Automi e Linguaggi Formali

Dai problemi indecidibili a quelli intrattabili, classi P e NP

Lamberto Ballan
lamberto.ballan@unipd.it

Padova, 30 Maggio 2017

Università degli Studi di Padova

Esercizi

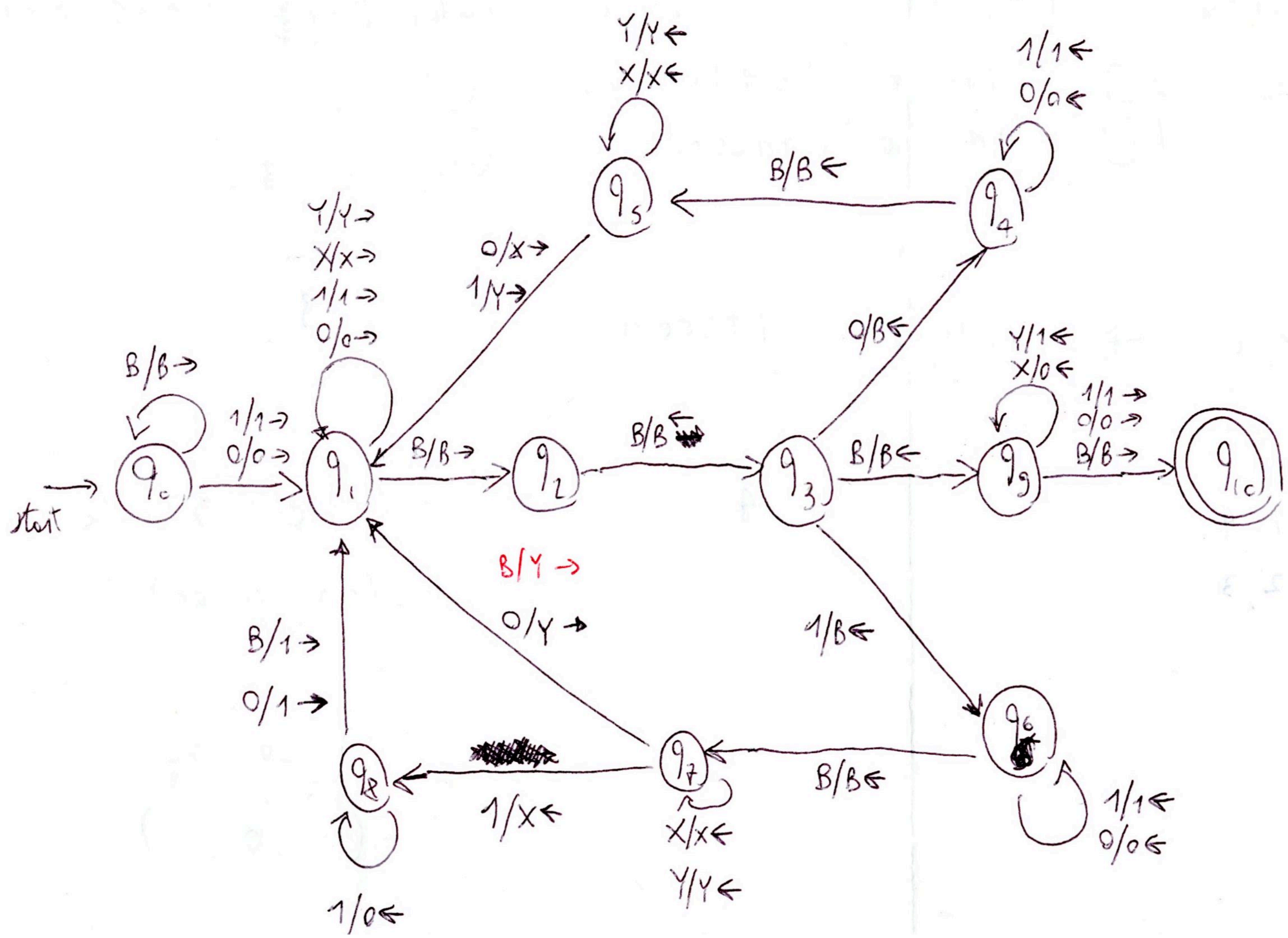
- Dire quali tra le seguenti istanze di PCP hanno soluzione:
  - *A*=(01, 001, 10); *B*=(011, 10, 00)
  - *A*=(01, 001, 10); *B*=(011, 01, 00)
  - *A*=(ab, a, bc, c); *B*=(bc, ab, ca, a)

Soluzione: La sequenza (1) non ha soluzione:1,3 sono le prime due scelte obbligate, successivamente non si può proseguire; la (2) ha come soluzione la sequenza di indici 1,3,2; nella (3) dobbiamo inizialmente scegliere 2,3 (ed eventualmente ripetere ancora sequenze di indici 2,3, ad es. 2,3,2,3); poi potremmo scegliere 1,4 ma questo ci porta ad un loop perché una delle due stringhe "avanza" di una lettera 'a' e l'altra stringa non potrà mai raggiungerla.

Esercizi

- Definire una TM che calcola la somma di due numeri binari *x* e *y* (l'input è una stringa binaria in cui si riporta la sequenza dei due numeri intervallati da un blank, cioè '*x y*' per x+y)

Soluzione vista a lezione: in rosso ho aggiunto la correzione necessaria per gestire la transizione mancante, come evidenziato dall'esempio 11 + 1001



Problemi intrattabili

- Abbiamo detto che la distinzione tra problemi *decidibili* (c'è un algoritmo) e *indecidibili* è più importante della distinzione tra RE (quelli per cui c'è una TM) e non RE (non hanno alcuna TM)
- Ci occuperemo adesso dei soli problemi *decidibili*, cioè ricorsivi
    - tra di loro alcuni sono detti *trattabili*; sono cioè quelli per cui si può provare che sono risolvibili in tempo polinomiale
    - tutti gli altri sono detti *intrattabili*

Problemi intrattabili

- Alcune “note introduttive” ed alcune distinzioni tra teoria dell’indecidibilità e teoria dell’intrattabilità:
    - distinguiamo tra tempo *polinomiale* (cioè $O(n^k)$, per un intero k) ed *esponenziale* (cioè $O(2^{cn})$, per una costante c>0): per esponenziale intendiamo qualunque tempo di esecuzione più grande di qualunque polinomio
    - poiché il tema è se i problemi possono essere risolti in tempo polinomiale, dobbiamo cambiare il concetto di riduzione (non basta trasformare istanze di P_1 in istanze di P_2)
    - i risultati sui problemi intrattabili che daremo d’ora in poi si fondano tutti su di un’ipotesi non dimostrata, ovvero $P \neq NP$

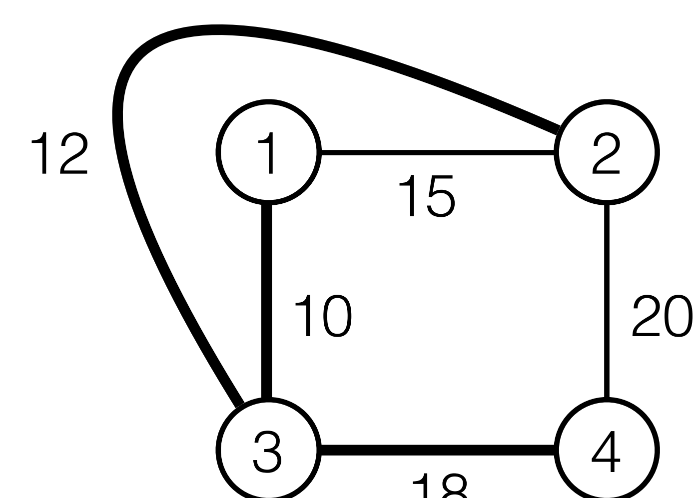
La classe P

- Una TM M ha complessità in tempo $T(n)$ se, dato una stringa in input w di lunghezza n, M si ferma dopo al massimo $T(n)$ passi
  - es. $T(n) = 5n^2 + 3n$, o $T(n) = 4n + 3n^2$
- Un linguaggio L è nella **classe *P*** se esiste un polinomio $T(n)$ tale che $L=L(M)$ per una TM *deterministica* M con complessità in tempo pari a $T(n)$
- Se la complessità in tempo non è polinomiale, si dice che è esponenziale, anche se $T(n)$ non è un esponenziale
  - es. $T(n) = n^{\log_2 n}$; la funzione cresce più velocemente di qualunque polinomio in n, ma più lentamente di qualunque esponenziale 2^{cn}

Un esempio di problema *P*

- Problema: trovare un albero di copertura di peso minimo in un grafo
  - grafo: composto da *nodi* ed *archi*, tra alcune coppie di nodi, con *peso* (intero)
  - albero di copertura: è un sottoinsieme degli archi che connetta tutti i nodi senza formare cicli
  - un albero di copertura si dice di *peso minimo* se il peso totale dei suoi lati è il min tra tutti gli alberi di copertura

Esempio:



Algoritmo di Kruskal

- Una soluzione *greedy* per il problema dell’albero di copertura minima è l’algoritmo di Kruskal:
  - per ciascun nodo si conserva la componente connessa (cioè i nodi raggiungibili da quel nodo con gli archi selezionati finora)
  - all’inizio: nessun arco, quindi ogni nodo è una componente connessa da solo
  - ad ogni passo si considera un altro arco di peso minimo: se unisce due nodi in componenti separate, lo scelgo e unisco le componenti, altrimenti non lo scelgo (si creerebbe un ciclo)
  - si continua a esaminare nuovi archi finché tutti gli archi sono stati esaminati, o il numero degli archi scelti è uguale al numero dei nodi meno 1 (tutti connessi)

La classe NP

- Un linguaggio L è nella **classe *NP*** se esiste una TM *non deterministica M* tale che $L=L(M)$ e, dato l'input w di lunghezza n, al massimo fa $T(n)$ mosse con $T(n)$ polinomiale
- Dato che ogni TM deterministica è una TM non-deterministica senza possibilità di scelta tra mosse, allora avremo che $P \subseteq NP$
  - una NTM polinomiale ha la capacità di congetturare un numero esponenziale di soluzioni e verificarle "in parallelo"
  - sembrerebbe perciò impossibile che $P=NP$; tuttavia nessuno è ancora riuscito a dimostrarlo!

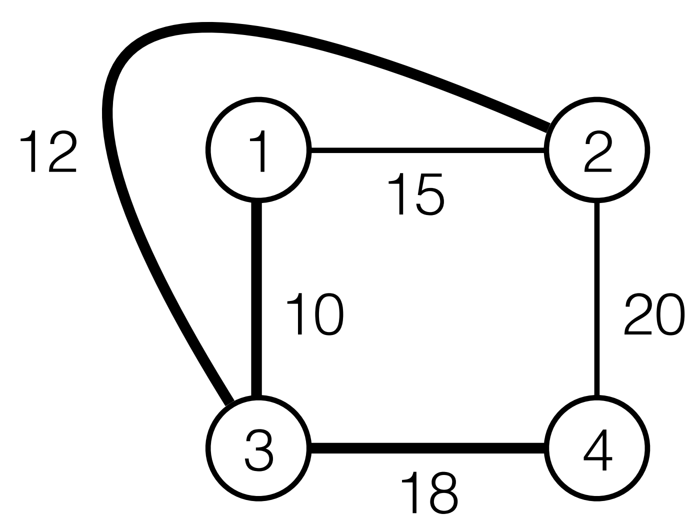
Algoritmo non-deterministico e T(n) polinomiale

- Cerchiamo di chiarire la portata della classe *NP* considerando un esempio che sembra essere in *NP* ma non in *P*
  - vedremo il problema del commesso viaggiatore (TSP) il cui input è uguale al problema dell'albero di copertura a peso min
- Prima però chiariamo come opera un algoritmo non-deterministico con tempo polinomiale:
  - fase 1 (non-deterministica): genero una stringa
  - fase 2: controllo se la stringa appartiene al linguaggio (algoritmo deterministico con tempo polinomiale)

Un esempio di problema *NP* (TSP)

- Problema del commesso viaggiatore (TSP): dato un grafo analogo a quello definito nel problema della ricerca di un albero di copertura minimo, dire se il grafo ha un *circuito hamiltoniano* di peso totale non superiore a *W*
  - circuito hamiltoniano: insieme di archi che connettono i nodi in un unico ciclo in cui ogni nodo appare una sola volta
  - in un grafo di *m* nodi, il numero di cicli distinti cresce come il fattoriale di *m*, che cresce più di 2^{cm}

Esempio:



L'esempio in figura ha un solo circuito hamiltoniano (1,2,4,3,1). Il suo peso totale è quindi 15+20+18+10=63.

Se W≥63 la risposta è "si", altrimenti è "no".

Un esempio di problema *NP* (TSP)

- qualsiasi algoritmo di soluzione del TSP sembra dover esaminare tutti i cicli (o un numero esponenziale) e calcolarne il peso totale
- con un computer non-deterministico potremmo scegliere in ogni ramo una permutazione dei nodi e calcolare il peso totale del ciclo corrispondente
- perciò, se la stringa in input fosse di lunghezza n, nessuna diramazione richiederebbe più di $O(n)$ passi
- abbiamo visto che su una NTM multinastro possiamo scegliere una permutazione in $O(n^2)$ passi; conseguentemente una NTM a nastro singolo può risolvere il TSP in un tempo massimo $O(n^4)$
- la conclusione è quindi che il problema del TSP è in *NP*

Riduzioni polinomiali

- Stesso concetto delle riduzioni già viste, solo con l'aggiunta del vincolo che deve essere fatta in tempo polinomiale
- Riduzione: da una istanza (stringa accettata) di P_1 ad una istanza di P_2
- Se P_1 è riducibile a P_2:
  - se P_2 è in P, allora lo è anche P_1
  - se P_1 non è in P, neanche P_2 lo è
- Intuizione: P_2 è difficile almeno quanto P_1

Problemi NP-completi

- Un linguaggio *L* é *NP*-completo se:
  - *L* è in *NP*
  - per ogni altro linguaggio *L’* in *NP*, esiste una riduzione polinomiale di *L’* a *L*

- Esempio: il problema del commesso viaggiatore (TSP)
  - nota: i problemi *NP-completi* sono i più difficili tra quelli in *NP*

Problemi NP-completi

- Teorema (10.4): se P_1 è *NP-completo*, P_2 è in *NP* e posso ridurre polinomialmente P_1 a P_2, allora P_2 è *NP-completo*
  - si dimostra sfruttando il fatto che la riduzione polinomiale è transitiva

- Teorema (10.5): se un problema *NP-completo* è in *P*, allora *P=NP*
  - nota: questo teorema "riassume" l'obiettivo dello studio della NP-completezza, cioè l'identificazione dei problemi la cui appartenenza a *P* implica che *P=NP*

Problemi NP-ardui

- Un problema è *NP-arduo* se ogni problema in *NP* è riducibile polinomialmente a lui
  - non è detto che sia in NP!
  - quindi potrebbe anche non avere un algoritmo non-deterministico polinomiale (se non è in NP)

- Abbiamo usato il termine *intrattabile* informalmente, riferendoci ai problemi che sembrano richiedere tempo esponenziale
  - l'uso del termine intrattabile al posto di NP-arduo è accettabile, anche se ci potrebbero essere problemi che richiedono tempo esponenziale pur non essendo formalmente NP-ardui