DOMANDA: Illustrare l'Algoritmo di Ordinamento per Selezione di Massimo (AOSMAX)

RISPOSTA:
Lo scopo di un Algoritmo di Ordinamento è ordinare (in senso crescente) un array di dati. I dati possono essere numeri, caratteri, stringhe di caratteri o qualsiasi altro insieme di dati su cui sia definita una relazione di ordine.

L'AOSMAX ha come dati di input un array e il suo size e ha come dato di output lo stesso array con i dati ordinati. L'AOSMAX non fa uso di altri array o strutture dati e per questo motivo si dice che opera "in situ" o "in place".

Diamo una rapida analisi della meccanica dell'AOSMAX. L'algoritmo effettua un numero di passi (iterazioni) pari al size dell'array meno 1. Cioè, detto n il size dell'array che chiamiamo A, l'algoritmo effettua n-1 passi per ordinare l'array A.
Ad ogni passo l'algoritmo risolve il sottoproblema della ricerca dell'elemento massimo (e del suo indice) di una opportuna "porzione" dell'array A. Una volta trovato l'elemento massimo e la sua posizione, tale elemento viene scambiato con l'elemento che si trova all'ultimo posto della porzione che si sta considerando.
Al passo successivo, si considera una nuova porzione dell'array, che avrà un numero di elementi diminuito di una unità rispetto al numero di elementi della porzione considerata precedentemente.
Un esempio aiuta a capire la dinamica dell'AOSMAX. Si consideri un array A di n=8 elementi interi

27	41	36	11	28	13	52	31
0	1	2	3	4	5	6	**i=n-1=7**

Per comodità, chiamiamo i l'indice che individua il termine della porzione di array che si sta considerando (tale indice decrescerà a ogni passo). Al primo passo (i=n-1=7), l'algoritmo determina il massimo elemento (e la sua posizione) dell'intero array, cioè della porzione che ha come primo elemento l'elemento di indice 0 e come ultimo elemento quello di indice i (cioè indice 7). Si noti che tale sottoproblema della determinazione del massimo può essere risolto mediante una opportuna chiamata della function `max_val_ind`.
Nell'esempio, elemento massimo è 52 e il suo indice è 6. Ora, si deve scambiare tale elemento con l'ulyimo elemento della porzione che è 31 e che ha indice i=7. Si noti che tale operazione può essere effettuata mediante una opportuna chiamata della function `scambiare`. Dopo tale operazione di scambio l'array A ha i seguenti valori

11	41	36	27	28	13	31	52
0	1	2	3	4	5	6	**I=n-1=7**

Nel secondo passo si considera la porzione dell'array A di lunghezza n-1 (cioè 7) che va dal primo elemento dell'array (indice 0) fino al penultimo elemento (cioè indice i=6)

11	41	36	27	28	13	31	52
0	1	2	3	4	5	**i=6**	n-1=7

e si determina l'elemento massimo (e il suo indice) di tale porzione di array, cioè la porzione 0..6. Anche questo sottoproblema può essere risolto mediante una opportuna chiamata alla function `max_val_ind`.
Nell'esempio, l'elemento massimo della porzione 0..6 è 41 e il suo indice (in A) è 1. Tale elemento deve essere scambiato con l'ultimo elemento della porzione, che è 31 e ha indice i=6. Dopo tale operazione di scambio il secondo passo è terminato e l'array A ha i seguenti valori

11	31	36	27	28	13	41	52
0	1	2	3	4	5	**i=6**	n-1=7

Nel terzo passo, si considera la porzione 0..5 dell'array (cioè i=5). Si noti che la porzione 6..7 dell'array è ordinata in modo definitivo e non sarà più modificata.
In generale, il modo migliore per sintetizzare il funzionamento dell'AOSMAX è quello di considerare la situazione al generico passo i. Nell'esempio, consideriamo i=4.

11	31	13	27	28	36	41	52
0	1	2	3	**i=4**	5	6	n-1=7

La porzione (i+1)..(n-1), cioè 5..7, è ordinata nella sua forma definitiva e non viene più modificata. La porzione 0..i, cioè 0..4, è ancora disordinata.
Si determina l'elemento massimo, e il suo indice, della porzione 0..4. Tale elemento si scambia con l'elemento di indice i, cioè l'elemento che si trova all'ultimo posto della porzione che si sta considerando. Nel nostro esempio 31 viene scambiato con 28, fornendo

11	28	13	27	31	36	41	52
0	1	2	3	**i=4**	5	6	n-1=7

La successiva porzione da considerare è la porzione 0..3. A ogni passo la lunghezza della porzione che viene considerata diminuisce di 1.
La struttura dell'algoritmo è costituita da un ciclo `for` per i passi. Questa è l'implementazione in C dell'AOSMAX, per un array di tipo `char`:

```
void ord_sel_max(char array[],int n)
{
  int i,indice_max;
  char max_array;
  for (i=n-1;i>0;i--)
     {
      max_val_ind(&array[0],i+1,&max_array,&indice_max);
      scambiare_c(&array[i],&array[indice_max]);
     }
 }
```

Un breve commento al codice. Il for su i da n-1 a 1 gestisce gli n-1 passi dell'AOSMAX. Al passo i, la determinazione dell'elemento massimo (e del suo indice) della porzione 0..i viene effettuata mediante la chiamata alla function `max_val_ind`, con 2 argomenti di input: l'indirizzo iniziale della porzione di array, cioè **`&array[0]`**, e il size della porzione, cioè **`i+1`** , e due argomenti di output, che essendo scalari devono essere passati per indirizzo: **`&max_array, &indice_max`**.
Si noti che il valore di `indice_max` restituito dalla function `max_val_ind` è il cosiddetto " valore locale" dell'indice, cioè si riferisce alla porzione che si sta considerando (0..i) ma coincide con il cosiddetto "valore globale" dell'indice, cioè quello che determina la corretta posizione all'interno dell'array completo, e quindi non si deve effettuare alcuna operazione di "spiazzamento dell'indice locale", che invece è necessaria nel caso dell'AOSMIN.
Lo scambio tra tale elemento e l'elemento finale della porzione (cioè l'elemento di indice `i`) viene effettuato dalla function `scambiare_c`, che è chiamata con i due argomenti di input/output, necessariamente passati per indirizzo: **`&array[i]`** , **`&array[indice_max]`** .
Se si avesse a disposizione una function che determina solo l'indice dell'elemento massimo di un array, chiamiamola **`max_ind`**, allora la function **`ord_sel_max`** assumerebbe la seguente forma:

```
void ord_sel_max(char array[],int n)
 {
  int i;
  for (i=n-1;  i>0; i--)
    scambiare_c(&array[i],&array[max_ind(&array[0],i+1)]);
 }
```

Con

```
int max_ind(char a[],int n)
 {
  int i,i_max;   i_max = 0;
  for (i=1;  i<n; i++)
       if (a[i] > a[i_max])
           i_max = i;
  return i_max;
 }
```

Analizziamo la complessità dell'AOSMAX. La complessità di spazio è n, perché l'algoritmo opera completamente "in place".
Passiamo alla complessità di tempo. Negli algoritmi di ordinamento si considerano come operazioni dominanti l'operazione di confronto tra due elementi dell'array e l'operazione di scambio tra 2 elementi dell'array. Cominciamo, per semplicità, col determinare il numero di operazioni di scambio. In pratica, si tratta di contare il numero di chiamate della function `scambiare_c`. L'attivazione della `scambiare_c` viene fatta a ogni iterazione del ciclo `for`, che va da (n-1) a 1, cioè per un totale di (n-1) iterazioni. Quindi l'AOSMAX effettua sempre (n-1) scambi; è una complessità cosiddetta assoluta, perché indipendente dal valore degli elementi dell'array.
Determiniamo ora il numero delle operazioni di confronto tra due elementi dell'array. Poiché né nel testo della function di ordinamento né nella function `scambiare_c` compaiono operazioni di confronto tra due elementi dell'array, allora i soli confronti sono quelli contenuti nella function `max_val_ind` . Ricordiamo che la complessità di tempo dell'algoritmo di determinazione del massimo di un array è data dal size dell'array -1 operazioni di confronto. Tale complessità è assoluta, cioè indipendente dal valore dei dati. Nel nostro caso la function `max_val_ind` è chiamata (n-1) volte, una per ogni iterazione del ciclo `for`, ma il size della porzione su cui agisce la function non è costante, perché diminuisce di 1 a ogni iterazione.
Alla prima iterazione, la porzione passata alla function `max_val_ind` coincide con l'intero array e ha lunghezza n; quindi la function effettua (n-1) confronti. Alla seconda iterazione, la porzione passata alla function `max_val_ind` ha una lunghezza diminuita di uno rispetto alla precedente, cioè ha lunghezza n-1; quindi la `max_val_ind` effettua n-2 confronti.
E così via, fino all'ultimo passo, in cui alla `max_val_ind` viene passata una porzione di lunghezza 2, per cui il numero di confronti è 1.
Per ottenere il numero totale di confronti bisogna sommare i numeri dei confronti di ogni passo, cioè (n-1) + (n-2) + (n-3) +…+ 2 + 1 . Ma tale numero è la somma dei primi (n-1) numeri naturali, che per la formula di Gauss è $\frac{n\cdot(n-1)}{2}$. Tale complessità è assoluta.
In conclusione, la complessità di tempo dell'AOSMAX è

$T(n)=\frac{n\cdot(n-1)}{2}=\frac{1}{2}(n^2-n)=O(n^2)$ confronti,

$T(n)=n=O(n)$ scambi.

Infine, poiché la complessità di tempo dell'AOSMAX non dipende dal valore dei dati, l'AOSMAX assume lo stesso costo anche nel caso in cui l'array sia già ordinato. Gergalmente, si dice che l'AOSMAX non si "accorge" del fatto che un array sia già ordinato.