Fondamenti di Informatica, A.A. 2012-2013

10/02/2014

Esercizio 1

È dato il frammento di codice Matlab

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

n = 3;

d = 0;

for  i=1:2:length(v)
   v = v+n+1
   d = d + diag(v)
end

v = diag(d)'+v;

disp(v)
```

Si chiede cosa viene visualizzato dall'interprete Matlab.

Soluzione

Viene stampato il vettore

```
23     26     29     32
```

Esercizio 2

Discutere della rappresentazione di un numero reale in un calcolatore elettronico odierno e delle sue proprietà principali. Descrivere chiaramente la differenza tra singola e doppia precisione; quale viene utilizzata in Octave/Matlab?

Soluzione

Su calcolatore si utilizzano i numeri floating-point: sono un sottoinsieme dei numeri reali (razionali) della forma (s, e, f) dove s è il segno, e è l'esponente e f è la parte frazionaria, o mantissa:

$$(s, e, f) = s \times f \times \beta^{e-t} = \pm\beta^e \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \dots \frac{d_t}{\beta^t} \right)$$

ed f è rappresentato su t cifre

$$.d_1 d_2 \dots d_t$$

L'idea dei numeri floating-point è mutuata dalla notazione scientifica, ad esempio $x = 2300 = 2.3 \times 10^3 = 0.23 \times 10^4$. Si assume normalmente che in numeri siano normalizzati:

$m \geq \beta^{t-1}$, ovvero $d_1 \neq 0$; all'insieme dei numeri normalizzati si aggiunge esplicitamente lo 0.

Un numero reale a non è in generale rappresentabile esattamente. Se $\mathrm{fl}(a)$ è il numero in virgola mobile che meglio approssima il numero reale a, si definisce la precisione di macchina tramite la espressione

$$\mathrm{fl}(a) = a(1+\epsilon), \qquad |\epsilon| \leq u,$$

dove la quantità

$$u = \frac{1}{2}\beta^{1-t}$$

è detta arrotondamento unitario. In Octave/Matlab si usa l'aritmetica IEEE 754:

Singola precisione $\beta = 2$, $t = 23 + 1$ (bit "fantasma" se normalizzato 1, se no 0), $e_{min} = -127$, $e_{max} = 127$, $u = 6 \times 10^{-8}$, range $10^{\pm 38}$

Doppia precisione $\beta = 2$, $t = 52 + 1$ (bit "fantasma" se normalizzato 1, se no 0), $e_{min} = -1023$, $e_{max} = 1023$, $u = 1 \times 10^{-16}$, range $10^{\pm 308}$

in base 2, un numero normalizzato necessariamente inizia per 1 quindi la prima cifra può essere sottintesa.

Le operazioni IEEE verificano

$$u \oplus v = \mathrm{round}(u + v)$$

ossia il risultato della singola operazione in virgola mobile è eguale all'arrotondamento del risultato esatto. Per le operazioni in virgola mobile, a causa dell'arrotondamento, cadono le proprietà di associatività e distributività della aritmetica sui numeri reali.

Esercizio 3

In una prova di esame di informatica viene richiesto di implementare l'algoritmo di ordinamento Insertion Sort; ricordiamo che l'algoritmo procede sui valori $i = 1, \ldots n$ in modo tale che ad ogni passo i tutti gli elementi di posto precedente ad i sono già ordinati, e l'elemento di posto i viene sistemato tra di essi dove appropriati. Viene proposta la seguente soluzione:

```
function v = insertionSort(v)

for i=1:length(v)
    j=1;
    while(v(j)<=v(i) && j<length(v))
        j = j+1;
    end
    v = [v(1:j-1) v(i) v(j:i-1) v(i+1:end)];
end

end
```

Si chiede di trovare l'errore presente nel codice e di proporre una correzione adeguata. L'array v è composto da numeri interi.

Soluzione

Il problema è nella condizione di uscita dal ciclo `while`; infatti, l'algoritmo prevede che l'elemento di posto i venga sistemato dove gli compete tra quelli già esaminati, quindi da 1 a $i-1$; pertanto il ciclo deve avere come condizione $j < i$. Inoltre è buona norma avere il controllo dell'indice prima dell'accesso al vettore di dati, anche se in questo caso particolare non darebbe grossi problemi. Il codice corretto è dunque

```
function v = insertionSort(v)
  for i=1:length(v)
      j=1;
      while ((j<i) &&  (v(j)<=v(i)))
          j = j+1;
      end
      v = [v(1:j-1) v(i) v(j:i-1) v(i+1:end)];
  end
end
```

Esercizio 4

È data la seguente funzione Matlab

```
function [v]=mistery(v,k,x)
  for i=1:k
      v = insertionSort(v);
      y = v*x';
      v = v./y;
  end
end
```

Si stimi il numero di operazioni aritmetiche ed il numero di confronti eseguito dal codice. Si assuma che v e x siano array di interi di lunghezza n.

Soluzione

Una stima può essere effettuata considerando che l'ordinamento attraverso InsertionSort prenderà in media $O(n^2/2)$ confronti se n è la dimensione del vettore. La seconda istruzione è un prodotto scalare che costa $2n$ operazioni aritmetiche, infine viene effettuato una scalatura del vettore per un costo di n operazioni. Tutte queste operazioni vengono ripetute per k volte, quindi

$$C = k\left(\frac{n^2}{2} + 3n\right).$$

Dopo la prima iterazione tuttavia il vettore v rimane ordinato; se l'algoritmo di InsertionSort è organizzato in modo tale da riconoscere che il suo input è ordinato con n confronti, allora il costo $n^2/2$ verrà pagato solo alla prima iterazione, e quindi si avrà

$$C = \frac{n^2}{2} + 3n + (k-1)(4n).$$

Esercizio 5

È dato il frammento di codice Matlab

```
n=length(v)
for i=1:n
    if (v(i) < 0) continue;
    if (v(i) > 1)
       v(i) = sqrt(v(i));
    else
       v(i) = v(i)^2;
    end
end
```

Riscriverlo facendo uso di operazioni su array.

Soluzione

Il ciclo `for` compie le seguenti operazioni:

1. Ignora gli elementi minori di 0;
2. Calcola la radice quadrata degli elementi maggiori di 1;
3. Eleva al quadrato gli elementi compresi tra 0 e 1.

Utilizzando l'indirizzamento con array booleani, le espressioni relazionali su array e le operazioni artimetiche termine a termine su vettori si può riscrivere il codice come segue:

```
idxg1   = (v>1);
idxg0l1 = (0<=v)&(v<1);
v(idxg1) = sqrt(v(idxg1));
v(idxg0l1) = (v(idxg0l1).^2);
```