Fondamenti di informatica II (prova di Modelli)

7 giugno 2022
120 minuti

1. Analisi di complessità.
   1.1. Che significa esattamente l'affermazione: "il programma *P* ha costo *O*(*n*)?
   1.2. Quali sono i principali vantaggi e svantaggi nell'utilizzo della notazione *O* per descrivere la complessità dei programmi?
   1.3. Determinare l'andamento asintotico, delimitandolo sia superiormente che inferiormente, della funzione T(*n*) la cui ricorrenza è:
   $T(n) = T(n/2) + n^2$ per $n > 0$; $T(0) = 1$.

2. Stabilire se la grammatica con alfabeto {*a*, *b*} le cui produzioni sono
   $S \rightarrow ab \mid aSb$
   sia LL(1). In caso negativo determinare una grammatica equivalente LL(1) (provando che sia LL(1)).

3. Data la seguente grammatica regolare:
   $S \rightarrow aA \mid aB$
   $A \rightarrow bS$
   $B \rightarrow bS \mid b$
   3.1. Definire il linguaggio generato dalla grammatica. (Notazione insiemistica o espressione regolare)
   3.2. Valutare l'ambiguità della grammatica, disegnando eventualmente gli alberi di derivazione da cui si deduce l'ambiguità.
   3.3. Costruire una grammatica regolare, equivalente e non ambigua.

4. Costruire un ASF che riconosca tutte e sole le stringhe **non** contenenti come sottostringa le ultime tre cifre del proprio numero di matricola. Se ad es. la matricola fosse 123456 allora la stringa 33845676 non sarebbe accettata, mentre 34345962 sì. L'alfabeto di riferimento è {0, …, 9}.

5. L'Indecidibilità del problema della fermata vuol dire che, dato un programma *P* e un input *x*:
   5.1. per nessun *x* posso decidere se *P* si ferma con input *x*
   5.2. posso decidere se *P* si ferma con input *x* solo per qualche *x*
   Discutere le due affermazioni 5.1 e 5.2; in particolare per ciascuna indicare se sia vera o falsa; motivare la propria risposta.

6. *Macchine di Turing*
   6.1. Fornire la definizione di una macchina di Turing con due nastri.
   6.2. Le macchine con più nastri sono in grado di eseguire calcoli primitivi non possibili alla macchina con un solo nastro (ma simulabili). Motivare l'affermazione.

7. *NP-Hardness*
   In quanto segue la nozione di *riducibilità* va definita formalmente.
   7.1. Sapendo che SAT, il problema della soddisfacibilità di formule logiche, è NP-completo come si può dimostrare che un altro problema sia NP-hard (o NP difficile)?
   7.2. Assumendo che SAT sia NP-completo stabilire la NP-hardness di PLI.

8. *Grammatiche e linguaggi CF*.
   8.1. Dare la definizione di grammatica di tipo 2. Fornire una grammatica che definisce il linguaggio $L = \{a^n b^{2n} , n \geq 1\}$.
   8.2. Stabilire se la seguente affermazione sia vera o falsa, motivando la risposta: "se L_1 è di tipo 2 ma non di tipo 3 e $L_1 \subseteq L_2$, (quindi ogni stringa in L_1 è anche stringa di L_2) allora L_2 non è di tipo 3"