ *tale che* $\forall w \in \Sigma^*$

$$w \in A \Leftrightarrow f(w) \in B$$

La funzione f `e chiamata una riduzione da A a B.



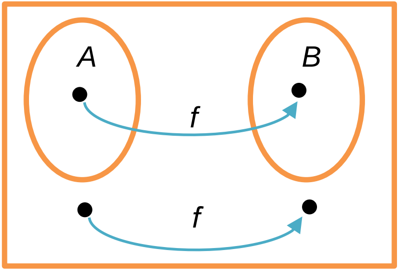
Riducibilità mediante funzione

Definizione

Un linguaggio $A \subseteq \Sigma^*$ \`e riducibile mediante funzione a un linguaggio $B \subseteq \Sigma^*$, e scriveremo $A \leq_m B$, se esiste una funzione calcolabile $f : \Sigma^* \to \Sigma^*$ tale che $\forall w \in \Sigma^*$

$$w \in A \Leftrightarrow f(w) \in B$$

La funzione f \`e chiamata una riduzione da A a B.



Riducibilit`a mediante funzione

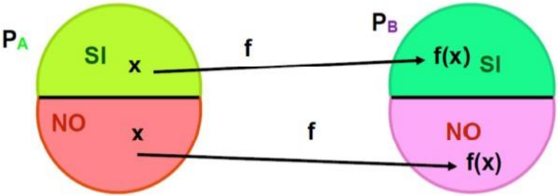


Immagine tratta dalle dispense della Prof.ssa Emanuela Fachini

Una riduzione fornisce un modo per convertire problemi di appartenenza ad *A* in problemi di appartenenza a *B*.

Se un problema *A* è riducibile a *B* e sappiamo risolvere *B* allora sappiamo risolvere *A* cioè *A* “non è piùdifficile” di *B*.

Teoremi

Teorema

$A \leq_m B$ *se e solo se* $\overline{A} \leq_m \overline{B}$.

Teoremi

Teorema
$A \leq_m B$ se e solo se $\overline{A} \leq_m \overline{B}$.

Dimostrazione
Per ipotesi $A \leq_m B$, quindi esiste una riduzione di A a B.

Poiché f è una riduzione, f è calcolabile e inoltre

$$\forall w \in \Sigma^* \quad w \in A \Leftrightarrow f(w) \in B$$

Proviamo che f è anche una riduzione da $\overline{A}$ a $\overline{B}$.

(cont.)

Infatti, poiché f è una riduzione, f è calcolabile e inoltre

$$\forall w \in \Sigma^* \quad w \in A \Leftrightarrow f(w) \in B$$

Quindi

$$\forall w \in \Sigma^* \quad w \notin A \Leftrightarrow f(w) \notin B$$

Cioè

$$\forall w \in \Sigma^* \quad w \in \overline{A} \Leftrightarrow f(w) \in \overline{B}$$

Quindi, per definizione, f è una riduzione da $\overline{A}$ a $\overline{B}$.

Teoremi

Teorema

Se $A \leq_m B$ *e B `e decidibile, allora A `e decidibile.*

Teoremi

Teorema

Se $A \leq_m B$ *e* B *`e decidibile, allora* $A$ *`e decidibile.*

Dimostrazione

Sia M_B un decider per B, f una riduzione da A a B e M_f una MdT che calcola f. Costruiamo un decider N per A: su input w

- simula M_f e calcola $f(w)$
- simula M_B su $f(w)$ e da lo stesso output.

$$N : w \rightarrow \boxed{\boxed{M_f} \rightarrow f(w) \rightarrow \boxed{M_B}}$$

N si ferma su ogni input.
N riconosce A. Infatti
$w \in L(N) \Leftrightarrow f(w) \in L(M_B) \Leftrightarrow f(w) \in B \Leftrightarrow w \in A$.

Quindi N decide A.

Teoremi

Teorema

Se $A \leq_m B$ e B `e Turing riconoscibile, allora $A$ `e Turing riconoscibile.

Dimostrazione

Sia M_B una MdT che riconosce B, f una riduzione da A a B e M_f una MdT che calcola f. Consideriamo la MdT M_A: su input w

- simula M_f e calcola $f(w)$
- simula M_B su $f(w)$
- se M_B accetta $f(w)$, accetta; se M_B rifiuta $f(w)$, rifiuta.

$$M_A : w \to \boxed{\boxed{M_f} \to f(w) \to \boxed{M_B}}$$

Ovviamente se M_B cicla, anche M_A cicla. Analogamente a prima, M_A riconosce A.

Teoremi

Corollario

Se $A \leq_m B$ e A *è* indecidibile, allora B *è* indecidibile.

(se B fosse decidibile lo sarebbe anche A in virtù del teorema precedente)

Teoremi

Corollario

Se $A \leq_m B$ e A è indecidibile, allora B è indecidibile.

(se B fosse decidibile lo sarebbe anche A in virtù del teorema precedente)

Corollario

Se $A \leq_m B$ e A non \`e Turing riconoscibile, allora $B$ non \`e Turing riconoscibile.

(se B fosse Turing riconoscibile lo sarebbe anche A in virtu\` del teorema precedente)