Algoritmi e Principi dell'Informatica
Soluzioni al Tema d'esame

12 luglio 2022

Informatica teorica - Tempo a disposizione: 1 ora

Esercizio 1 (8 punti)

Si costruisca una grammatica o automa o formula logica a potenza minima per il seguente linguaggio:

$$\{a^{2n}\Sigma^+b^{3n} \mid n > 0\} \cup \{\Sigma^+b^{3n} \mid n > 0\},$$

con $\Sigma = \{a, b\}$.

Soluzione

Il linguaggio è aperiodico (o "star-free"), in quanto $\{a^{2n}\Sigma^+b^{3n} \mid n > 0\} \subset \{\Sigma^+b^{3n} \mid n > 0\} = \Sigma^+.b^3$. Si può dunque esprimere con la seguente formula MFO:

$$\exists y(y > 0 \wedge b(y) \wedge b(y+1) \wedge b(y+2) \wedge last(y+2)).$$

Esercizio 2 (8 punti). Chi ha diritto alla riduzione del 30% della prova svolga solo il punto 3.

1. Dire se è decidibile il problema di stabilire se due programmi, uno scritto in C e uno scritto in Java, calcolano la stessa funzione.
2. Dire se è decidibile il problema di stabilire se due programmi, entrambi scritti in C, sono identici, salvo per il fatto che i nomi delle variabili usate sono diversi tra un programma e l'altro.
3. Dire se è decidibile il problema di stabilire se, dato un programma C che fa uso di puntatori, esiste un'esecuzione in cui, ad un certo punto, due variabili puntano entrambe allo stesso indirizzo (fenomeno dell'aliasing).

Soluzione

1. L'equivalenza di programmi C e Java è indecidibile, si può vedere riducendo il problema dell'equivalenza tra due programmi C a quella tra un programma C e un programma Java: ogni programma C può essere trasformato in un equivalente programma Java, e se fosse decidibile l'equivalenza tra programma Java e programma C, allora potremmo anche decidere l'equivalenza tra programmi C, che non è possibile.

2. Il problema è decidibile, in quanto si tratta di leggere i 2 programmi parola per parola. Laddove si trovano dichiarazioni di variabili (che devono essere in corrispondenza una con l'altra), si memorizza una corrispondenza tra la variabile v_1 del programma P_1 e quella (chiamiamola v_2) del programma P_2, e si controlla poi che ogni volta che nel programma P_1 compare la variabile v_1, nel programma P_2 compare v_2. In questo modo, dopo aver letto per intero i programmi P_1 e P_2, se non si sono trovate discrepanze, si conclude che i programmi sono effettivamente identici a meno dei nomi delle variabili, altrimenti no.

3. Il problema non è decidibile, lo si può vedere per riduzione dal problema della terminazione dei programmi C. Più precisamente, si consideri un generico programma P, espresso in C. Nel caso in cui P usi puntatori, è possibile sempre ri-esprimere lo stesso calcolo in C senza usare puntatori: assumiamo dunque che P non ne faccia uso.

   Dato quindi il programma P ed un generico suo input a, possiamo scrivere un nuovo programma P' che prende come input a e, in sequenza

   - dichiara due puntatori, x e y (che non compaiono in P, dato che in P non compare alcun puntatore)
   - assegna loro due valori diversi
   - esegue P su a e, se la computazione termina, assegna x a y, creando quindi un alias tra x e y.

   Chiaramente P termina su input a se, e solo se, in P' ad un certo punto x e y puntano allo stesso indirizzo. Quindi, se il problema desiderato fosse decidibile, lo sarebbe anche quello della terminazione di P su un generico input a, il che è impossibile.

Algoritmi e Principi dell'Informatica
Soluzioni al Tema d'esame

12 luglio 2022

Algoritmi e strutture dati - Tempo a disposizione: 1 ora

Esercizio 3 (8 punti)
Sia dato un albero binario di ricerca, contenente n chiavi intere, dove sono presenti chiavi duplicate.

- Si realizzi la procedura TrueSuccessor che, dato un riferimento ad un nodo x dell'albero restituisce un elemento contenente la più piccola chiave strettamente maggiore di $x.key$.
- Si realizzi la procedura BSTToQueue che, dato un BST con duplicati, lo trasforma in una coda di elementi, senza duplicati, ordinata, dove il primo elemento in coda è l'elemento con chiave minima.

Si fornisca lo pseudocodice di tutte le procedure e se ne valuti la complessità temporale.

Soluzione

La procedura TrueSuccessor(x) si limita ad invocare la procedura Successor(x) fino a quando il nodo successore non ha effettivamente chiave diversa da quello di partenza.

```
TrueSuccessor(x)
1  y ← Successor(x)
2  while y ≠ NIL and y.key = x.key
3        y ← Successor(y)
4  return y
```

La complessità risulta $T(n) \in \mathcal{O}(n)$: nel caso pessimo l'albero è completamente sbilanciato e l'unica chiave è ripetuta in ogni nodo.

La procedura BSTToQueue utilizza TrueSuccessor per scandire il BST in una singola passata, dall'elemento più piccolo al più grande, accodando gli elementi.

```
BSTToQueue(T)
1  if T = NIL
2        return NIL
3  i ← min(T)
4  Enqueue(Q, i)
5  i ← TrueSuccessor(i)
6  while i ≠ NIL
7        Enqueue(Q, i)
8        i ← TrueSuccessor(i)
9  return Q
```

L'attraversamento indotto dall'algoritmo corrisponde ad un attraversamento simmetrico, che visita i nodi in ordine crescente di chiave e richiede tempo $T(n) \in \mathcal{O}(n)$.

Esercizio 4 (8 punti). Chi ha diritto alla riduzione del 30% della prova analizzi unicamente il punto 3.
Si considerino le seguenti ricorrenze e se ne fornisca un limite superiore asintotico

1. $T(n) = 3T(\frac{n}{27}) + \sqrt[2]{n}$
2. $T(n) = 16T(\frac{n}{2}) + n^5$
3. $T(n) = 2T(\frac{n}{2}) + (n\cos(n))^2$

Soluzione

1. Master theorem, caso 3, $n_{crit} = \log_{27}(3) = \frac{1}{3}$. Condizione di regolarità automaticamente soddisfatta in quanto $f(n) = n^k$. $T(n) \in \Theta(\sqrt[2]{n})$.
2. Master theorem, caso 3, $n_{crit} = \log_2(16) = 4$, condizione di regolarità automaticamente soddisfatta in quanto $f(n) = n^k$. $T(n) \in \Theta(n^5)$.
3. Master theorem, caso 3, $n_{crit} = \log_2(2) = 1$, condizione di regolarità $2\left(\frac{n}{2}\cos(\frac{n}{2})\right)^2 \leq k(n\cos(n))^2$ non soddisfacibile. Si osservi che $\cos(n)$ è limitato superiormente da 1 (ovviamente anche per $n \to +\infty$). Abbiamo quindi che $T(n)$ sarà limitato superiormente da $2T(\frac{n}{2})+n^2$, il che si mostra semplicemente per induzione. Applicando il MT a $T'(n) = 2T'(\frac{n}{2}) + 2n^2$ otteniamo che $T'(n) \in \Theta(n^2)$ e quindi $T(n) \in \mathcal{O}(n^2)$.