Algoritmi e Principi dell'Informatica
Appello del 18 Febbraio 2020

Durata: 2 ore e 15 minuti.

Esercizio 1 (7 punti)
Sia L un qualunque linguaggio regolare costruito su un alfabeto A e sia `a` un simbolo di A.
Il linguaggio L' che consiste di tutte le stringhe di L che contengono la `a` è regolare?
Motivare opportunamente la risposta.

Esercizio 2 (8 punti)
È decidibile il problema della soddisfacibilità di una formula MFO? Motivare adeguatamente la risposta.

Esercizio 3 (4 punti per la prima parte e fino a un massimo di 6 punti –in caso di soluzione particolarmente brillante– per la seconda parte)
Indicare la complessità asintotica della seguente funzione (si rammenta che in questo caso si fa riferimento al criterio di costo costante).

```
F(n)
1  sum = 0
2  for (i = 1; i < n; i++)
3      for (j = 1; j < i * i; j++)
4          if (j % i == 0)
5              for (int k = 0; k < j; k++)
6                  sum++
7  return sum
```

È possibile riscrivere la funzione `F(n)` in modo che, a parità di risultato calcolato, si abbassi la complessità?

Esercizio 4 (8 punti)
Si consideri un sistema di assegnazione di pacchi a corrieri presente in un centro smistamento.
Il sistema gestisce pacchi caratterizzati da una data e ora di spedizione (il momento in cui il cliente ha consegnato il pacco da spedire) e da un livello di urgenza rappresentato da un intero positivo compreso tra 1 (meno urgente) e 10 (più urgente) e da un identificativo numerico unico.
I pacchi sono acquisiti dal sistema già contrassegnati da una data/ora di spedizione e da un livello di urgenza assegnato.

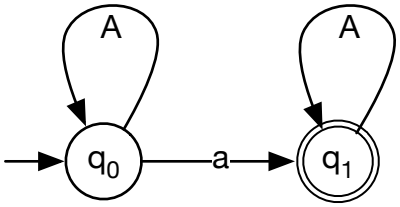
Il sistema di smistamento deve selezionare quali pacchi devono essere assegnati ad un corriere nel modo quanto più efficiente possibile considerando i seguenti vincoli:

- Nessun pacco con urgenza x deve essere assegnato ad un corriere se ne esiste un altro nel sistema con urgenza maggiore di x
- Nel caso siano presenti pacchi allo stesso livello di urgenza, viene deciso di assegnare prima al corriere il pacco con data di spedizione più vecchia tra quelli con lo stesso livello di urgenza tra pacchi
- Si assuma, per semplicità, che non vi siano due pacchi con data e ora di spedizione identiche
- Il sistema di gestione può acquisire un numero arbitrario di pacchi tra una decisione di assegnamento di un pacco al corriere e la successiva decisione (e, analogamente, può effettuare un numero arbitrario di decisioni di assegnamento tra l'acquisizione di un pacco e la successiva acquisizione).

Si descrivano le strutture dati necessarie e le procedure `prossimo_pacco()` e `acquisisci_pacco(identificativo, data, urgenza)`, e si valuti la loro complessità asintotica in funzione del numero di pacchi attualmente in carico al sistema di gestione del magazzino.

Tracce delle soluzioni

Esercizio 1
Sia L'' il linguaggio di tutte le stringhe contenenti la `a`. Tale linguaggio è chiaramente regolare, essendo ad esempio riconosciuto dal seguente automa nondeterministico a stati finiti (l'etichetta "A" indica un qualunque simbolo di A, incluso `a`).



Il linguaggio L' si può ottenere facendo l'intersezione di L e L'', che sono entrambi regolari.
Poiché la famiglia dei linguaggi regolari è chiusa rispetto all'intersezione, anche L' è regolare.
ATTENZIONE: affermare *un sottoinsieme di un linguaggio regolare è regolare* è profondamente errato: se ne potrebbe dedurre che ogni linguaggio è regolare, affermazione chiaramente falsa.

Esercizio 2
La soddisfacibilità di una formula MFO è decidibile: infatti questa logica definisce una sottofamiglia dei linguaggi regolari ed esistono algoritmi che, data una formula MFO, costruiscono un FSA ad essa equivalente, ossia che riconosce l'insieme delle stringhe che soddisfano la formula. Quindi una formula F è soddisfacibile se e solo se un FSA ad essa equivalente riconosce un linguaggio non vuoto, problema notoriamente decidibile, grazie al pumping lemma.

Esercizio 3
La complessità è O(n^4). Infatti, il ciclo più esterno esegue O(n) iterazioni. Il secondo ciclo ne esegue O(i^2), ossia O(n^2) per ogni iterazione del primo ciclo. Il terzo ne esegue O(j) ossia O(i^2), ma viene eseguito solo ogni i iterazioni del secondo ciclo, quindi complessivamente esegue O(i) (ossia O(n)) iterazioni per ogni iterazione del secondo ciclo. L'iterazione interna ha una complessità di O(1). Complessivamente abbiamo O(n)·O(n^2)·O(n) = O(n^4).

La parte ciclica della funzione si può riscrivere come segue, abbassando la complessità a O(n^2).
```
for (i = 1; i < n; i++)
    for (j = i; j < i * i; j+=i)
        sum += j
```

Una riscrittura equivalente, ottenuta raccogliendo il fattore `i` dal secondo ciclo, è la seguente:
```
for (i = 1; i < n; i++)
    for (k = 1; k < i; k++)
        sum += k*i
```

Dalla formula di Gauss, secondo cui $\sum_{k=1}^{i-1} k = \frac{i(i-1)}{2}$, evinciamo che $\sum_{k=1}^{i-1} ki = \frac{i(i-1)}{2} i$. Questo consente di sostituire il ciclo interno con una formula esplicita. La riscrittura seguente porta a una complessità di O(n).
```
for (i = 1; i < n; i++)
    sum += i*(i-1)/2*i
```

Si può ricavare una semplificazione estrema mediante le formule per la somma di cubi e per la somma di quadrati, ottenendo un'unica formula (polinomio di quarto grado in n) che dà direttamente il risultato. La complessità risultante è O(1).
```
return (n-2)*(n-1)*n*(3*n-1)/24
```

Esercizio 4
La struttura dati necessaria è un vettore di min-heap avente tante celle quanti sono i livelli di priorità (10). Ogni min-heap contiene i pacchi, utilizzando come valore su cui operare i confronti la data di acquisizione del pacco stesso e considerando l'ordine cronologico tra le date. La procedura `prossimo_pacco` scorre il vettore di

heap in ordine decrescente di priorità fino a quando non ne individua uno non vuoto. Quando è stato trovato uno heap non vuoto, estrae il minimo da esso e riorganizza il min-heap in tempo O(log(*n*)) con *n* numero dei pacchi nello heap.
La procedura `acquisisci_pacco` invece inserisce il pacco da acquisire nello heap corrispondente alla priorità dello stesso; anche essa ha costo O(log(*n*)) con *n* numero dei pacchi nello heap; nel caso pessimo il numero di pacchi in un singolo heap è proporzionale al totale dei pacchi, quindi la complessità di entrambe le procedure è O(log(*n*)) anche in funzione del numero totale di pacchi.