Progetto e analisi di algoritmi

Roberto Cordone

DTI - Università degli Studi di Milano

Polo Didattico e di Ricerca di Crema

Tel. 0373 / 898**089**

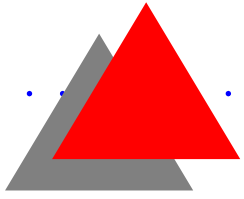
E-mail: **cordone@dti.unimi.it**

Ricevimento: **su appuntamento**

Web page: **http://www.dti.unimi.it/~cordone**

Lezioni: **Martedì dalle 14.00 alle 16.00**

Giovedì dalle 11.00 alle 13.00



Algoritmi e macchine

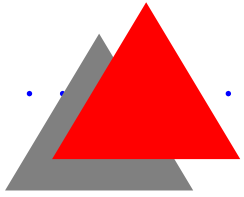
Una macchina interamente specificata definisce un programma, cioè una sequenza di operazioni

1. elementari (di durata limitata)
2. determinate meccanicamente dall'istanza

Un algoritmo è un programma le cui operazioni sono in numero limitato, cioè che termina

$$\mathcal{A} \subset \mathcal{M}$$

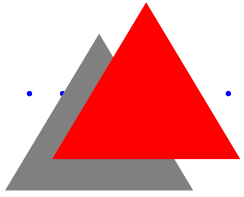
I programmi (macchine) che non terminano non sono algoritmi



Macchine e modelli

Un modello computazionale è l'insieme di tutte le macchine che hanno una data struttura

- la macchina di Turing che risolve il problema della parità è una ben precisa 7-upla $(Q, q_0, A, \Sigma, -, \Gamma, \delta)$ con Q insieme finito, $q_0 \in Q$, $A \subseteq Q$, ecc...
- la Macchina di Turing è l'insieme di tutte le 7-uple $(Q, q_0, A, \Sigma, -, \Gamma, \delta)$ con...



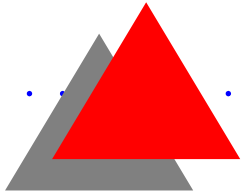
Relazioni fra modelli computazionali (1)

Ogni modello computazionale risolve una classe di problemi

Un modello computazionale è più potente di un altro quando risolve una classe di problemi più ampia, cioè per ogni problema risolto da un esemplare del secondo modello esiste un esemplare del primo modello che risolve lo stesso problema

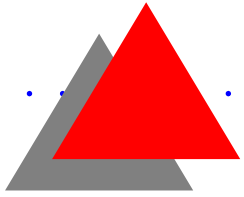
Una macchina più potente può anche impiegare più tempo!

- il PDA è più potente del FSA
- la PN è più potente del FSA
- la TM è più potente del PDA e della PN



Relazioni fra modelli computazionali (2)

- il PDA con due pile è equivalente alla TM
- la TM con fermata è equivalente alla TM
- la TM con più nastri è equivalente alla TM
- la TM e la macchina RAM sono equivalenti, cioè sono vicendevolmente una più potente dell'altra
- la macchina PRAM (*Parallel Random Access Memory*) è un insieme illimitato di macchine RAM comunicanti in tempo istantaneo è più potente della macchina RAM



Equivalenza fra modelli

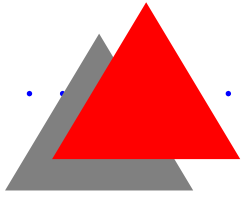
Molti modelli computazionali risolvono gli stessi problemi

- Macchina di Turing
- Macchina RAM
- λ-calcolo
- Macchina di Markov

Ogni problema risolto da uno di questi modelli computazionali può essere risolto da ognuno degli altri

Allora non è essenziale sapere quale macchina si usi

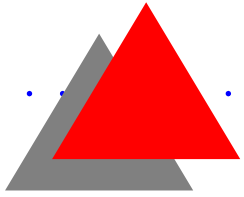
Tesi di Church/Turing: se un problema è risolubile in pratica, allora lo è con una di queste macchine



Codifica di una TM

Anche una TM si può codificare in una stringa di simboli. Si può

1. definire una macchina di Turing universale (UTM), che riceve la codifica di una macchina TM_2 e di un'istanza I e fornisce la risposta che TM_2 darebbe su I
2. risolvere problemi su una macchina di Turing TM_2 fornendo la sua codifica a un'opportuna macchina TM_1 (per es., sapere se TM_2 si ferma)
3. associare ogni macchina di Turing a un numero naturale



La macchina di Turing universale

La codifica di una TM comincia col simbolo B e termina col simbolo E

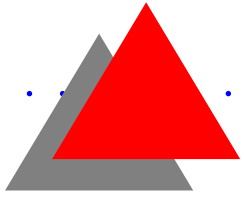
In mezzo ci sono le quintuple della funzione di transizione:

1. stato corrente, descritto in unario col simbolo S ($q_0 = S$, $q_1 = SS$, $\ldots$)
2. simbolo in ingresso
3. stato futuro, ancora in unario
4. simbolo in uscita
5. direzione di spostamento (0 per sinistra, 1 per destra)

All'inizio il nastro contiene

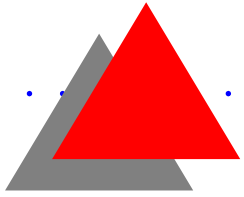
- la codifica della TM da simulare
- il simbolo T seguito dall'istanza su cui operare

Il simbolo T precede la posizione corrente della testina



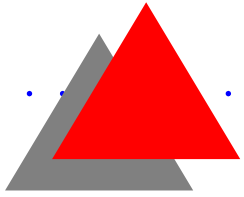
La macchina di Turing universale

1. Inizializzazione: scrive S a sinistra della prima posizione
   (dove si conserva lo stato corrente, che ora è $q_0 = S$)
2. Lettura dell'istanza:
   (a) legge il simbolo che segue la T, lo salva nello stato e torna a B
   (b) va in fondo allo stato corrente, scrive il simbolo letto e torna a B
3. Ricerca della regola:
   (a) confronta stato corrente e stato iniziale della prima regola non barrata, sostituendo S con $\bar{S}$
   (b) Se sono uguali, passa al punto 4
   altrimenti, barra stato iniziale e finale della regola, ripristina lo stato corrente, torna a B e passa al punto $3a$



La macchina di Turing universale

4. Esecuzione della regola:
   (a) copia lo stato finale della regola nello stato corrente
   (b) legge il simbolo in uscita e la direzione di spostamento, copia il simbolo sul nastro nella posizione che segue T e sposta T
   (c) ripristina tutte le regole cancellate, torna a B e passa al punto $2a$
5. Terminazione: riconosce se lo stato corrente è accettante o no



Risolubilità

Abbiamo uno strumento per risolvere problemi

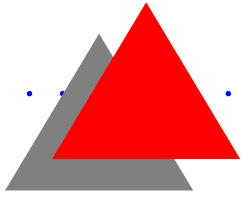
Tesi di Church/Turing: se un problema è risolubile in pratica,
allora lo è con una di queste macchine

Ammesso (e non concesso) che sia vero, risolve tutti i problemi?
Se non tutti, quanti problemi risolve?

- Quanti sono i problemi?
- Quante sono le macchine?

Infiniti entrambi. Siamo a posto?

Meno di quanto sembri. . .



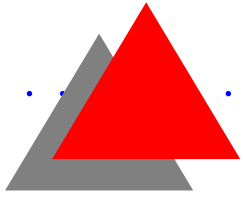
Insiemi finiti, infiniti, numerabili

- insieme infinito è un insieme in corrispondenza biunivoca con una sua parte propria (es.: numeri naturali e pari)
- insieme finito: gli altri insiemi

Un insieme numerabile è un insieme in corrispondenza biunivoca con una parte di $\mathbb{N}$

- i numeri pari
- le potenze di 2
- i numeri interi
- i numeri razionali

Le macchine di Turing formano un insieme numerabile



Quanti sono i problemi?

Problemi (di decisione) e sottoinsiemi di $\{0,1\}^*$
sono in corrispondenza biunivoca

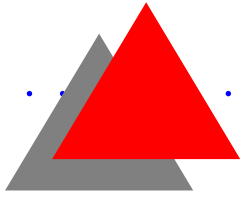
Sottoinsiemi di $\{0,1\}^*$ e sottoinsiemi di $\mathbb{N}$
sono in corrispondenza biunivoca

Problemi (di decisione) e sottoinsiemi di $\mathbb{N}$
sono in corrispondenza biunivoca

Quanti sono i sottoinsiemi di $\mathbb{N}$?

I sottoinsiemi di $\mathbb{N}$ non sono numerabili

$\Rightarrow$ i problemi di decisione non sono numerabili

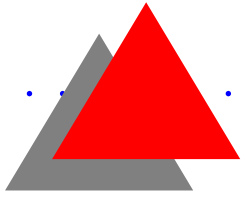


Innumerabilità dei problemi

Vediamolo con la diagonalizzazione di Cantor

1. supponiamo che i sottoinsiemi di $\mathbb{N}$ siano numerabili
2. sulle colonne di una tabella elenchiamo i numeri naturali
3. sulle righe elenchiamo i sottoinsiemi: 1 indica che il sottoinsieme contiene il numero, 0 che non lo contiene

0	0	1	1	…
1	1	1	0	…
1	0	0	1	…
1	0	1	0	…
…	…	…	…	…

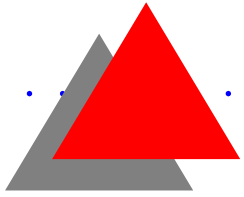


Innumerabilità dei problemi

4. la diagonale della tabella individua un sottoinsieme

0	0	1	1	…
1	1	1	0	…
1	0	0	1	…
1	0	1	0	…
…	…	…	…	…

5. consideriamo il suo complemento



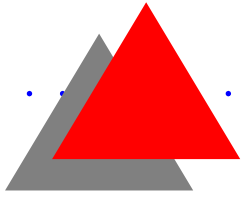
Innumerabilità dei problemi

4. la diagonale della tabella individua un sottoinsieme

0	0	1	1	…
1	1	1	0	…
1	0	0	1	…
1	0	1	0	…
…	…	…	…	…

5. consideriamo il suo complemento

1	0	1	1	…



Innumerabilità dei problemi

6. Il complemento della diagonale non è una riga della tabella!
   - non è la prima riga: cambia il primo simbolo...
   - non è la seconda riga: cambia il secondo simbolo...

1	0	1	1	...
0	0	1	1	...
1	1	1	0	...
1	0	0	1	...
1	0	1	0	...
...	...	...	...	...

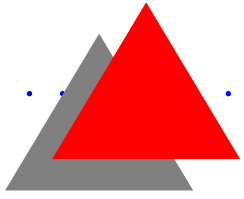
7. Quindi i sottoinsiemi di $\mathbb{N}$ non si possono tabellare

Problemi e macchine

Ci sono molti più problemi che macchine

L'unica speranza è che i problemi non risolubili siano problemi non interessanti

Purtroppo, non è così: molti problemi interessanti su macchine di Turing non possono essere risolti con macchine di Turing

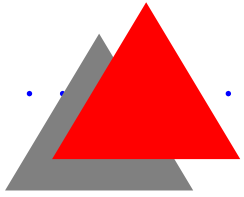


Problemi indecidibili

- Terminazione: Data una macchina M e un'istanza I, $M(I)$ termina in tempo finito?
- Verifica: Data una macchina M e una specifica f (relazione fra ingresso I e uscita S), M è coerente con f?
- Equivalenza: Date due macchine $M1$ e $M2$, sono equivalenti ($M1(I) = M2(I)$ per ogni I)?

Non significa che è impossibile decidere se un programma termina o è corretto, oppure se due programmi sono equivalenti!

Significa che in alcuni casi non avremo risposta



Problema di Collatz (1)

Dato il seguente programma...

Collatz(n)

$t := 1$;

While $n \neq 1$ **do**

If *Odd*(n)

then $n := 3n + 1$;

else $n := n/2$;

$t := t + 1$;

Return t;

...risolvere il seguente problema

Il programma *Collatz* termina sull'istanza n?

Istanza: $n \in \mathbb{N}$

Soluzione:

- *Vero* se esiste t^* t.c.

$$t \leq t^* \quad \forall n \in \mathbb{N}$$

- *Falso* altrimenti

Problema di Collatz (2)

- Nessuno ha mai trovato un intero n che lo faccia funzionare per sempre ($n \leq 3 \cdot 2^{53}$)
- Nessuno ha mai dimostrato che non esista tale intero

Non si sa se è decidibile o indecidibile

Servirebbe un algoritmo che, per ogni n, risponda *Vero* o *Falso*

Perché non usiamo le istruzioni del programma stesso?

Perché non costituiscono una sequenza di operazioni sicuramente finita, dunque non costituiscono un algoritmo

`http://mathworld.wolfram.com/CollatzProblem.html`