Fondamenti di Informatica, A.A. 2014-2015 - FILA A

08/07/2015

Esercizio 1

È dato il frammento di codice Matlab

```
v = [1 1 2 3];
n = length(v);
for x = v(end:-2:1)
   v = [v mod(sum(v),n)];
   n = length(v);
   k = n-x;
   v(k) = v(end);
end

disp(v)
```

Descrivere in dettaglio le operazioni effettuate dall'interprete Matlab/Octave e cosa viene visualizzato.

Soluzione

L'espressione `v(end:-2:1)` prima dell'inizio del ciclo ha il valore `[3 1]`; pertanto la variabile del ciclo `x` prende i valori 3 ed 1, in successione, e questo anche se `v` viene alterato nel ciclo (per via della semantica della istruzione `for`). All'inizio vale $n = 4$ per cui avremo $\mod(7,4) = 3$ che viene aggiunto al vettore; poi avremo $n = 5$ e $k = n - x = 2$, per cui $v(2) = v(5) = 3$. Alla seconda iterazione, la somma di v vale 12, $n = 5$ per cui $\mod(sum(v), n) = 2$, che viene aggiunto al vettore; n diventa 6, $n - x = 5$, e $v(5) = v(6) = 2$.

L'interprete Matlab stampa quindi

```
1     3     2     3     2     2
```

Esercizio 2

Descrivere le funzionalità di I/O in Matlab/Octave.

Soluzione

Matlab fornisce varie funzionalità di I/O. Esempi di funzioni di Input sono:

- `x=input(prompt)` or `x=input(prompt,'s')` visualizzano a video la stringa `prompt` ed assegnano alla variabile `x` un dato che verrà inserito da tastiera. Nel primo caso il dato sarà numerico o convertito in dato numerico. Nel secondo caso si da disposizione di interpretare il dato inserito come una stringa.
- `A = fread(fileID)` legge il contenuto del file identificato dal descrittore `fileID`. Tale descrittore si ottiene come output della funzione `fileID=fopen(filename)` che apre il file `filename`

- `num = xlsread(filename)` apre il file `filename` in formato `.xls` o `.xlsx` (Excel) e ne memorizza il contenuto nella variabile num.

Esempi di funzioni di output sono:

- `disp(x)` che visulizza il contenuto della variabile `X`
- `fprintf(formatSpec,A1,...,An)` applica la stringa di formattazione `formatSpec` a tutte le variabili `A1,...,An` e li visualizza sullo schermo
- `fprintf(fileID, formatSpec,A1,...,An)` applica la stringa di formattazione `formatSpec` a tutte le variabili `A1,...,An` e le scrive in un file di testo identificato dal descrittore `fileID` e precedentemento aperto con una `fopen`.
- `fwrite(fileID,A)` scrive il contenuto di A come interi senza segno a 8-bit in un file binario identificato da `fileID` e precedentemente aperto con `fopen`. Finita la scrittura il file va chiuso con la `fclose`
- `xlswrite(filename,A)` scrive la variabile `A` nel file excel `filename` a partire dalla cella A1.

Esercizio 3

In una prova di informatica viene chiesto di realizzare una funzione ricorsiva che conti le occorrenze di uno specifico numero intero x in un vettore v. Viene proposta la seguente soluzione:

```
function res = occ(v,x)
  res = 0;
  if(length(v) == 1)
    if(v(1) == x)
      res = 1;
      return;
    end
  end
  res = occ(v(1:floor(length(v)./2)),x) + occ(v(floor(length(v)./2)+1:end),x);
end
```

Si chiede di identificare gli eventuali errori presenti e di proporre una versione corretta del codice.

Soluzione

Il codice proposto presenta due problemi:

- Nel caso base si dovrebbe arrestare la esecuzione sia che $v(1) == x$ sia altrimenti, invece nel codice l'istruzione `return` è eseguita solo nel caso affermativo. Questo comporta che ci potrebbe essere una successiva chiamata ricorsiva che viene effettuata con una stringa di dimensione 0;
- Non viene considerato (appunto) il caso di stringa di dimensione 0.

Una possibile versione corretta del codice è:

```
function res = occ(v,x)
  res=0
  if (length(v) == 0)
     return;
  elseif (length(v) == 1)
     res = (v(1) == x);
  else
     res = occ(v(1:floor(length(v)./2)),x) + occ(v(floor(length(v)./2)+1:end),
  end
end
```

Esercizio 4

È data la seguente funzione Matlab

```
function [y]=mistery(m,a,k)
  i=1; j=1;
  y=0;
  while (j <= 4*k)
       y = y + m*a;
       j = j + 2*i + 1;
       i = i+1;
  end
end
```

Si stimi il numero di operazioni aritmetiche eseguito dal codice, assumendo che m e a siano matrici quadrate $n \times n$, e k un intero positivo.

Soluzione

Il ciclo `while` contiene una operazione su array $y = y + m * a$ e due operazioni su scalari; per n grande il costo delle operazioni scalari sarà trascurabile. In

$$y = y + m * a$$

abbiamo una moltiplicazione tra due matrici m ed a di dimensione $n \times n$, per un costo di $2n^3$ operazioni; il risultato viene poi sommato ad y, termine a termine, per un costo di n^2 operazioni. Il costo è indipendente dalla iterazione i, pertanto dobbiamo solo capire quante iterazioni verranno eseguite.

Al primo passo abbiamo $i = 1$ e $j = 1$; al secondo passo abbiamo $i = 2$ e $j = 4$; al terzo $i = 3$ e $j = 9$. In effetti abbiamo sempre $j = i^2$; infatti, la affermazione è vera per $i = 1$; inoltre se $j_i = i^2$, abbiamo che $j_{i+1} = j + 2i + 1 = i^2 + 2i + 1 = (i + 1)^2$ e quindi è vera sempre. Pertanto il ciclo si arresterà quando

$$j > 4k \Rightarrow i^2 > 4k \Rightarrow i = \lceil 2\sqrt{k} \rceil.$$

La complessità sarà pertanto

$$opcnt = 2\sqrt{k}\left(2n^3 + n^2\right) = O(n^3\sqrt{k})$$

Esercizio 5

È dato il frammento di codice Matlab

```
n = 8; m = 100; v = [];

for i = 100:-2:n
  v = [v, i./10];
  for j = 1:4:m
    for k = 1:length(v)
      v(k) = v(k) + 4;
    end
  end
end
```

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

Soluzione

Una possibile soluzione è

```
n = 8; m = 100; v = [];

i=100;
while (i >= n)
  v = [v, i./10];
  j=1;
  while (j <= m)
    k = 1;
    while (k <= length(v))
      v(k) = v(k) + 4;
      k=k+1;
    end
    j = j + 4;
  end
  i = i - 2;
end
```

Fondamenti di Informatica, A.A. 2014-2015 - FILA B

08/07/2015

Esercizio 1

È dato il frammento di codice Matlab

```
v = [1 1 2 3];
n = length(v);
for x = v(1:2:end)
  v = [v, mod(sum(v),n)];
  n = length(v);
  k = n-x;
  v(k) = v(end);
end

disp(v)
```

Descrivere in dettaglio le operazioni effettuate dall'interprete Matlab/Octave e cosa viene visualizzato.

Soluzione

L'espressione `v(1:2:end)` prima dell'inizio del ciclo ha il valore `[1 2]`; pertanto la variabile del ciclo `x` prende i valori 1 e 2, in successione, e questo anche se `v` viene alterato nel ciclo (per via della semantica della istruzione `for`). All'inizio vale $n = 4$ per cui avremo $\mathrm{mod}\,(7, 4) = 3$ che viene aggiunto al vettore; poi avremo $n = 5$ e $k = n - x = 4$, per cui $v(4) = v(5) = 3$. Alla seconda iterazione, la somma di v vale 10, $n = 5$ per cui $\mathrm{mod}\,(sum(v), n) = 0$, che viene aggiunto al vettore; n diventa 6, $n - x = 4$, e $v(4) = v(6) = 0$.

L'interprete Matlab stampa quindi

1 1 2 0 3 0

Esercizio 2

Descrivere le funzionalità di I/O in Matlab/Octave.

Soluzione

Matlab fornisce varie funzionalità di I/O. Esempi di funzioni di Input sono:

- `x=input(prompt)` or `x=input(prompt,'s')` visualizzano a video la stringa `prompt` ed assegnano alla variabile `x` un dato che verrà inserito da tastiera. Nel primo caso il dato sarà numerico o convertito in dato numerico. Nel secondo caso si da disposizione di interpretare il dato inserito come una stringa.
- `A = fread(fileID)` legge il contenuto del file identificato dal descrittore `fileID`. Tale descrittore si ottiene come output della funzione `fileID=fopen(filename)` che apre il file `filename`

- `num = xlsread(filename)` apre il file `filename` in formato `.xls` o `.xlsx` (Excel) e ne memorizza il contenuto nella variabile num.

Esempi di funzioni di output sono:

- `disp(x)` che visulizza il contenuto della variabile `X`
- `fprintf(formatSpec,A1,...,An)` applica la stringa di formattazione `formatSpec` a tutte le variabili `A1,...,An` e li visualizza sullo schermo
- `fprintf(fileID, formatSpec,A1,...,An)` applica la stringa di formattazione `formatSpec` a tutte le variabili `A1,...,An` e le scrive in un file di testo identificato dal descrittore `fileID` e precedentemento aperto con una `fopen`.
- `fwrite(fileID,A)` scrive il contenuto di A come interi senza segno a 8-bit in un file binario identificato da `fileID` e precedentemente aperto con `fopen`. Finita la scrittura il file va chiuso con la `fclose`
- `xlswrite(filename,A)` scrive la variabile `A` nel file excel `filename` a partire dalla cella A1.

Esercizio 3

In una prova di informatica viene chiesto di realizzare una funzione ricorsiva che verifichi se una stringa è palindroma (ossia, se è eguale alla propria immagine speculare). Viene proposto il seguente codice:

```
function res = palindrome(x)
  res = 0;
  if(length(x) == 1)
    res = 1;
  end
  res = (x(1)==x(end)) || palindrome(x(2:end-1)) ;
end
```

Si chiede di identificare gli eventuali errori presenti e di proporre una versione corretta del codice.

Soluzione

Il codice proposto presenta due problemi:

- Nel caso base si deve considerare che una stringa è palindroma non solo per lunghezza 1 ma anche per lunghezza 0; infatti la stringa iniziale potrebe avere lunghezza sia pari che dispari;
- La condizione realizzata con OR `||` dovrebbe in realtà essere un AND `&&`

Una possibile versione corretta del codice è dunque:

```
function res = palindrome(x)
  res=0
  if (length(x) <= 1)
     res = 1;
  else
     res = (x(1)==x(end)) && palindrome(x(2:end-1)) ;
  end
end
```

È inoltre possibile riorganizzare il codice in modo da avere una funzione *tail recursive*:

```
function res = palindrome(x)
  res=0
  if (length(x) <= 1)
     res = 1;
  else
     if (x(1)==x(end))
       res =  palindrome(x(2:end-1)) ;
     end
  end
end
```

Esercizio 4

È data la seguente funzione Matlab

```
function [y]=mistery(m,x,k)
  i=1; j=1;
  y=0;
  while (j <= 4*k)
       y = y + m*x;
       j = j + 2*i + 1;
       i = i+1;
  end
end
```

Si stimi il numero di operazioni aritmetiche eseguito dal codice, assumendo che m sia una matrice quadrata $n \times n$, x un vettore $n \times 1$ e k un intero positivo.

Soluzione

Il ciclo `while` contiene una operazione su array $y = y + m * x$ e due operazioni su scalari; per n grande il costo delle operazioni scalari sarà trascurabile. In

$$y = y + m * x$$

abbiamo una moltiplicazione tra una matrice m di dimensione $n \times n$ ed un vettore x di dimensione n, per un costo di $2n^2$ operazioni; il risultato viene poi sommato ad y, termine a termine, per un costo di n operazioni. Il costo è indipendente dalla iterazione i, pertanto dobbiamo solo capire quante iterazioni verranno eseguite.

Al primo passo abbiamo $i = 1$ e $j = 1$; al secondo passo abbiamo $i = 2$ e $j = 4$; al terzo $i = 3$ e $j = 9$. In effetti abbiamo sempre $j = i^2$; infatti, la affermazione è vera per $i = 1$; inoltre se $j_i = i^2$, abbiamo che $j_{i+1} = j + 2i + 1 = i^2 + 2i + 1 = (i + 1)^2$ e quindi è vera sempre. Pertanto il ciclo si arresterà quando

$$j > 4k \Rightarrow i^2 > 4k \Rightarrow i = \lceil 2\sqrt{k} \rceil.$$

La complessità sarà pertanto

$$opcnt = 2\sqrt{k}\left(2n^2 + n\right) = O(n^2\sqrt{k})$$

Esercizio 5

È dato il frammento di codice Matlab

```
n = 40; m = 100; v = [];

for i = 1:2:m
  v = [v, i./10];
  for j = n:-4:1
    for k = 1:length(v)
      v(k) = v(k) + 4;
    end
  end
end
```

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

Soluzione

Una possibile soluzione è

```
n = 8; m = 100; v = [];

i=1;
while (i <= m)
  v = [v, i./10];
  j=n;
  while (j >= 1)
    k = 1;
    while (k <= length(v))
      v(k) = v(k) + 4;
      k=k+1;
    end
    j = j - 4;
  end
  i = i + 2;
end
```