Automi e Linguaggi Formali

Parte 13 – Algoritmi per macchine di Turing

Davide Bresolin
Ultimo aggiornamento: 4 maggio 2022

Università degli Studi di Padova

Sommario

Contesto Storico: David Hilbert

David Hilbert, discorso al Secondo Congresso Internazionale di Matematica, Parigi, 1900

- Definisce 23 problemi matematici come sfida per il nuovo secolo.
- Decimo problema: creare un algoritmo per determinare se un polinomio ha una radice intera
- Il presupposto era che l'algoritmo dovesse esistere, e bastava trovarlo
- Ora sappiamo che questo problema è non risolvibile algoritmicamente

Che cos'è un Algoritmo?

- La nozione intuitiva di algoritmo esiste da migliaia di anni.
- La definizione formale di algoritmo è stata data per la prima volta nel XX secolo
- Senza una definizione formale, è quasi impossibile provare che un algoritmo non può esistere.

Contesto Storico: Tesi di Church-Turing

- 1936 Church pubblica un formalismo chiamato λ-calcolo per definire algoritmi.
- 1936 Turing pubblica le specifiche per una "macchina astratta" per definire algoritmi.
- 1952 Kleene mostra che i due modelli sono equivalenti
- 1970 Matiyasevich dimostra che l'algoritmo per stabilire se un polinomio ha radici intere non esiste

- Il decimo teorema di Hilbert con la nostra terminologia:

$$D = \{p \mid p \text{ è un polinomio avente radice intera}\}$$

- Il problema diventa "D è un insieme decidibile?"
- Possiamo mostrare che D è Turing-riconoscibile
- Partiamo da un problema più semplice:

$$D_1 = \{p \mid p \text{ è un polinomio su } x \text{ avente radice intera}\}$$

Come descrivere una Turing Machine

- Descrizione formale
  - Dichiara esplicitamente tutto quanto
  - Estremamente dettagliata
  - Da evitare a tutti i costi !!!
- Descrizione implementativa
  - Descrive a parole il movimento della testina e la scrittura sul nastro
  - Nessun dettaglio sugli stati
- Descrizione di alto livello
  - Descrizione a parole dell'algoritmo
  - Nessun dettaglio implementativo
  - Da utilizzare sempre, se non indicato altrimenti

Notazione formale per macchine di Turing

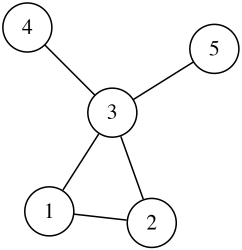
- L'input è sempre una stringa.
- Se l'input è un oggetto, deve essere rappresentato come una stringa.
  - Polinomi, grammatiche, automi, ecc.
  - L'input può essere una combinazione di diversi tipi di oggetti.
- Un oggetto O codificato come stringa è $\langle O \rangle$.
- Una sequenza di oggetti $O_1, O_2, \ldots, O_k$ è codificata come $\langle O_1, O_2, \ldots, O_k \rangle$.
- L'algoritmo viene descritto con un testo, indentato e con struttura a blocchi.
- La prima riga dell'algoritmo descrive l'input macchina.

Sommario

I grafi e le loro applicazioni

- I grafi sono strutture dati che vengono usate estensivamente in informatica
- Ci sono migliaia di problemi computazionali che sono importanti per le applicazioni e che si possono modellare con i grafi.
- In questa lezione vedremo che cos'è un grafo, e studieremo alcuni problemi sui grafi che sono interessanti per la loro classe di complessità.

Un grafo è definito da un'insieme di nodi (o vertici) e da un'insieme di archi che collegano i nodi.



Definition (Grafo non orientato)

Un grafo non orientato (detto anche indiretto) G è una coppia (V, E) dove:

- $V = \{v_1, v_2, \ldots, v_n\}$ è un insieme finito e non vuoto di vertici;
- $E \subseteq \{\{u, v\} \mid u, v \in V\}$ è un insieme di coppie non ordinate, ognuna delle quali corrisponde ad un arco non orientato del grafo.

Un problema di connessione

- Un grafo è connesso se ogni nodo può essere raggiunto da ogni altro nodo tramite gli archi del grafo

Problema

Il linguaggio $A = \{\langle G \rangle \mid G \text{ è un grafo connesso}\}$ è decidibile?

Una TM che decide A

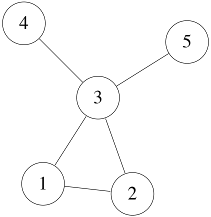
Descrizione di alto livello:

$M =$ "Su input $\langle G \rangle$, la codifica di un grafo G:

1. Seleziona il primo nodo di G e lo marca.
2. Ripeti la fase seguente fino a quando non vengono marcati nuovi nodi:
3. per ogni nodo in G, marcalo se è connesso con un arco ad un nodo già marcato.
4. Esamina tutti i nodi di G: se sono tutti marcati, accetta, altrimenti rifiuta."

Codifica del grafo

- Codifica di G: lista dei nodi + lista degli archi

$G =$ 

$\langle G \rangle =$ `(1,2,3,4,5)((1,2),(1,3),(2,3),(3,4),(3,5))`

- M verifica che l'input sia sia una codifica di un grafo:
  - Se l'input non è nella forma corretta, rifiuta
  - Se l'input codifica un grafo, prosegue con la fase 1