Fondamenti di Informatica, A.A. 2012-2013

06/09/2013

Esercizio 1

È dato il frammento di codice Matlab

```
v = [1 3 4 2 5 2];
n = length(v);
for i=1:n
  x = v(1:i).^2;
  if (any(x>20))
    break;
  end
end
disp(x);
```

Si chiede cosa viene visualizzato dall'interprete Matlab. Giustificare la risposta.

Soluzione

Il codice riempie il vettore `x` con porzioni progressivamente più grandi del quadrato, elemento per elemento, del vettore `v`; tuttavia arrivato a dimensione 5 si arresta in quanto il quadrato del quinto elemento vale $25 > 20$ e quindi la condizione di uscita anticipata dal ciclo viene soddisfatta. L'interprete Matlab stampa

```
1     9     16     4     25
```

Esercizio 2

Discutere qualitativamente delle classi di complessità di esecuzione, in particolare degli algoritmi con complessità esponenziale ovvero polinomiale. Riportare inoltre un esempio per ciascuna delle due categorie di algoritmi.

Soluzione

Un algoritmo si dice di complessità $O(f(n))$ se il suo tempo di esecuzione cresce con la dimensione del problema rimanendo asintoticamente limitato da un multiplo di $f(n)$, ossia $T(n) \leq cf(n)$ per una qualche costante c.

Il tempo d'esecuzione di un algoritmo esponenziale aumenta esponenzialmente all'aumentare dell'input che deve elaborare; pertanto ad un aumento della velocità di un elaboratore corrisponde, a parità di tempo, un aumento di una quantità fissa della dimensione del problema risolubile nel tempo dato. Ad esempio, per un problema di complessità $O(2^n)$ ad un raddoppio della velocità corrisponde l'aumento di 1 unità della dimensione del problema risolubile. Un esempio tipico di algoritmo esponenziale è l'algoritmo ricorsivo per il calcolo dei numeri di Fibonacci. Gli algoritmi polinomiali hanno complessità polinomiale relativamente alla dimensione dell'input che devono elaborar, e sono considerati trattabili.

Un tipico esempio di algoritmo polinomiale è quello della ricerca sequenziale; tale algoritmo ha infatti complessità lineare $O(n)$. Per un algoritmo lineare ad un raddoppio della velocità di calcolo corrisponde un raddoppio della massima dimensione risolubile; più in generale per un algoritmo $O(n^k)$, ad un raddoppio della velocità corrisponde un aumento delle dimensioni del problema a parità di tempo di un fattore $2^{\frac{1}{k}}$. Un algoritmo polinomiale è asintoticamente, ossia per input di grandi dimensioni, da preferire ad uno esponenziale, ed un algoritmo con un grado k più basso ad uno di grado più alto; tuttavia ciò è vero in senso asintotico, mentre per input di piccole dimensioni la scelta migliore dipende dalle costanti c e potrebbe non coincidere con quella asintotica.

Esercizio 3

In una prova di esame di informatica viene richiesto di realizzare un programma che simuli un distributore automatico di bevande. Uno studente propone la seguente soluzione:

```
beverages = [0.1 1 0.8 0.5];
avail = [3 5 1 8];

importo = input('Inserisci importo ');

if(importo < min(beverages))
    disp('Importo insufficiente')
else
    sel = input('Seleziona id bevanda ');
    if(beverages(sel)>importo | (sel>length(avail) || sel<1))
        disp('Bevanda non selezionabile')
    else
        if(avail(sel)>0)
            avail(sel)=avail(sel)-1;
            disp('Erogazione in corso')
        else
            disp('Non disponibile')
        end
    end
end
```

Si chiede di trovare l'errore presente nel codice e di proporre una correzione adeguata.

Soluzione

Il problema è nella condizione del blocco `if` che verifica la correttezza dell'id della bevanda appena inserito, in quanto si usa l'operatore `|` singolo invece che doppio `||`, ed inoltre il controllo viene fatto in un ordine che non garantisce un accesso corretto all'array `beverages`. Una possibile soluzione è quindi la seguente:

```
beverages = [0.1 1 0.8 0.5];
avail = [3 5 1 8];

importo = input('Inserisci importo ');
```

```
if(importo < min(beverages))
    disp('Importo insufficiente')
else
    sel = input('Seleziona id bevanda ');
    if((sel>length(avail) || sel<1) || beverages(sel)>importo)
        disp('Bevanda non selezionabile')
    else
        if(avail(sel)>0)
            avail(sel)=avail(sel)-1;
            disp('Erogazione in corso')
        else
            disp('Non disponibile')
        end
    end
end
```

Esercizio 4

È data la seguente funzione Matlab

```
function [x]=mistery(a,b,k,p)
  for j=1:p
    c = a*a';
    for i=1:k
      a(i) = a(i)+c*b(i);
    end
  end
  x = a;
end
```

Si stimi il numero di operazioni aritmetiche eseguite assumendo che `a` e `b` siano vettori riga di lunghezza `n` e che `k` e `p` siano interi minori o uguali a `n`.

Soluzione

L'array `a` moltiplicato per se stesso in colonna produce uno scalare. Tale operazione richiede $2n$ operazioni aritmetiche; il ciclo su `k` comporta due operazioni aritmetiche per iterazione, quindi nel ciclo più esterno si avranno $2(n+k)$ operazioni aritmetiche, ripetute p volte, quindi in totale avremo $2p(n+k)$ operazioni. Nell'ipotesi data $k, p \leq n$ si ha quindi una complessità non superiore a $O(n^2)$.

Esercizio 5

È dato il frammento di codice Matlab

```
vect = [0.2  0.3  0.5];

for i=1:10
  r = (vect*vect')^2;
  vect = vect-r;
  s = vect < 0;
  if(sum(s) > 0)
      vect(s) = 0.1;
      break;
  end
end
```

Riscrivere il codice facendo uso di uno o piu' cicli `while`.

Soluzione

Una possibile soluzione è la seguente:

```
vect = [0.2  0.3  0.5];
s = vect < 0;
```

```
i=1;
while((sum(s)<=0)&&(i<=10))
   r = (vect*vect')^2;
   vect=vect-r;
   s = vect < 0;
   if(sum(s) > 0)
      vect(s) = 0.1;
   end
   i = i + 1;
end
```