Una raccolta di algoritmi

Marco Lapegna - Bozza di lavoro

May 6, 2021

1. il Massimo Comun Divisore
2. Ricerca sequenziale
3. Ricerca Binaria
4. Fusione di due array merging
5. Ricerca parola più lunga in un array
6. Ordinamento per selezione
7. Ordinamento per scambi
8. Ordinamento per inserzione
9. Trasposta di una matrice
10. Prodotto scalare di due vettori
11. Aggiornamento "saxpy" di un vettore
12. Prodotto "righe per colonne" tra una matrice e un vettore
13. Prodotto "righe per colonne" tra due matrici
14. Verifica di una matrice a diagonale dominante per righe
15. Verifica di una matrice con al più P colonne aventi almeno Q elementi nulli
16. Sistema di equazioni triangolare
17. Ordinamento delle righe di una matrice con indirizzamento indiretto
18. Report di occorrenze in una matrice
19. Valutazione di un polinomio con il metodo di Horner
20. L'epsilon macchina
21. Il minimo numero rappresentabile
22. La radice quadrata di X (con $0 \leq X \leq 10$)

1 Il Massimo Comun Divisore

Algoritmo 1: macodi

```
1  procedure macodi (in: a, b; out: mcd)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var a, b, q, r, mcd: integer;
4  |
   |  /* inizio esecuzione dell'algoritmo                 */
5  |  while ( b ≠ 0 ) do
6  |  |  q = a/b;
7  |  |  r = a - b*q;
8  |  |  a = b;
9  |  |  b = r;
10 |  end
11 |  mcd = a;
12 |
13 end
```

2 Ricerca sequenziale

Algoritmo 2: Ricerca sequenziale

```
1  procedure Ricerca sequenziale (in: X, n, w ; out: pos)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, pos, i: integer;
4  |  var X( n ): real;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  i = 0;
7  |  pos = -1;
8  |  repeat
9  |  |  i = i + 1;
10 |  |  if ( w == X( i ) ) then
11 |  |  |  pos = i;
12 |  |  end
13 |  until (pos ≠ =1 or i == n);
14 |
15 end
```

3 Ricerca binaria

Algoritmo 3: Ricerca sequenziale

```
1  procedure Ricerca binaria (in: X, n, w ; out: pos)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, pos, m, fisrt, last: integer;
4  |  var X( n ): real;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  first = 1;
7  |  last = n;
8  |  pos = -1;
9  |  repeat
10 |  |  m = (fisrt + last)/2;
11 |  |  if ( w == X( m ) ) then
12 |  |  |  pos = i;
13 |  |  else
14 |  |  |  if ( w > X( m ) ) then
15 |  |  |  |  first = m + 1;
16 |  |  |  else
17 |  |  |  |  last = m - 1;
18 |  |  |  end
19 |  |  end
20 |  until (pos ≠ =1 or first > last);
21 |
22 end
```

4 Fusione di due array (merging)

Algoritmo 4: Merging

```
1  procedure Merging (in: A, B, n, m; out: C)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, m, i, j, k: integer;
4  |  var A( n ), B( m ), C( n+m ): real;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  i = 1;
7  |  j = 1;
8  |  for k = 1 to n+m do
9  |  |  if ( i ≤ n and j ≤ m ) then
10 |  |  |  if ( A( i ) < B( j ) ) then
11 |  |  |  |  C( k ) = A( i );
12 |  |  |  |  i = i + 1;
13 |  |  |  else
14 |  |  |  |  C( k ) = B( j );
15 |  |  |  |  j = j + 1;
16 |  |  |  end
17 |  |  else
18 |  |  |  if ( i > n ) then
19 |  |  |  |  C( k ) = B( j );
20 |  |  |  |  j = j + 1;
21 |  |  |  else
22 |  |  |  |  C( k ) = A( i );
23 |  |  |  |  i = i + 1;
24 |  |  |  end
25 |  |  end
26 |  end
27 |
28 end
```

5 Ricerca parola più lunga in un array

Algoritmo 5: LongestWord

```
1  procedure LongestWord (in: X, n; out: len, pos)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, i, len, pos, c: integer;
4  |  var X( n ): char;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  len = 0;
7  |  pos = 0;
8  |  c = 0;
9  |  for k = 1 to n+m do
10 |  |  if ( X( i ) ≠ " " or X( i ) ≠ "." ) then
11 |  |  |  c = c + 1;
12 |  |  else
13 |  |  |  if ( c > len ) then
14 |  |  |  |  len = c;
15 |  |  |  |  pos = i;
16 |  |  |  end
17 |  |  |  c = 0;
18 |  |  end
19 |  end
20 |
21 end
```

6 Ordinamento per selezione

Algoritmo 6: Ordinamento per selezione

```
1  procedure Ordsel (in: A, n; out: A)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, i, j, p: integer;
4  |  var A[ n ], min, t: real;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  for i = 1 to n-1 do
7  |  |  p = i;
8  |  |  min = A[ i ];
9  |  |  for j = i+1 to n do
10 |  |  |  if ( A[ j ] < min ) then
11 |  |  |  |  p = j;
12 |  |  |  |  min = A[ j ];
13 |  |  |  end
14 |  |  end
15 |  |  t = A [ p ];
16 |  |  A[ p ] = A[ i ];
17 |  |  A[ i ] = t;
18 |  end
19 |
20 end
```

7 Ordinamento per scambi

Algoritmo 7: Ordinamento per scambi

```
1  procedure Ordsca (in: A, n; out: A)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, i, j, p: integer;
4  |  var A[ n ], min, t: real;
5  |  var scambi: logical;
6  |
   |  /* inizio esecuzione dell'algoritmo                 */
7  |  i = 0;
8  |  repeat
9  |  |  i = i+1;
10 |  |  scambi = .false.;
11 |  |  for j = 1 to n-i do
12 |  |  |  if ( A[ j ] > A[ j+1 ] ) then
13 |  |  |  |  scambi = .true.;
14 |  |  |  |  t = A[ j ];
15 |  |  |  |  A[ j ] = A[ j+1 ];
16 |  |  |  |  A[ j+1 ] = t;
17 |  |  |  end
18 |  |  end
19 |  until ( i == n-1 or scambi == .false. );
20 |
21 end
```

8 Ordinamento per inserzione

Algoritmo 8: Ordinamento per inserzione

```
1  procedure Ordins (in: A, n; out: A)
2  |
3  |  dichiarazione delle variabili;
4  |  var n, i, k: integer;
5  |  var A( n ), t: real;
6  |  var stop: logical;
7  |
8  |  inizio esecuzione dell'algoritmo;
9  |  for i = 2 to n do
10 |  |  t = A( i );
11 |  |  stop = .false.;
12 |  |  k = i;
13 |  |  repeat
14 |  |  |  if ( t < A( k-1 ) ) then
15 |  |  |  |  A( k ) = A( k-1 );
16 |  |  |  |  k = k-1;
17 |  |  |  else
18 |  |  |  |  stop = .true.;
19 |  |  |  end
20 |  |  until ( stop = .true. or k == 1 );
21 |  |  A( k ) = t;
22 |  end
23 |
24 end
```

9 Trasposta di una matrice

Algoritmo 9: Trasposta di una matrice

```
1  procedure traspmat (in: A, n; out: A)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, i, j: integer;
4  |  var A( n , n ), t: real;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  inizio esecuzione dell'algoritmo;
7  |  for i = 2 to n do
8  |  |  for i = 1 to n - 1 do
9  |  |  |  t = A( i , j );
10 |  |  |  A( i , j) = A( j , i );
11 |  |  |  A( j , i ) = t;
12 |  |  end
13 |  end
14 |
15 end
```

10 Prodotto scalare di due vettori

Algoritmo 10: prodscal

```
1  procedure prodscal (in: X, Y, n; out: s)
2  |
3  |  dichiarazione delle variabili;
4  |  var n, i: integer;
5  |  var X(n), Y(n), s: real;
6  |
7  |  inizio esecuzione dell'algoritmo;
8  |  s = 0;
9  |  for i = 1 to n do
10 |  |  s = s + X(i)*Y(i);
11 |  end
12 |
13 end
```

11 Aggiornamento "saxpy" di un vettore

.

Algoritmo 11: Aggiornamento saxpy

```
1  procedure saxpy (in: a, X, Y, n; out: Y)
2  |
3  |  dichiarazione delle variabili;
4  |  var n, i: integer;
5  |  var X(n), Y(n), a: real;
6  |
7  |  inizio esecuzione dell'algoritmo;
8  |  for i = 1 to n do
9  |  |  Y(i) = Y(i) + a*X(i);
10 |  end
11 |
12 end
```

12 Prodotto "righe per colonne" tra una matrice e un vettore

Algoritmo 12: matvet

```
1  procedure matvet (in: A, X, n, m; out: Y)
2  |
3  |  dichiarazione delle variabili;
4  |  var n, m, i, j: integer;
5  |  var A(n,m), X(m), Y(n): real;
6  |
7  |  inizio esecuzione dell'algoritmo;
8  |  for i = 1 to n do
9  |  |  y(i) = 0;
10 |  |  for j = 1 to m do
11 |  |  |  y(i) = y(i) + A(i,j)*X(j);
12 |  |  end
13 |  end
14 |
15 end
```

Osservazioni e ulteriori esercizi:

13 Prodotto "righe per colonne" tra due matrici

Algoritmo 13: matmat

```
1  procedure matmat (in: A, B, n, m, p; out: C)
2  |
3  |  dichiarazione delle variabili;
4  |  var n, m, i, j: integer;
5  |  var A(n,m), B(m,p), C(n,p): real;
6  |
7  |  inizio esecuzione dell'algoritmo;
8  |  for i = 1 to n do
9  |  |  for j = 1 to p do
10 |  |  |  C(i,j) = 0;
11 |  |  |  for k = 1 to m do
12 |  |  |  |  C(i,j) = C(i,j) + A(i,k)*B(k,j);
13 |  |  |  end
14 |  |  end
15 |  end
16 |
17 end
```

14 Verifica di una matrice a diagonale dominante per righe

Algoritmo 14: matdiagdom

```
1  procedure matdiagdom (in: A, n; out: test)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, i, j, test: integer;
4  |  var A( n , n ), diag, sum: real;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  test = -1;
7  |  i = 0;
8  |  repeat
9  |  |  i = i + 1;
10 |  |  sum = 0;
11 |  |  for j = 1 to n do
12 |  |  |  if ( i == j ) then
13 |  |  |  |  diag = |A( i , i )|;
14 |  |  |  else
15 |  |  |  |  sum = sum + |A( i , j )|;
16 |  |  |  end
17 |  |  end
18 |  |  if ( diag < sum ) then
19 |  |  |  test = i;
20 |  |  end
21 |  until ( i == nor test / = -1 );
22 |
23 end
```

15 Verifica di una matrice con al più P colonne aventi almeno Q elementi nulli

Algoritmo 15: verificaPQ

```
1  procedure verificaPQ (in: A, n, m, P, Q; out: test)
2  |
   |  /* dichiarazione delle variabili                          */
3  |  var n, m, i, j, P, Q, test, contacol, contanulli: integer;
4  |  var A( n , m ): real;
5  |
   |  /* inizio esecuzione dell'algoritmo                       */
6  |  contacol = 0;
7  |  j = 0;
8  |  repeat
9  |  |  j = j + 1;
10 |  |  contanulli = 0;
11 |  |  i = 0;
12 |  |  repeat
13 |  |  |  i = i + 1;
14 |  |  |  if ( A( i , j ) == 0 ) then
15 |  |  |  |  contanulli = contanulli +1;
16 |  |  |  end
17 |  |  until ( i == n or contanulli == Q );
18 |  |  if ( contanulli == Q ) then
19 |  |  |  contacol = contacol + 1;
20 |  |  end
21 |  until ( j == m or contacol > P );
22 |  if ( contacol ≤ P ) then
23 |  |  test = 1;
24 |  else
25 |  |  test = -1;
26 |  end
27 |
28 end
```

16 Sistema di equazioni triangolare

Algoritmo 16: sistri

```
1  procedure sistri (in: A, b, n; out: test, x)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, i, j, test: integer;
4  |  var A( n , n ), b( n ), x( n ): real;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  test = -1;
7  |  i = 0;
8  |  repeat
9  |  |  i = i + 1;
10 |  |  if ( A( i , i ) == 0 ) then
11 |  |  |  test = i;
12 |  |  else
13 |  |  |  sum = 0;
14 |  |  |  for j = 1 to n do
15 |  |  |  |  sum = sum + A( i , j )*x( j );
16 |  |  |  end
17 |  |  |  x( i ) = (b( i ) - sum)/A( i , i );
18 |  |  end
19 |  until ( i == nor test / = -1 );
20 |
21 end
```

17 Ordinamento delle righe di una matrice con indirizzamento indiretto

Algoritmo 17: ordindind

```
1  procedure ordindind (in: A, n, m; out: ind)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, m, i, j, p, t, ind( n ): integer;
4  |  var A( n , n ), min: real;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  for i = 1 to n do
7  |  |  ind( i ) = i;
8  |  end
9  |  for i = 1 to n-1 do
10 |  |  p = i;
11 |  |  min = A( ind( i ) , 1);
12 |  |  for j = i+1 to n do
13 |  |  |  if ( A( ind( j ) , 1 ) < min ) then
14 |  |  |  |  p = j;
15 |  |  |  |  min = A( ind( j ) , 1 );
16 |  |  |  end
17 |  |  end
18 |  |  t = A [ p ];
19 |  |  A[ p ] = A[ i ];
20 |  |  A[ i ] = t;
21 |  end
22 |
23 end
```

18 Report di occorrenze in una matrice

Algoritmo 18: occorrenze

```
1  procedure occorrenze (in: A, n, m; out: P, B)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, m, P, k, pos: integer;
4  |  var A( n , n ), B( P , 2): integer;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  P = 0;
7  |  for i = 1 to n do
8  |  |  for j = 1 to m do
9  |  |  |  pos = -1;
10 |  |  |  k = 0;
11 |  |  |  while ( k < P and pos == -1 ) do
12 |  |  |  |  k = k + 1;
13 |  |  |  |  if ( A( i , j ) == B( k , 1 ) then
14 |  |  |  |  |  pos = k;
15 |  |  |  |  end
16 |  |  |  end
17 |  |  |  if ( pos ≠ -1 then
18 |  |  |  |  B( pos , 2 ) = B( pos , 2 ) + 1;
19 |  |  |  else
20 |  |  |  |  P = P + 1;
21 |  |  |  |  B( P , 1 ) = A( i , j );
22 |  |  |  |  B( P , 2 ) = 1;
23 |  |  |  end
24 |  |  end
25 |  end
26 |
27 end
```

19 Valutazione di un polinomio con il metodo di Horner

Algoritmo 19: horner

```
1  procedure horner (in: A, n, x; out: pol)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var n, i: integer;
4  |  var A( 0 : n ), x, pol: integer;
5  |
   |  /* inizio esecuzione dell'algoritmo                 */
6  |  pol = A(n);
7  |  for i = n-1 to 0 step -1 do
8  |  |  pol = pol * x + A( i );
9  |  end
10 |
11 end
```

20 L'epsilon macchina

Algoritmo 20: epsmac

```
1  procedure epsmac (out: eps)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var eps, a, sum : real;
4  |
   |  /* inizio esecuzione dell'algoritmo                 */
5  |  a = 1;
6  |  repeat
7  |  |  eps = a;
8  |  |  a = a/2;
9  |  |  sum = 1 + a;
10 |  until ( sum == 1 );
11 |
12 end
```

21 Il minimo numero rappresentabile

Algoritmo 21: minrap

```
1  procedure minrap (out: minr)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var minr, a : real;
4  |
   |  /* inizio esecuzione dell'algoritmo                 */
5  |  a = 1;
6  |  repeat
7  |  |  minr = a;
8  |  |  a = a/2;
9  |  until ( a == 0. );
10 |
11 end
```

22 La radice quadrata di X (con $0 \leq X \leq 10$)

Algoritmo 22: sqroot

```
1  procedure sqroot ( in: X; out: Rad)
2  |
   |  /* dichiarazione delle variabili                    */
3  |  var X, Rad, a, oldrad : real;
4  |
   |  /* inizio esecuzione dell'algoritmo                 */
5  |  if ( X < 0 or X > 10 ) then
6  |  |  Rad = -1;
7  |  else
8  |  |  epsmac(out: eps);
9  |  |  Rad = 0;
10 |  |  a = 1;
11 |  |  repeat
12 |  |  |  repeat
13 |  |  |  |  oldrad = Rad;
14 |  |  |  |  Rad = Rad + a;
15 |  |  |  until ( Rad * Rad > X );
16 |  |  |  Rad = oldrad;
17 |  |  |  a = a/2;
18 |  |  until ( a < Rad * eps );
19 |  end
20 |
21 end
```