Fondamenti di Informatica, A.A. 2013-2014 - FILA A

28/07/2014

Esercizio 1

È dato il frammento di codice Matlab

```
a = [16 32 64 128 256 512 1024 2048];
b = [8 16 24 32 40 48 56 64];

x = find(a>100);
y = b(end:-2:1);

res = [a(2:length(a)-2) y x(1:end-1)];

n = length(res)

disp(res(1:n-1))
```

Si chiede cosa viene visualizzato dall'interprete Matlab.

Soluzione

Ricordando che `find(a>100)` restituisce gli indici per i quali l'espressione è vera, in questo caso `4 5 6 7 8`, il risultato stampato è

```
octave:7>     disp(res(1:n-1))
    32     64    128    256    512     64     48     32     16      4      5
6
```

Esercizio 2

Dato il seguente algoritmo se ne stimi la complessità.

```
function v = myfunction( v )
 for i=2:length(v)
     x=v(i);
     for j=1:i
         if v(j)>x
             break;
         end
     end
     if j < i
         for l=i-1:-1:j
             v(l+1)=v(l);
         end
         v(j)=x;
     end
end
```

Soluzione

Il codice propone una variante dell'algoritmo di Insertion Sort. Il ciclo più esterno contiene due cicli; il primo ciclo interno viene eseguito un numero di volte `j` che non è prevedibile a priori (in quanto a seconda dei valori in `v` viene eseguita l'istruzione `break`), ma nel caso peggiore vale `j=i`. Il secondo ciclo interno viene eseguito un numero di volte pari a `i-j`. Se *non* contiamo le operazioni di spostamento dati nel secondo ciclo interno, allora nel caso peggiore il costo complessivo vale

$$\sum_{i=2}^{n} i = \left(\sum_{i=1}^{n} i\right) - 1 = \frac{n(n+1)}{2} - 1 = O(n^2)$$

Se invece consideriamo nel conteggio le operazioni di spostamento dati, allora possiamo valutare che si abbia *sempre* (e non solo nel caso peggiore)

$$\sum_{i=2}^{n} j + (i-j) = \sum_{i=2}^{n} i = \left(\sum_{i=1}^{n} i\right) - 1 = \frac{n(n+1)}{2} - 1 = O(n^2)$$

Esercizio 3

Il seguente codice viene proposto per effettuare la ricerca binaria di un elemento in un vettore. Si identifichi l'errore contenuto e si proponga una soluzione adeguata.

```
function pos=bseach(key,v)

n=length(v);
l=1;
u=n
pos=-1;
while (l<=u)
    m = floor((u+l)/2);
    if (key==v(m))
        pos=m;
        return;
    else if (key <v(m))
        u=m;
    else
        l=m;
    end
end
```

Soluzione

Il problema sta nel fatto che negli aggiornamenti degli estremi non viene eliminato l'elemento nella posizione `m`. Questo può dar luogo alla non terminazione dell'algoritmo; infatti se abbiamo $u = l + 1$ il punto medio vale $m = l$ per via dell'arrotondamento all'intero più basso tramite la funzione `floor`, e se vale $v(l) < key < v(u)$, i valori di `l` e `u` non vengono mai più aggiornati. La soluzione consiste nel porrre

```
else if (key <v(m))
  u=m-1;
else
  l=m+1;
end
```

Esercizio 4

È data la seguente funzione Matlab/Octave

```
function [x n]=simple(a,n)
  for i=1:n
    d = min(a);
    d = d.*2;
    a = a+ sum(a*d');
 end
```

Si stimi il numero di operazioni aritmetiche eseguite, assumendo che a sia una matrice $n \times n$ e che n sia un intero positivo.

Soluzione

Il ciclo principale della funzione si compone di tre parti che non dipendono direttamente dal'indice `i`. La prima istruzione Matlab/Octave calcola il minimo della matrice per colonne, producendo un vettore riga; per calcolare il minimo vengono eseguiti n^2 confronti. Nella seconda istruzione il vettore riga viene elevato al quadrato elemento per elemento,per altre n operazioni. Infine, nella terza istruzione, viene effettuato un prodotto della matrice `a` per il trasposto del vettore `d`, al costo di $2n^2$ operazioni, seguito dalla somma del vettore risultante con n operazioni e la somma dello scalare risultante con la matrice `a` per altre n^2 addizioni. Considerando che il ciclo viene eseguito n volte, in totale avremo

$$opcnt = n(n^2 + n + 2n^2 + n + n^2) = n(4n^2 + 2n) = O(n^3)$$

Esercizio 5

È dato il frammento di codice Matlab

```
sel = input('Write ft to convert meters to feet or mt to convert feet to meter
dist = input('Insert the value\n');
if strcmp(sel,'ft') || strcmp(sel,'FT')
        dist = dist*3.2808;
        str = [num2str(dist) ' ft'];
        disp(str);
elseif strcmp(sel,'mt') || strcmp(sel,'MT')
        dist = dist/3.2808;
        str = [num2str(dist) ' m'];
        disp(str);
else
```

```
        disp('Error');
end
```

Riscrivere il codice facendo uso del costrutto `switch`.

Soluzione

```
sel = input('Write ft to convert meters to feet or mt to convert feet to meter
dist = input('Insert the value\n');
switch(sel)
case  {'ft','FT'}
        dist = dist*3.2808;
        str = [num2str(dist) ' ft'];
        disp(str);
case {'mt','MT'}
        dist = dist/3.2808;
        str = [num2str(dist) ' m'];
        disp(str);
otherwise
        disp('Error');
end
```

Fondamenti di Informatica, A.A. 2013-2014 - FILA B

28/07/2014

Esercizio 1

È dato il frammento di codice Matlab

```
v = [1 2 3 4];

for x=v
  v = [sum(v) x];
  k = max(size(v))-1;
  v(k) = v(1);
  v = v(2:end);
end

disp(v)
```

Si chiede cosa viene visualizzato dall'interprete Matlab.

Soluzione

L'unica difficoltà nella soluzione del quesito sta nel ricordare che la semantica di Matlab/Octave per il ciclo `for` fa assumere in successione alla variabile `x` i valori `1, 2, 3, 4` che il vettore `v` conteneva all'inizio, anche se il vettore stesso viene alterato durante il ciclo. In particolare, dopo la prima istruzione `v = [sum(v) x];` il vettore `v` contiene solo due elementi, specificamente `v=[10, 1]`, e dopo `v=v(2:end)` ne rimane solo uno `v=[1]`; alla iterazione successiva il vettore diventa prima `v=[1,2]` e poi `v=[2]` e così via. Al termine della esecuzione l'interprete stampa

```
octave:13>    disp(v)
 4
```

Esercizio 2

Si dia la definizione di complessità computazionale, dando almeno un esempio di un algoritmo avente complessità $n\log(n)$.

Soluzione

Per complessità computazionale si intende il numero di operazioni necessario affinché un algoritmo produca la soluzione ad una istanza di un determinato problema, in funzione della dimensione dei dati del problema stesso. La complessità viene solitamente intesa in senso asintotico, a meno di una costante moltiplicativa; pertanto dire che un algoritmo è $O(f(n))$ vuol dire che per n abbastanza grande si ha $T(n) \leq Cf(n)$ per una qualche costante C. Due esempi di algoritmi di complessità $O(n\log(n))$ sono la trasformata rapida di Fourier (FFT) e l'algoritmo di *merge-sort*.

Esercizio 3

Viene proposto il seguente codice per il calcolo del valore di un polinomio in un punto `x` dati i valori dei suoi coefficienti in un array `cf`. Si identifichi l'errore e si proponga una correzione adeguata.

```
function y=polyeval(cf,x)

  n=length(cf);
  y=0; v=x;
  for i=1:n
    y=y+cf(i).*v;
    v=v.*x;
  end
```

Soluzione

Il codice si propone di costruire iterativamente le potenze della variabile `x`, per poi moltiplicarle per l'opportuno coefficiente contenuto nel vettore `cf`. L'errore sta nel fatto che le potenze vengono fatte partire da x^1 invece che da $x^0 = 1$, per cui il codice proposto valuta in effetti $y = x * p(x)$. È sufficiente correggere la inizializzazione `v=1` al posto di `v=x`.

Esercizio 4

È data la seguente funzione Matlab

```
function [x]=mistery(m,y,k)
  for i=1:k
      y=y+sum(min(m))
      m = m*min(m)' + diag(m);
  end
end
```

Si stimi il numero di operazioni aritmetiche eseguito dal codice, considerando che m sia una matrice quadrata $n \times n$, che y sia uno scalare positivo.

Soluzione

Il codice contiene un ciclo che dovrebbe essere eseguito k volte; le istruzioni nel ciclo non dipendono direttamente dall'indice `i`. Nella prima istruzione si calcola il vettore che contiene i minimi nelle colonne di `m` al costo di n^2 confronti, e poi se ne sommano gli elementi con n addizioni; infine la somma con lo scalare `y` comporta una ulteriore addizione. La matrice viene poi moltiplicata per lo stesso vettore di `min(m)` al costo di $2n^2$ operazioni, e gli viene aggiuta la diagonale, risultando in un vettore al costo di altre n operazioni. Alla iterazione successiva, la seconda operazione darà errore di conformità in quanto avremo `m*min(m)'` che darà un vettore, e `diag(m)` che darà una matrice. In definitiva verranno eseguite solo $3n^2 + 2n$ operazioni.

Esercizio 5

```
sel = input('Write hour to convert days to hours or day to convert hours to da
val = input('Insert the value\n');
if strcmp(sel,'day') || strcmp(sel,'DAY')
        val = val/24;
        str = [num2str(val) ' days'];
        disp(str);
elseif strcmp(sel,'hours') || strcmp(sel,'HOURS')
        val = val*24;
        str = [num2str(val) ' hours'];
        disp(str);
else
        disp('Error');
 end
```

Riscrivere il codice facendo uso di uno del costrutto `switch`.

Soluzione

```
sel = input('Write hour to convert days to hours or day to convert hours to da
val = input('Insert the value\n');
swicth(sel)
case {'day','DAY'}
        val = val/24;
        str = [num2str(val) ' days'];
        disp(str);
case {'hours','HOURS'}
        val = val*24;
        str = [num2str(val) ' hours'];
        disp(str);
otherwise
        disp('Error');
end
```