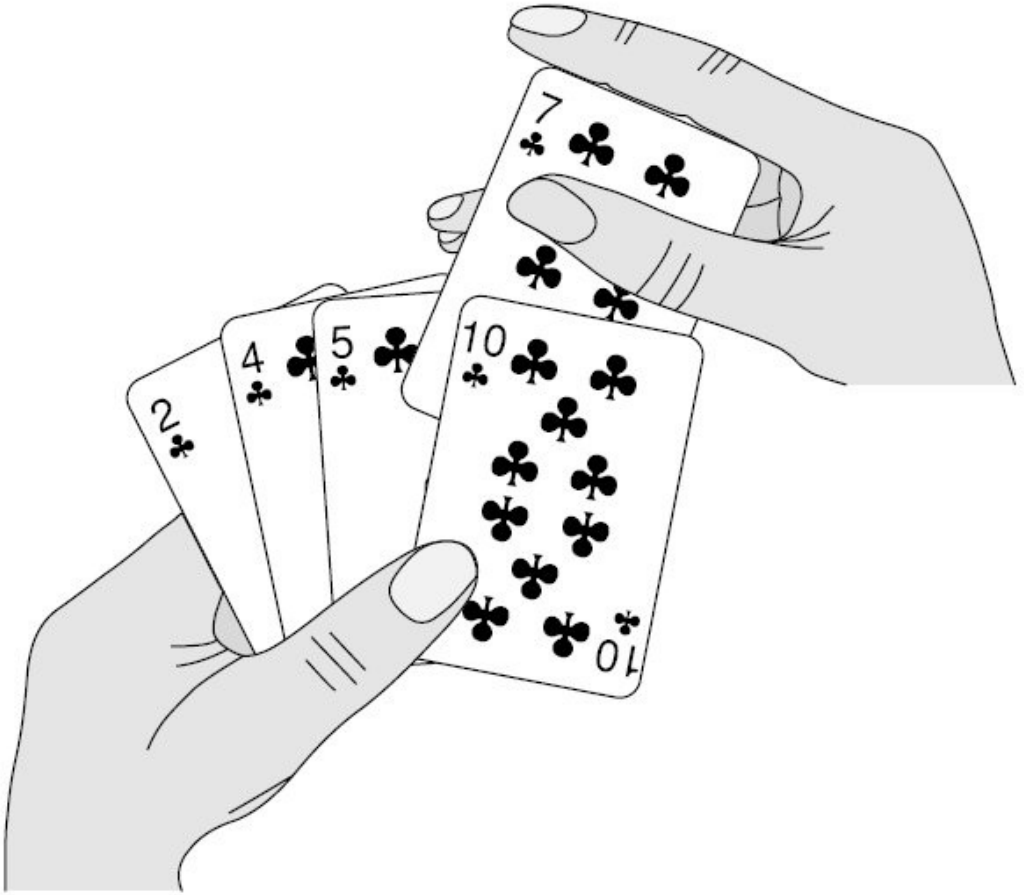
Ripasso algoritmi di ordinamento da IP



Selection Sort

https://it.wikipedia.org/wiki/Selection_sort

L'algoritmo seleziona di volta in volta il numero minore nella sequenza di partenza e lo sposta nella sequenza ordinata; di fatto la sequenza viene suddivisa in due parti: la sottosequenza ordinata, che occupa le prime posizioni dell'array, e la sottosequenza da ordinare, che costituisce la parte restante dell'array.

Selection Sort

https://it.wikipedia.org/wiki/Selection_sort

```
void selectionSort(vector<int>& v)
{
   int current_min_index;
   unsigned int size = v.size();
   for (unsigned int i=0; i<size; ++i)
   {
     current_min_index = i;
     for (unsigned int j=i+1;j<size; ++j)
        if (v[current_min_index] > v[j])
            current_min_index = j;
     scambia(v, i, current_min_index);
   }
}
```

Selection Sort

https://it.wikipedia.org/wiki/Selection_sort

$\Theta(n^2)$ nel caso migliore

$\Theta(n^2)$ nel caso peggiore

Motivazione: la ricerca del minimo nella sottosequenza ancora da ordinare non trae alcun giovamento dal fatto che essa sia già ordinata oppure no: caso migliore e caso peggiore coincidono

Insertion Sort

https://it.wikipedia.org/wiki/Insertion_sort

Si assume che la sequenza da ordinare sia partizionata in una sottosequenza già ordinata, all'inizio composta da un solo elemento, e una ancora da ordinare. Alla k-esima iterazione, la sequenza già ordinata contiene k elementi. In ogni iterazione, viene rimosso un elemento dalla sottosequenza non ordinata (scelto, in generale, arbitrariamente) e inserito (da cui il nome dell'algoritmo) nella posizione corretta della sottosequenza ordinata, estendendola così di un elemento.

Insertion Sort
https://it.wikipedia.org/wiki/Insertion_sort

```
void insertionSort(vector<int>& v)
{
   int current, prev;
   unsigned int size = v.size();
   for (unsigned int i=1; i<size; ++i)
   { current=i;
     prev=i-1;
     while(prev>=0 && v[current]<v[prev])
        {
         scambia(v, current, prev);
         --current;
         --prev;
        }
   }
}
```

Insertion Sort

https://it.wikipedia.org/wiki/Insertion_sort

$\Theta(n)$ nel caso migliore

$\Theta(n^2)$ nel caso peggiore

Bubble Sort

https://it.wikipedia.org/wiki/Bubble_sort

La singola iterazione dell'algoritmo prevede che gli elementi dell'array siano confrontati a due a due, procedendo in un verso stabilito (che si scorra l'array a partire dall'inizio in avanti, o a partire dal fondo all'indietro, è irrilevante; d'ora in poi ipotizzeremo che lo si scorra partendo dall'inizio).

Per esempio, saranno confrontati il primo e il secondo elemento, poi il secondo e il terzo, poi il terzo e il quarto, e così via fino al confronto fra il penultimo e l'ultimo elemento. Ad ogni confronto, se i due elementi confrontati non sono ordinati secondo il criterio prescelto, vengono scambiati di posizione. Durante ogni iterazione almeno un valore viene spostato rapidamente fino a raggiungere la sua collocazione definitiva; in particolare, alla prima iterazione il numero più grande raggiunge l'ultima posizione dell'array, alla seconda il secondo numero più grande raggiunge la penultima posizione, e così via.

Bubble Sort

https://it.wikipedia.org/wiki/Bubble_sort

```
void bubbleSort(vector<int>& v)
{
   unsigned int size = v.size();
   bool scambiati;
   for (unsigned int i=1; i<size; ++i)
      {
      scambiati = false;
      for (unsigned int j=0; j<size-i; ++j)
          if(v[j]>v[j+1])
            {
              scambia(v, j, j+1);
              scambiati = true;
            }
      if (!scambiati) return;
      }
}
```

Bubble Sort

https://it.wikipedia.org/wiki/Bubble_sort

$\Theta(n)$ nel caso migliore

$\Theta(n^2)$ nel caso peggiore