Algoritmi e Principi dell'Informatica
Appello del 12 Luglio 2019
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 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)
Un linguaggio L viene detto *senza prefissi* se non contiene alcuna stringa che sia un prefisso proprio di un'altra stringa di L.
È decidibile il problema di stabilire se un generico automa a stati finiti A riconosca un linguaggio senza prefissi?
Cambierebbe la risposta se, al posto di un automa a stati finiti, fosse data una generica macchina di Turing?

Esercizio 2 (8 punti)
Si consideri la seguente grammatica G1:

S -> ab A b | S b | abb
A -> ab S b

1) Si descriva formalmente, come insieme di stringhe, il linguaggio L(G1).
2) È possibile rendere il precedente linguaggio regolare, aggiungendo una sola produzione a G1? In caso negativo, si motivi esaurientemente la risposta; in caso positivo, si indichi tale produzione e si definisca il linguaggio risultante usando una grammatica o automa a potenza espressiva minima.

Esercizio 3 (7 punti)
Si consideri una funzione $f(n)$ tale che

$$\lim_{n \to \infty} \frac{f(n)}{n^c} = 0$$

dove c è una costante positiva arbitraria.
Si può concludere che $f(n) = O(n^{c-\varepsilon})$, per una qualche costante positiva $\varepsilon < c$? Motivare opportunamente la risposta.

Esercizio 4 (9 punti)
Siano dati k BST (Binary search tree, o alberi binari di ricerca), ciascuno contenente un numero n_i di nodi, anche fortemente variabile tra i vari alberi. Si descriva in modo chiaro, sintetico e preciso, preferibilmente anche mediante pseudocodice, un algoritmo che produca una *lista concatenata monodirezionale*, ossia con il solo puntatore all'elemento successivo, e ordinata in ordine crescente, contenente tutte e sole le foglie dei k alberi. Nel caso si faccia uso di altre strutture dati oltre ai k BST in input e la lista finale da produrre in output, queste dovranno comunque essere liste concatenate monodirezionali: né array, né alberi, liste bidirezionali, ecc. Si valutino la complessità temporale e spaziale dell'algoritmo in funzione del numero totale di nodi degli alberi e del numero k degli alberi.
NB1: si assuma che le chiavi degli oggetti contenuti nei nodi degli alberi possano essere confrontate mediante le normali relazioni, <, >, = .
NB2: non vi è bisogno di ricodificare operazioni base per le normali strutture dati note in letteratura: esse possono essere semplicemente invocate, precisandone la relativa complessità ai fini del calcolo della complessità totale dell'algoritmo.

Algoritmi e Principi dell'Informatica, *English version*
July 12, 2019
The total available time is 2h.
If you are registered for the first module only (*Informatica Teorica*), you must solve Exercises 1 and 2 in 1h.
If you are registered for the second module only (*Informatica 3*), you must solve Exercises 3 and 4 in 1h.
NB: points assigned to individual exercises are intended w.r.t. the entire exam and are only indicative.

Exercise 1 (8 points)
A formal language L is called *prefix-free* if it does not contain any string that is the strict prefix of another string of L.
Is it decidable to state whether a generic finite state automaton A accepts a prefix-free language? Would your answer be different if, instead of a finite state automaton, a generic Turing machine is given?
Explain your answers.

Exercise 2 (8 points)

Consider the following gramma G1:

S -> ab A b | S b | abb
A -> ab S b

1. Formally describe the language generated by G1, L(G1), as a set of strings.
2. Is it possible to make L(G1) a regular language, say L', by adding just one production to G1? If not, explain why; if yes, provide a suitable automaton or grammar describing L', of course belonging to the family with minimum expressive power.

Exercise 3 (7 points)
Consider a function $f(n)$ such that

$$\lim_{n \to \infty} \frac{f(n)}{n^c} = 0$$

where c is an arbitrary, positive, constant.
Is it possible to conclude that $f(n) = O(n^{c-\varepsilon})$, for some positive constant $\varepsilon < c$? Clearly explain your answer.

Exercise 4 (9 points)

Given k BST (Binary search trees) each one containing n_i nodes (NB: the various trees can have strongly different size), precisely describe an algorithm, possibly by using some pseudocode, that delivers as output a singly linked list, i.e., a list where each elements points only to the next one, that stores all and only the leaves of all trees in non-decreasing order. If you use further data structures besides the trees given as input and the list requested as output, such data structures must necessarily be singly linked lists: no arrays, no other trees, etc. Evaluate the space and time complexity of your algorithm as a function of the total number of nodes of the k trees.
NB1: you may assume that the keys of the objects stored in the tree nodes can be compared by means of the usual arithmetic relations: <, >, =.
NB2: you may use (call, in terms of pseudocode) any basic operation on traditional data structures well-known from the literature, without need to copy their pseudocode.

Tracce delle soluzioni

Esercizio 1
Per l'automa a stati finiti è decidibile. Basta infatti effettuare qualche controllo strutturale sull'automa.
Per preparare l'analisi che segue, si eliminano tutti gli stati che non sono raggiungibili dallo stato iniziale (mediante ad esempio una ricerca in profondità). Quindi, per ogni stato finale q, si verifica se c'è un altro stato finale q' raggiungibile da q oppure se ci sia un ciclo che da q ritorna a q. Se esiste un tale cammino, allora il linguaggio riconosciuto non è senza prefissi, altrimenti lo è.
Nel caso della macchina di Turing il problema sarebbe indecidibile per il teorema di Rice, trattandosi di una proprietà della funzione calcolata dalla macchina.

Esercizio 2
1) L(G1) = $\{(ab)^{2k+1}.b^{2k+1}.b^* \mid k >= 0\}$
2) Con S -> ab S il linguaggio diviene $(ab)^+.b^+$. Una grammatica regolare per esso è la seguente:
S -> a B
B -> b S | b C
C -> b C | b
Una soluzione alternativa è S -> S a. In questo caso il linguaggio diviene abb.{a,b}*, con grammatica
S -> a A
A -> b B
B -> b C | b
C -> a C | b C | a | b

Esercizio 3
No. La funzione
$$f(n)=\frac{n^c}{\log n}$$
soddisfa chiaramente il limite imposto. Tuttavia non esistono 3 costanti positive M, n_0 e ε tali che per, $n>n_0$, risulti sempre $f(n) \leq M\, n^{c-\varepsilon}$, ossia:
$$\frac{n^c}{\log n} \leq M\, n^{c-\varepsilon}, \text{ il che avviene se e solo se}$$
$$\frac{n^\varepsilon}{\log n} \leq M,$$
ma, come noto, n^ε cresce più velocemente di log n per qualunque ε >0, pertanto il loro rapporto non può essere limitato da una costante M.

Esercizio 4
Un algoritmo naturale per raggiungere lo scopo consiste in:
1. Visitare ogni BST "in-order" ossia: sottoalbero sinistro, contenente gli elementi a chiave minore, radice, sottoalbero destro.
2. Quando la suddetta visita incontra una foglia, caratterizzata dal fatto che entrambi i figli sono NIL, la foglia viene memorizzata in una lista concatenata (preferibile ad un array perché le dimensioni dell'albero non sono note a priori e possono essere fortemente variabili).
3. Usando la normale operazione di inserimento in testa di una lista monodirezionale, che si esegue in tempo O(1) si ottengono k liste *ordinate in ordine inverso*.

4. Si procede quindi con un algoritmo di fusione che preleva ad ogni iterazione l'elemento *massimo* tra esse e lo inserisce, in testa, nella lista finale, producendo quindi una lista ordinata in ordine crescente.

In questo modo occorrono:

- Per ogni albero la visita di n_i nodi e, al più, altrettanti inserimenti in lista, quindi n di queste operazioni dove $n = \sum_{i=1}^{k} n_i$.
- k.n scansioni delle k liste (a meno di ottimizzazioni che evitino di riesaminare una lista già svuotata) e relativo inserimento in testa dell'elemento massimo.

Nel caso pessimo perciò l'algoritmo ha complessità temporale O(k.n) e spaziale O(n).