L'algoritmo Bubble Sort

Questo algoritmo usa una strategia un po' diversa, generalmente più efficiente del insertion Sort: infatti esso effettua lo stesso numero di confronti dell'Insertion Sort solo nel caso peggiore, cioè quando i dati sono ordinati al contrario.

Nel caso medio esso svolge un numero di confronti minore di

$$\frac{(N-1)N}{2}$$ (numero di confronti per l'InsertionSort)

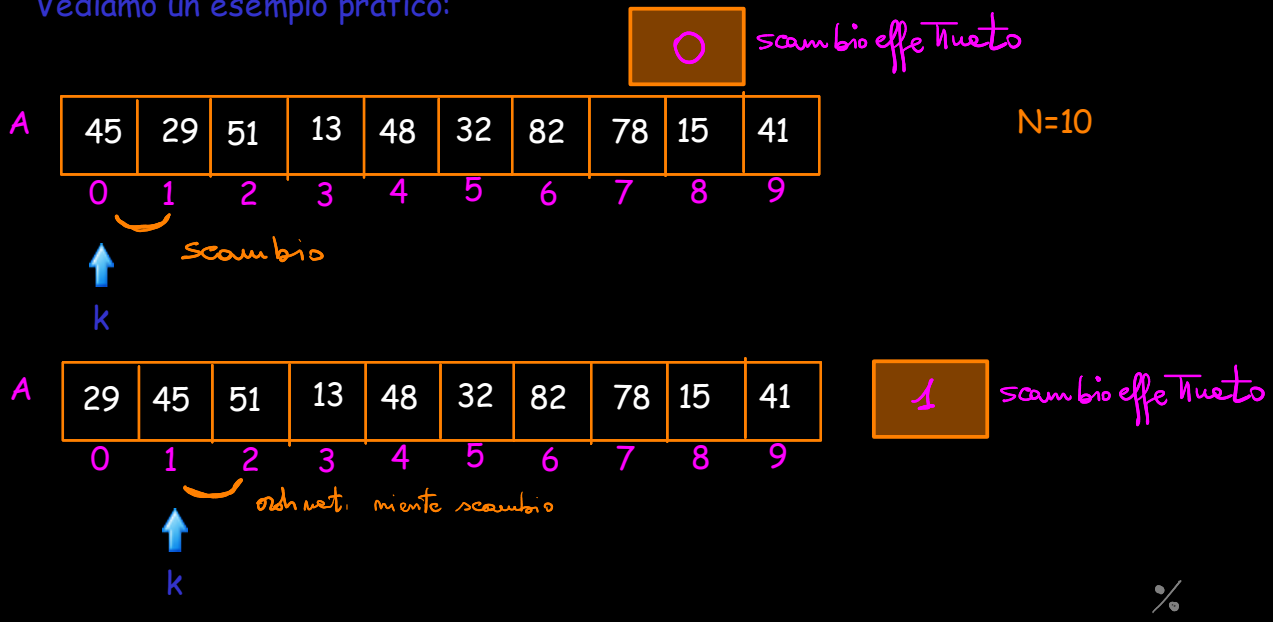
Strategia

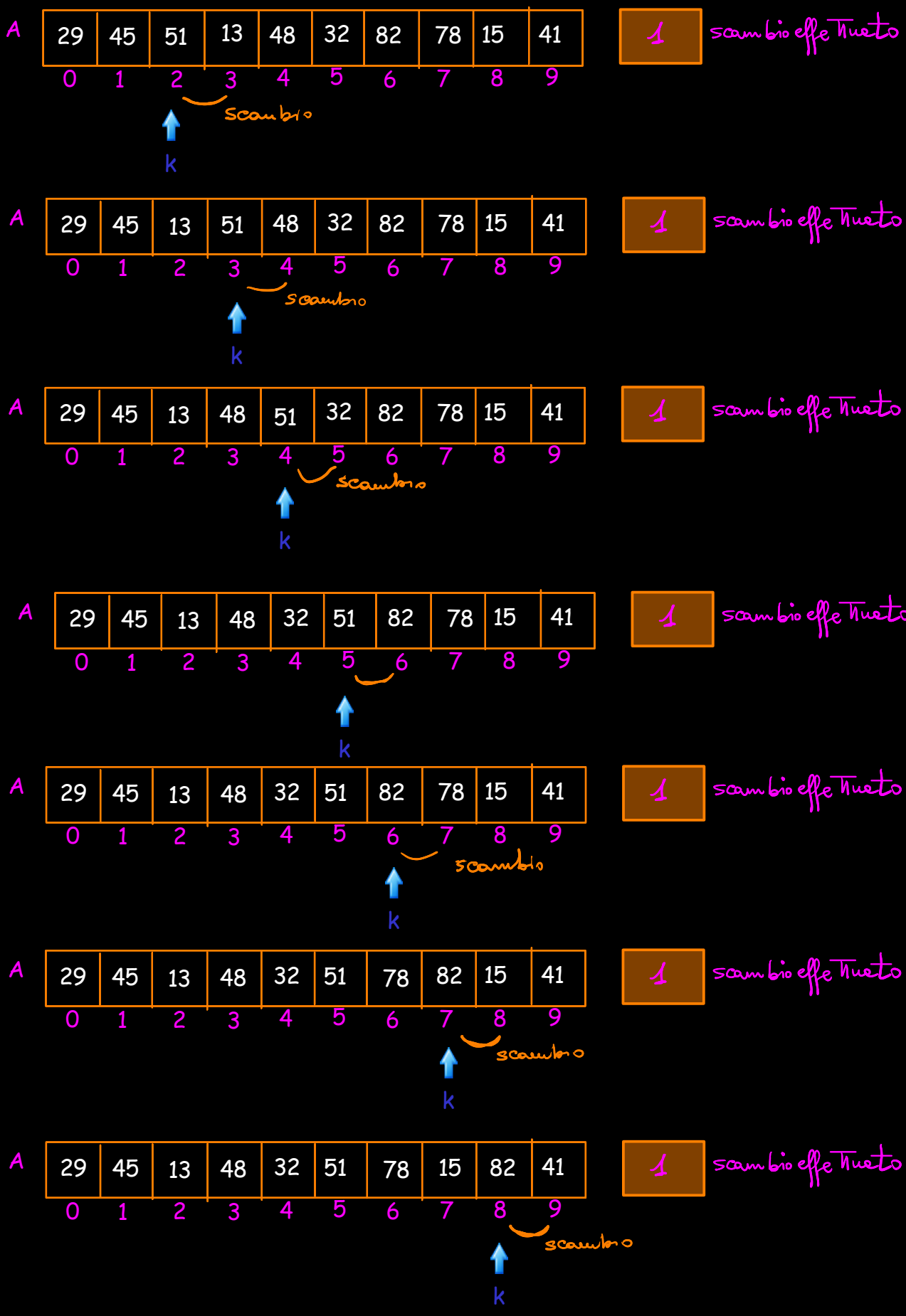
Ogni elemento, a partire dal primo, viene confrontato con l'elemento successivo e se non sono ordinati vengono scambiati fra loro.
Viene usato un solo indice che scorre fino al penultimo elemento del vettore cosicchè si possa confrontare con l'elemento successivo che è l'ultimo.
Quando la prima scansione sarà terminata, avremmo portato il massimo del vettore in ultima posizione, cioè nella posizione N-1

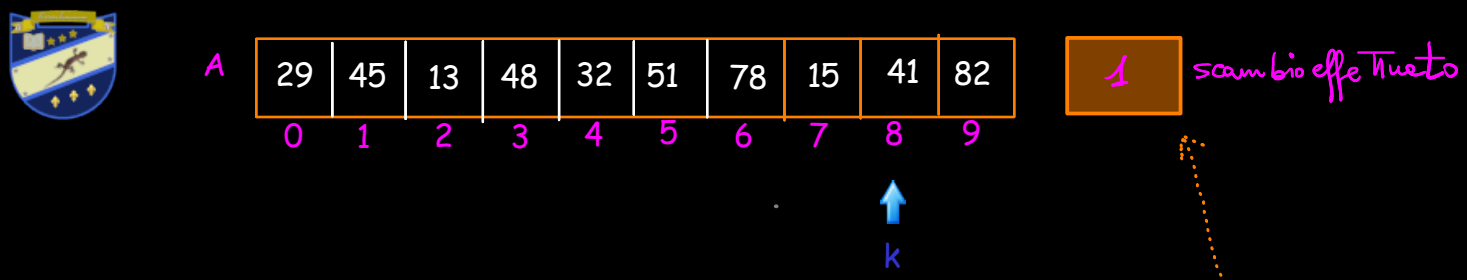
Per ottimizzare l'efficienza di questo algoritmo utilizzeremo una variabile chiamata scambioeffettuato che sarà impostata a 0 prima di incominciare ogni scansione e sarà impostata a 1 quando, durante loa scansione, si verificherà la necessità di uno scambio.

In questo modo saremo in grado di accorgerci se durante la scansione del vettore è avvenuto o meno uno scambio.
Se non è avvenuto alcuno scambio (cioè se la variabile scambioeffettuato vale 0) vuol dire che gli elementi sono tutti ordinati e l'algoritmo potrà terminare senza ulteriori confronti.

Vediamo un esempio pratico:




A
29 45 51 13 48 32 82 78 15 41
0 1 2 3 4 5 6 7 8 9
scambio
k
1
scambio effettuato
A
29 45 13 51 48 32 82 78 15 41
0 1 2 3 4 5 6 7 8 9
scambio
k
1
scambio effettuato
A
29 45 13 48 51 32 82 78 15 41
0 1 2 3 4 5 6 7 8 9
scambio
k
1
scambio effettuato
A
29 45 13 48 32 51 82 78 15 41
0 1 2 3 4 5 6 7 8 9
k
1
scambio effettuato
A
29 45 13 48 32 51 82 78 15 41
0 1 2 3 4 5 6 7 8 9
scambio
k
1
scambio effettuato
A
29 45 13 48 32 51 78 82 15 41
0 1 2 3 4 5 6 7 8 9
scambio
k
1
scambio effettuato
A
29 45 13 48 32 51 78 15 82 41
0 1 2 3 4 5 6 7 8 9
scambio
k
1
scambio effettuato



fine della prima scansione!!!

Osservazioni:

Come già previsto il massimo è finito in ultima posizione, Cioè nella posizione N-1

Guardiamo la variabile scambioeffettuato. Siccome il suo valore è 1, vuol dire che è stato effettuato uno scambio e quindi bisogna ripetere la procedura cominciando da capo.

useremo un ciclo per effettuare la ripetizione del tipo:

ripeti
scambioeffettuato = 0
{ ---------- istruzioni per la scansione, il confronto e lo scambio. }
Se durante la scansione avviene uno scambio, scambioeffettuato sarà messo a 1
mentre (scambioeffettuato == 1)

Siccome in ultima posizione c'è ormai il massimo, la prossima scansione la fermo all'elemento N-3 che sarà confrontato con l'elemento N-2

Questo vuol dire che ad ogni scansione completata, ridurrò virtualmente il vettore di una casella, risparmiando così sul numero di confronti necessari.

ultimo valore dell'indice K durante la prima scansione

Utilizzerò a questo scopo una variabile ultimo, che inizialmente sarà impostata a N-2 e che sarà decrementata alla fine di ogni scansione, permettendomi così di ridurre (virtualmente) il vettore

Pseudo codifica:

```
ultimo = N-2
ripeti
    scambioEffettuato = 0
    Per K da 0 a ultimo incluso
        Se A[K] > A[K+1] allora
            Scambia(A[K], A[K+1])
            scambioEffettuato = 1
        fine se
    fine per
    ultimo = ultimo - 1
mentre (scambioEffettuato = 1)
```

Valutazione dell'efficienza dell'algoritmo:

Caso migliore: i dati sono ordinati; vengono effettuati N-1 confronti

Caso peggiore: i dati sono ordinati al contrario; vengono effettuati (N-1)N/2 confronti

Codifica

```
/*
dichiarazioni a livello i modulo
*/
#define MAX 100
int nconfronti=0;
```

è una macro: una direttiva al completatore, che sostituisce ogni occorrenza "MAX" con il valore 100

variabile globale, visibile in tutto il modulo

```
int main (){
 //int n, a[MAX],i;
 //n = carica_vettore(a);
 int n, i;
 int a[] = { 52, 25, 35, 53, 63, 36, 66, 48, 12, 12, 88, 84, 48, 78, 26 };
 n = 15;
 printf ("Vettore iniziale: \n");
 stampa_vettore (a, 0, n - 1);
 BubbleSort (a, n);
 printf ("Vettore ordinato con Bubble sort: \n");
 stampa_vettore (a, 0, n - 1);
 printf ("sono stati effettuati %d confronti", nconfronti);
 return 0;
}
```

Vettore statico da usare solo in fase di Test nel caso reale commentare le linee evidenziate in giallo e togliere il commento alle linee evidenziate col verde

quanti dati contiene il vettore?

stampa il contenuto del vettore non ordinato dalla posizione 0, alla posizione N-1 (estremi inclusi)

ordina il vettore

stampa il vettore ordinato,

↳ mostra quanti confronti sono stati effettuati.

la funzione non ha un valore di ritorno

```
void stampa_vettore (int a[], int p, int u)
{
  int i;
  for (i = p; i <= u; i++)
    {
      printf ("%d ", a[i]);
    }
  printf ("\n");
  return;
}
```

la funzione ha 3 parametri di input
1. il vettore
2. la prima posizione da stampare p
3. l'ultima posizione da stampare

ad ogni ciclo stampa un elemento del vettore

Quale? a[i] è quello stampato!

e i varia da p a u estremi inclusi

poiché non c'è "\n" sono stampati tutti sulla stessa riga

alla fine dei dati andiamo a capo con "\n"

next page

non ha valore di ritorno

```
void BubbleSort (int a[], int n)
{
   int k, scambioeffettuato, ultimo;
   ultimo=n-2;
   do {
      scambioeffettuato=0;
      for (k=0;k<=ultimo;k++){
         nconfronti++;

         if (a[k]>a[k+1]){
            scambia (a, k, k+1);
            scambioeffettuato=1;
         }
      }
      ultimo--;
   } while (scambioeffettuato==1);
}
```

prende 2 parametri in input: il vettore e il numero di elementi che contiene

k → indice generico

indica se è stato effettuato uno scambio (1) oppure no (0)

indice fino a quale elemento bisogna effettuare la scansione; ad ogni ciclo diminuisce di 1. Parte dal valore n-2

ciclo postcondizionato

azzeriamo il segnalatore di scambio (importante: se non ci fosse si avrebbe un loop infinito)

per tutti gli elementi della porzione di vettore da prendere in considerazione

Confronta ogni elemento con il successivo

Se necessario, effettua lo scambio e alziamo la bandierina per ricordarcelo.

Se nel ciclo precedente è stato effettuato almeno uno scambio comincia da capo.

diminuiamo di 1 il numero di elementi da considerare

```
void scambia (int a[], int x, int y)
{
  int copia;
  copia = a[x];
  a[x] = a[y];
  a[y] = copia;

}
```

Questa non è degna di essere commentata.

Troppo banale!

... e comunque, per chi ne avesse bisogno, ste spiegate nell'algoritmo dell'

INSERTION SORT

che avreste dovuto già aver studiato!

GN