Problema:

Data una sequenza di elementi in ordine qualsiasi, ordinarla.

- Questo è un problema fondamentale, che si presenta in moltissimi contesti, ed in diverse forme.
- Nel nostro caso formuliamo il problema in termini di ordinamento di vettori:

  Dato un vettore A di n elementi, ordinarlo in modo crescente
- Per semplicità faremo sempre riferimento a vettori di interi.

$$\begin{array}{ccc} & 5\ 2\ 4\ 6\ 1\ 3 & \\ \Longrightarrow & & \\ & 1\ 2\ 3\ 4\ 5\ 6 & \end{array}$$

Ordinamento per inserzione (insertion sort)

- **Esempio:** Ordinamento di una mano di ramino
  - si inizia con la mano sinistra vuota e le carte coperte sul tavolo
  - si prende dalla tavola una carta alla volta e la si inserisce nella corretta posizione nella mano sinistra
  - ...
  - si termina quando si sono finite tutte le carte sul tavolo.
- Stesso procedimento per ordinare un vettore:
  - inizialmente il vettore rappresenta il mazzo sul tavolo
  - si usa un ciclo per analizzare uno alla volta gli elementi del vettore
  - Alla generica iterazione la situazione è la seguente

mano sinistra	carte ancora da scoprire

    ↑ nuova carta
  - per inserire la nuova carta al posto giusto nella mano sinistra dobbiamo
    - scorrere gli elementi che lo precedono per decidere la posizione che gli compete
    - spostare di un posto verso destra gli elementi maggiori per fargli spazio.

Esempio

In verde le carte ancora da esaminare, in rosso quelle già esaminate (mano sinistra). La nuova carta da esaminare è sottolineata.

<u>5</u>	2	4	6	1	3
5	<u>2</u>	4	6	1	3
2	5	<u>4</u>	6	1	3
2	4	5	<u>6</u>	1	3
2	4	5	6	<u>1</u>	3
1	2	4	5	6	<u>3</u>
1	2	3	4	5	6

- Una volta individuata la posizione k in cui inserire la nuova carta dobbiamo farle spazio, ovvero spostare verso destra di una posizione tutte le carte rosse da k in poi.

  Esempio:

1	2	4	5	6	3
1	2	4	5	6	3
1	2	4	5		6
1	2	4		5	6
1	2		4	5	6

- A questo punto possiamo piazzare la carta in modo ordinato

1	2	3	4	5	6

- Definiamo allora una procedura che sposta tutti gli elementi di un vettore verso destra di una posizione tra due indici dati `from` e `to`

```
void shiftR(int v[], int from, int to)
{
int i;
for (i=to-1; i>=from; i--)
  v[i+1] = v[i];
}
```

- L'elemento in posizione `to` viene perso
- Bisogna procedere da destra verso sinistra (perché?)
- se `to` è minore o uguale a `from` non succede nulla

Possiamo allora definire la procedura di ordinamento per inserzione come segue

```
void sort(int v[], int dim)
{
int h, curr, j=1;
while (j<dim)
  { h=0;
    curr=v[j];
          /* curr e' l'elemento da piazzare */

    while((v[h]<curr) && (h<j))
       h++;
          /* curr va inserito in posizione h */

    shiftR(v,h,j);
     v[h]=curr;
    j++;
   }
}
```

Possiamo anche rendere la procedura più efficiente, nel modo seguente

```
void sort (int v[], int dim) {
  int i, j, prossimo;
  for (i = 1; i < dim; i++) {
     prossimo = v[i];
     j = i;
     while ((j > 0) && (v[j-1] > prossimo)) {
        v[j] = v [j-1];
        j--;
     }
     v[j] = prossimo;
  }
}
```

Ordinamento per selezione del minimo (selection sort)

- **Esempio:** Ordinamento di un mazzo di carte
  - si seleziona la carta più piccola e si mette da parte
  - delle rimanenti si seleziona la più piccola e si mette da parte
  - ...
  - si termina quando rimane una sola carta
- Ordinamento di un vettore:
  - per selezionare l'elemento più piccolo tra quelli rimanenti si utilizza un ciclo
  - mettere da parte significa scambiare con l'elemento che si trova nella posizione che compete a quello selezionato

- in verde la parte che rimane da analizzare
  in blu l'elemento minimo selezionato
  in marrone lo scambio effettuato
  in rosso la parte ordinata

5	2	4	6	1	3	
	5	2	4	6	1	3
	1	2	4	6	5	3
1	2	4	6	5	3	
	1	2	4	6	5	3
	1	2	4	6	5	3
1	2	4	6	5	3	
	1	2	4	6	5	3
	1	2	3	6	5	4
1	2	3	6	5	4	
	1	2	3	6	5	4
	1	2	3	4	5	6
1	2	3	4	5	6	
	1	2	3	4	5	6
	1	2	3	4	5	6
1	2	3	4	5	6	
1	2	3	4	5	6	

Implementazione

```
int minPos(int v[], int from, int to);
/* calcola la posizione del minimo elemento di
   v nella porzione [from,to]                */

void swap(int *p, int *q);
/* scambia le variabili puntate da p e q */

/** PROCEDURA DI ORDINAMENTO PER SELEZIONE **/

void sort(int v[], int dim)
{
  int i, min;
  for(i=0; i<dim-1; i++)
    {
      min = minPos(v, i, dim-1);
      swap(v+i, v+min);
    }
}
```

Scrivere per **esercizio** le procedure `swap` e `minpos`

```
int minPos(int v[], int from, int to) {
/* calcola la posizione del minimo elemento di
   v nella porzione [from,to]                */

    int i, pos;
    pos = from;
    for (i=from+1; i<=to; i++)
        if (v[i] < v[pos])
            pos = i;
    return pos;
}

void swap(int *p, int *q) {
/* scambia le variabili puntate da p e q */
    int temp = *p;
    *p = *q;
    *q = temp;
}
```

Ordinamento a bolle (bubble sort)

- Si fanno salire gli elementi più piccoli ("più leggeri") verso l'inizio del vettore ("verso l'alto"), scambiandoli con quelli adiacenti.
- L'ordinamento è suddiviso in n-1 fasi:
  - fase 0: 0^o elemento (il più piccolo) in posizione 0
  - fase 1: 1^o elemento in posizione 1
  - ...
  - fase n-2: $(n-2)^o$ elemento in posizione n-2, e quindi $(n-1)^o$ elemento in posizione n-1
- Nella fase i: cominciamo a confrontare dal basso e portiamo l'elemento più piccolo (più leggero) in posizione i

5	2	4	6	1	3	
	5	2	4	6	1	3
	5	2	4	1	6	3
	5	2	1	4	6	3
	5	1	2	4	6	3
	1	5	2	4	6	3
1	5	2	4	6	3	
	1	5	2	4	3	6
	1	5	2	3	4	6
	1	5	2	3	4	6
	1	2	5	3	4	6
1	2	5	3	4	6	
1	2	3	5	4	6	
1	2	3	4	5	6	
1	2	3	4	5	6	

```
/** PROCEDURA BUBBLE SORT **/


void sort(int v[], int dim)
{
  int temp,i,j;
  for (i = 0; i < dim-1; i++)      /* fase i-esima */

    for (j = dim-1; j > i; j--)    /* bolla piu' leggera in posizione i */
      if (v[j] < v[j-1])
        swap(v+j, v+j-1);
}
```