ESERCIZI E DOMANDE

NB: Queste sono ulteriori domande ed esercizi oltre a quelli presenti nei lucidi

Analisi della complessità

Nell'analisi di complessità e, in particolare nella notazione O(), si ignorano le costanti moltiplicative e additive. Discutere questa scelta indicandone i vantaggi e gli svantaggi.

Cosa si intende quando diciamo che un algoritmo A è asintoticamente più efficiente dell'algoritmo B?

1. A è sempre meglio di B per tutti gli input
2. A è sempre meglio di B per input di grandi dimensioni
3. A è sempre meglio di B per input di piccole dimensioni
4. B è sempre meglio di A per input di piccole dimensioni

Verificare e motivare la correttezza o meno delle seguenti affermazioni.

a) $(n!) \in \Omega(n^2)$ (Corretto)

(b) $n^{3/2} \in O(n \log n)$ (Errato)

(c) $2^{(\log_2 n)} \in \Theta(n)$ (Corretto)

Spiega la validità della seguente espressione e illustrare la differenza fra costo di un programma e andamento asintotico rappresentato con la notazione O()

if f(n) <= g(n) then O(f(n) + g(n)) = O(g(n))

Per n > 1, sia T(n) il numero esatto di esecuzioni del comando a =a + 2 nel seguente frammento codice:

```
for i  = 1 to n – 1 do
        for j  =  n – i + 1 to n do
                a =a + 2
```

Utilizzando la notazione O() si esprima il costo di T(n) in funzione di n; si calcoli il valore esatto di T(n) per n=10.

Ripetere l'esercizio per il programma seguente

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
   for (j = 2; j <= n; j = j * 2) {
      k = k + n / 2;
   }
}
```

Esprimere il costo asintotico dell'algoritmo usuale per la moltiplicazione di due numeri di n bit. Assumi che ciascuna operazione fra due bit abbia costo unitario.

```
          1........101  x            n bit
          1...........11             n bit
          ____________
          1 .......101 +      n                                    operazioni
           ..... 1010  +       n+1    }   Somma di n valori        operazioni
          ...........00   +                parziali
                    ...
  ________________
  ……......00........0         2n bit nel caso peggiore
```

Quante operazioni sono necessarie con l'usuale algoritmo per eseguire la moltiplicazione di una matrice (n x m) per una matrice (m x k)?

Analizza il numero di istruzioni eseguito dal seguente frammenti programma per il calcolo del numero di Fibonacci ed esprimere il costo come funzioni di n. Se teniamo conto delle dimensioni dell'input il programma è polinomiale?

Sia dato un array ordinato A che memorizza solo 0 e 1, tale che ogni zero appare in una componente prima di componenti che memorizzano 1; in altre parole l'array è del tipo {0, 0, 0, ..., 0, 0, 1, 1, ..., 1,1,1}. Progetta un algoritmo che trova la componente più piccola tale che A[i] = 1 e abbia costo O(log n) nel caso peggiore (n rappresenta il numero degli elementi dell'array.

Computabilità

Enunciare e discutere la tesi di Church-Turing, chiarendo perché viene formulata come tesi e non come teorema.

Definire il concetto di numerabilità (o equivalentemente contabilità) e mostrare utilizzando la tecnica della diagonalizzazione che l'insieme dei linguaggi sull'alfabeto {0, 1} non è numerabile.

Assumi che i linguaggi B e C siano decidibili (cioè dato x esistono due algoritmi A1 e A2 che decidono se x appartiene a B e se x appartiene a C rispettivamente. Mostra che anche
B ∪ C (linguaggio unione) , B ∩ C (linguaggio intersezione) ,
B \ C (linguaggio differenza formato dalle stringhe che appartengono a B ma non a C)
B* sono linguaggi decidibili.

Dati A1 e A2 abbiamo che A1(x) = vero se x appartiene a B falso altrimenti
Analogamente per A2 e C

Dimostra che il seguente insieme è decidibile
{B: B è la codifica di un automa a stati finiti deterministico che non accetta nessuna stringa}

Soluzione: La prova per decidere se X appartiene al linguaggio si basa sui seguenti passi: utilizzando X ssi costruisce l'automa deterministico codificato. Quindi si verifica per ogni stato finale q se esiste un cammino dallo stato iniziale a q. Se un tale cammino esiste per almeno uno stato finale allora l'automa accetta stringhe. Per verificare che per ogni stato finale q esiste un cammino dallo stato iniziale a q è sufficiente verificare l'esistenza di cammini di lunghezza al più $|Q| - 1$, dove Q è il numero di stati dell'automa. Perché è sufficiente considerare cammini di lunghezza $|Q| - 1$?

Mostra che l'insieme delle Macchine di Turing è numerabile.

Quante sono le possibili macchine di Turing con alfabeto binario e con 4 stati?

Grammatiche, espressioni regolari e automi a stati finiti

Descrivere le fasi di analisi lessicale e di analisi sintattica di un compilatore; per ciascuna delle due fasi descrivi l'input e l'output e come viene elaborato il programma.

Illustrare la gerarchia di Chomsky delle grammatiche; indicare per ciascuna categoria la proprietà/caratteristica a vostro giudizio maggiormente rilevante.

Descrivi l'algoritmo per eliminare le ricorsioni sinistre di una grammatica di tipo 2.

Fornire una grammatica non ambigua per generare il linguaggio delle espressioni aritmetiche avente cinque simboli terminali: id (che rappresenta un identificatore), i simboli di operazione + e – e le parentesi tonde (e). Ovviamente la grammatica deve generare tutte e sole le stringhe che sono corrette (ad esempio non deve generare la stringa id+((id-id) o la stringa id++id-id.

Motivare inoltre perché questo linguaggio non può essere generato da una grammatica di tipo 3. Quali sono le proprietà di una grammatica LL(1) e per quale ragione sono molto adatte per descrivere i linguaggi di programmazione.

Quali sono le proprietà di una grammatica LR(0) e per quale ragione sono molto adatte per descrivere i linguaggi di programmazione.

Descrivere un analizzatore sintattico di tipo top-down specificando l'input dell'analizzatore, i possibili risultati calcolati e un possibile algoritmo per l'analisi.

Descrivere l'algoritmo di analisi sintattica di una grammatica LL(0)

Descrivi l'algoritmo di analisi sintattica di una grammatica LR(0)

Descrivi il linguaggio generato dalla seguente grammatica (S assioma, S e B simboli non terminali)
$S \rightarrow abc$
$S \rightarrow aSBc$
$cB \rightarrow Bc$
$bB \rightarrow bb$
Nella gerarchia di Chomsky di che tipo è la grammatica precedente? Prova a trovare una grammatica di tipo 3 che genera lo stesso linguaggio.

Si considerino le seguenti espressioni regolari sull'alfabeto {a,b,c}:

(1) R1= a (b | aa)* c
(2) R2= abb (a | cc) (ab)*

Per ciascuna costruire un automa a stati finiti deterministico o nondeterministico che riconosce il linguaggio definito dall'espressione regolare.

Data una Grammatica regolare G, è possibile stabilire se G genera il linguaggio vuoto? Come?

Data una Grammatica regolare G, è possibile stabilire se G genera un linguaggio finito o infinito? Come?

(Sugg. Per i due esercizi precedenti utilizzare il fatto che per ogni grammatica regolare G esiste un automa a stati finiti che riconosce tutte e sole le stringhe che appartengono a G)

Alice e Biagio discutono animatamente. Stanno esaminando una grammatica regolare G in cui V_T è l'insieme dei terminali, V_N è l'insieme dei non terminali, S è l'assioma e P l'insieme delle produzioni (di tipo 3). Carla sta per fornire loro un ulteriore insieme di produzioni P', sempre di tipo 3, che tuttavia Alice e Bob non hanno ancora visto. Carla ha avvisato che P è anch'esso basato su V_T e V_N e che chiederà ad Alice e Biagio di esaminare la grammatica regolare G' in cui V_T è l'insieme dei terminali, V_N è l'insieme dei non terminali, S è l'assioma e l'insieme delle produzioni (di tipo 3) è l'unione di P e P'.
Siano L(G) e L(G') i linguaggi generati da G e G'. Alice sostiene che $L(G) \subseteq L(G')$ poiché l'introduzione di nuove produzioni potrebbe consentire di generare nuove stringhe, mentre Biagio pensa che $L(G) \supseteq L(G')$, dato che le nuove produzioni introdurranno restrizioni sulle possibilità di generazione.

Per i seguenti linguaggi definisci un'espressione regolare o una grammatica regolare che li genera:
(1) l'insieme delle stringhe binarie che contengono le stringhe 101 o 010 (o ambedue) come sottostringhe.
(2) l'insieme delle stringhe binarie che contengono le stringhe 00 e 11 come sottostringhe.
(3) l'insieme delle stringhe binarie che contengono la stringa 00 ma non la stringa 11 come sottostringhe.
(4) l'insieme delle stringhe binarie che contengono iniziano con 00 e finiscono con 11.
(5) l'insieme delle stringhe sull'alfabeto {x,y,z} in cui ogni x è immediatamente seguita da una y.
(6) l'insieme delle stringhe sull'alfabeto {x,y,z} che contengono un numero dispari di y.

Definire un ASF che, avendo in input stringhe costruite sull'alfabeto $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, riconosce il linguaggio delle stringhe che descrivono un numero multiplo di 5 in base 10. (È ammissibile che un numero inizi per 0, come ad esempio 0051 o 012345)

Sono dati M1 e M2, due automi a stati finiti che riconoscono i linguaggi L1 e L2 rispettivamente;
(1) definire un automa M che riconosce il linguaggio $L1 \cup L2$ (formato dalle stringhe cha appartengono a L1 o a L2 o a entrambi). Analogamente per intersezione

Grammatiche libere dal contesto

Considera la seguente grammatica G (assioma S), con simboli terminali {a, b, c}:
S→ aAb | AE
A→ aAbE | ab
B → AB |c
D→ AAcE
E → BA
F→ FA | a
(1) Identifica i simboli non terminali che non sono raggiungibili a partire dall'assioma.
(2) Trasforma la grammatica eliminando le produzioni inutili.
(3)
(Risposta (1): D,F - non esiste la possibilità di generare una stringa che contiene D o F a partire da S e ottenendo una sequenza di terminali;
(2) bisogna eliminare tutte le produzioni che coinvolgono D e F)

Considera la seguente grammatica G (assioma S), con simboli terminali {a, b, c}:

S→ aABb | AC
A→ aAb | ab
B → abB
C → ABC | c
(1) Identifica i simboli non terminali che non sono utilizzati per definire stringhe del linguaggio.
(2) Trasforma la grammatica eliminando le produzioni inutili.
(Risposta (1): B (non esiste la possibilità di eliminare B trasformandolo in una sequenza di terminali; (2) bisogna eliminare tutte le produzioni che coinvolgono B, ottenendo
S→ AC
A→ aAb | ab
C → c

Trova una grammatica libera dal contesto per i seguenti linguaggi
L1={ $0^m 1^n$ con n≠m, n,m>0} L2 ={ $a^m b^n c^p d^q$ con m+n= p+q>0}
Sugg. Parti da grammatica per ={ $0^n 1^n$ con n>0}

Costruire un analizzatore sintattico predittivo a discesa ricorsiva per il linguaggio delle parentesi bilanciate, generato dalla grammatica S (assioma), simboli terminali '(' e ')' e produzioni S→(S) S | ε
E' possibile svolgere l'esercizio anche su una grammatica equivalente a quella data.

Data la grammatica per il linguaggio delle parentesi bilanciate, generato dalla grammatica S (assioma), simboli terminali '(' e ')' e produzioni S→(S) S | ε dare l'albero di derivazione per la stringa (()())()

Si consideri la grammatica S→ aSbS S→ bSaS S→λ
La grammatica è ambigua? Quanti differenti alberi di derivazione esistono per la sequenza abab ? Mostrare le derivazioni sinistre e destre.

Sia data la grammatica - assioma E, simboli nonterminali caratteri maiuscoli, simboli terminali +, -, *, /, (,), id -
E→TQ
Q→+TQ Q→ -TQ Q→ λ
T → FR
R→ *FR R→ /FR R→ λ
F→ (E) F→ id
(a) Fornire l'albero di derivazione per le stringhe
(1) id + ((id* id)- id) * id (2) ((id-id) /id) + (id –(id *id))
(b) Questa grammatica è LL(1) ma la tabella di parsing non può essere calcolata usando solo i simboli FIRST; spiegare perché.

Dimostrare che la seguente grammatica
S→ aB S→ aC S→ B B→ bB B→d C→ B
non è LL(1). Costruire una grammatica LL(1) equivalente alla precedente.
Soluz. La grammatica non è LL(1) a causa delle produzioni con S nella parte sinistra con stesso simbolo FIRST (S→ aB, S→ aC). Infatti non sappiamo scegliere quale delle due produzioni. Osserviamo che si può introdurre un nuovo non terminale T e modificare la grammatica nel seguente modo
S→ aT T→B T→ C S→ B B→ bB B→d C→ B
A questo punto è facile verificare che le produzioni T→C e C→ B equivalgono a T→ B che è già presente; quindi possono essere eliminate. In questo modo otteniamo la grammatica
S→ aT T→B S→ B B→ bB B→d

Verifichiamo che questa grammatica è LL(1); calcoliamo i simboli FIRST
FIRST(S → aT)= {a} FIRST(S→B)= {b,d} FIRST(B→bB)= {b} FIRST(B→d)= {d} FIRST(T→B)= ε
Per produzioni con stessa parte sinistra i simboli FIRST sono disgiunti e quindi la grammatica è LL(1).

NP-completezza

Definire le classi: P, NP e indicare le relazioni di contenimento fra le classi

Definire la classe dei problemi NP-completi e indicare le relazioni di contenimento rispetto alle classi P e NP

Indicare un possibile risultato che permetterebbe di stabilire che $P \neq NP$

Indicare un possibile risultato che permetterebbe di stabilire che $P = NP$

Per dimostrare che un problema A è NP-completo cosa è necessario dimostrare?

Definire il problema della soddisfacibilità di una formula logica e cosa è una formula in forma normale congiuntiva (CNF). A quale classe di complessità temporale appartiene il problema della soddisfacibilità di una formula CNF?

Definire il problema della soddisfacibilità di una formula logica e cosa è una formula in forma normale congiuntiva (DNF). A quale classe di complessità temporale appartiene il problema della soddisfacibilità di una formula DNF?

Nel caso di problemi NP-completi non rinunciamo a cercare soluzioni. Illustrare un possibile approccio per il problema della soddisfacibilità di formule logiche.

Illustrare una riduzione – a vostra scelta - che mostra che un problema A è NP-completo.

Per mostrare che un problema B è NP-difficile è sufficiente fornire una riduzione da un problema A a B. Quali proprietà deve verificare A e quali deve verificare la riduzione?