Le parole dell'informatica: algoritmo e decidibilità

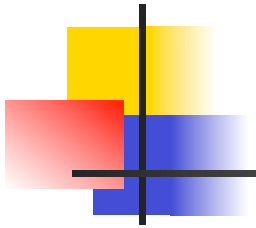
Angelo Montanari

Dipartimento di Matematica e Informatica

Università degli Studi di Udine

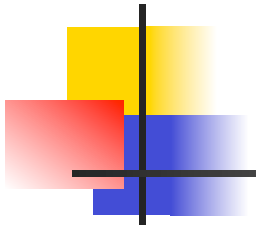
Ciclo di seminari su un Vocabolario Filosofico dell'Informatica

Udine, 14 gennaio, 2016



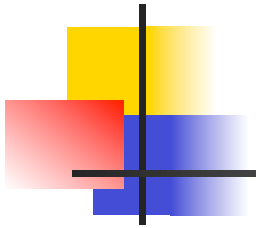
Le parole dell'informatica

- **algoritmi** (**teoria degli**)
- **decidibilità/indecidibilità**
- modelli di calcolo
- complessità (teoria della)
- trattabilità



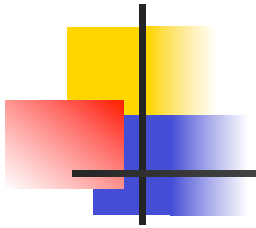
Algoritmi (teoria degli)

- Il concetto di **algoritmo** non è di per sé un contributo originale dell'informatica. Gli algoritmi sono noti da sempre non solo in matematica (l'algoritmo per il calcolo del massimo comun divisore di due numeri interi positivi fu inventato da Euclide nel quarto secolo a.C.), ma anche in numerose altre discipline (architettura)
- Ciò che è nuovo è il ruolo centrale che essi assumono nell'informatica (**teoria degli algoritmi** o algoritmica)



La nozione di algoritmo

- Un **algoritmo** è una descrizione finita e non ambigua di una sequenza di passi che consente di risolvere un determinato problema
- In generale un problema può essere risolto da vari algoritmi (esempi: il problema della **ricerca del massimo** di un insieme di numeri naturali distinti e il problema dell'ordinamento)
- Due possibili algoritmi per la ricerca del massimo di un insieme di numeri naturali distinti

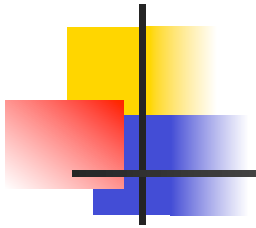


Proprietà degli algoritmi

Caratteristiche essenziali di un algoritmo:

Uniformità: un algoritmo si applica a tutte le istanze di un dato problema, potenzialmente infinite, e non ad una singola istanza, e la sua formulazione non dipende dalla singola istanza, ma è la stessa per ogni istanza

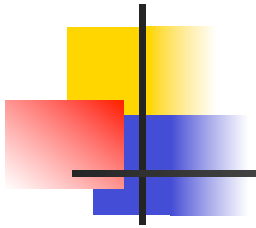
Effettività: la raggiungibilità della soluzione voluta in un tempo finito o, equivalentemente, in un numero finito di passi



Algoritmi e programmi

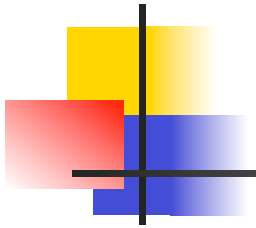
Un **programma** è una formulazione di un algoritmo in uno specifico linguaggio (linguaggio di programmazione) che può essere letto e interpretato da un computer

Osservazione: la formulazione di un algoritmo dipende dal soggetto (macchina) che lo deve eseguire, ossia un algoritmo deve essere formulato utilizzando i comandi (istruzioni) di un **linguaggio** che il soggetto (macchina) è in grado di comprendere ed eseguire



Gerarchia di linguaggi

Tale dipendenza è all'origine della **gerarchia di linguaggi di programmazione**, che spaziano dai linguaggi di programmazione di alto livello (C, JAVA, Prolog, Haskell) che utilizzano istruzioni di controllo complesse, ai linguaggi macchina, che utilizzano istruzioni direttamente eseguibili dalla macchina, e di algoritmi/programmi (**compilatori**/**interpreti**) che consentono di passare dagli uni agli altri

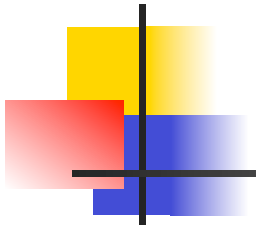


Decidibilità e indecidibilità

La risposta alla domanda (ingenua): “esiste un algoritmo per ogni problema?” è negativa

L'esistenza di problemi per i quali non esistono algoritmi si può mostrare facendo vedere che esistono **funzioni non computabili** (funzioni per le quali non esiste un algoritmo in grado di computarle).

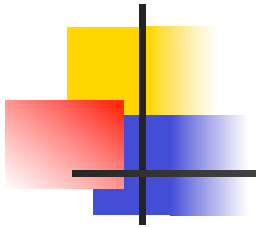
Esistono, infatti, più funzioni che nomi per designarle e algoritmi per computarle. L'argomento può essere formalizzato usando il metodo della diagonale di Cantor



Una semplice dimostrazione

Dimostrazione per assurdo. Si assuma che per ogni funzione computabile esista un algoritmo/programma che la computa. Ogni algoritmo altro non è che una stringa di lunghezza finita su un qualche alfabeto finito ed è, quindi, finitamente specificabile. Da ciò segue che l'insieme di tutti gli algoritmi è numerabile.

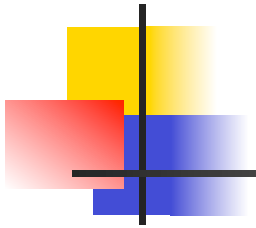
Si considerino ora tutte le funzioni che mappano gli interi positivi in 0 e 1. Si supponga, per assurdo, che l'insieme di tali funzioni sia numerabile e che i suoi elementi (le funzioni) siano stati messi in corrispondenza con l'insieme dei numeri interi positivi. Sia f_i la funzione che corrisponde all'intero i-esimo. E' immediato vedere come la funzione f che assegna il valore 0 all'i-esimo intero se f_i assegna ad esso valore 1, e il valore 1 altrimenti, non corrisponda ad alcun intero, ossia come f sia diversa da f_k per ogni intero k (contraddizione).



I problemi “algoritmici”

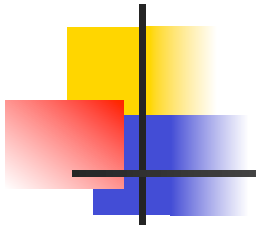
- Una domanda meno ingenua è: “che tipo di problemi intendono risolvere gli algoritmi (ossia quali sono i problemi algoritmici) e quali fra questi problemi sono effettivamente in grado di risolvere?”

- Esistono varie categorie di problemi algoritmici (numerici, di ordinamento, di ricerca su grafi, di ottimizzazione)



I problemi di decisione

- Un ruolo particolare è quello dei cosiddetti **problemi decisionali**, il cui algoritmo deve decidere se una data proprietà è vera o falsa e fornisce, quindi, una risposta di tipo sì/no
- Esempio di problema decisionale: stabilire se un dato numero intero positivo n sia o meno primo
- Un problema algoritmico privo di soluzione è detto problema non computabile; nel caso dei problemi decisionali, un problema privo di soluzione è detto **indecidibile**



Problemi indecidibili

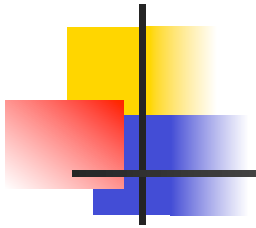
Esistono importanti problemi decisionali **indecidibili** nell'ambito della (logica) matematica

Esempio 1 (Goedel). Il **teorema di incompletezza di Goedel** dimostra che all'interno di ogni sistema formale contenente l'aritmetica esistono proposizioni che il sistema non riesce a decidere (non riesce, cioè, a dare una dimostrazione né di esse né della loro negazione)

Il programma hilbertiano

Osservazione (importante).

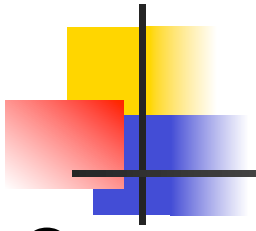
- Fra le proposizioni che un sistema formale contenente l'aritmetica non riesce a dimostrare c'è anche quella che, in termini numerici, esprime la non-contraddittorietà del sistema (**fallimento del programma hilbertiano** che voleva dimostrare la non-contraddittorietà dell'intera teoria formale dei numeri sfruttando la sola aritmetica finitistica)



Il problema della decisione

Esempio 2 (Turing-Church). L'Entscheidungsproblem (**problema della decisione**), posto sempre da Hilbert, era il problema di trovare un algoritmo che, data una formula della logica del primo ordine, fosse in grado di determinare se essa era o meno valida

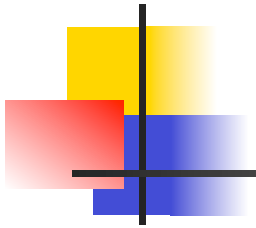
L'indecidibilità dell'Entscheidungsproblem fu dimostrata da Turing facendo ricorso alla cosiddetta macchina di Turing e, contemporaneamente, da Church attraverso il lambda calcolo



Alcune ricadute sorprendenti

Osservazione.

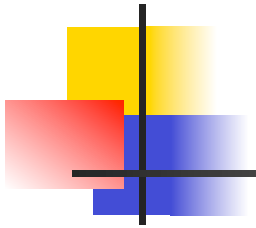
I risultati “negativi” di Turing e Church hanno avuto conseguenze importanti e imprevedibili: il modello proposto da Church è alla base del paradigma della **programmazione funzionale**, mentre il modello di Turing si riflette nel paradigma della **programmazione imperativa**



Il problema della terminazione

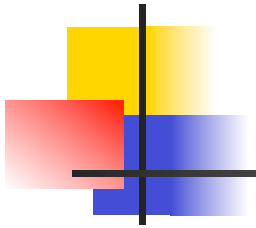
Esistono importanti problemi decisionali indecidibili in **informatica**

Esempio 3 (Turing). Il **problema della terminazione**, ossia il problema di stabilire, ricevuti in ingresso un qualsiasi programma e un suo possibile input, se l'esecuzione di tale programma sullo specifico input termina o meno, è indecidibile.



Teorema di Rice

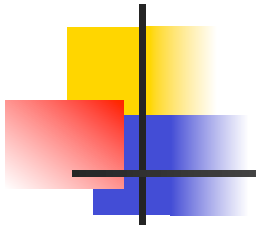
Esempio 4 (Rice). Il **teorema di Rice** mostra che non esiste un algoritmo in grado di stabilire se un dato programma gode di una qualsiasi proprietà non banale (una proprietà non banale dei programmi è una proprietà che è soddisfatta da alcuni programmi, ma non da tutti, e che non dipende dallo specifico linguaggio di programmazione utilizzato, ma dagli algoritmi in sè).



Il problema del ricoprimento

Fra i problemi indecidibili ve ne sono alcuni molto semplici. E' questo il caso di molti problemi di ricoprimento (**tiling problem**)

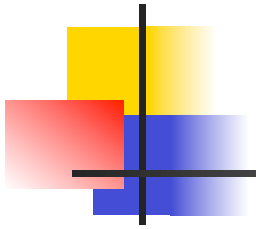
- Un problema di ricoprimento richiede di stabilire, ricevuto in ingresso un insieme finito di tipi diversi di piastrelle colorate (quadrati di dimensioni unitarie divisi in quattro parti dalle diagonali, ognuna delle quali colorata in un certo modo), se sia possibile o meno coprire una determinata superficie del piano con piastrelle dei tipi dati, rispettando alcune semplici condizioni sui colori delle piastrelle adiacenti (ad esempio, stesso colore)



La tecnica della riduzione

E' molto comune mostrare l'indecidibilità di un problema riducendo ad esso un altro problema di cui già si conosce l'indecidibilità (**tecnica di riduzione**)

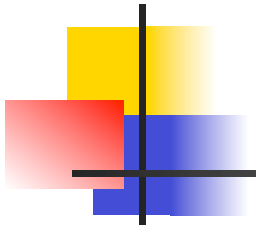
Tale tecnica sfrutta molto spesso le varianti indecidibili del problema del ricoprimento



Equivalenza di problemi

E' possibile definire una **relazione di equivalenza** sull'insieme dei problemi indecidibili in termini di riducibilità: due problemi indecidibili si dicono equivalenti se l'algoritmo immaginario in grado di decidere l'uno (oracolo) potrebbe essere utilizzato per risolvere l'altro, e viceversa

Esempio. Il problema della terminazione e il problema del ricoprimento sono equivalenti



Gerarchia di problemi

- Non tutti i problemi indecidibili sono equivalenti: ve ne sono alcuni più indecidibili di altri. Vi sono cioè situazioni in cui la riduzione precedente vale in una direzione, ma non nell'altra
- E' possibile definire una gerarchia infinita di problemi sempre più indecidibili (**livelli di indecidibilità**)