Algoritmi e Principi dell'Informatica

Appello del 17 gennaio 2018

Chi deve sostenere l'esame integrato (API) deve svolgere tutti gli esercizi in 2 ore.

Chi deve sostenere solo il modulo di *Informatica teorica* deve svolgere gli Esercizi 1 e 2 in 1 ora.

Chi deve sostenere solo il modulo di *Informatica 3* deve svolgere gli Esercizi 3 e 4 e 5 in 1 ora.

NB: i punti attribuiti ai singoli esercizi hanno senso solo con riferimento all'esame integrato e hanno valore puramente indicativo.

Esercizio 1 (8 punti)

Si consideri il linguaggio L costruito su un alfabeto che comprende, tra gli altri, anche i simboli *i*, *f*, e *u*, e fatto di tutte e sole le parole in cui ogni simbolo *f* è preceduto da almeno un simbolo *i*, e tra un simbolo *i* ed il successivo *f* c'è esattamente un simbolo *u* (oltre potenzialmente ad altri simboli, ovviamente diversi da *f*). Inoltre, ogni parola del linguaggio L deve terminare con un simbolo *f*.

1. Dire qual è il formalismo (o quali sono i formalismi, se più di uno) a potere espressivo minimo tra quelli visti a lezione (automi, grammatiche, logiche) che è in grado di esprimere il linguaggio L.
2. Formalizzare il linguaggio L nel formalismo (o in uno solo dei formalismi, se più di uno) individuato al punto 1.

Esercizio 2 (8 punti)

1.

Dire se è decidibile il problema di stabilire se, data una generica grammatica regolare G, esiste una sequenza di produzioni che portano ad una parola fatta solo di simboli terminali.

Dire se il problema è semidecidibile.

2.

Dire se è decidibile il problema di stabilire se, data una generica grammatica non ristretta G, esiste una sequenza di produzioni che portano ad una parola fatta solo di simboli terminali.

Dire se il problema è semidecidibile.

Esercizio 3 (6 punti)

Sia A = {a, b, c} un alfabeto; si considerino gli omomorfismi h', h" : $A^* \rightarrow A^*$ definiti da h'(x) = c, per ogni $x \in A$, e h"(a) = b, h"(b) = a, h"(c) = c, estesi naturalmente a parole (es. h"(abbcac) = baacbc).

Si descrivano una MT e una RAM che accettano il linguaggio $\{h'(w) \cdot w \cdot h''(w) \mid w \in \{a, b\}^+\}$, valutandone le complessità temporali e spaziali, nel caso della RAM con entrambi i criteri di costo.

Esercizio 4 (6 punti)

Si consideri la mappa della viabilità di una città rappresentabile come un grafo orientato, dove un nodo rappresenta un incrocio e un arco la presenza di una strada che va da un dato incrocio ad un altro. Il verso dell'arco indica il verso di percorrenza della strada; in caso di strada a doppio senso di percorrenza sono presenti due archi.

1. Si argomenti qual è la miglior scelta di struttura dati per rappresentare il grafo in questione volendo ottenere la più bassa complessità spaziale. Si indichi la complessità spaziale in funzione del numero di nodi |V| e di archi |E|.
2. Date due persone situate in due incroci, si descriva un algoritmo che calcoli il percorso più breve affinché una raggiunga l'altra e se ne indichi la complessità temporale. La lunghezza del percorso è misurata il numero di archi che lo compongono. Solo una delle due persone può muoversi per raggiungere l'altra (non possono incontrarsi in un incrocio diverso da quello di partenza di una delle due).
3. Nel caso si sia certi di essere in una città priva di sensi unici, è possibile progettare un algoritmo più efficiente del precedente? Se sì, lo si descriva e se ne calcoli la complessità asintotica, altrimenti si indichi cosa previene la progettazione di una soluzione più efficiente.

Esercizio 5 (4 punti)

Si calcoli la complessità temporale asintotica del seguente programma in funzione del valore `n` in input:

```
myfun (n)
1  if n ≤ 1
2    return n
3  res := 0
4  for i := 1 to 3
5    res := res + myfun (n/4)
6  j := 1
7  while j ≤ n*n
8    res := res / 2
9    j := j*2
10 return res
```

Tracce delle soluzioni

Esercizio 1

1.

Il formalismo in questione è la logica MFO, che è strettamente meno potente degli FSA.

2.

$$\forall x\,(\,\mathrm{f}(x) \rightarrow \exists y\,(\,y < x \wedge \mathrm{i}(y)\,)\,)$$
$$\wedge$$
$$\forall x, y\,(\,x < y \wedge \mathrm{i}(x) \wedge \mathrm{f}(y) \wedge \forall z\,(\,x < z \wedge z < y \rightarrow \neg\mathrm{f}(z)) \rightarrow \exists k\,(\,x < k \wedge k < y \wedge \mathrm{u}(k) \wedge \forall t\,(\,x < t \wedge t < y \wedge t \neq k \rightarrow \neg\mathrm{u}(t))\,)\,)$$
$$\wedge$$
$$\exists x\,(\,\mathrm{last}(x) \wedge \mathrm{f}(x)\,)$$

Esercizio 2

1.

Siccome per ogni grammatica regolare G possiamo costruire un'equivalente FSA che accetta lo stesso linguaggio generato da G (e viceversa), questo problema è lo stesso del problema di decidere se il linguaggio accettato da un FSA è vuoto o meno.

Il problema è quindi decidibile (quindi anche semidecidibile), perché lo è il problema di stabilire se il linguaggio accettato da un FSA è vuoto o no, per risolvere il quale è stato brevemente introdotto un algoritmo a lezione (basta controllare l'accettazione di tutte le parole con lunghezza $\leq |Q|$, con Q lo spazio di stato dell'automa, che sono in numero finito).

2.

E' noto che, data una qualunque grammatica non ristretta G, è possibile costruire un'equivalente MT che accetta lo stesso linguaggio generato da G (e viceversa). Quindi, questo problema è equivalente al problema di decidere se il linguaggio accettato da una generica MT è vuoto o no.

Il problema non è decidibile, lo si può mostrate per riduzione dal problema della terminazione del calcolo. Per esempio, si può costruire una MT M che, data in input una stringa *x*, funziona in questo modo: M prende un'altra generica MT M' ed una stringa di input *x'*, simula M' su *x* e, se M' accetta *x*, allora termina anch'essa (altrimenti va avanti all'infinito). Per ogni coppia di M' e *x*', il linguaggio accettato da M è vuoto se e solo se M' termina su *x*', che sappiamo essere un problema indecidibile, quindi anche il problema di partenza (la vuotezza del linguaggio accettato da una MT).

Il problema è invece semidecidibile, in quanto basta usare il solito procedimento diagonale per enumerare le stringhe in input ed il numero di passi da simulare e, se esiste una stringa in input per cui la macchina termina per un certo numero di passi, si conclude che il linguaggio accettato dalla macchina non è vuoto, altrimenti, se il linguaggio è vuoto, la computazione continua all'infinito.

Esercizio 3

La MT legge la prima parte di stringa di caratteri c, copiandola nel primo nastro; finite le c, riposiziona la testina del nastro all'inizio e sovrascrive il contenuto con la successiva

sottostringa di a e b. Arrivato in fondo alle c, ritorna all'inizio e legge la parte finale, controllando che le ad ogni a corrisponda una b in memoria e viceversa.

La RAM si comporta sostanzialmente allo stesso modo, con la differenza di dover copiare l'intera stringa non essendo in grado di ritornare indietro sul nastro di ingresso.

Le complessità spaziale e temporale a costo costante sono $\Theta(n)$, mentre a criterio di costo logaritmico sono rispettivamente $\Theta(n)$ e $\Theta(n \log(n))$.

Esercizio 4

1. Il grafo della viabilità di una città è sparso (è planare, a meno di qualche sottopassaggio) quindi una rappresentazione a liste di adiacenza risulta essere quella più efficiente in termini di spazio costando $O(|V|+|E|)$.
2. E' possibile risolvere il problema di trovare il percorso più breve che conduce una persona all' altra calcolando separatamente i percorsi a partire da ognuna di esse e scegliendo il più breve. Ognuno dei percorsi può essere calcolato tramite una visita in ampiezza del grafo che inizia dalla persona che si sposta e termina quando il nodo incontrato è quello dove risiede la persona (rimasta ferma) da incontrare. La distanza si ricava tenendo traccia di quante volte si è espanso l' orizzonte di visita nella ricerca in ampiezza.
3. Nel caso la città contenga unicamente strade a doppio senso, è sufficiente effettuare una singola ricerca in ampiezza a partire da uno qualunque dei due incroci per ottenere il percorso più breve in $O(|V|+|E|)$. Partire simultaneamente da entrambi gli incroci non cambia la complessità asintotica nel caso pessimo.

Esercizio 5

La ricorrenza associata alla funzione è la seguente:
$T(n) = 3T(n/4) + \Theta(\log(n))$

La ricorrenza può essere risolta sfruttando il master theorem: occorre confrontare $n^{(\log_4(3))}$ con $\log(n)$. $n^{(\log_4(3))}$ è polinomialmente maggiore di $\log(n)$, quindi siamo nel caso 1 del teorema, e la soluzione della ricorrenza è $\Theta(n^{(\log_4(3))})$.