Domanda 1

Completo

Punteggio ottenuto 4,00 su 6,00

Definite le classi di complessità di linguaggi P, NP, EXPTIME. Per quanto possibile, aggiungete le definizioni necessarie per spiegare queste definizioni. Fornite le inclusioni note e congetturate per queste classi.

La classe di complessità di un linguaggio è valutata secondo la seguente definizione:

TIME(f(n)) = l'insieme di tutti i linguaggi decidibili da una macchina di Turing single-tape in tempo O(f(n)), dove f è una fuzione da $N \rightarrow R^+$

Gli studi nel campo della complessità in tempo dei linguaggi hanno portato alla definizione di tre diverse classi di complessità. Le classi:

- P

Sono tutti i linguaggi decidibili in tempo polinomiale da una macchina di Turing single-tape, scritto sotto forma di insieme sarebbe: $U_k TIME(n^k)$. Solitamente quando si trova una soluzione polinomiale per un problema (ad es. $O(n^{100})$) è possibile attraverso certe tecniche ridurre la complessità di questo problema fino ad una più trattabile. Un esempio di problema polinomiale può essere PATH = {| c'è un cammino da s a t in G}

- NP

Sono tutti i linguaggi decidibili in tempo polinomiale da una macchina di Turing non-deterministica, scritto sotto forma di insieme sarebbe: $U_k NTIME(n^k)$ dove NTIME(f(n)) = {L | L è un linguaggio decidibile in tempo O(f(n)) da una MT non deterministica}. La soluzione di forza bruta per questi problemi impiega tempo esponenziale.

Un'ulteriore caratteristica che li distingue dai problemi da quelli esponenziali è che i problemi in NP sono verificabili in tempo polinomiale, ovvero: Dato A problema in NP e w un input per A

A = {w| V accetta per un qualche c} e V è un decisore che esegue in tempo polinomiale.

Questa ultima definizione è importante perché mostra la differenza tra la classe NP e la classe EXP. Se noi trovassimo un modo polinomiale di generare certificati in tempo polinomiale allora tutti i problemi in P sarebbero risolvibili in tempo polinomiale.

- EXPTIME

Sono tutti i linguaggi decidibili in tempo esponenziale da una macchina di Turing deterministica, scritto sotto forma di insieme sarebbe: $U_k TIME(n^{O(f(n))})$.

Nel campo della complessità il problema P = NP è un dibattito ancora aperto. Per ora si ha la forte convinzione che P != NP, ovvero che i problemi in NP siano insolvibili in tempo polinomiale. Questa però è solo una congettura perché non possiamo sapere se un domani vengano implementati degli algoritmi fuori dalla nostra cognizione che trovino una soluzione polinomiale per tutti i problemi in NP.

Commento:
manca l'inclusione NP in EXPTIME, il resto va bene

Domanda 2

Completo

Punteggio ottenuto 6,00 su 6,00

Spiegate cosa si intende per forma normale congiuntiva di una formula proposizionale. Esprimente il problema CNF-SAT della soddisficibilità di una forma normale congiuntiva tramite un linguaggio e spiegate cosa sappiamo su di esso e sulla relazione tra CNF-SAT e i problemi fondamentali visti nel corso.

Con forma normale proposizionale congiuntiva intendiamo una formula booleana formata da letterali (variabili booleane) e clausole (un insieme di disgiunzioni di letterali delimitati da delle paretesi) che rispetta le seguenti caratteristiche:

- i letterali risiedono solo dentro delle clausole e sono in disgiunzione tra di loro
- le clausole possono essere solo in congiunzione con altre clausole

Il problema della soddisfacibilità di una formula in forma normale congiuntiva (CNF-SAT), consiste nel decidere su una certa formula f in CNF può essere soddisfatta, ovvero se è possibile rendere vero almeno un letterale per clausola. Il problema può essere espresso da questo linguaggio:

CNF-SAT = { | dove f è una formula CNF soddisfacibile}

Sappiamo che questo problema è NP-completo perché (essendo un caso particolare di SAT) si riduce polinomialmente a SAT. Con riduzione polinomiale di A a B intendiamo che:

Esiste una funzione f computabile in tempo polinomiale , che dato un qualsiasi input w per A:

w appartiene ad $\leftrightarrow$ a f(w) appartiene a B.

Una funzione computabile non è altro che una funzione f : $\sum^*$ -> $\sum^*$ per cui esiste una MT single-tape M che dato un input w termina con f(w) sulla propria tape.

Grazie alla definizione di NP-completezza e ad una suo proprietà che asserisce che se B è NP completo è B $\leq p$ C allora anche C è NP-compelto (perché essenzialmente esiste una riduzione polinomiale g (A $\leq p$ C con A un qualsiasi problema in NP) che che altro non è che la composizione di due riduzioni polinomiali) possiamo costruire una catena di riduzioni che mostra che alla fin fine qualsiasi problema in NP è anche NP-completo (Teorema di Cook).

SAT $\leq_p$CNF-SAT $\leq_p$ 3SAT $\leq_p$CLIQUE $\leq_p$ ...

Commento:
ok

Domanda 3

Completo

Punteggio ottenuto 6,00 su 6,00

Enunciate il problema del linguaggio vuoto per linguaggi riconoscibili da una macchina di Turing, esprimete questo problema tramine un linguaggio, e provate che non è decidibile. Nella prova potete usare risultati di base del corso.

Il problema del linguaggio vuoto consiste nel decidere se una certa MT M produce il linguaggio vuoto.

E_{tm} = {M | M è una TM e L(M) = Ø }

Possiamo dimostrare che il problema non è decidibile semplicemente dimostrando che, se lo fosse, allora sarebbe decidibile anche A_{tm} , il problema dell'accettazione.

Supponiamo che esista una MT J che decide l'emptiness di un linguaggio. Dato un input una MT M

J accetta se L(M) = Ø

J rifiuta se L(M) != Ø

A questo punto costruiamo una macchina Y che decide l'accettazione basandosi su J. Dato in input un problema dell'accettazione Y costruisce una MT M' che si comporta come segue: Su un qualsiasi input x

Se x = w allora M' simula M su w

accetta se M accetta

rifiuta se M rifiuta

Se x != w allora M' rifiuta

In questo modo ho creato una MT che ha come linguaggio L(M') = {w} ∩ {L(M)} , ovvero

L(M') = w se e solo se M accetta w

L(M') = Ø se e solo se M rifiuta w o x != w

Avendo questa nuova macchina M' ora Y può lanciare J su M' e

Y accetta se e solo se J rifiuta, ma J rifiuta solo L(M') = {w} ∩ {L(M)} != Ø ma allora w appartiene a L(M)

Y rifiuta se e solo se J accetta, ma J accetta solo quando L(M') = {w} ∩ {L(M)} = Ø ma allora w non appartiene a L(M)

Ho dimostrato che il problema dell'accettazione si riduce al problema del linguaggio vuoto Atm ≤ Etm, ma sapendo che l'accettazione non è decidibile so che J a sua volta **non può** essere decidibile, altrimenti potrei decidere anche l'accettazione.

Commento:
ok

Domanda 4

Completo

Punteggio ottenuto 5,00 su 6,00

Definite le classi di dei linguaggi decidibili, positivamente decidibili, negativamente decidibili. Per quanto possibile, aggiungete le definizioni necessarie per spiegare queste definizioni. Fornite le tutte le relazioni importanti tra queste tre classi e tra linguaggi positivamente e negativamente decidibili. Fornite esempi di linguaggi che appartengono a una classe e non a un'altra.

Supponiamo di avere un qualsiasi linguaggio A e una stringa w. Vogliamo verificare se w appartiene ad A.

A è un linguaggio decidibile se esiste una MT M che dato un input w

M accetta se w appartiene L(A)

M rifiuta se w non appartiene a L(A)

A è un linguaggio positivamente decidibile se esiste un decisore positivo X che siamo sicuri termini solo nel caso in cui un qualsiasi input w appartiene L(A).

A è un linguaggio negativamente decidibile se esiste un decisore negativo Y che siamo sicuri termini sono nel caso in cui un qualsiasi input w non appartiene a L(A)

Tutte e tre queste classi di linguaggi formano la classe dei linguaggi Turing-recognizable, ovvero quei linguaggi riconosciuta da una qualche MT di Turing.

Un teorema molto importante che lega tra di loro queste classi è il problema di Post. Esso asserisce che:
Un linguaggio A è decidibile se e solo se è sia positivamente decidibile che negativamente decidibile.

Dimostrazione:

Se A è decidibile allora è positivamente e negativamente decidibile. Questo aspetto è ovvio perché sappiamo che qualunque linguaggio decidibile A è Turing-recognazible e lo stesso vale per il suo complemento ¬A.

Se A è positivamente e negativamente decidibile allora è decidibile.

Supponiamo di avere due decisori positivi

M1 decide positivamente L(A)

M2 decide positivamente L(¬A)

Basandomi su queste due macchine posso costruire una macchina F che, dato un input w

Lancia in parallelo M1 e M2 sull'input w

F accetta se M1 accetta

F rifiuta se M2 accetta

Così facendo ho appena dimostrato che F decide A, perché la stringa w o appartiene a L(A) (in questo caso M1 accetta e F termina accettando) o non appartiene a L(A), cioè appartiene a L(¬A) (in questo caso M2 accetta e F termina rifiutando).

Esempio di linguaggio decidibile: Acfg

Esempio di linguaggio negativamente decidibile: WANG

Esempio di linguaggio positivamente decidibile: Atm

Esempio di linguaggio ne positivamente ne negativamente decidibile : EQtm

Commento:
dovevi anche spiegare che quando un decisore positivo (o negativo) per A termina allora dice la verita', cioe' se accetta una stringa allora la stringa sta in A e se rifiuta allora la stringa non sta in A

Domanda 5

Completo

Punteggio ottenuto 6,00 su 6,00

Definite la nozione di Linguaggio decidibile, positivamente decidibile e negativamente decidibile, e spiegate la differenza tra queste tre nozioni, le loro proprietà fondamentali, e introducete esempi significativi.

Riprendo i concetti già espressi nella precedente domanda. Appoggiandoci alla definizione di mapping reducibility possiamo trarre alcune importanti proprietà sulla decidibilità e sulla turing-recognizabilità.

Un linguaggio A si dice riducibile (mapping reducible) ad un problema B A $\leq_m$ B se esiste una funzione computabile f: $\sum^* \rightarrow \sum^*$ che, per qualsiasi w

w appartiene a A $\leftrightarrow$ f(w) appartiene a B

- Se A $\leq_m$ B e B è decidibile allora anche A è decidibile.

Supponiamo che una macchina X decida il linguaggio B. Posso costruire una MT F che:

Su un input w del linguaggio A

Applica la funzione f su w e si calcola f(w)

Lancia X su f(w)

F restituisce il risultato di X

Possiamo affermare che F decide A proprio perché F accetta se X accetta, ma X accetta le f(w) appartiene a L(B). Come conseguenza della mapping reducibility possiamo affermare che le f(w) appartiene a L(B) allora w appartiene a L(A). Lo stesso discorso vale all'opposto se F rifiuta.

- Se A $\leq_m$ B e B è positivamente decidibile allora anche A lo è.

La dimostrazione è uguale a quella sopra descritta l'unica differenza è che X e F non sono più decisori ma solamente decisori positivi.

- Se A $\leq_m$ B e A non è positivamente decidibile allora neanche B lo è.

Possiamo dimostrarlo supponendo che A sia il linguaggio L(¬Atm). A $\leq_m$ B è equivalente a ¬A $\leq_m$ ¬B. Ma visto che A = ¬Atm ora la relazione diventa $A_{tm} \leq_m$ ¬B. Dalla proprietà sopra descritta sappiamo che ¬B fosse negativamente decidibile (ovvero B fosse positivamente decidibile) allora anche Atm lo sarebbe. Ma noi sappiamo che Atm non può essere negativamente decidibile altrimenti Atm sarebbe decidibile

Esempio Eqtm non è ne positivamente ne negativamente decidibile. Tramite il concetto di mapping reducibility posso dimostrare che:

- $A_{tm} \leq_m$ Eqtm. Grazie alla proprietà sopra descritta posso dedurre che Eqtm non è negativamente decidibile.
- ¬$A_{tm} \leq_m$ Eqtm. Grazie alla proprietà sopra descritta posso dedurre che Eqtm non è positivamente decidibile.

Commento:
ok

Domanda 6

Completo

Punteggio ottenuto 6,00 su 6,00

Descrivete con parole vostre il Teorema di Savitch. Spiegate la differenza tra le proprietà delle classi non deterministiche in tempo e classi non deterministiche in spazio.

Per definire il teorema di Savitch devo definire due classi di complessità **spaziale.**

Data una funzione f(n) con $f(n) \geq n$

SPACE(f(n)) = { L | L è un linguaggio deciso da una MT deterministica che impiega spazio O(f(n))}

NSPACE(f(n)) = { L | L è un linguaggio deciso da una MT non-deterministica che impiega spazio O(f(n))}

Il teorema di Savitch enuncia che, data una funzione f: N -> R+, con $f(n) \geq n$

NSPACE(f(n)) è inclusa in SPACE ($f^2(n)$). Ovvero una MT non deterministica che impiega spazio O(f(n)) può essere ridotta ad una MT deterministica che impiega al più $O(f^2(n))$.

Se definiamo come PSPACE = UkSPACE(n^k) e NPSPACE = UkNSPACE(n^k) possiamo osservare (grazie al teorema di Savitch) queste due classi convergono nella stessa classe PSPACE in quanto ogni MT non deterministica in spazio può essere tradotta in una MT deterministica che utilizza sempre spazio polinomiale.

Commento: