UNIVERSITÀ DEGLI STUDI DI TRIESTE

MODULO 2: Algoritmi

Prof.ssa Giulia Cisotto

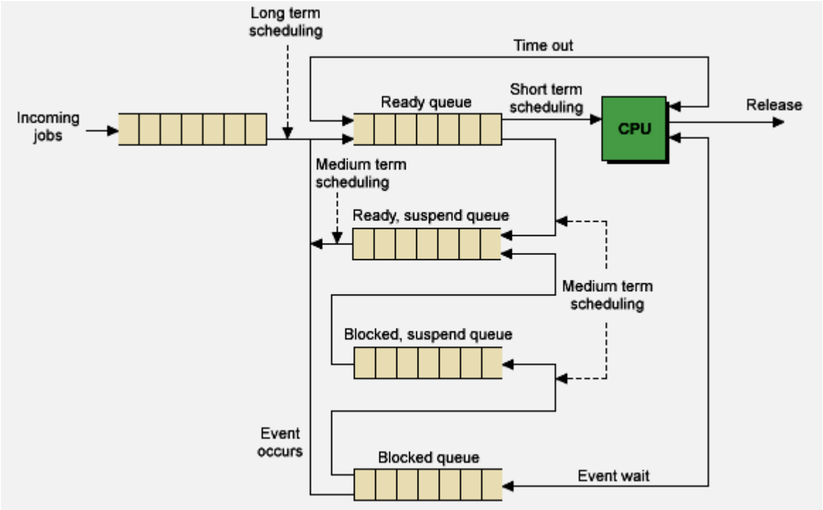
giulia.cisotto@units.it

Trieste, 5 maggio 2026

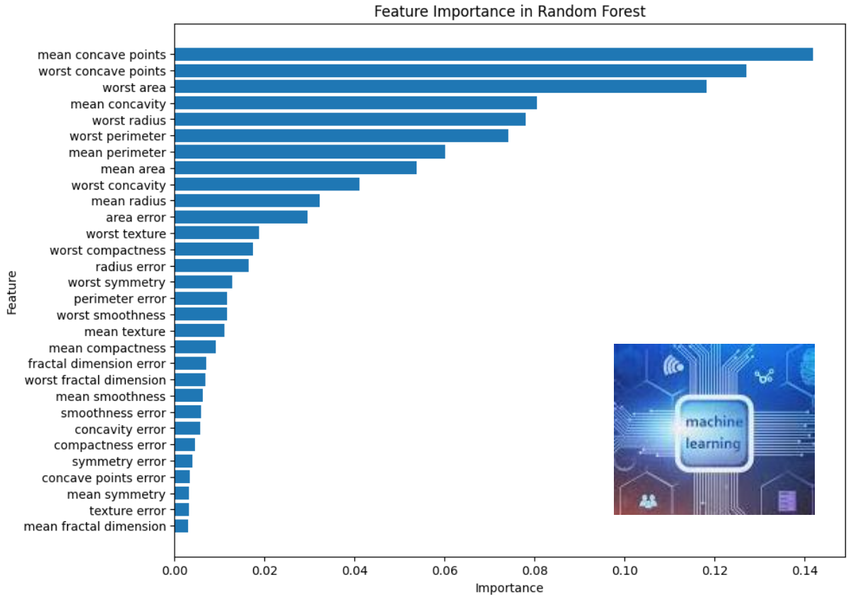
UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

- Algoritmi di ordinamento: introduzione
- Bogosort, Insertion sort, Merge Sort

PROBLEMI «CLASSICI» DELL'INFORMATICA



Scheduling processi da eseguire su CPU



Features più importanti in un task di classificazione

Product Sub Class	Customers	Value
SHIRTS	16,231	$1,950
SPORTS JACKETS	15,678	$2,027
SNEAKERS	9,685	$817
SPORT SHIRTS	7,818	$1,814
DENIM	7,298	$2,660
T-SHIRTS	7,117	$786
SWEATERS	5,408	$2,172
DRESS SHOES	5,060	$2,419
SKIN CARE	4,935	$743
LEATHER JACKETS	4,054	$3,277

Preferenze utenti su un sito di e-commerce

Cos'hanno in comune questi tre scenari?

Devono tutti risolvere un PROBLEMA DI ORDINAMENTO

PROBLEMI «CLASSICI» DELL'INFORMATICA

Il **problema dell'ORDINAMENTO** è uno dei classici problemi dell'informatica:

- **Database**: ordinamento per eseguire query ORDER BY ed efficientare join.
- **Motori di ricerca**: ordinamento dei risultati per rilevanza.
- **E-commerce**: ordinamento dei prodotti per prezzo, popolarità, recensioni.
- **Grafica computerizzata**: ordinamento dei poligoni per il rendering (painter's algorithm).
- **Compressione dati**: algoritmi come Burrows-Wheeler usano ordinamento.
- **Machine Learning**: selezione delle feature più rilevanti, ranking.
- **Sistemi operativi**: scheduling dei processi ordinati per priorità o tempo di arrivo.

Altri problemi: **RICERCA**, **INSERIMENTO, SELEZIONE**, **PERCORSO MINIMO** (per grafi), …

Proprio perché sono «CLASSICI», **esistono degli ALGORITMI FAMOSI che li risolvono.**
Non bisogna scoprire l'acqua calda ogni volta!

ORDINARE UN MAZZO DI CARTE

Abbiamo un mazzo di carte che è stato mescolato

Vogliamo disporre le carte in ordine crescente

Siamo in grado di dare un **algoritmo**?

Iniziamo a formalizzare il problema

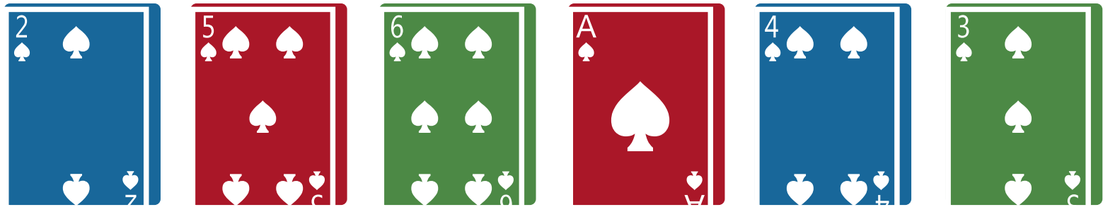
ORDINARE UN MAZZO DI CARTE

Dati in ingresso (**input**)
Un mazzo di carte non necessariamente ordinate

CI SERVE UNA PROCEDURA MECCANICA
CHE DALL'INPUT PRODUCA L'OUTPUT

Dati in uscita (**output**)
Una permutazione del mazzo di carte fornito in input
in cui tutte le carte sono ordinate in ordine crescente

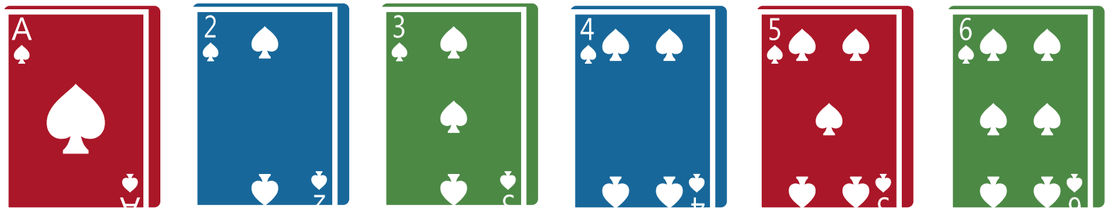
ORDINARE UN MAZZO DI CARTE



Per ogni permutazione delle carte del mazzo

1. Permuta le carte
2. Se le carte sono ordinate termina l’esecuzione
3. Altrimenti riportale alla configurazione iniziale

MOLTO TEMPO DOPO…



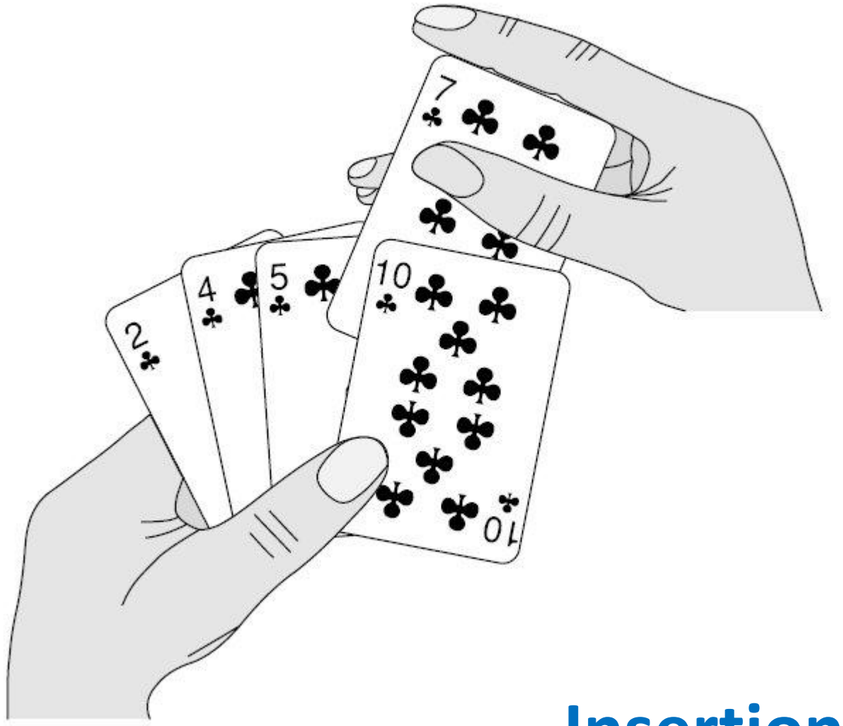
PRESTAZIONI

- **CORRETTEZZA**: l'algoritmo funziona (necessariamente troverà una permutazione per cui le carte sono ordinate).
- **EFFICIENZA**: quante comparazioni dobbiamo fare con n carte?
  - Caso migliore: circa n
    (per verificare che le carte siano ordinate)
  - Caso peggiore: fino ad $n!$
    (ci sono $n!$ permutazioni e potremmo riuscire ad avere le carte ordinate solo all'ultima)

Algoritmo bogosort o stupidsort o monkeysort

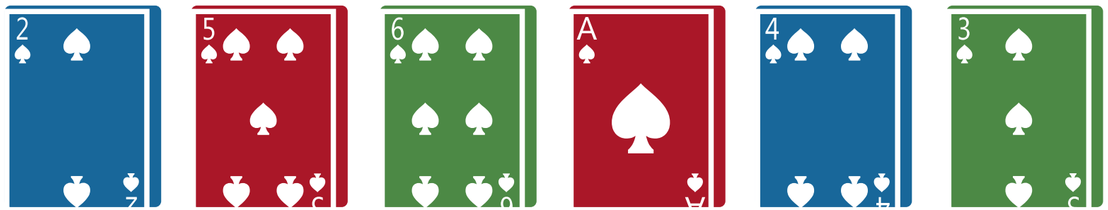
ORDINARE UN MAZZO DI CARTE

Possiamo pensare ad un modo alternativo?



Insertion sort!

INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

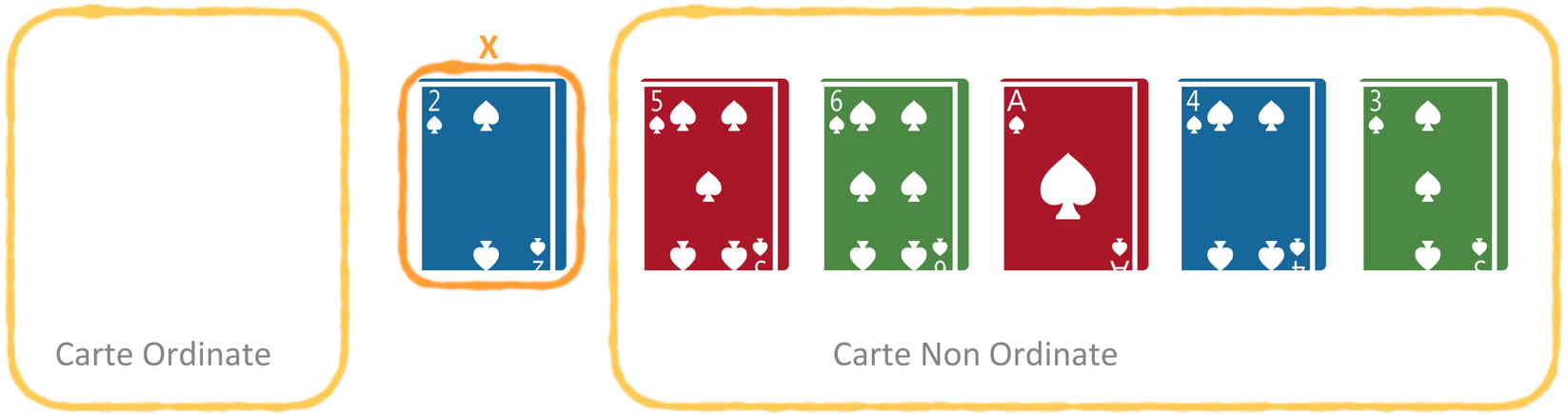
INSERTION SORT: ESEMPIO

1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO

1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO

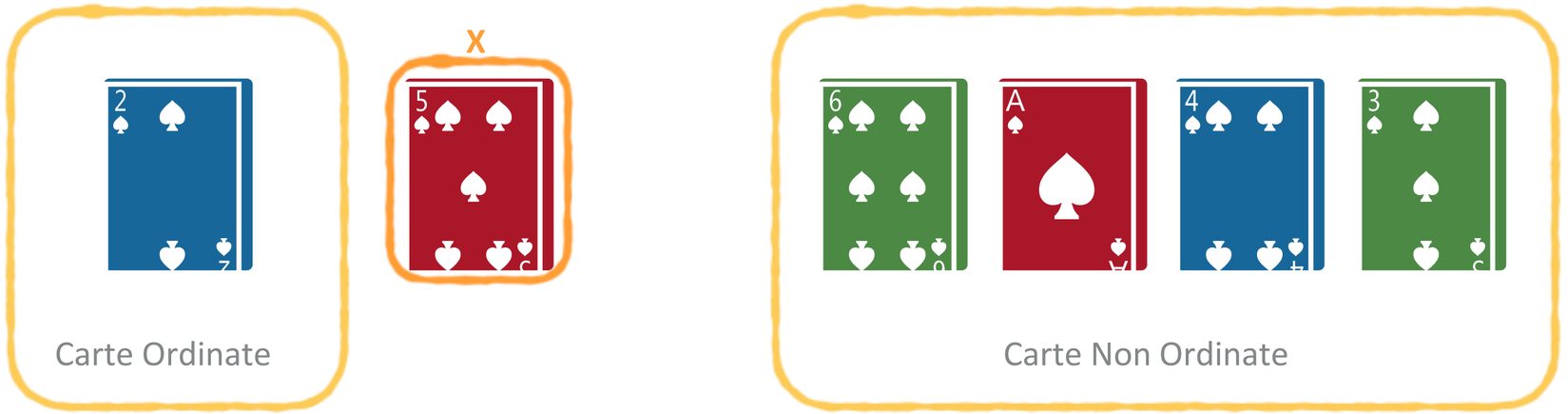
1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO

1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

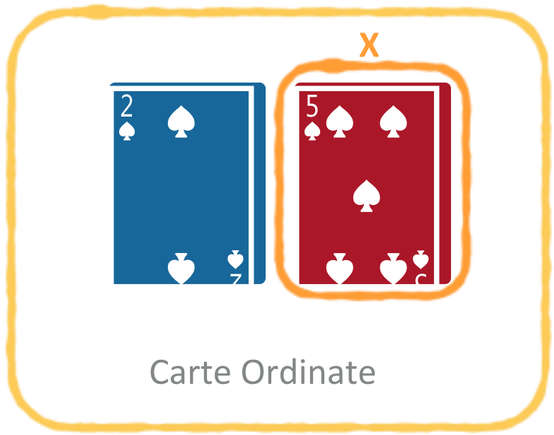
NON DOBBIAMO FARE NIENTE
(NON CI SONO ALTRE CARTE GIÀ IN ORDINE)

INSERTION SORT: ESEMPIO



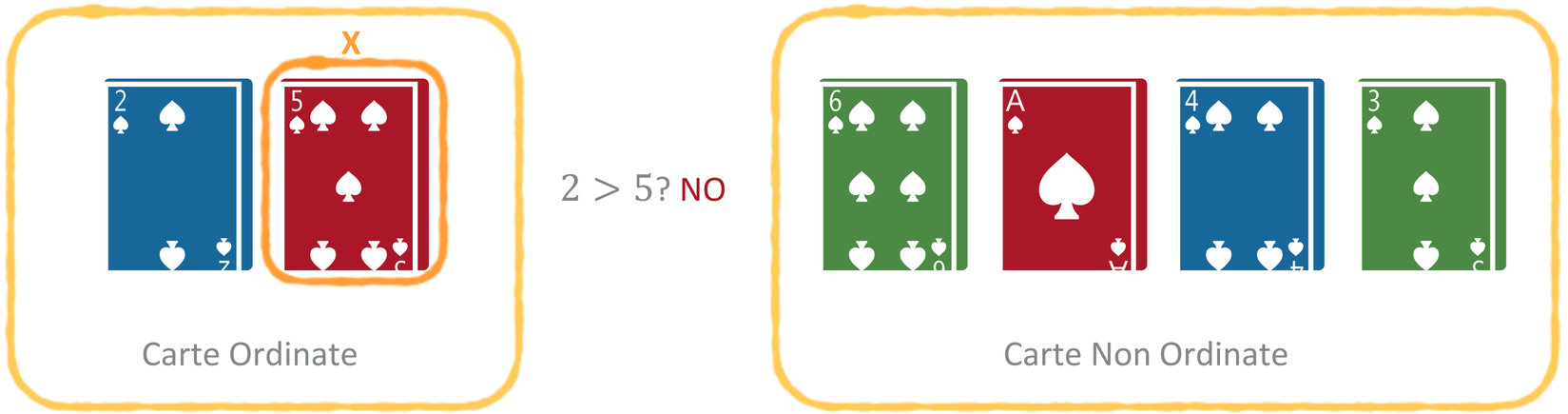
1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

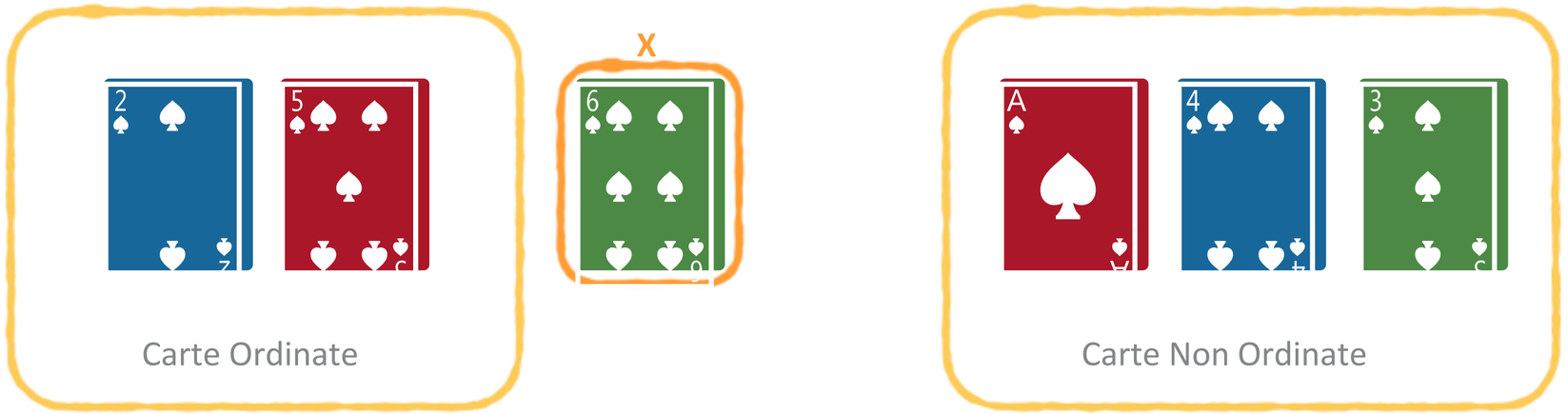
INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

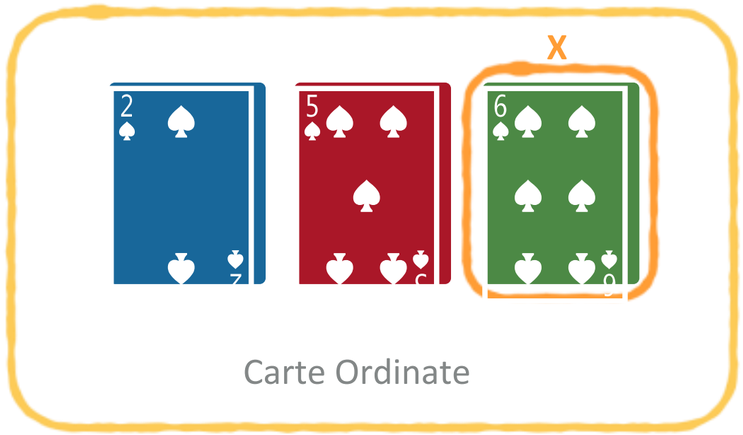
IL TEST FALLISCE
E QUINDI PROSEGUIREMO
CON L'ISTRUZIONE NUMERO…

INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO

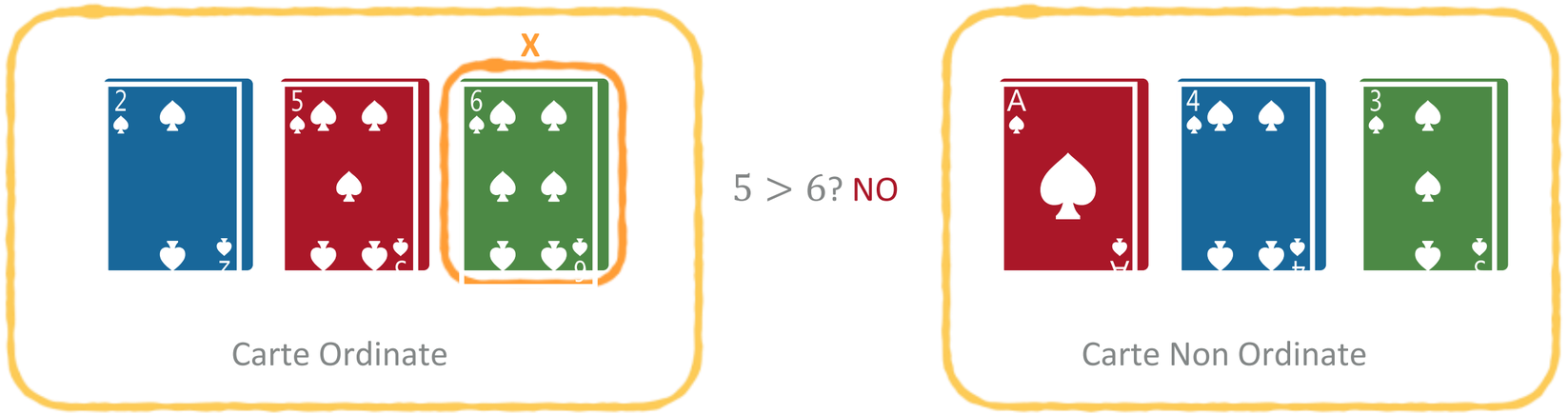


Carte Ordinate

Carte Non Ordinate

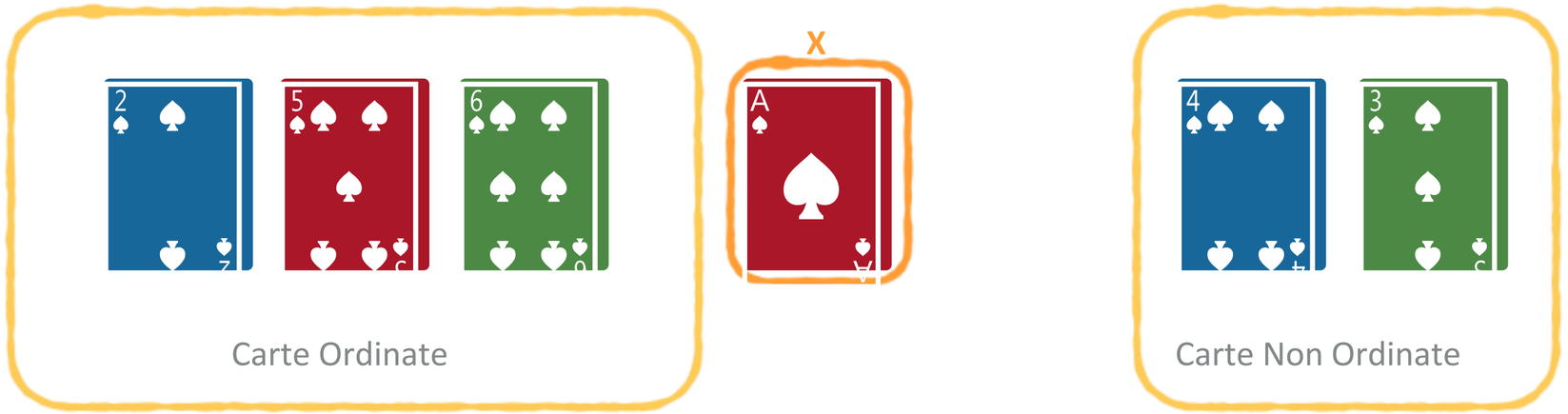
1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO

1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO

$6 > 1$? SI

1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate

QUI INVECE IL TEST HA SUCCESSO QUINDI DOBBIAMO ESEGUIRE L'OPERAZIONE DI SCAMBIO

    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

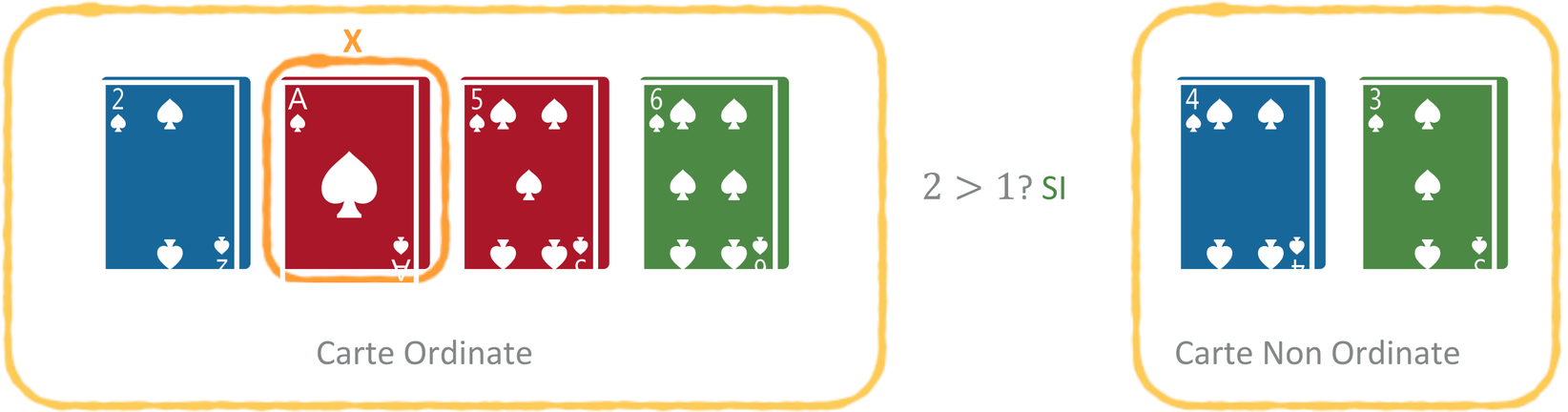
INSERTION SORT: ESEMPIO

$5 > 1$? SI

1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

ORA CHE DOBBIAMO FARE?
CONTINUIAMO CON L'ISTRUZIONE 2.3
O TORNIAMO ALLA 2.1?

INSERTION SORT: ESEMPIO

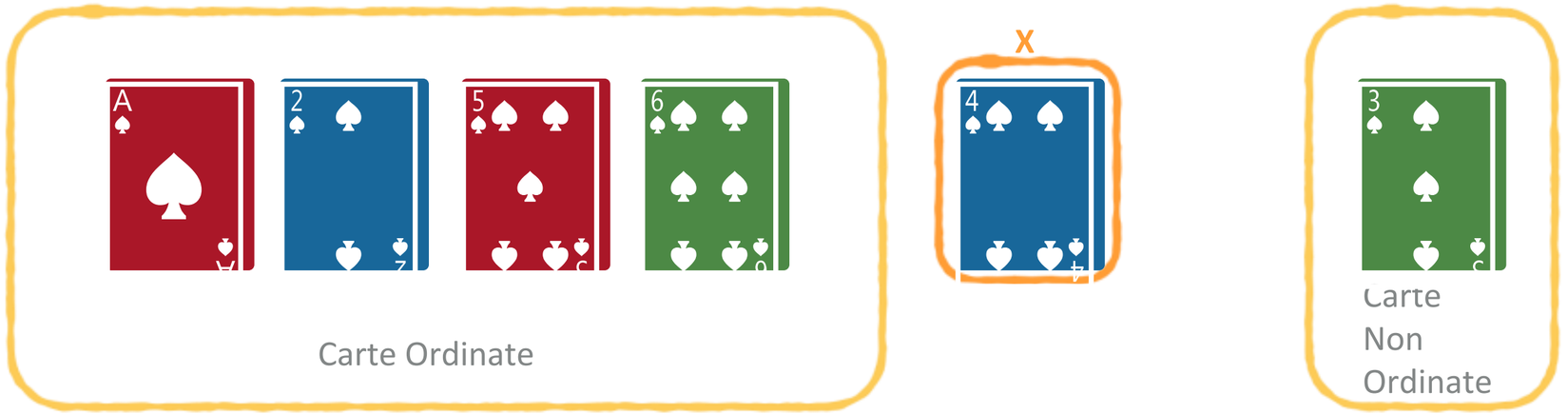


1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO

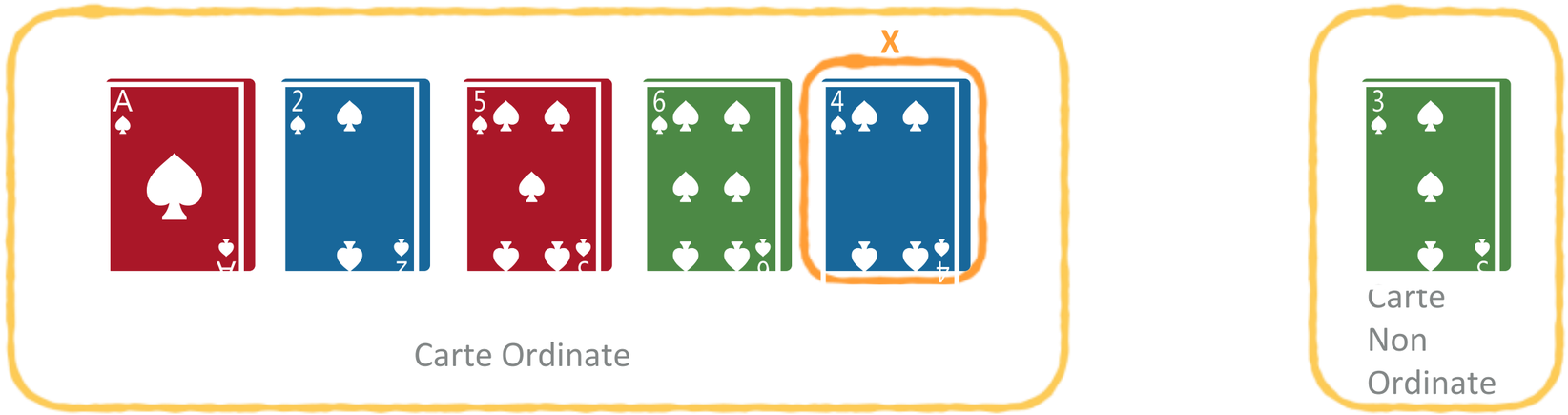
1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



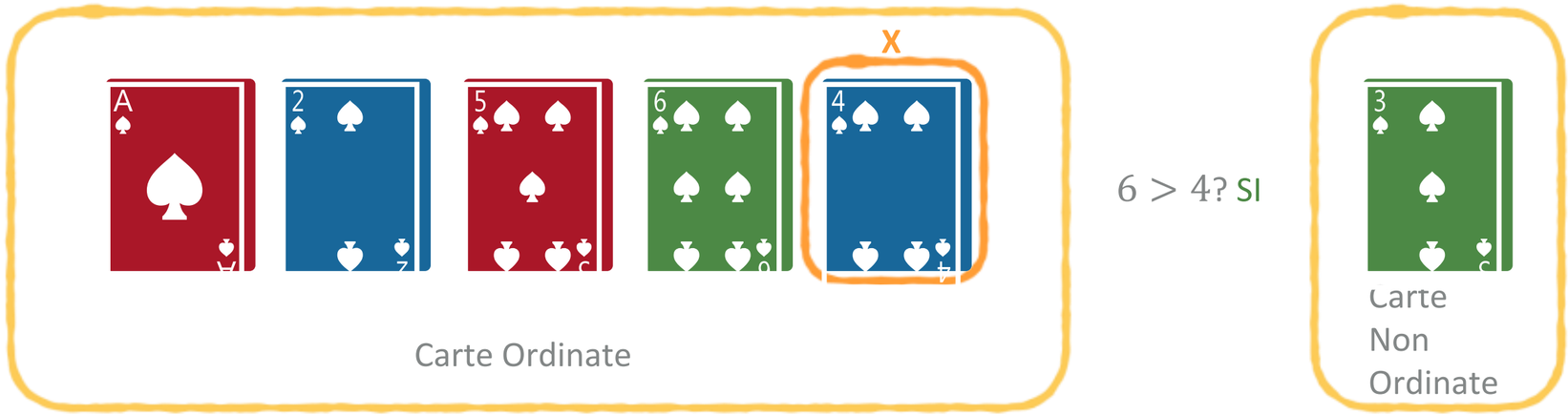
1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



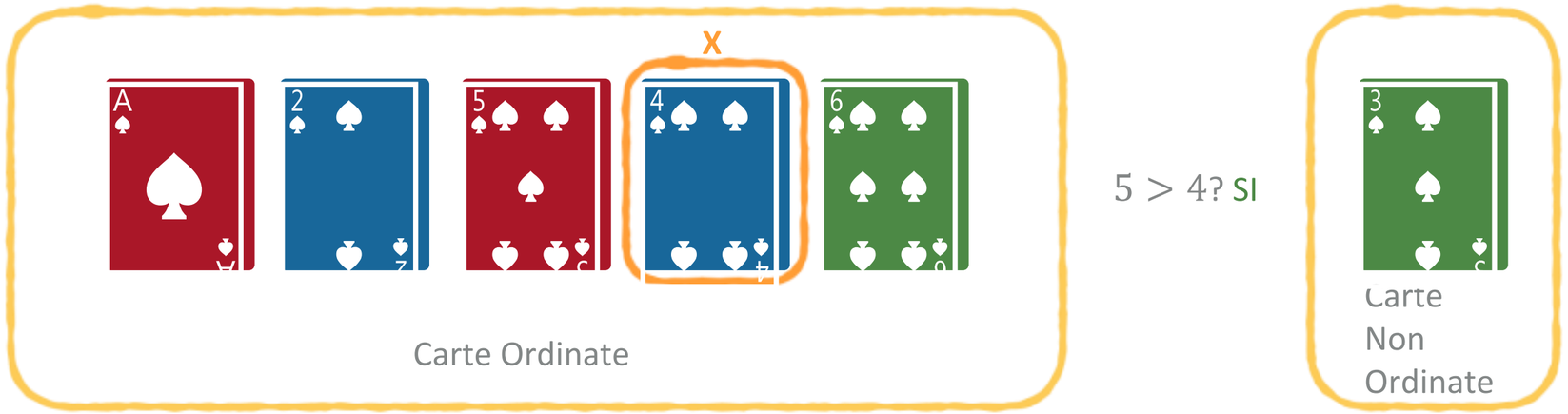
1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



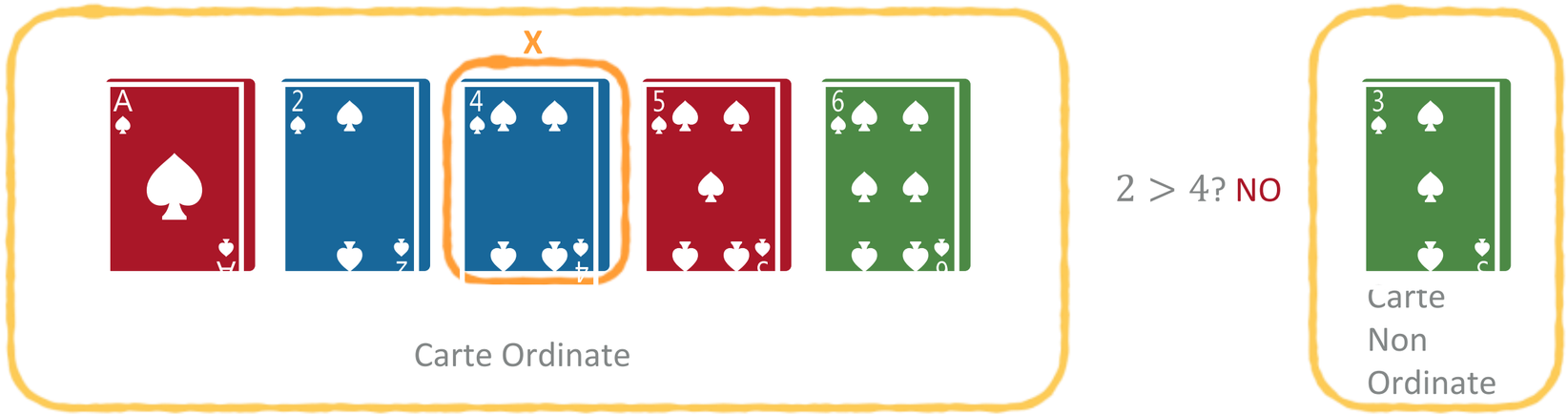
1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



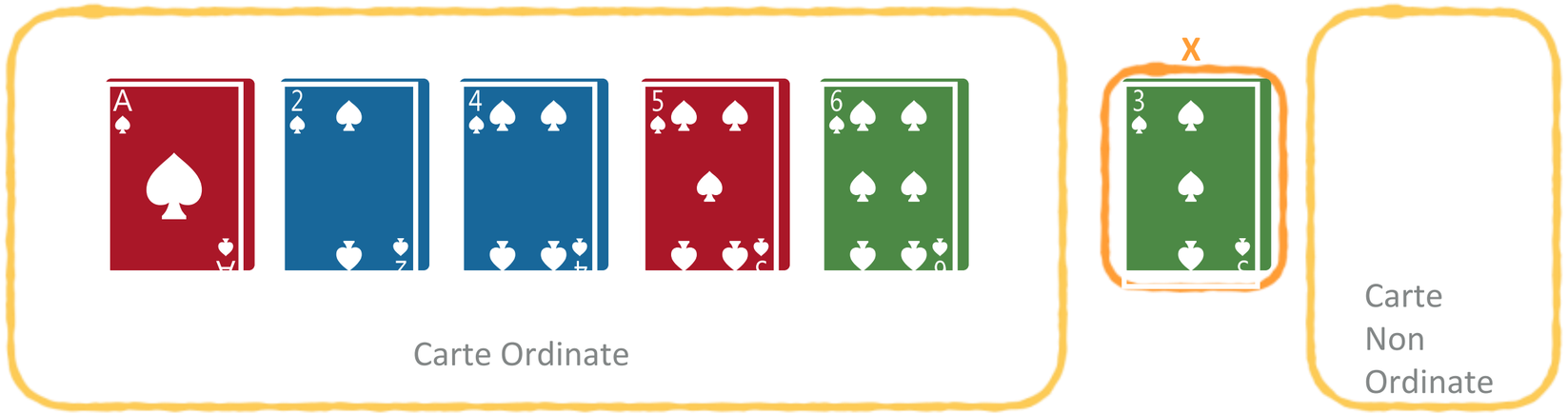
1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO

Carte
Non
Ordinate

1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

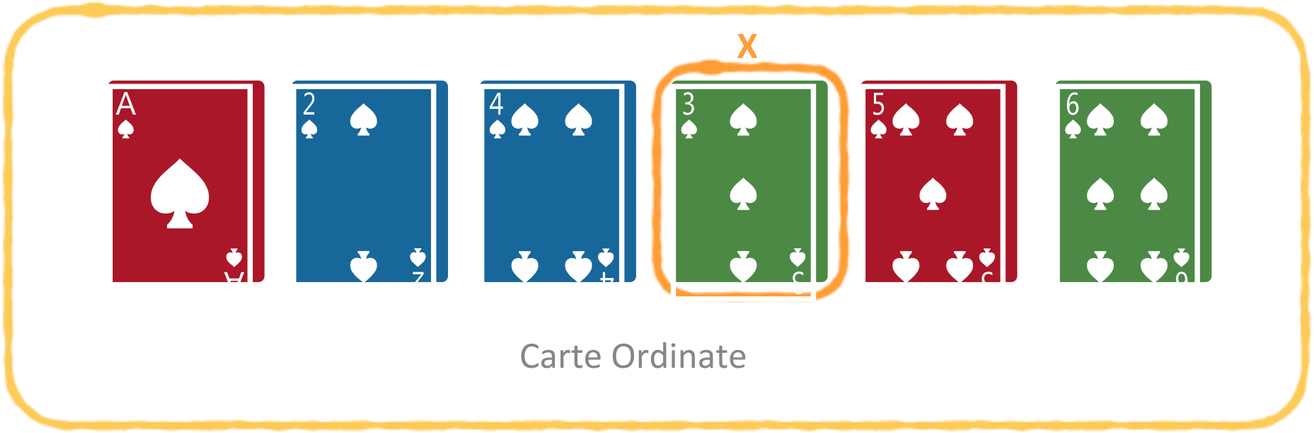
INSERTION SORT: ESEMPIO

1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO

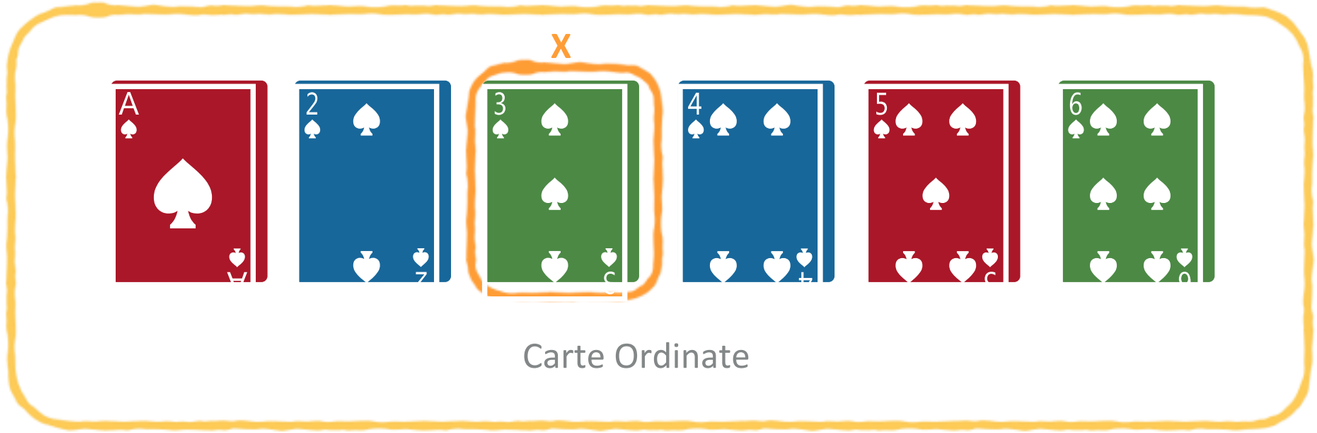
1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

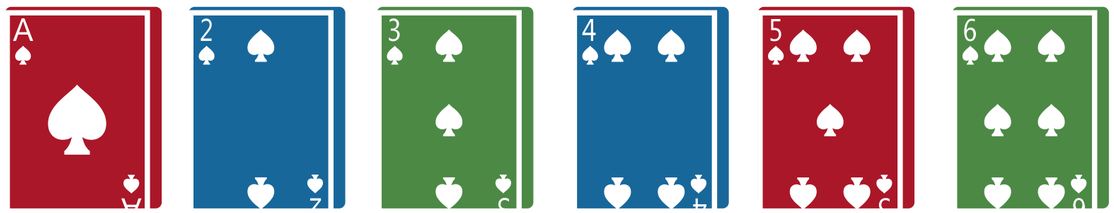
INSERTION SORT: ESEMPIO



1. Inizialmente non si hanno carte ordinate
2. Finché ci rimangono carte non ordinate
    1. Prendi una carta dalla lista di carte non ordinata e chiamala X
    2. Metti X in fondo alla lista di carte ordinate
    3. Finché la carta che precede X ha valore maggiore del valore di X, scambia X con la carta che lo precede

ORA LA CONDIZIONE NON È PIÙ VERA E ABBIAMO TERMINATO

INSERTION SORT: ESEMPIO



Carte Ordinate

Siamo riusciti ad ottenere un mazzo di carte ordinato

Rimangono alcune domande:

- Correttezza: questo algoritmo funziona con ogni input?
- Efficienza: quanti passi ci mettiamo ad ordinare un mazzo di n carte al crescere di n?

PRESTAZIONI

- **CORRETTEZZA**: l'algoritmo funziona, esiste senz'altro una soluzione per ogni possibile *istanza* del problema
- **EFFICIENZA**: quante comparazioni dobbiamo fare con n carte?
  - Caso migliore: circa n
    (le estraiamo dall'insieme di partenza già ordinate)
  - Caso peggiore: al massimo $n * n$
    (dobbiamo al massimo fare n confronti per ogni carta per posizionarla nell'ordine giusto)

PSEUDOCODICE

```
INSERTION-SORT(A,n)
for i = 2 to n          //array indicizzato a partire da 1
     key = A[i]         //memorizza A[i] nella variabile key
     j = i-1            //elemento precedente
     while j>0 && A[j]>key
          A[j+1] = A[j]
          j=j-1
     A[j+1] = key       //inserisce A[i] nella sequenza ordinata A[1:i-1]
```

Note:
- Attenzione che nel libro c'è un typo (una «i» che invece deve essere «j»)
- Non vengono usate due liste separate ma si fa tutto all'interno dello stesso array A

ESEMPIO (ITERAZIONE #1)

A = [2 4 5 3]

i = 2

key ← A[i] = A[2] = 4

j = i-1 = 1

Controllo j>0 && A[**j**]=A[1]=2 > key=4: No! ➔ non entro nel ciclo

A[j+1] = A[2] ← key = 4

A = [2 4 5 3] //rimasto invariato

ESEMPIO (ITERAZIONE #2)

```
i = 3

key ← A[i] = A[3] = 5

j = i-1 = 2

Controllo j>0 && A[j]=A[2]=4 > key=5: No! ➔ non entro nel ciclo

A[j+1] = A[3] ← key = 5


A = [2 4 5 3] //rimasto invariato
```

ESEMPIO (ITERAZIONE #3)

```
i = 4
key ← A[i] = A[4] = 3
j = i-1 = 3
Controllo j>0 && A[j]=A[3]=5 > key=3: Sì! ➔ questa volta entro nel ciclo
   A[j+1] = A[4] ← A[j]=A[3]=5   //diventa: A[2 4 3 5]
   j = 2
Controllo j>0 && A[j]=A[2]=4 > key=3: Sì! ➔ di nuovo entro nel ciclo
   A[j+1] = A[3] ← A[j]=A[2]=4 //diventa: A[2 3 4 5]
   j = 1
A[j+1] = A[2] ← key = 3
A = [2 3 4 5] //ordinato!
```

ORDINARE UN MAZZO DI CARTE

Possiamo pensare ad un algoritmo più efficiente?

Mergesort!

MERGESORT: ALGORITMO

1. Se il mazzo ha 1 carta è ordinato
2. Altrimenti spezza il mazzo in due parti $\lceil n/2 \rceil$ e $\lfloor n/2 \rfloor$
3. Applica questa procedura ad entrambe le parti
4. Unisci le due parti A e B seguendo la procedura MERGE,
   per unire due mazzi ordinati in un unico mazzo anch'esso ordinato

DOPO QUESTO PASSO
LE DUE PARTI SONO ORDINATE

LA PROCEDURA DI MERGE VERRÀ
DETTAGLIATA NELLE PROSSIME SLIDE

MERGESORT: PROCEDURA MERGE

***Ipotesi**: i due mazzi sono ordinati*

Mazzo A

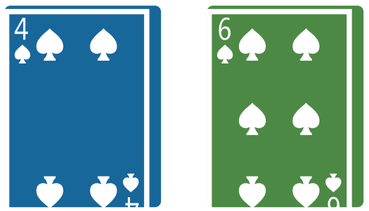
Mazzo B

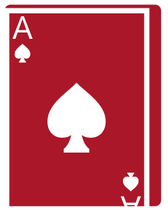
Mazzo unificato

MERGESORT: PROCEDURA MERGE

Mazzo A

Mazzo B

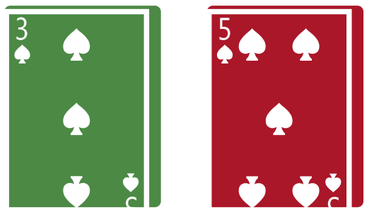




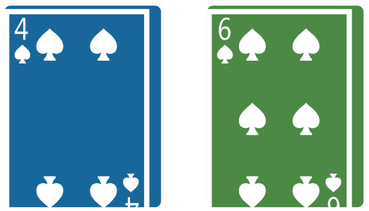
Mazzo unificato

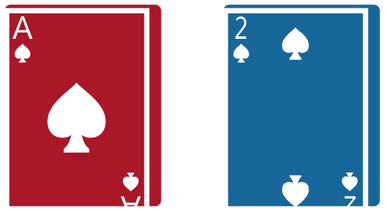
MERGESORT: PROCEDURA MERGE

Mazzo A



Mazzo B

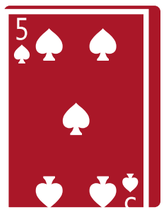




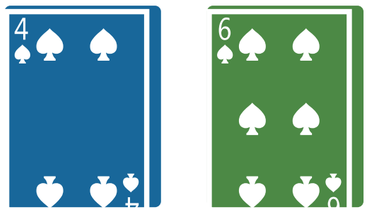
Mazzo unificato

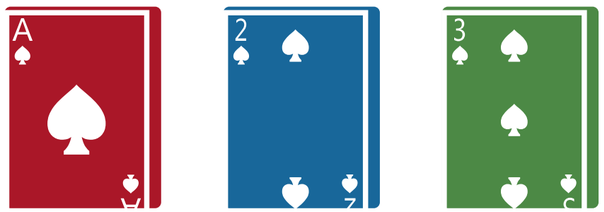
MERGESORT: PROCEDURA MERGE

Mazzo A



Mazzo B

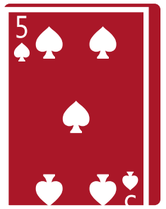




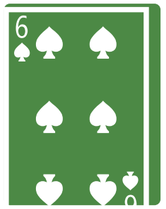
Mazzo unificato

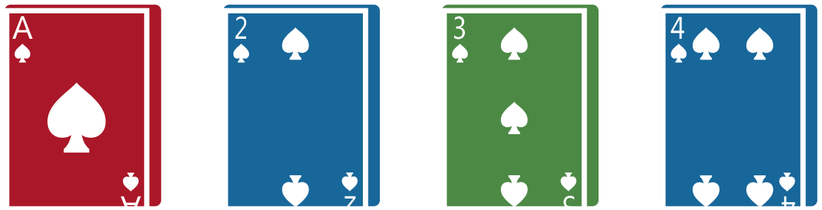
MERGESORT: PROCEDURA MERGE

Mazzo A



Mazzo B



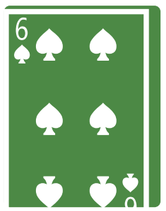


Mazzo unificato

MERGESORT: PROCEDURA MERGE

Mazzo A

Mazzo B

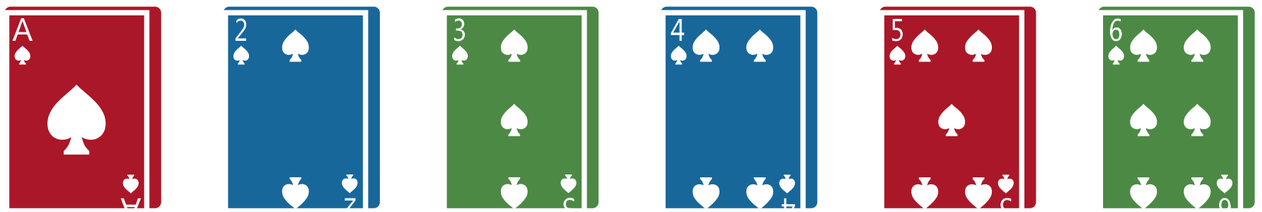


Mazzo unificato

MERGESORT: PROCEDURA MERGE

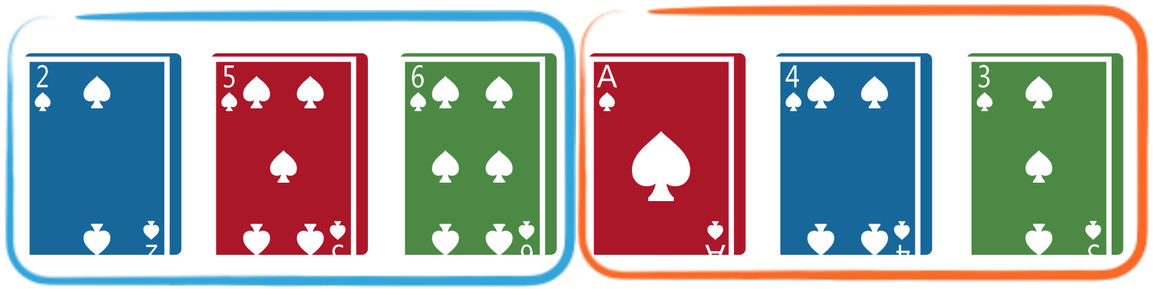
Mazzo A

Mazzo B



Mazzo unificato

MERGESORT: ESEMPIO

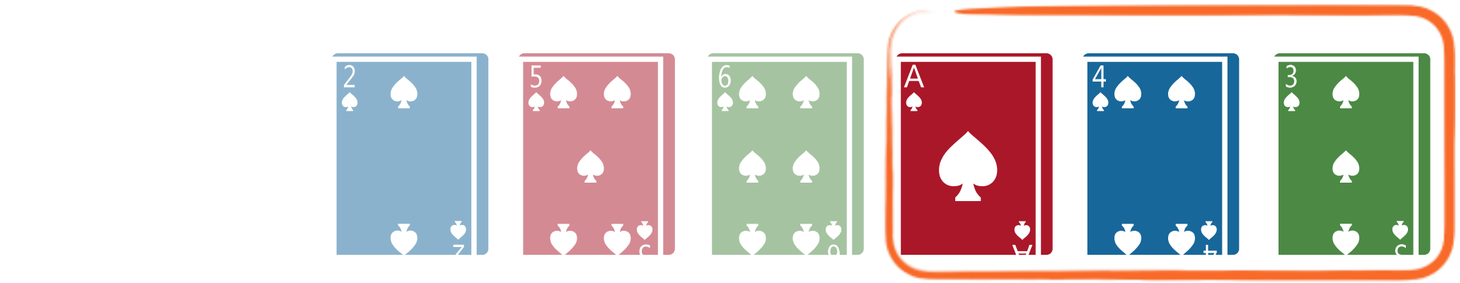


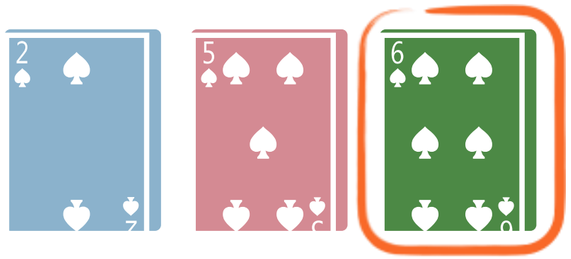
SPEZZARE IL MAZZO

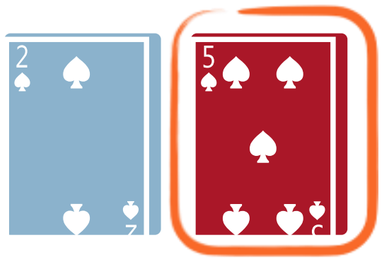
MERGESORT: ESEMPIO

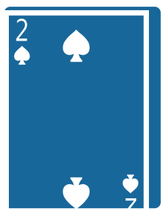
MERGESORT: ESEMPIO

MERGESORT: ESEMPIO



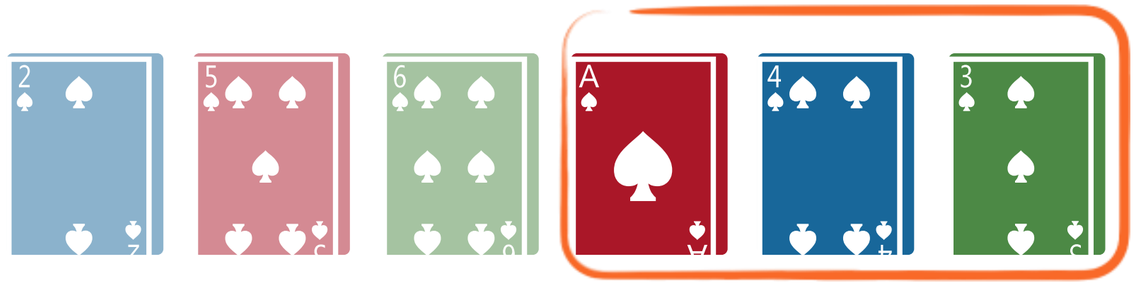


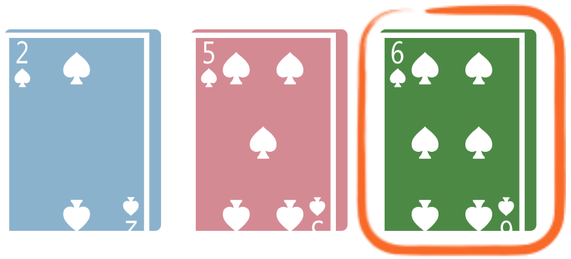


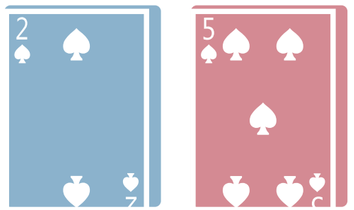


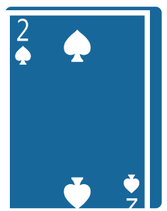
QUESTO MAZZO È COMPOSTO
DA UNA CARTA SOLA
E QUINDI È ORDINATO

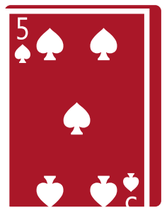
MERGESORT: ESEMPIO





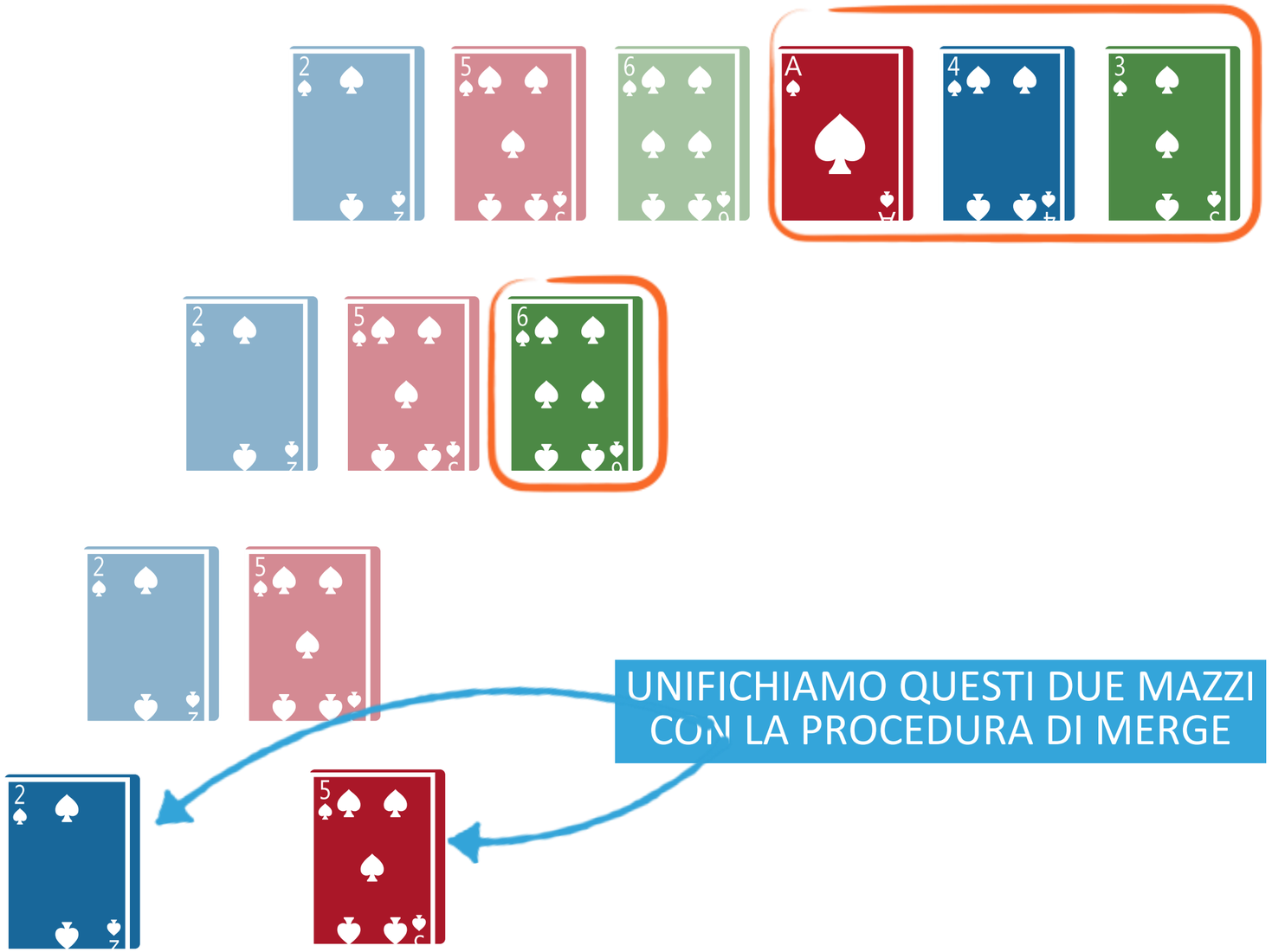




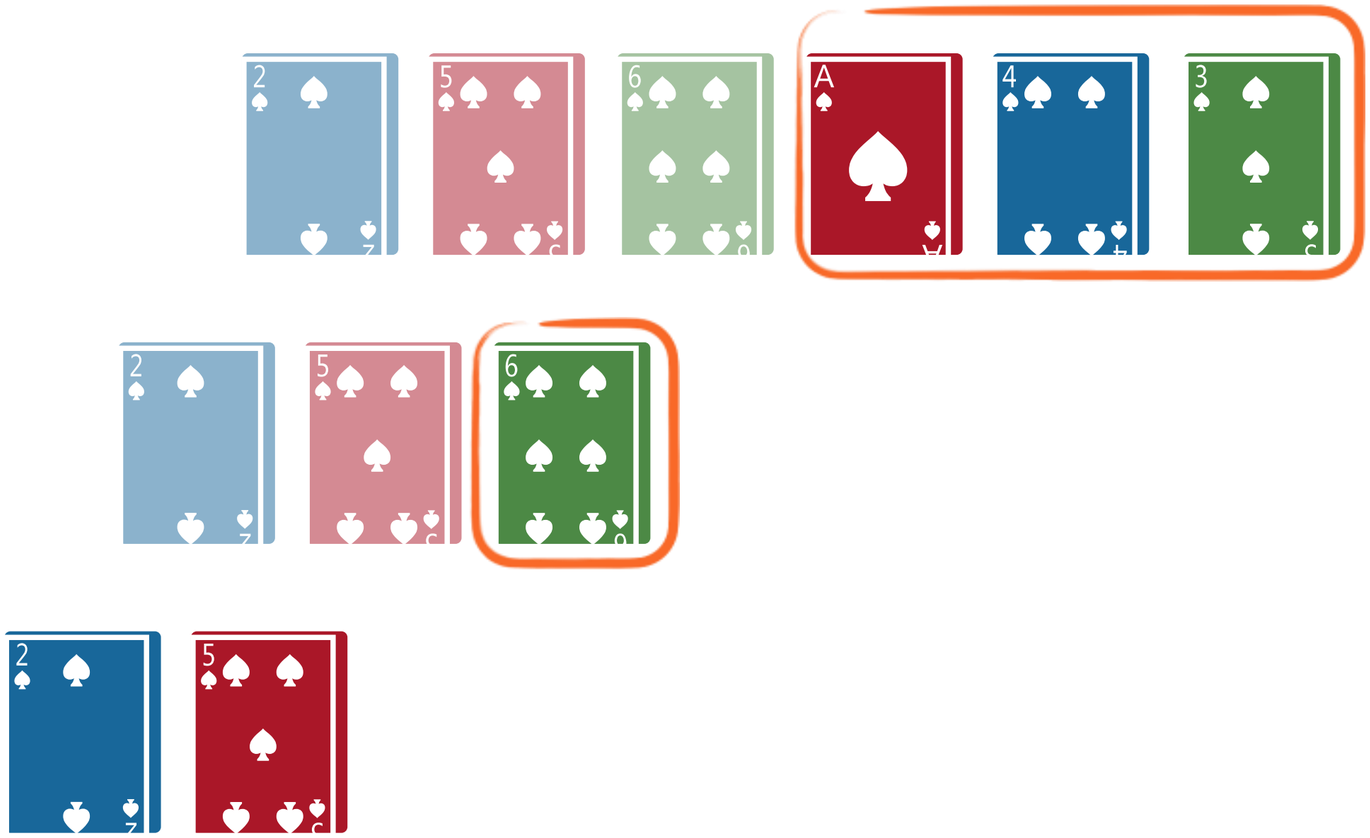


QUESTO MAZZO È COMPOSTO
DA UNA CARTA SOLA
E QUINDI È ORDINATO

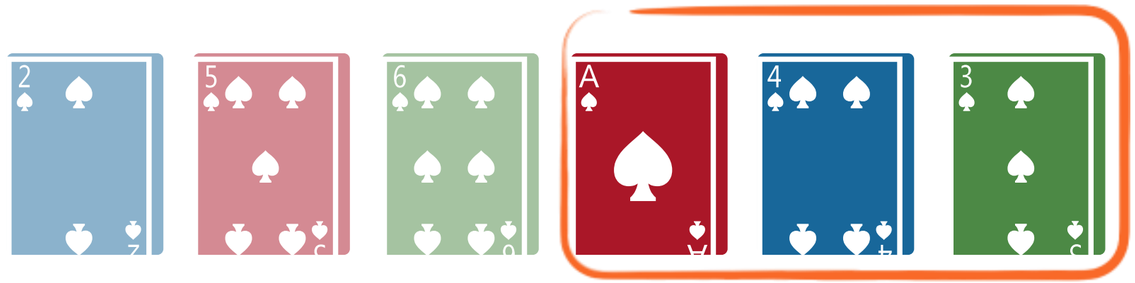
MERGESORT: ESEMPIO

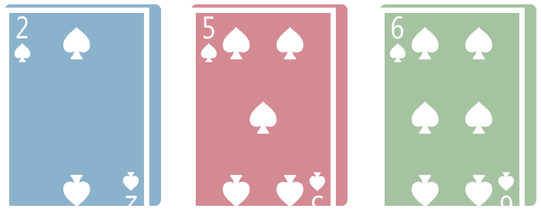


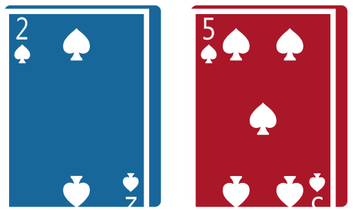
MERGESORT: ESEMPIO

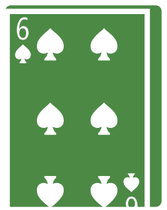


MERGESORT: ESEMPIO



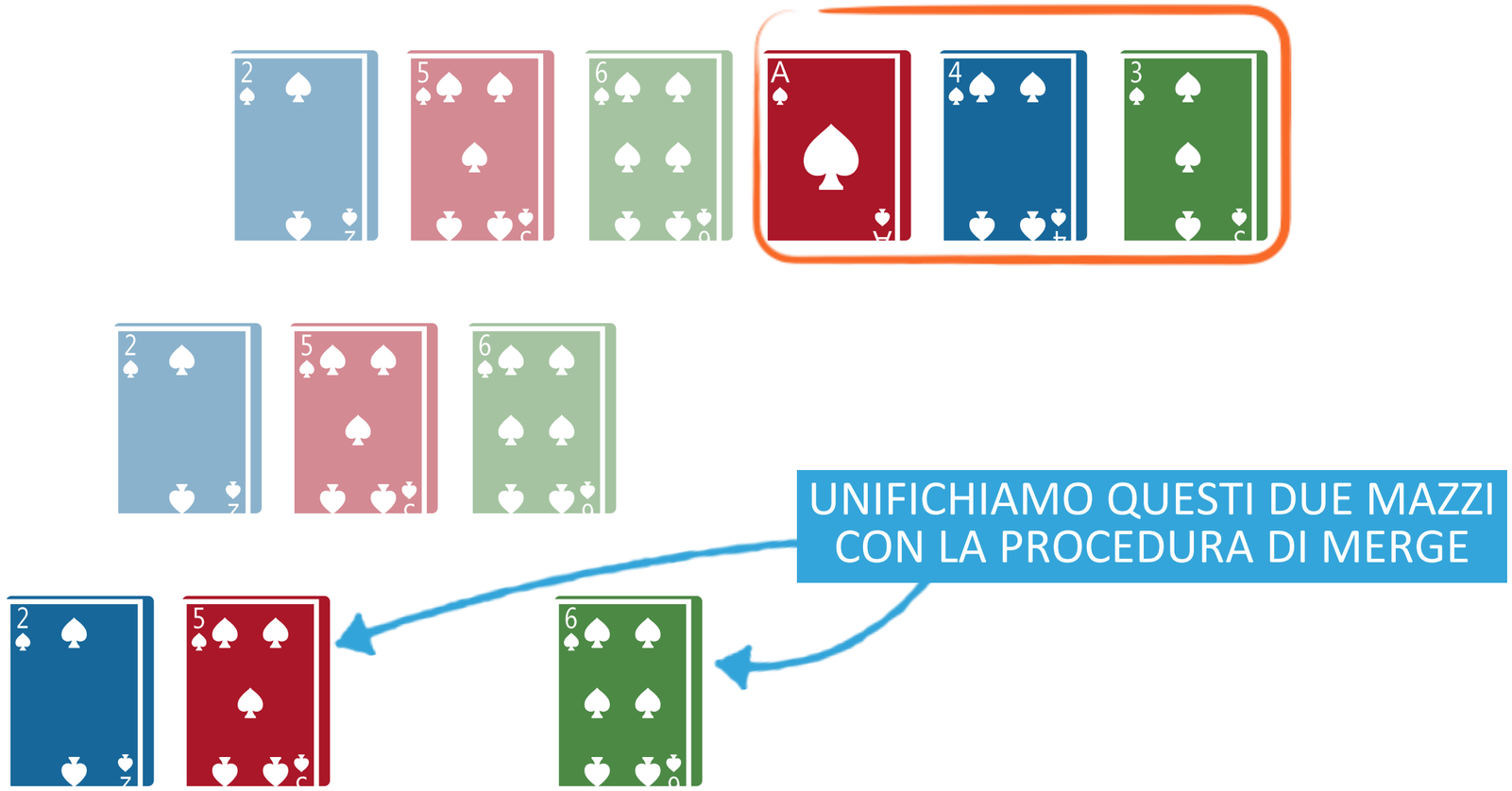




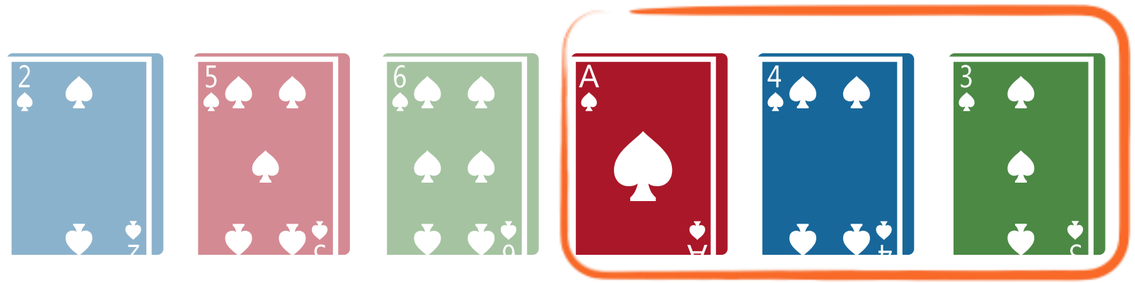


QUESTO MAZZO È COMPOSTO
DA UNA CARTA SOLA
E QUINDI È ORDINATO

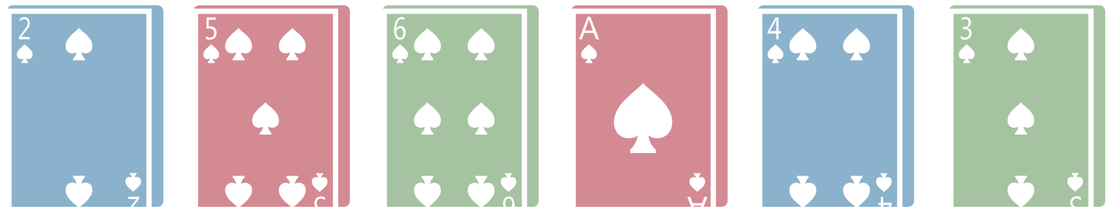
MERGESORT: ESEMPIO

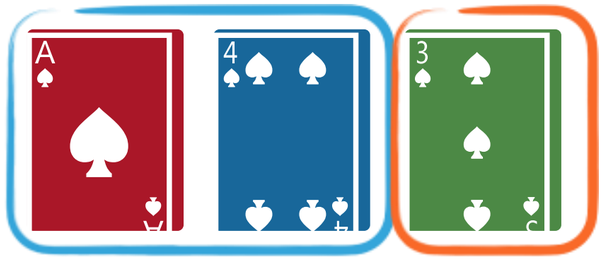


MERGESORT: ESEMPIO



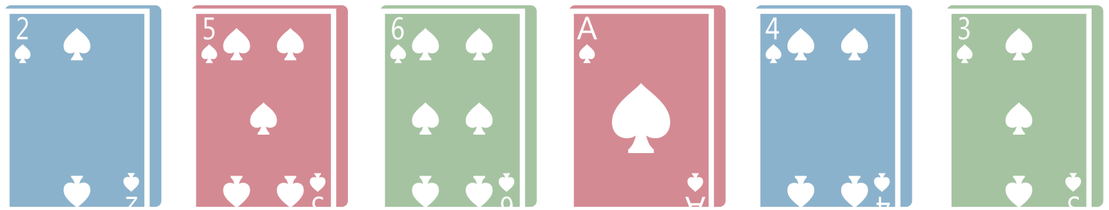
MERGESORT: ESEMPIO

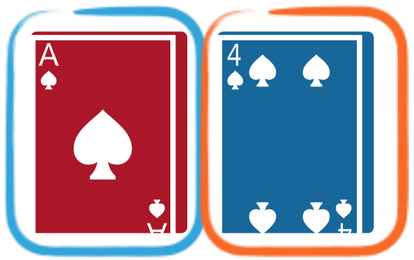




SPEZZARE IL MAZZO

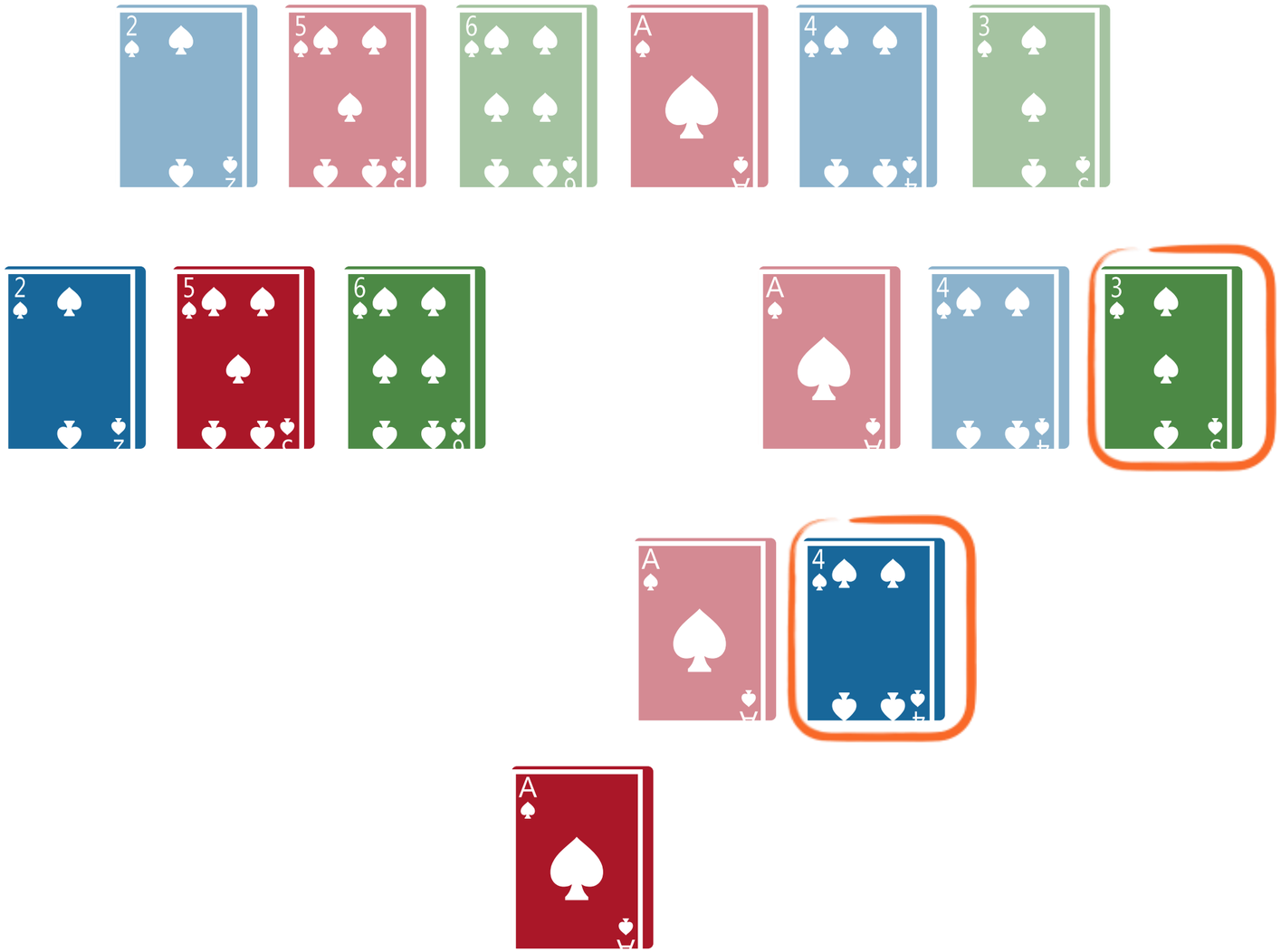
MERGESORT: ESEMPIO



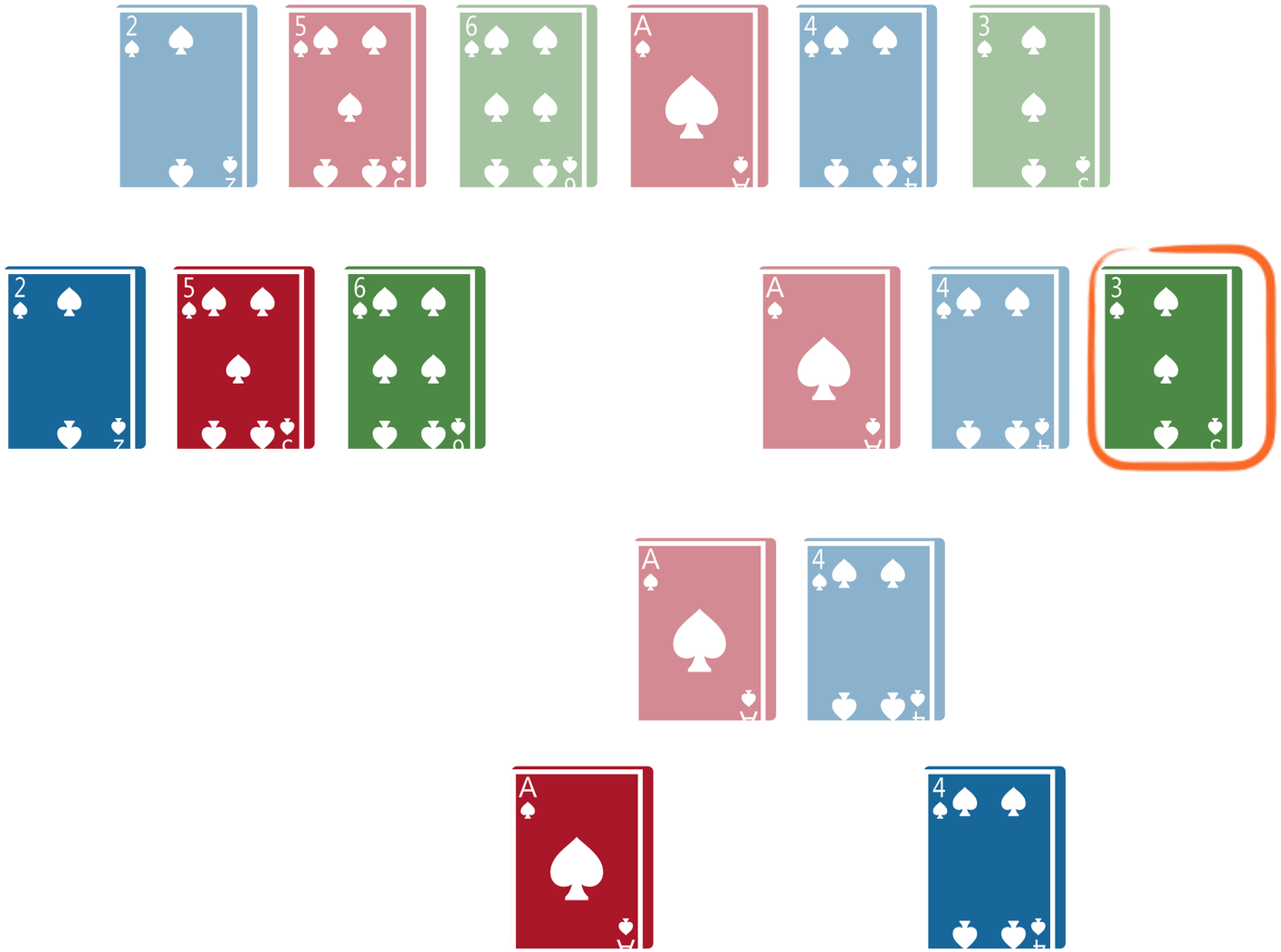


SPEZZARE IL MAZZO

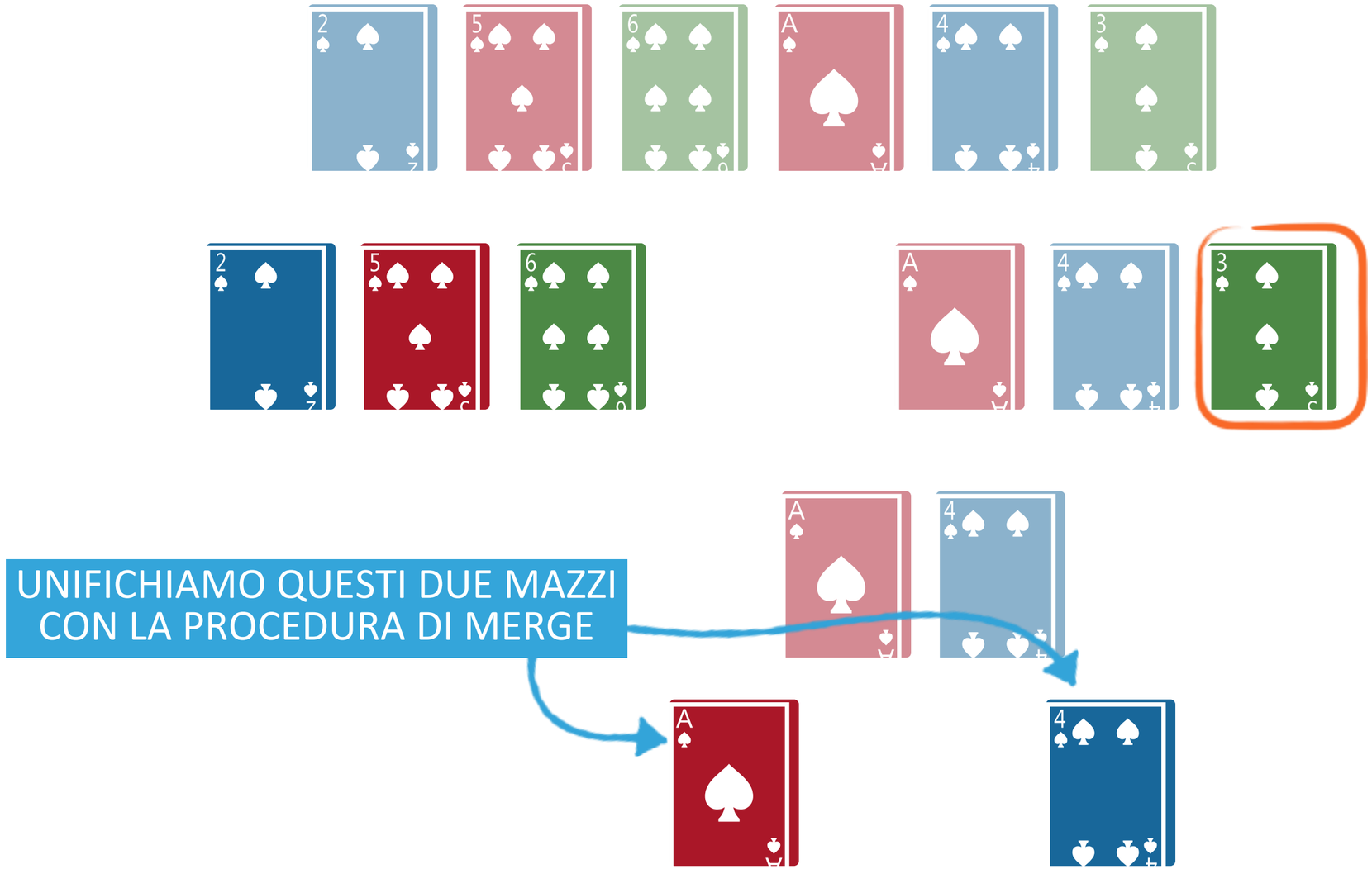
MERGESORT: ESEMPIO



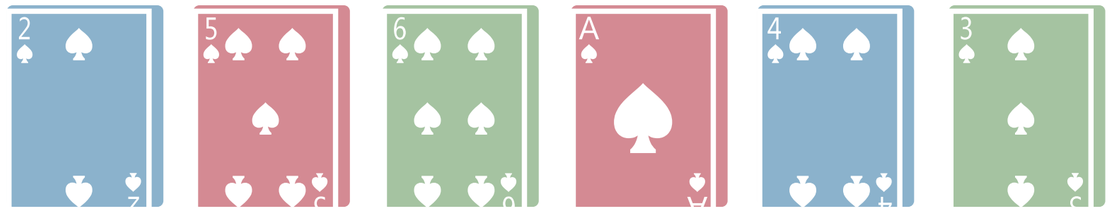
MERGESORT: ESEMPIO

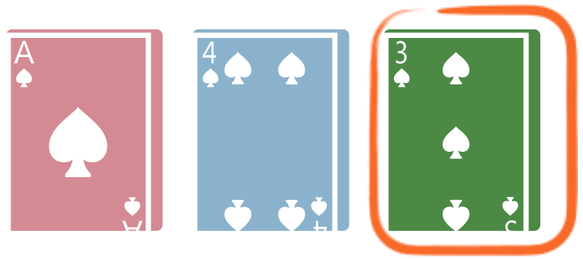


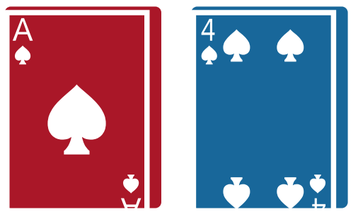
MERGESORT: ESEMPIO



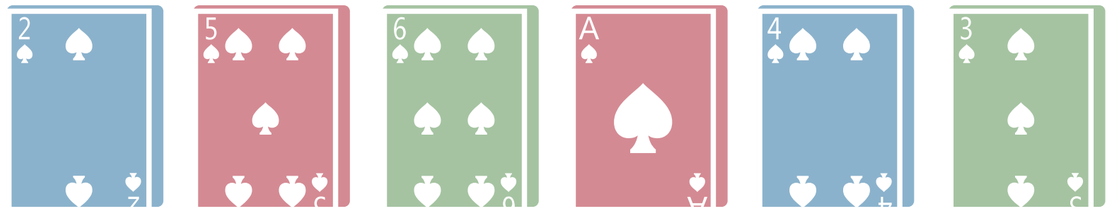
MERGESORT: ESEMPIO

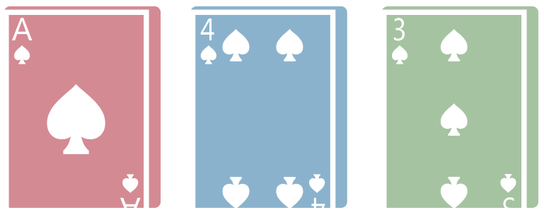


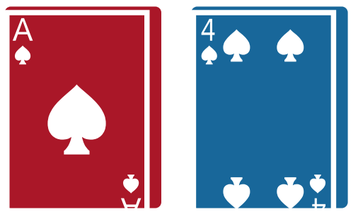


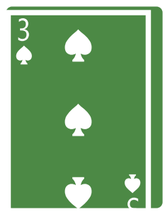


MERGESORT: ESEMPIO

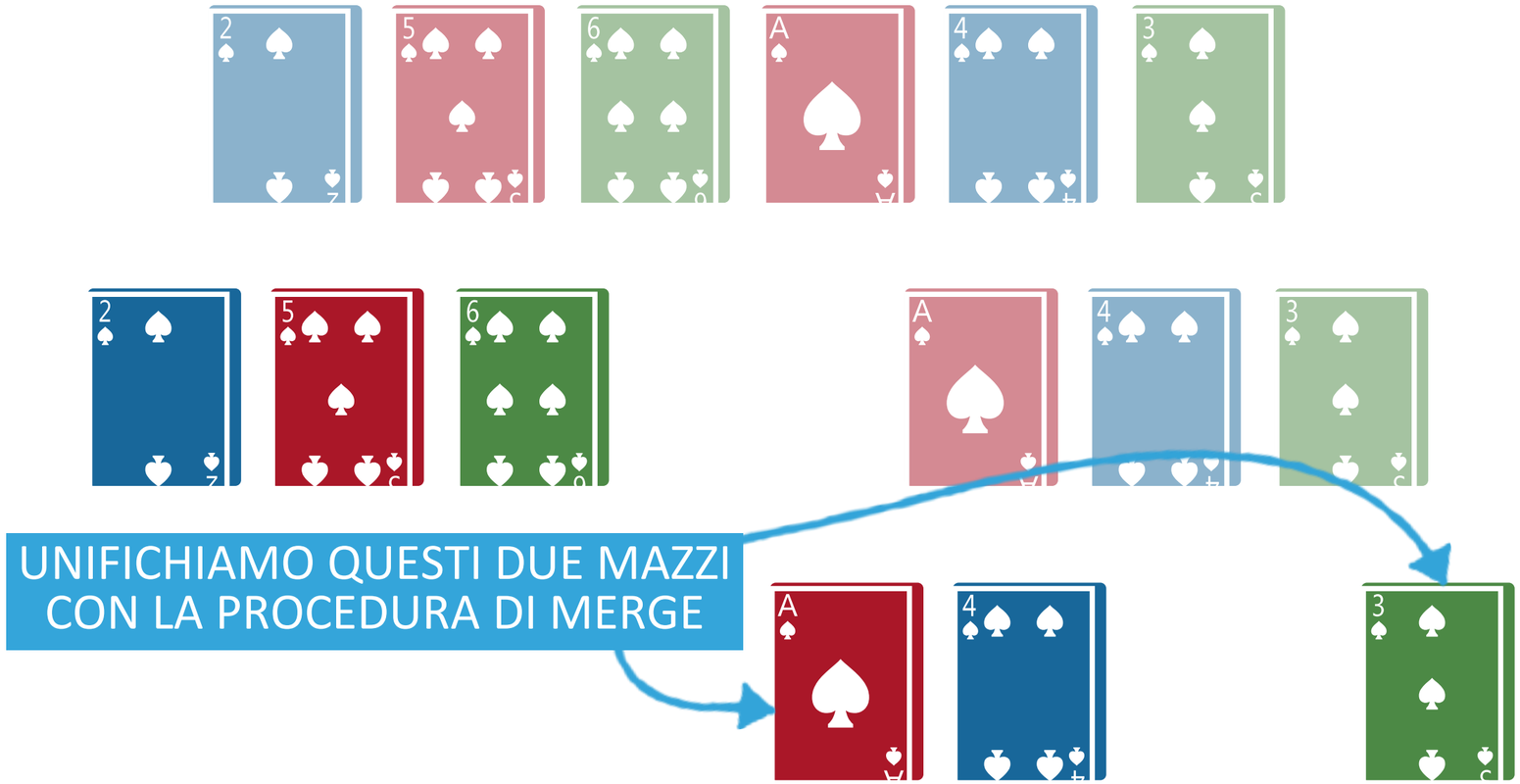




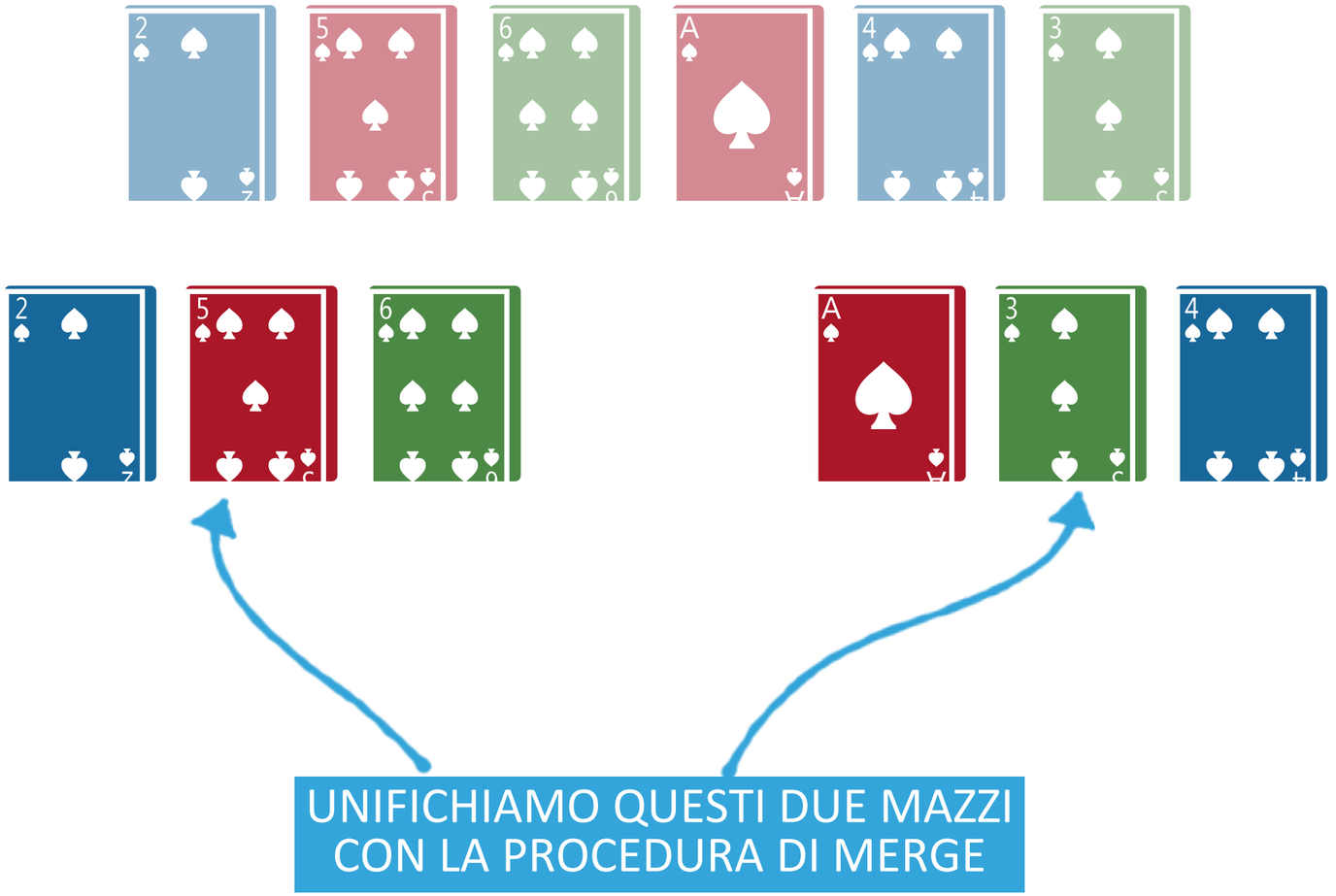




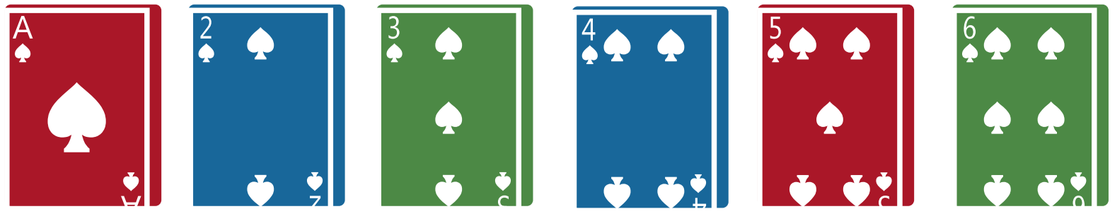
MERGESORT: ESEMPIO



MERGESORT: ESEMPIO



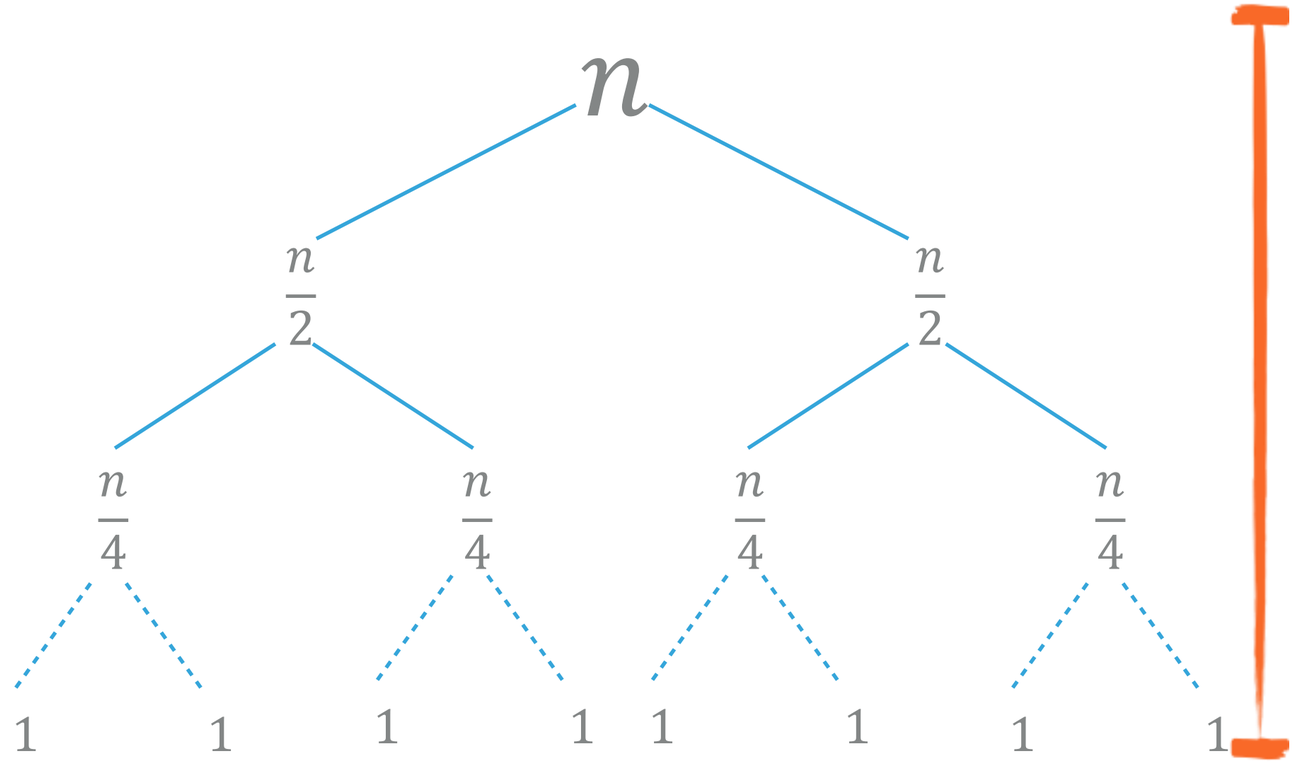
MERGESORT: ESEMPIO



Abbiamo ottenuto il mazzo ordinato!

Ma quale è il costo in numero di comparazioni?

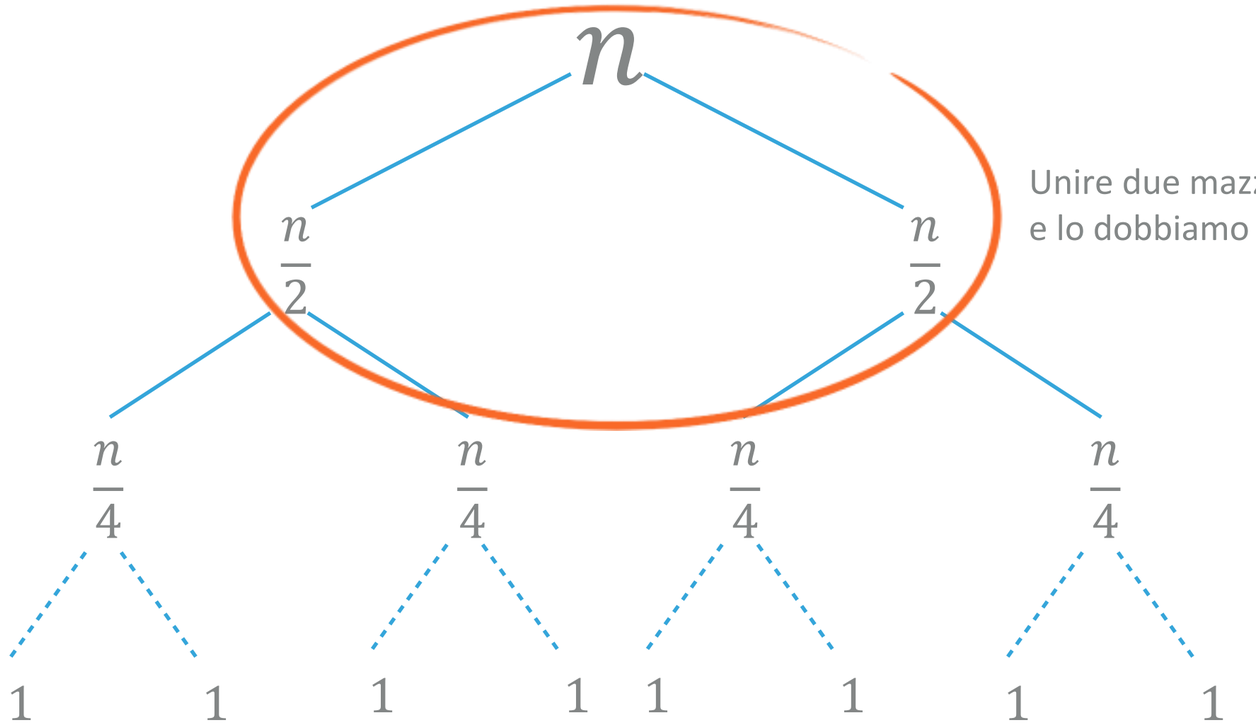
CALCOLO DEL COSTO



Possiamo dimezzare la dimensione del mazzo al più $\log_2(n)$ volte prima di avere un solo elemento

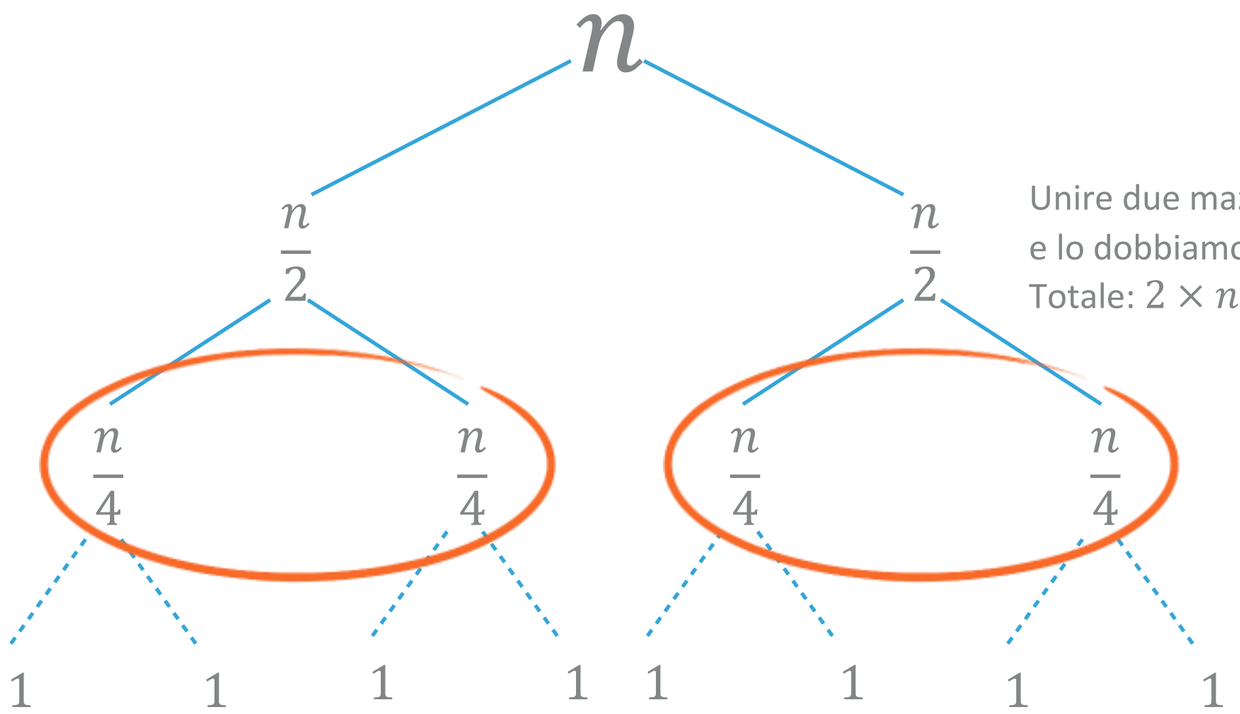
Nota. Questo particolare albero si chiama «albero di ricorsione».

CALCOLO DEL COSTO



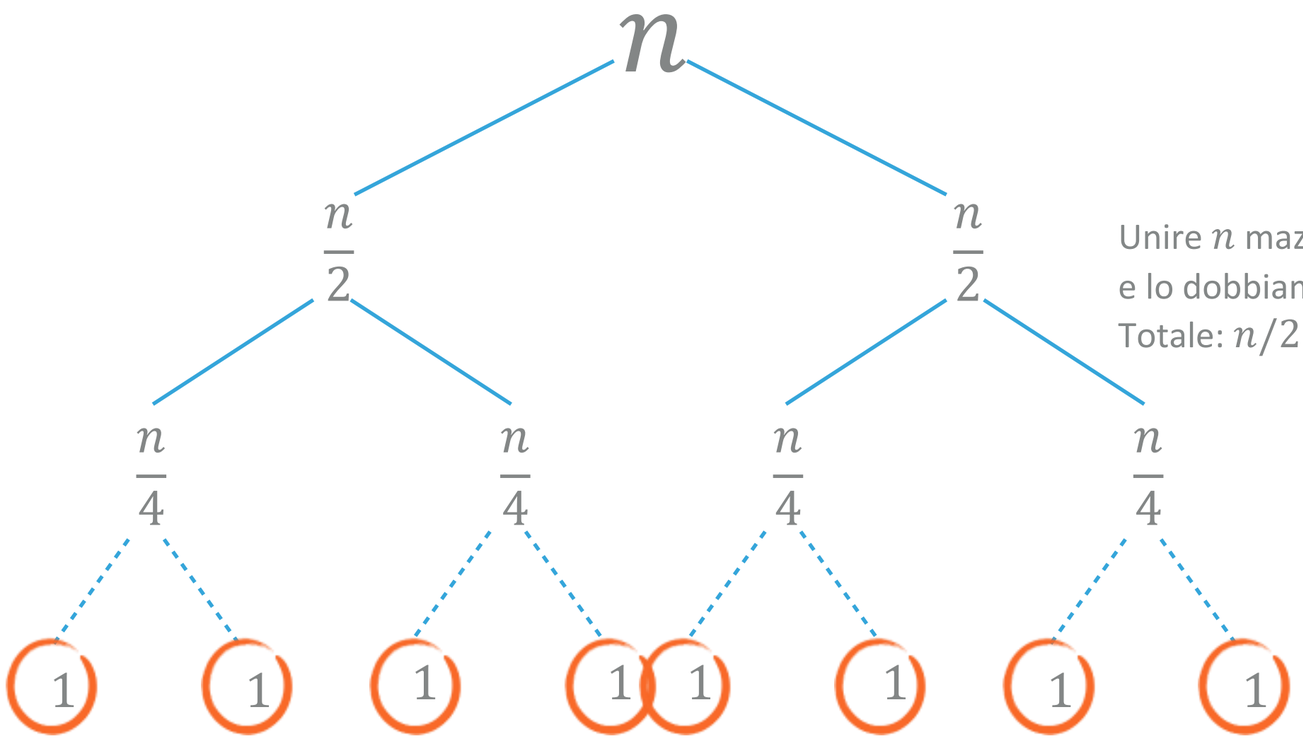
Unire due mazzi di lunghezza $n/2$ costa n
e lo dobbiamo fare una volta sola (PROCEDURA DI MERGE)

CALCOLO DEL COSTO



Unire due mazzi di lunghezza $n/4$ costa $n/2$
e lo dobbiamo fare 2 volte.
Totale: $2 \times n/2 = n$

CALCOLO DEL COSTO



Unire n mazzi di lunghezza 1 costa 2
e lo dobbiamo fare $n/2$ volte.
Totale: $n/2 \times 2 = n$

PSEUDOCODICE: PROCEDURA MERGE (1/2)

```
MERGE(A,p,q,r)
n_L = q-p-1     //lunghezza della sottosequenza L = A[p:q]
n_R = r-q       //lunghezza della sottosequenza R = A[q+1:r]

//si creano due nuovi array: L[0:n_L-1] ed R[0:n_R-1]
for i=0 to n_L -1
    L[i] = A[p+i] //copia A[p:q] in L[0:n_L-1]
for j=0 to n_R -1
    R[j] = A[q+1+j] //copia A[q+1:r] in R[0:n_R-1]
```

PSEUDOCODICE: PROCEDURA MERGE (2/2)

```
i=0     //indice che scorre su L
j=0     //indice che scorre su R
k=0     //indice che scorre su A
while i < n_L & j < n_R//finchè le due sottosequenze non sono vuote
      if L[i] ≤ R[j]
        A[k] = L[i]
        i = i+1
      else
        A[k] = R[j]
        j = j+1
      k = k+1
while i < n_L          //finisco di copiare il resto di L in A
      A[k] = L[i]
      i = i+1
      k = k+1
while j < n_R          //finisco di copiare il resto di R in A
      A[k] = R[j]
      j = j+1
      k = k+1
```

Nota. C'è un typo sul libro (j=1 invece che j=0).

PSEUDOCODICE: ALGORITMO MERGESORT COMPLETO

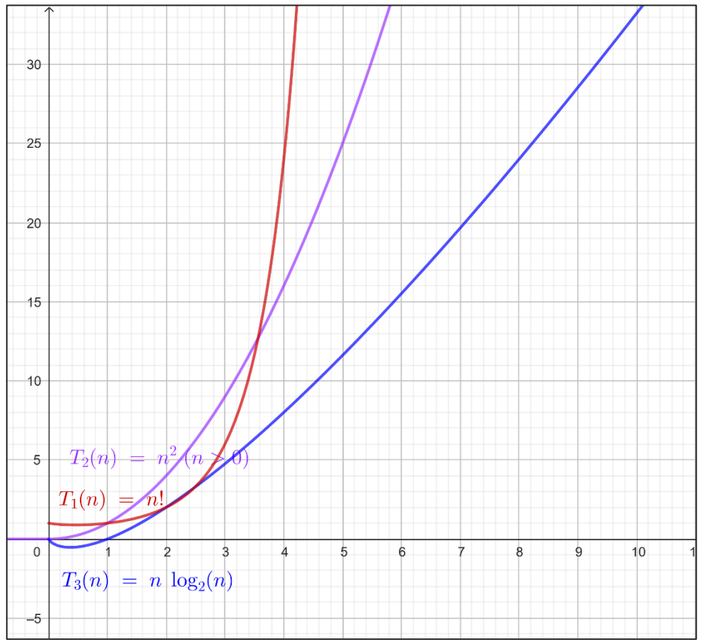
```
MERGE-SORT(A,p,r)
if p ≥ r                    //0 o 1 solo elemento (caso base)
   return
q = ⌊(p+r)/2⌋                //calcolo punto medio di A[p:r]
MERGE-SORT(A,p,q)            //ordina ricorsivamente A[p:q]
MERGE-SORT(A,q+1,r)          //ordina ricorsivamente A[q+1:r]
MERGE(A,p,q,r)               //fonde A[p:q] con A[q+1:r] per ottenere A[p:r] ordinato
```

CALCOLO DEL COSTO

- Ogni “livello” ci costa al più n comparazioni.
- Abbiamo $\log_2(n)$ livelli.
- Il costo totale in termini di comparazioni è $\boldsymbol{nlog_2n}$.
- Abbiamo migliorato ancora il tempo necessario ad ordinare un mazzo di carte:

Algoritmo	T(n)
Bogosort	n!
Insertion sort	$n*n = n^2$
Mergesort	$n*\log_2 n$

bogosort insertionsort mergesort



Annunci

Modifica alle date di alcuni appelli: *stay tuned on Moodle!*

Annuncio seminari

14/05 h. 14-16 Prof. A. Raganato (Univ. Milano-Bicocca)

Elaborazione del Linguaggio Naturale e Large Language Models

Nota. Ospite online, ma noi in presenza.

19/05 h. 14-16 Prof. S. Ghidoni (Univ. Padova)

Intelligenza artificiale: dall'emulazione del ragionamento umano all'apprendimento dei dati

I seminari valgono fino a 1 punto per il voto finale. Verrà registrata la presenza

UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Prossima lezione: 6 ***maggio, h.16:00, aula 3B***