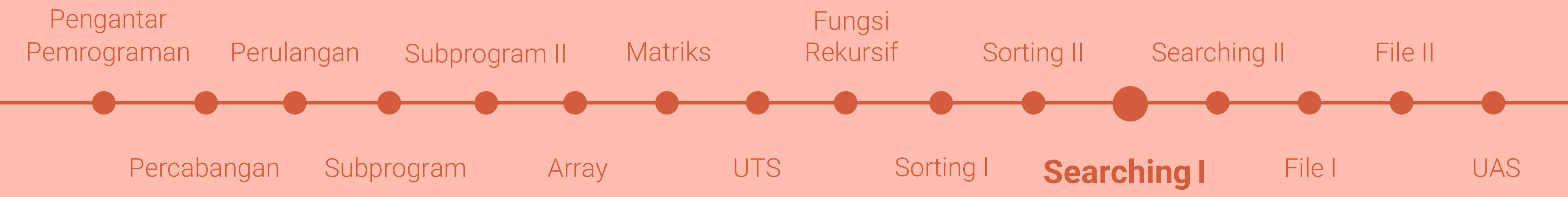
DASAR PEMROGRAMAN

Pertemuan XII



Pengantar Pemrograman

Percabangan

Perulangan

Subprogram

Subprogram II

Array

Matriks

UTS

Fungsi Rekursif

Sorting I

Sorting II

Searching I

Searching II

File I

File II

UAS

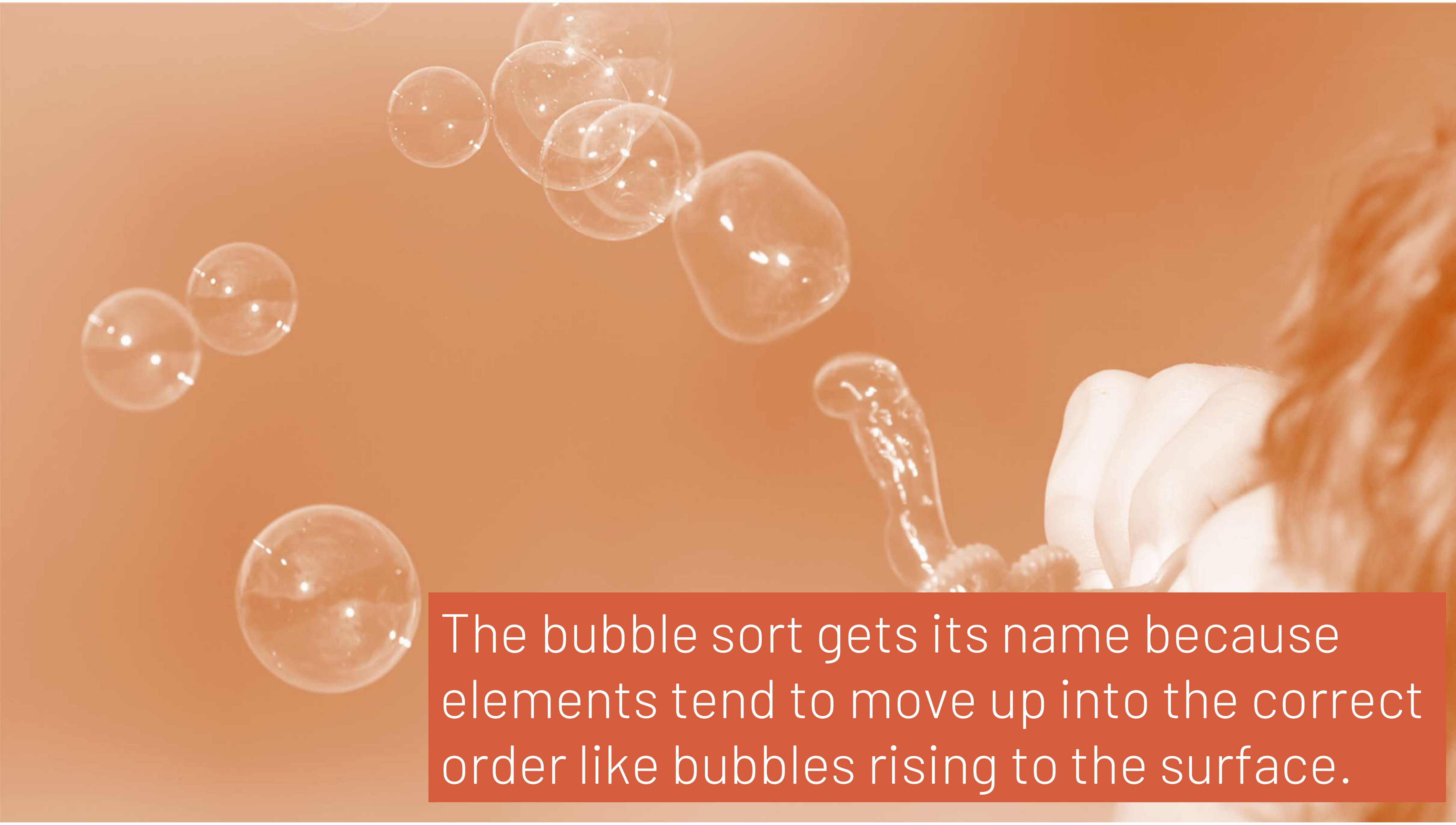
Tujuan

- Mahasiswa mampu melakukan pengurutan elemen di dalam array dengan algoritma pengurutan tertentu.

Materi

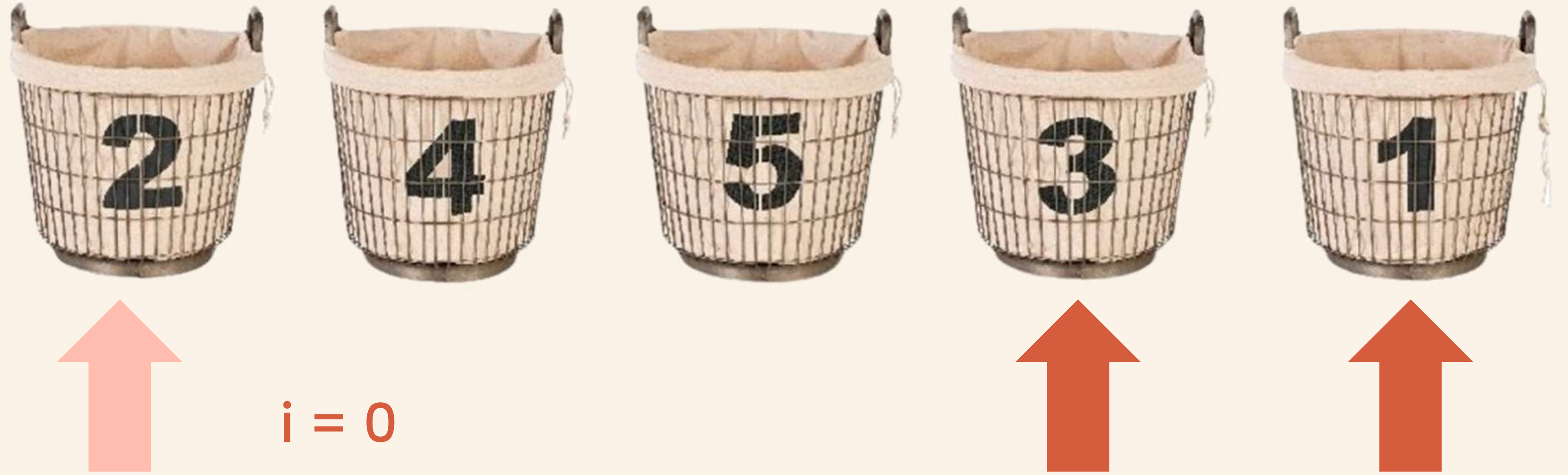
- Bubble Sort
- Selection Sort
- Insertion Sort

BUBBLE SORT



The bubble sort gets its name because elements tend to move up into the correct order like bubbles rising to the surface.

```
Bubble Sort ()

for $i$=0 to $n$ - 2
    for $j$=$n$ - 1 downto $i$ + 1
        if $L[j] < L[j-1]$
            swap($L[j]$, $L[j-1]$)
```

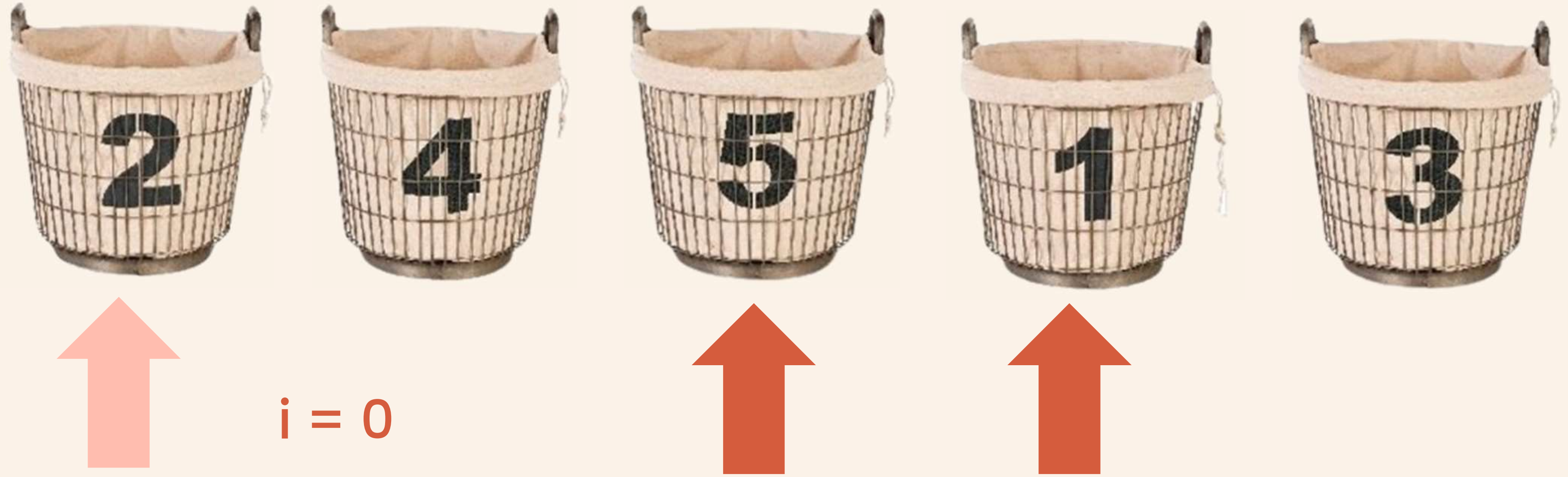


i = 0

j = 4

Tukar? YES

```
BUBBLE SORT ()
    for $i$=0 to $n$ - 2
        for $j$=$n$ - 1 downto $i$ + 1
            if $L[j] < L[j-1]$
                swap($L[j]$, $L[j-1]$)
```



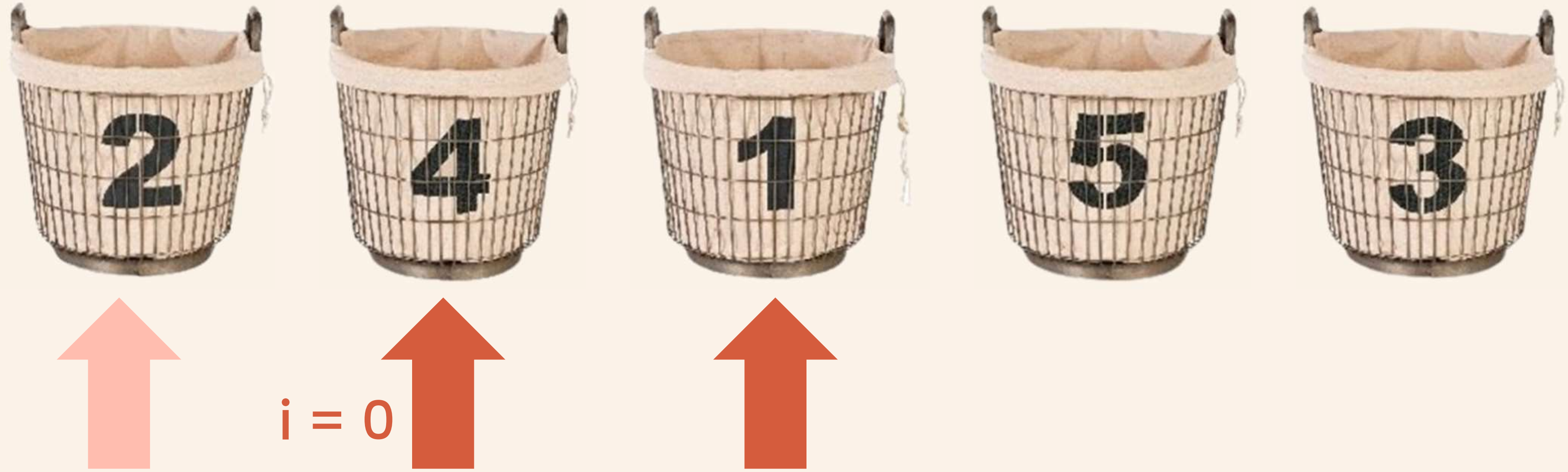
i = 0

j = 3

Tukar? YES

BUBBLE SORT ()

```
for $i$=0 to $n$ - 2
    for $j$=$n$ - 1 downto $i$ + 1
        if $L[j] < L[j-1]$
            swap($L[j]$, $L[j-1]$)
```

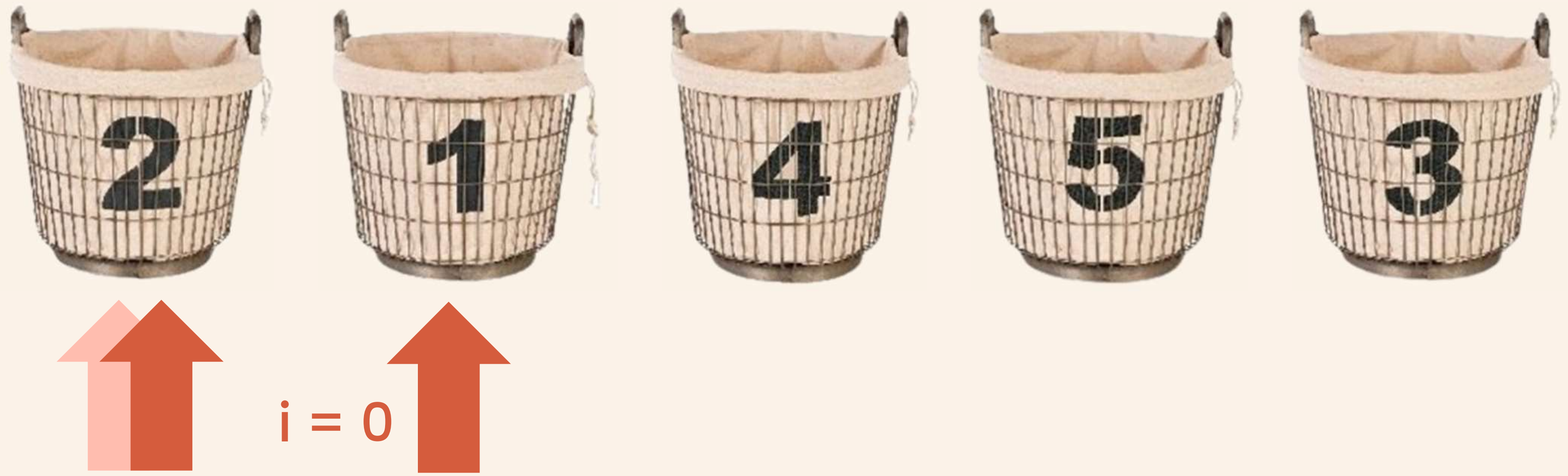


i = 0

j = 2

Tukar? YES

```
BUBBLE SORT ()

for $i$=0 to $n$ - 2
    for $j$=$n$ - 1 downto $i$ + 1
        if $L[j] < L[j-1]$
            swap($L[j]$, $L[j-1]$)
```



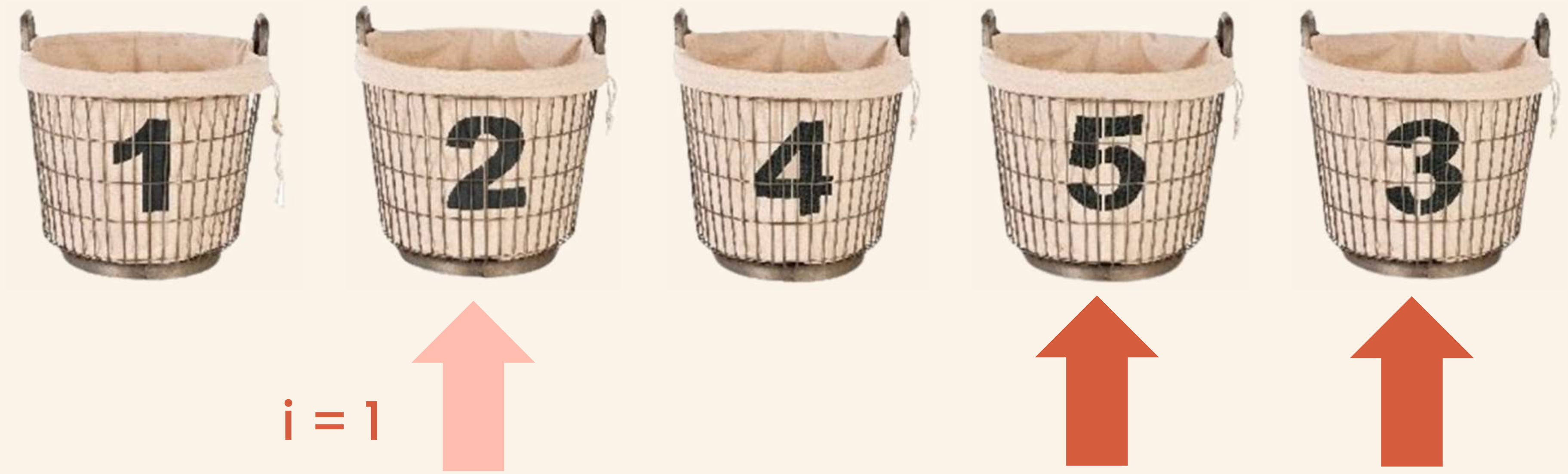
i = 0

j = 1

Tukar? YES

Bubble Sort ()

```
for $i$=0 to $n$ - 2
    for $j$=$n$ - 1 downto $i$ + 1
        if $L[j] < L[j-1]$
            swap($L[j]$, $L[j-1]$)
```

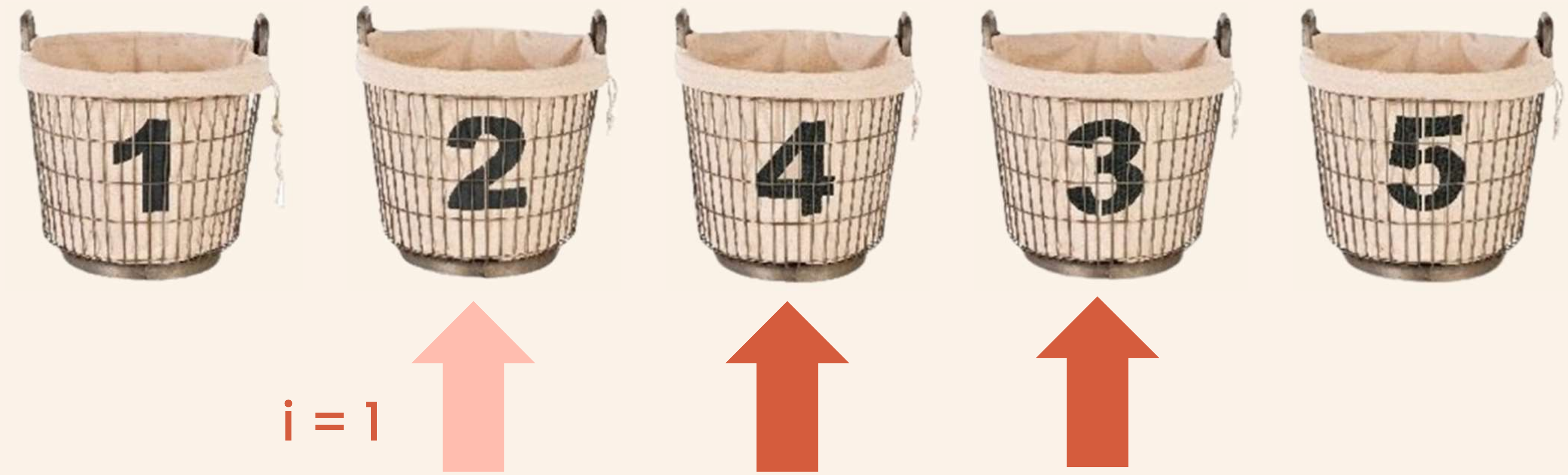


i = 1

j = 4

Tukar? YES

```
BUBBLE SORT ()
for $i$=0 to $n$ - 2
    for $j$=$n$ - 1 downto $i$ + 1
        if $L[j] < L[j-1]$
            swap($L[j]$, $L[j-1]$)
```

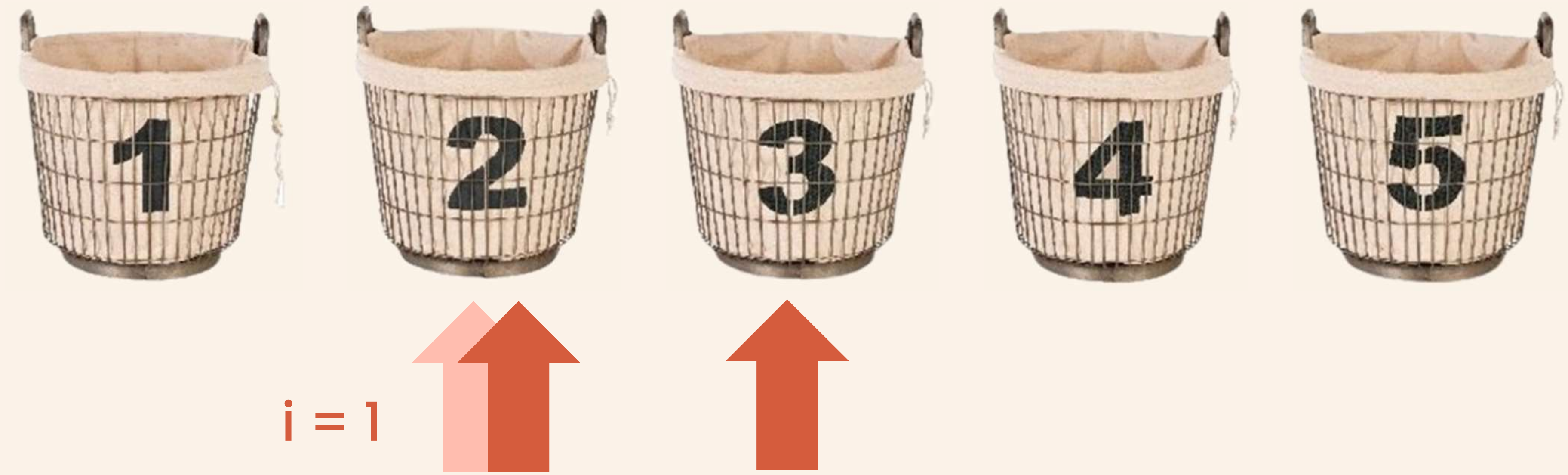


i = 1

j = 3

Tukar? YES

```
BUBBLE SORT ()
for $i=0$ to $n - 2$
    for $j=n - 1$ downto $i + 1$
        if $L[j] < L[j-1]$
            swap($L[j]$, $L[j-1]$)
```

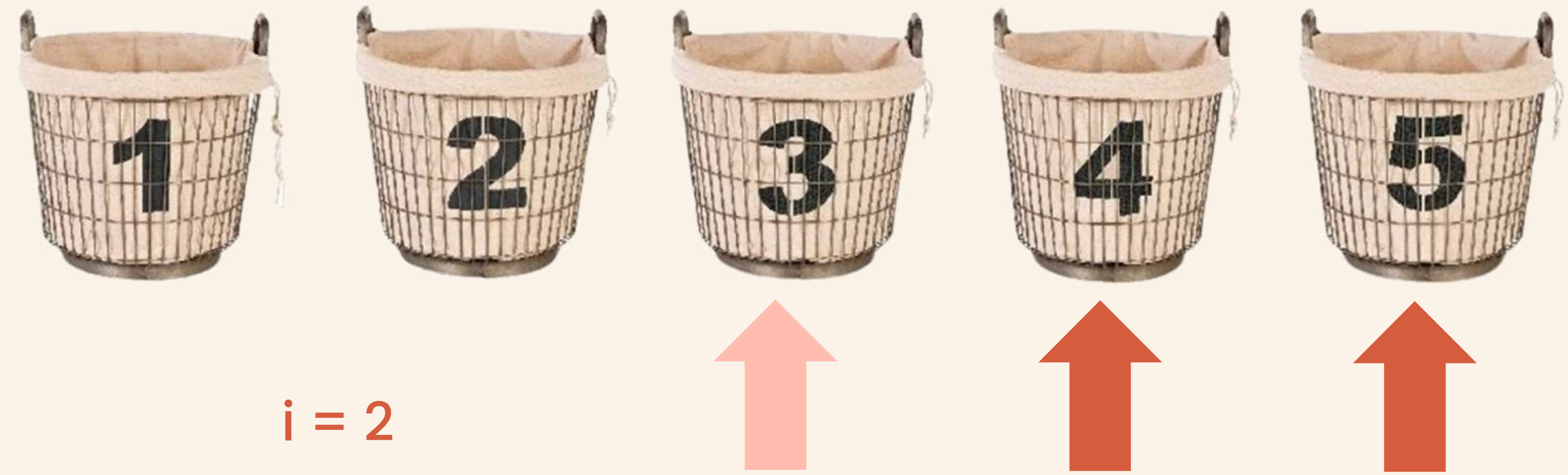


i = 1

j = 2

Tukar? NO

```
BUBBLE SORT ()
for i=0 to n - 2
    for j=n - 1 downto i + 1
        if L[j] < L[j-1]
            swap(L[j], L[j-1])
```

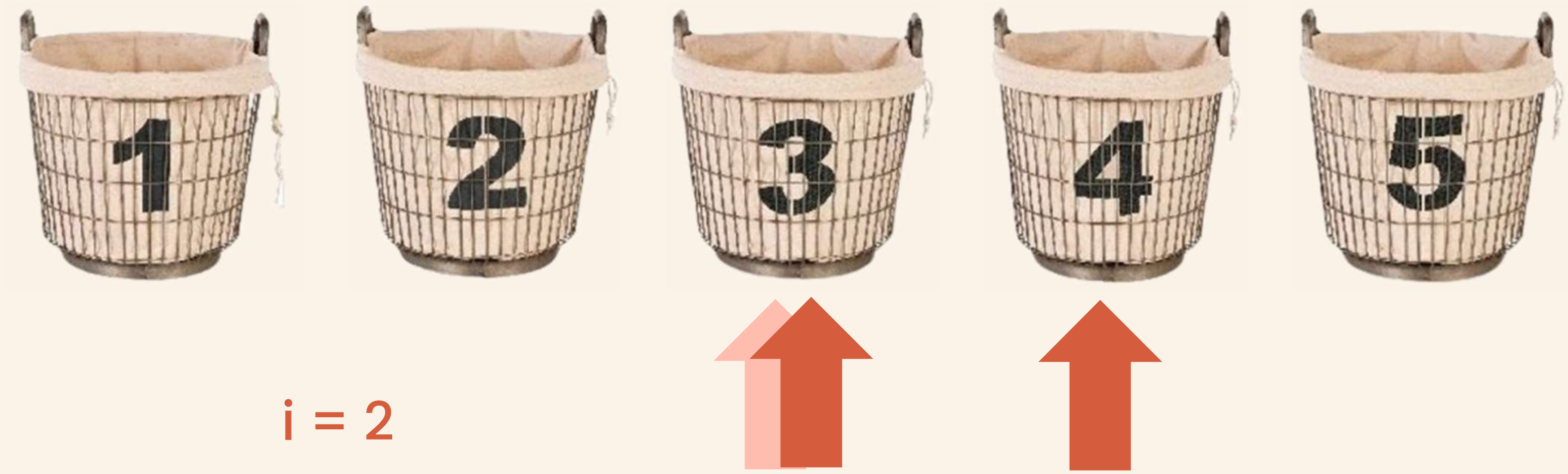


i = 2

j = 4

Tukar? NO

```
BUBBLE SORT ()
for $i$=0 to $n$ - 2
    for $j$=$n$ - 1 downto $i$ + 1
        if $L[j] < L[j-1]$
            swap($L[j]$, $L[j-1]$)
```



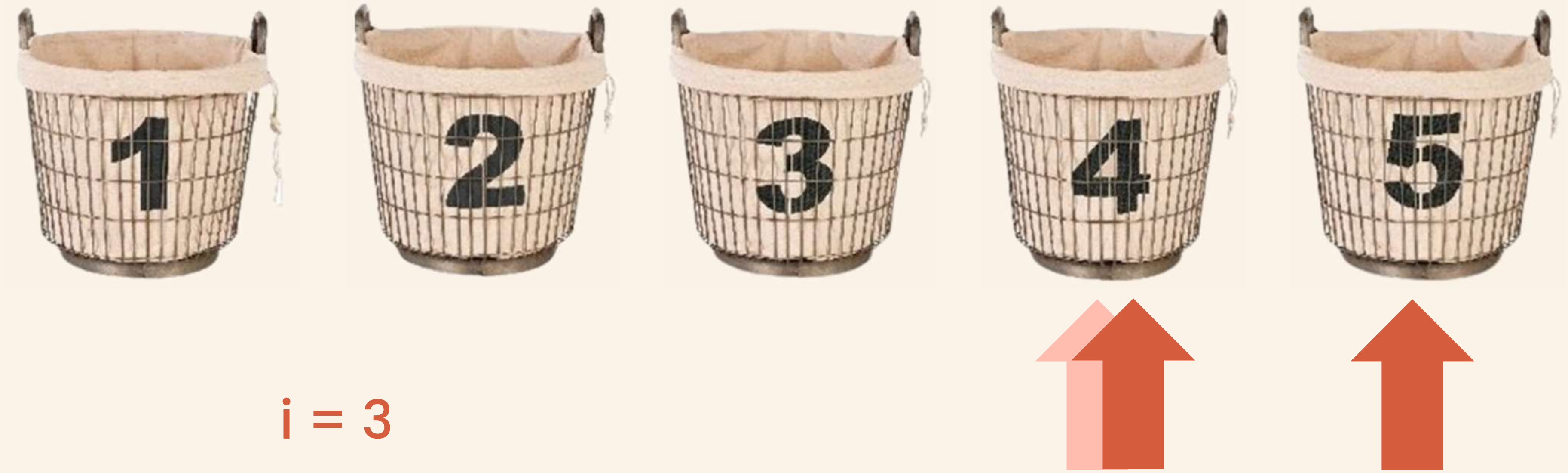
i = 2

j = 3

Tukar? NO

BUBBLE SORT ()

```
for $i$=0 to $n$ - 2
    for $j$=$n$ - 1 downto $i$ + 1
        if $L[j] < L[j-1]$
            swap($L[j]$, $L[j-1]$)
```



i = 3

j = 4

Tukar? NO

Bubble Sort ()

```
for i=0 to n - 2
    for j=n - 1 downto i + 1
        if L[j] < L[j-1]
            swap(L[j], L[j-1])
```

BUBBLE SORT ()

```
for $i$=0 to $n$ - 2
    for $j$=$n$ - 1 downto $i$ + 1
        if $L[j] < L[j-1]$
            swap($L[j]$, $L[j-1]$)
```

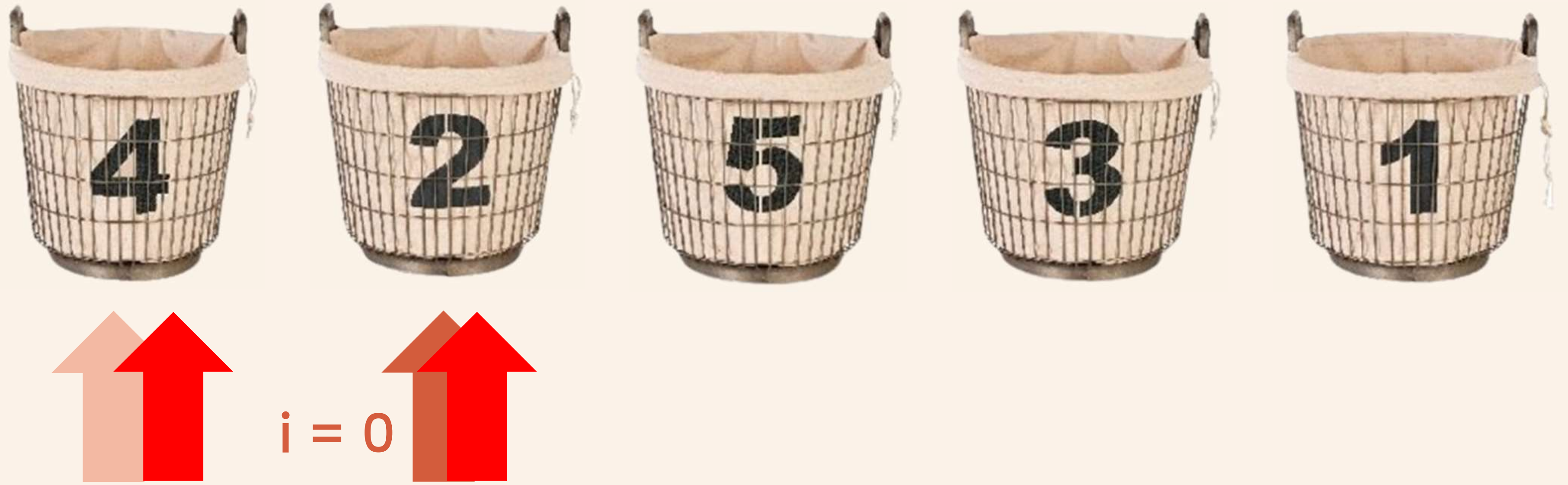
BUBBLE SORT ()

```
for $i$=0 to $n$ - 1
    for $j$=$0$ downto $n$- $i$ -$1$
        if $L[j] > L[j+1]$
            swap($L[j]$, $L[j+1]$)
```

SELECTION SORT

Selection sort works by repeatedly element. First find the smallest in the array and exchange it with the element in the first position, then find the second smallest element and exchange it with the element in the second position and continue in this way until the entire array is sorted.

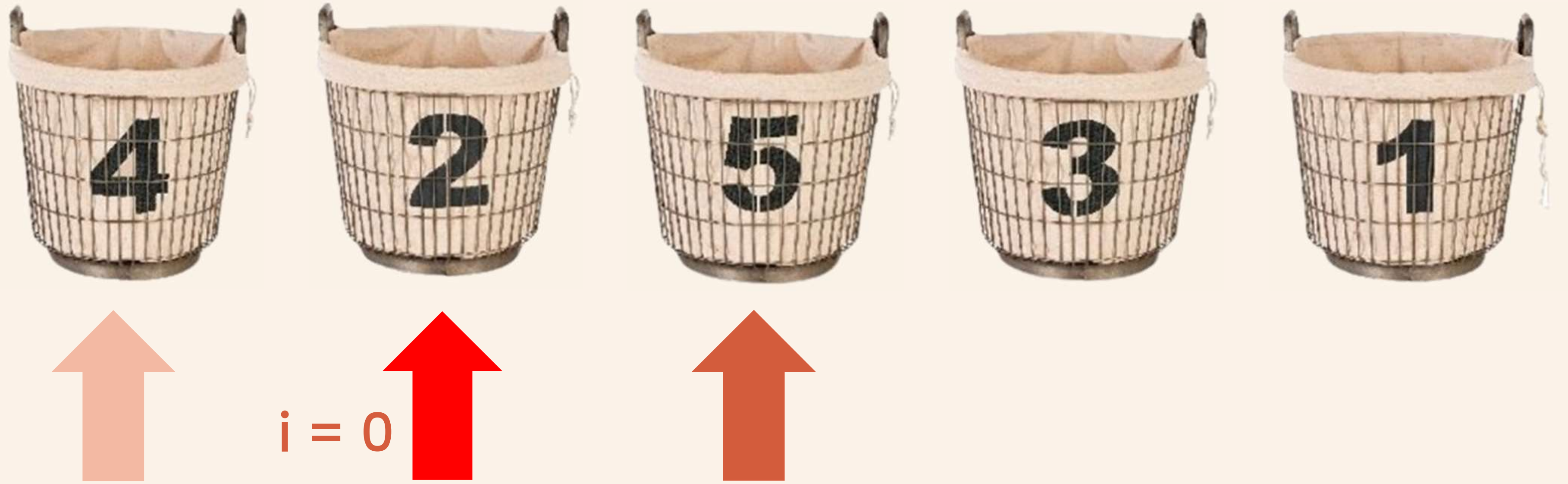
```
SELECTION SORT ()

for i=0 to n - 2
    indeks_min = i
    for j=i + 1 to n - 1
        if L[j] < L[indeks_min]
            indeks_min = j
    swap(L[i], L[indeks_min])
```



i = 0

j = 1

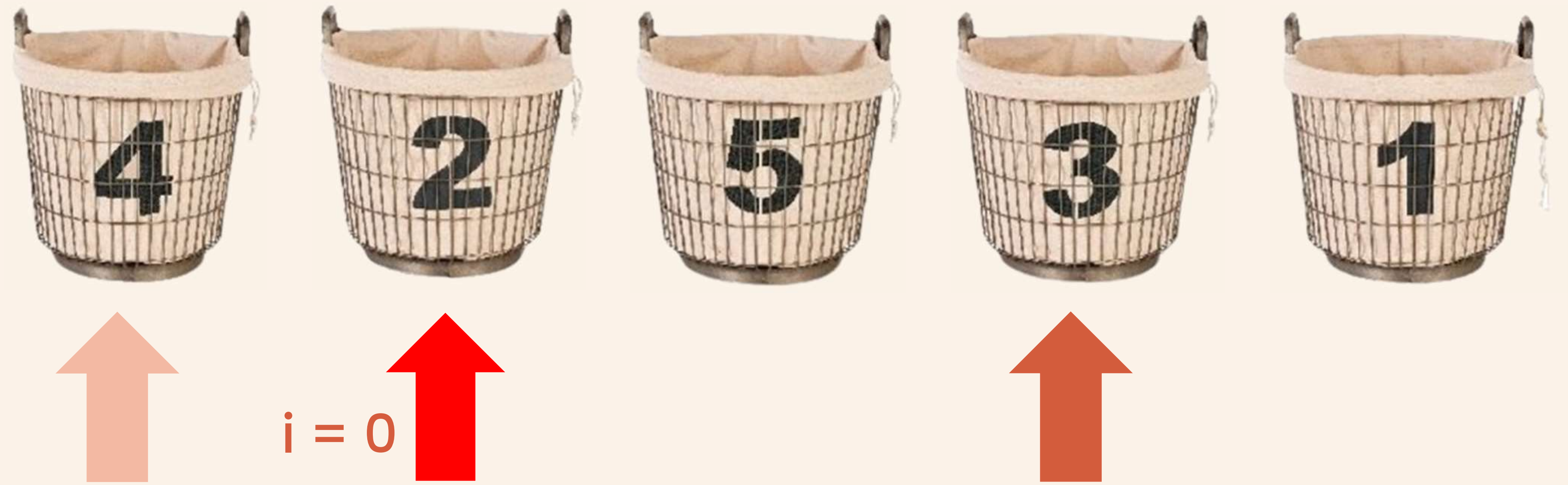
```
SELECTION SORT ()

for i=0 to n - 2
    indeks_min = i
    for j=i + 1 to n - 1
        if L[j] < L[indeks_min]
            indeks_min = j
    swap(L[i], L[indeks_min])
```



i = 0

j = 2

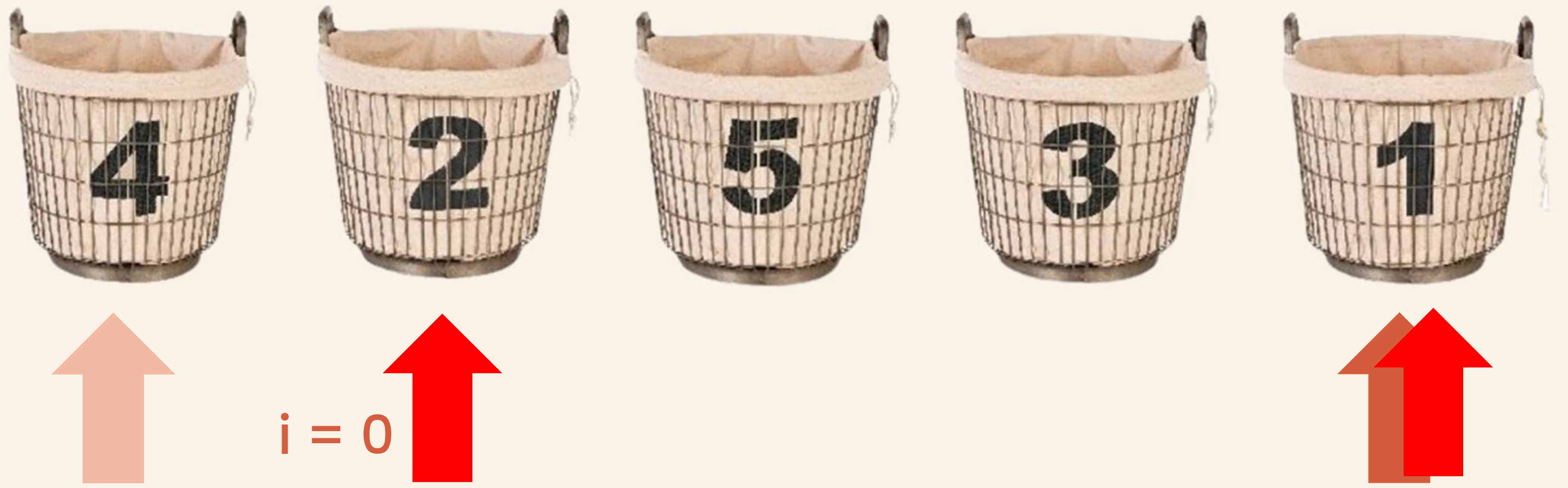
```
SELECTION SORT ()
    for i=0 to n - 2
        indeks_min = i
        for j=i + 1 to n - 1
            if L[j] < L[indeks_min]
                indeks_min = j
        swap(L[i], L[indeks_min])
```



i = 0

j = 3

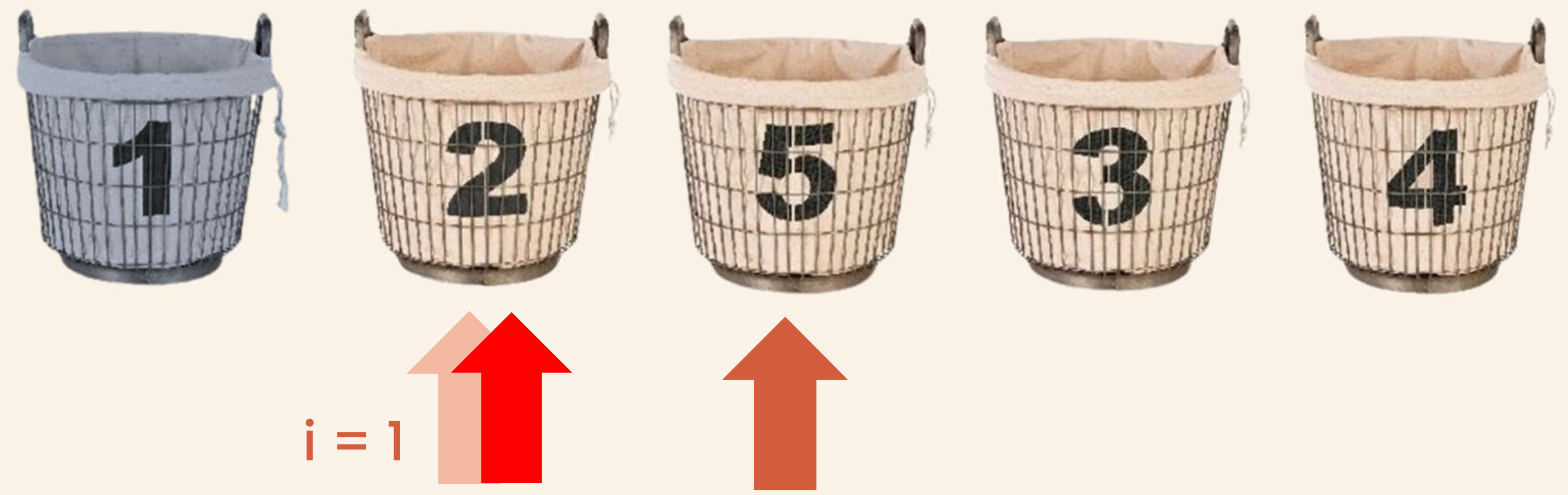
```
SELECTION SORT ()

for i=0 to n - 2
    indeks_min = i
    for j=i + 1 to n - 1
        if L[j] < L[indeks_min]
            indeks_min = j
    swap(L[i], L[indeks_min])
```



i = 0

j = 4

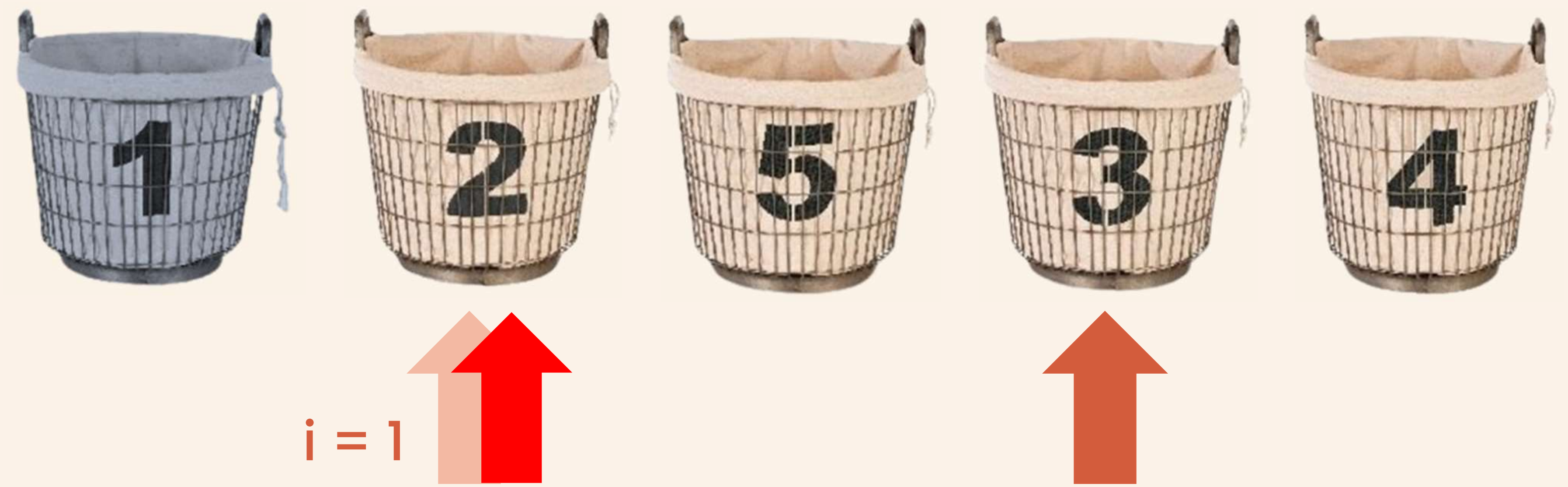
```
SELECTION SORT ()

for i=0 to n - 2
    indeks_min = i
    for j=i + 1 to n - 1
        if L[j] < L[indeks_min]
            indeks_min = j
    swap(L[i], L[indeks_min])
```



i = 1

j = 2

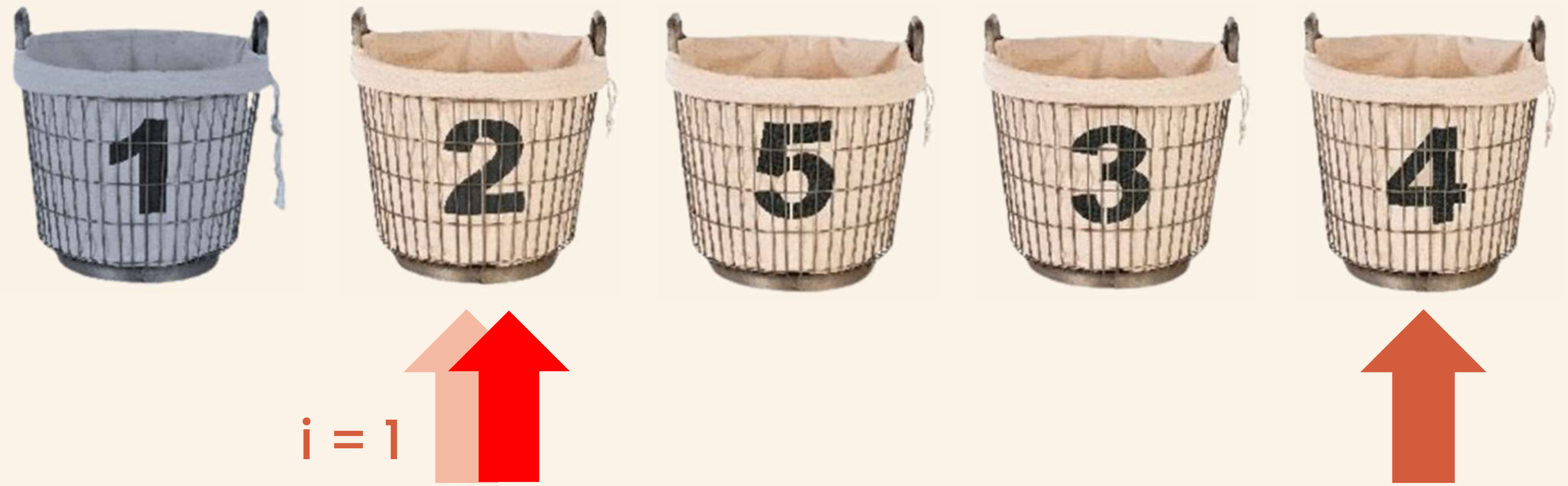
```
SELECTION SORT ()

for i=0 to n - 2
    indeks_min = i
    for j=i + 1 to n - 1
        if L[j] < L[indeks_min]
            indeks_min = j
    swap(L[i], L[indeks_min])
```



i = 1

j = 3

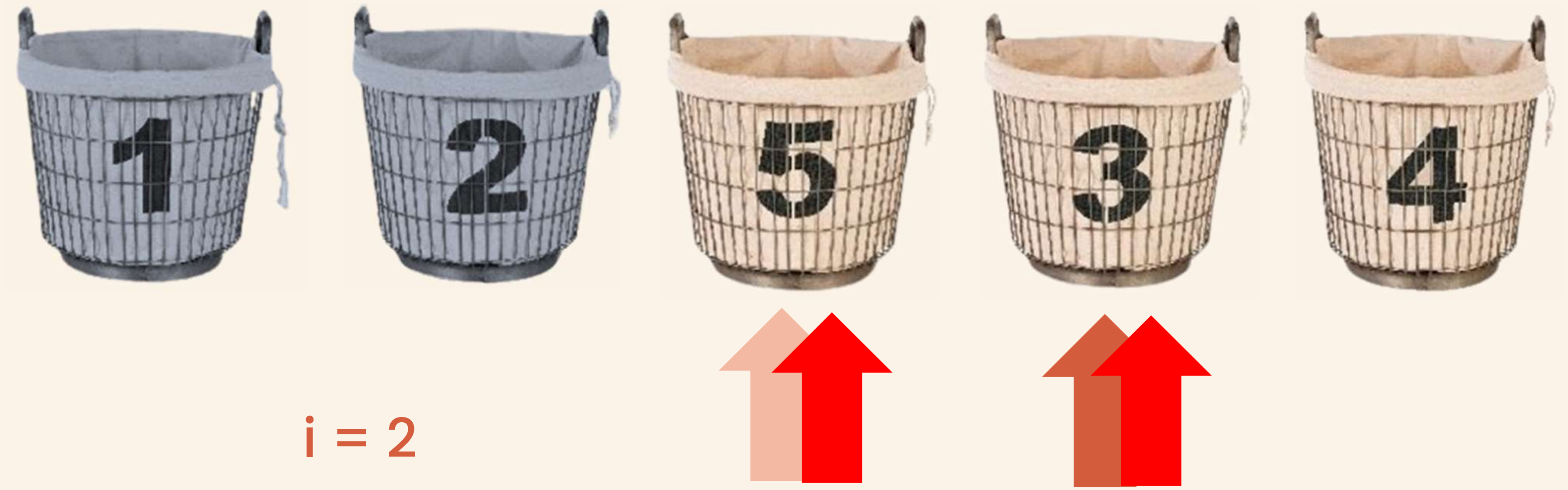
```
SELECTION SORT ()

    for i=0 to n - 2
        indeks_min = i
        for j=i + 1 to n - 1
            if L[j] < L[indeks_min]
                indeks_min = j
        swap(L[i], L[indeks_min])
```



i = 1

j = 4

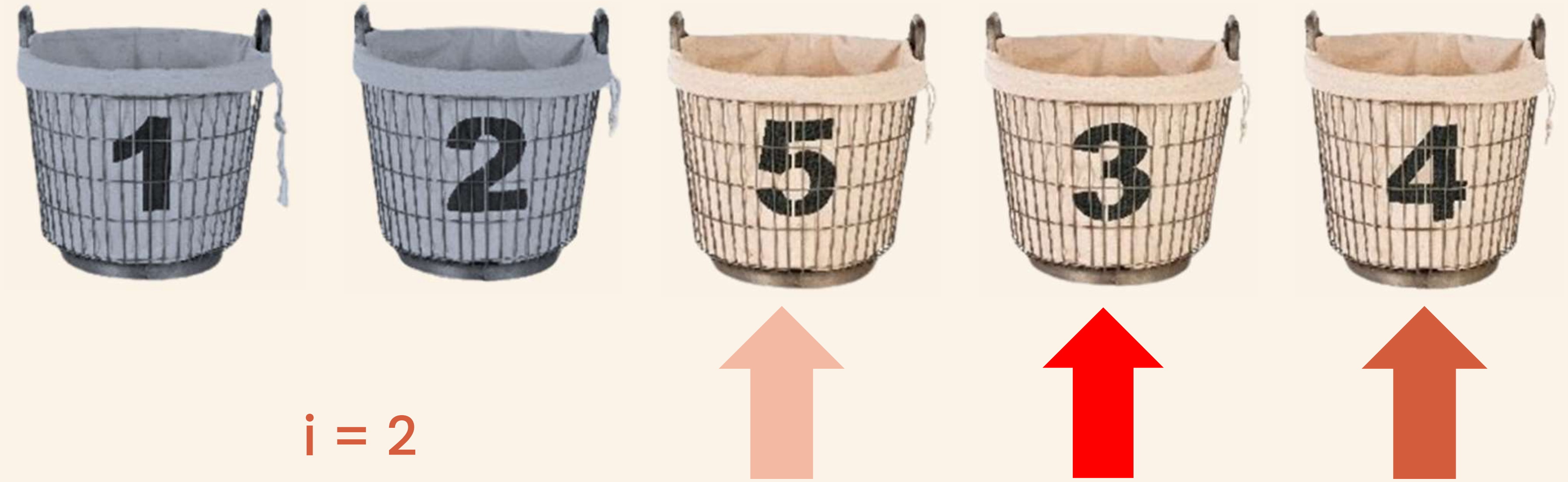
```
SELECTION SORT ()
for i=0 to n - 2
    indeks_min = i
    for j=i + 1 to n - 1
        if L[j] < L[indeks_min]
            indeks_min = j
    swap(L[i], L[indeks_min])
```



i = 2

j = 3

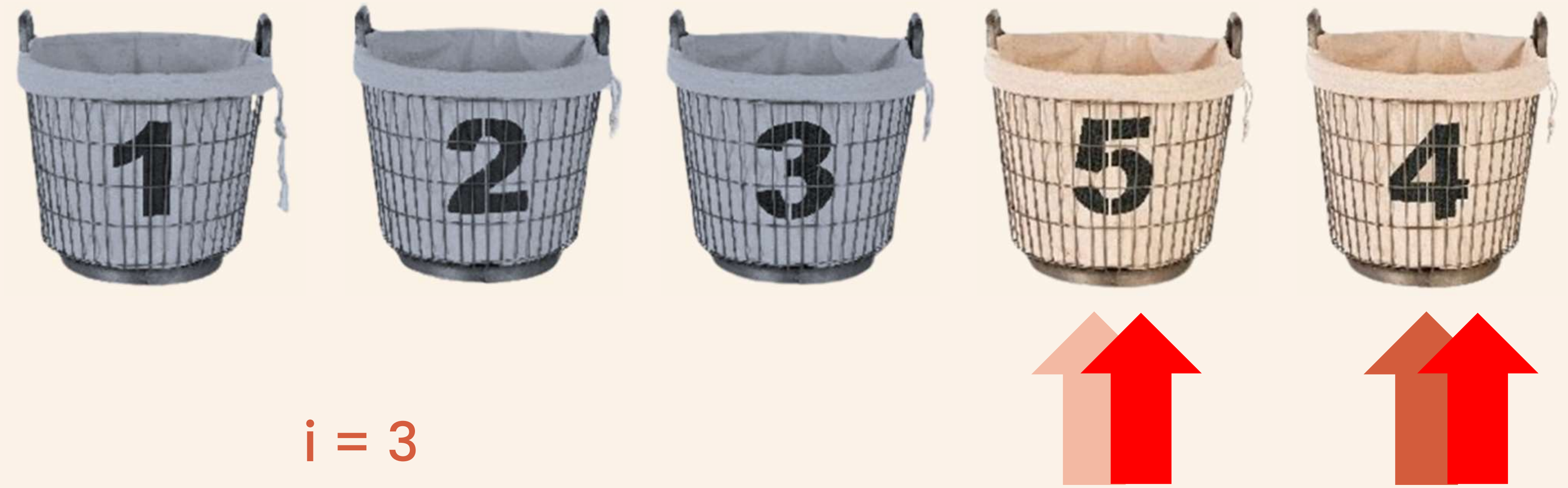
```
SELECTION SORT ()

for i=0 to n - 2
    indeks_min = i
    for j=i + 1 to n - 1
        if L[j] < L[indeks_min]
            indeks_min = j
    swap(L[i], L[indeks_min])
```



i = 2

j = 4

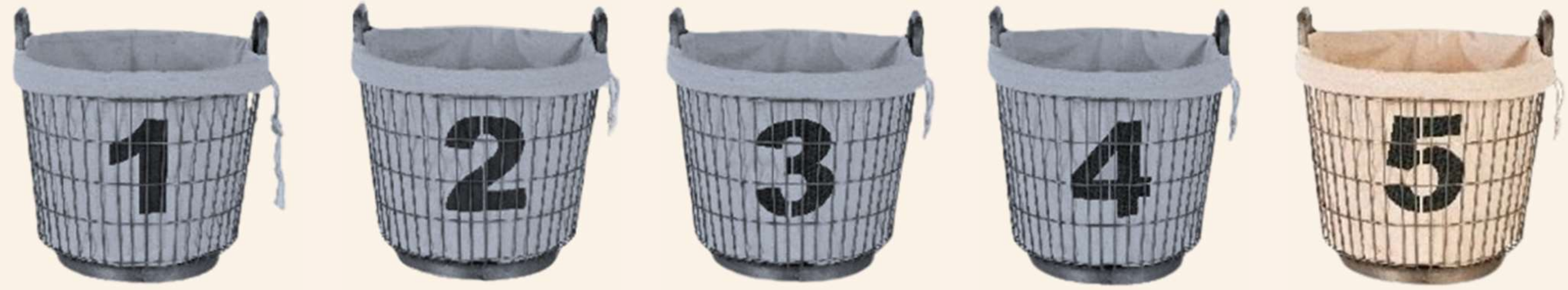
```
SELECTION SORT ()
    for i=0 to n - 2
        indeks_min = i
        for j=i + 1 to n - 1
            if L[j] < L[indeks_min]
                indeks_min = j
        swap(L[i], L[indeks_min])
```



i = 3

j = 4

```
SELECTION SORT ()
for i=0 to n - 2
    indeks_min = i
    for j=i + 1 to n - 1
        if L[j] < L[indeks_min]
            indeks_min = j
    swap(L[i], L[indeks_min])
```

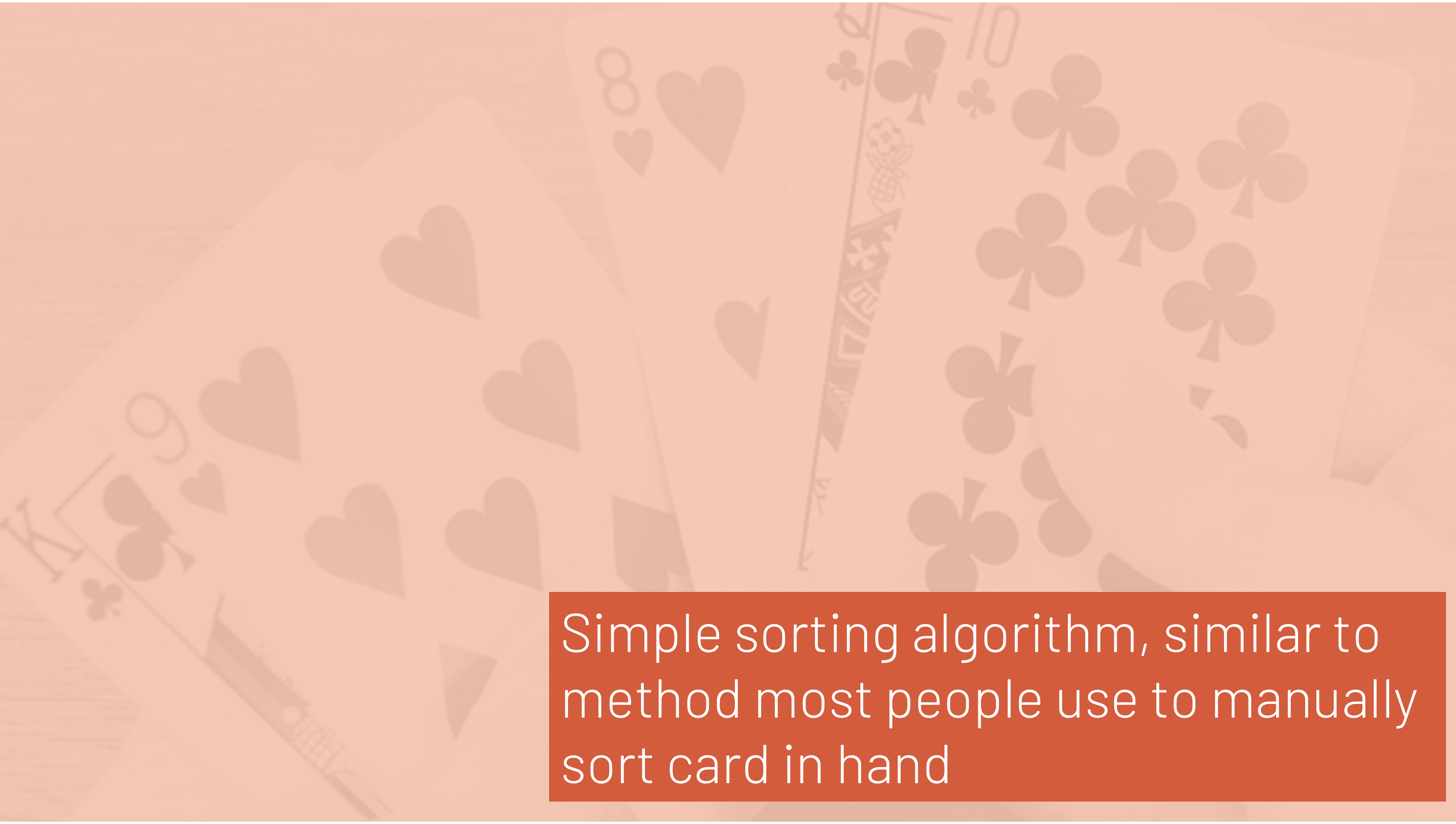


i = 3

j = 4

```
SELECTION SORT ()

for i=0 to n - 2
    indeks_min = i
    for j=i + 1 to n - 1
        if L[j] < L[indeks_min]
            indeks_min = j
    swap(L[i], L[indeks_min])
```

INSERTION SORT



Simple sorting algorithm, similar to method most people use to manually sort card in hand

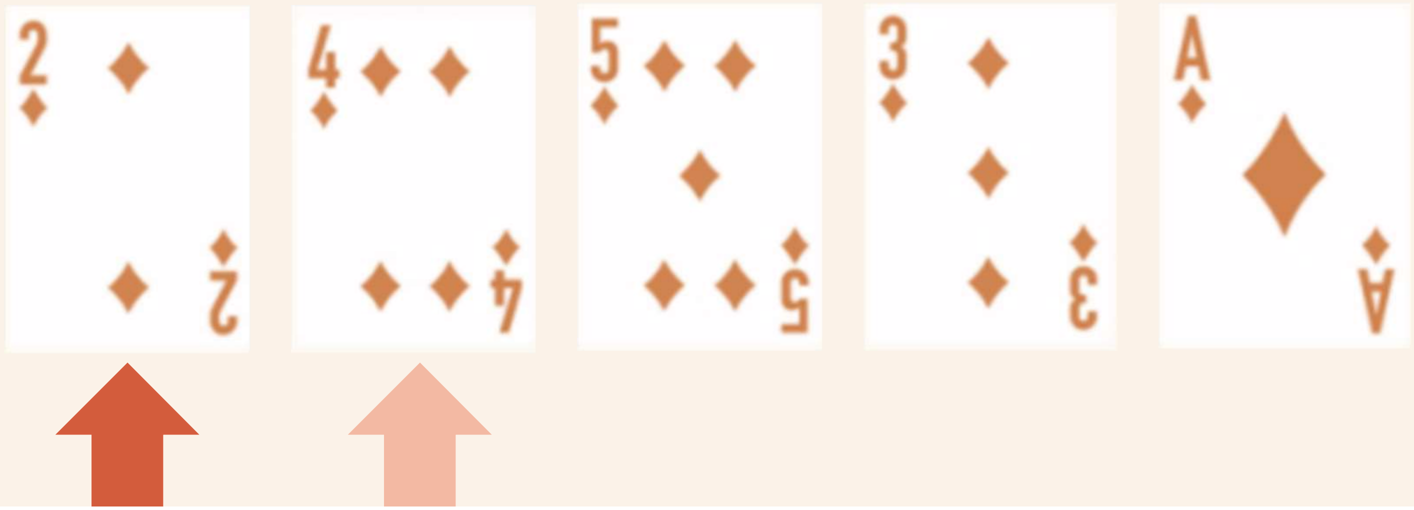
```
INSERTION SORT (L)

for i=1 to n - 1
      y = L[i]
      j = i - 1
      found = false
      while j >= 0 and found = false
            if y < L[j]
                  L[j+1] = L[j]
                  j = j - 1
            else
                  found = true
      L[j+1] = y
```

i = 1

y = 2

j =0−1

found = false



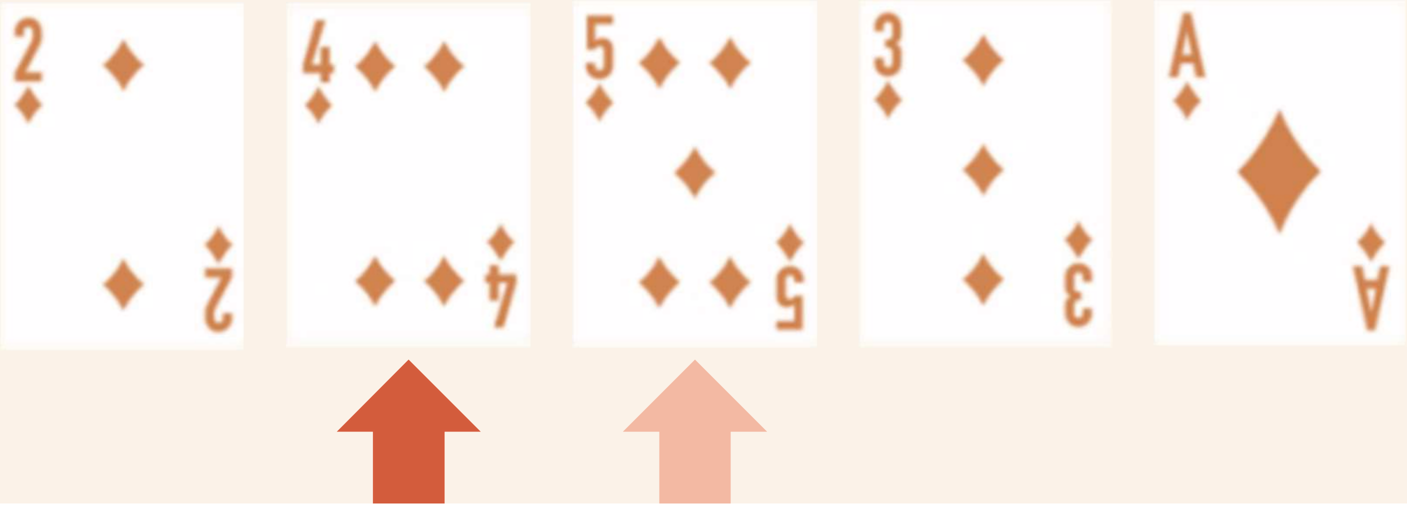
```
INSERTION SORT (L)

for i=1 to n - 1
      y = L[i]
      j = i - 1
      found = false
      while j >= 0 and found = false
            if y < L[j]
                  L[j+1] = L[j]
                  j = j - 1
            else
                  found = true
      L[j+1] = y
```

i = 2

y = 5

j = 1

found = false true



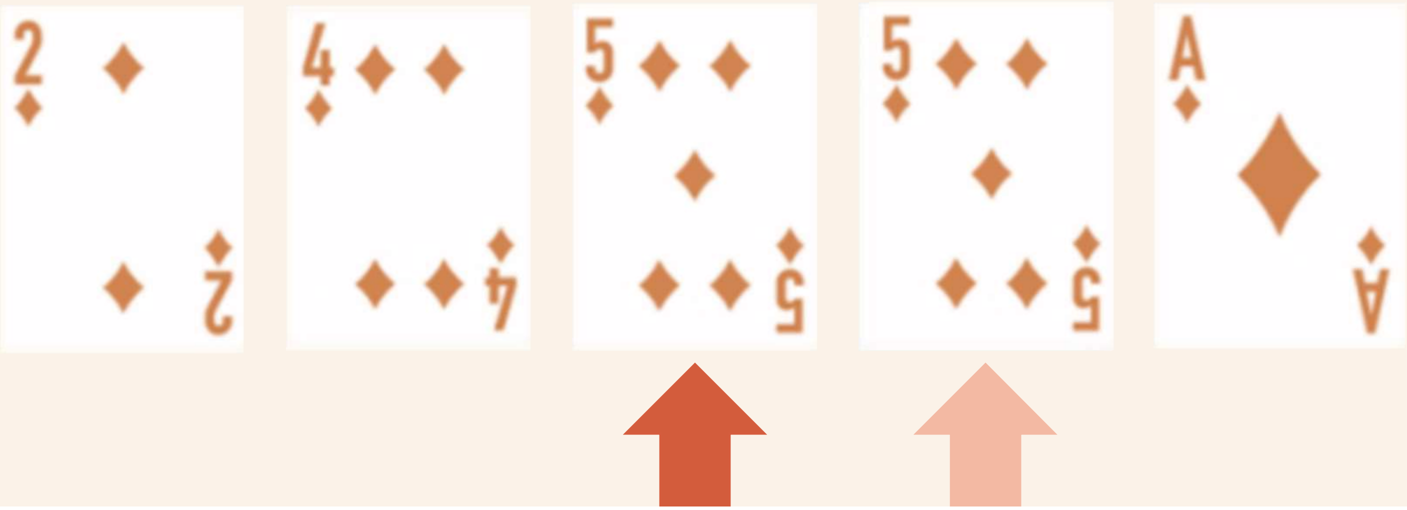
```
INSERTION SORT (L)

for i=1 to n - 1
    y = L[i]
    j = i - 1
    found = false
    while j >= 0 and found = false
        if y < L[j]
            L[j+1] = L[j]
            j = j - 1
        else
            found = true
    L[j+1] = y
```

i = 3

y = 3

j = 12

found = false



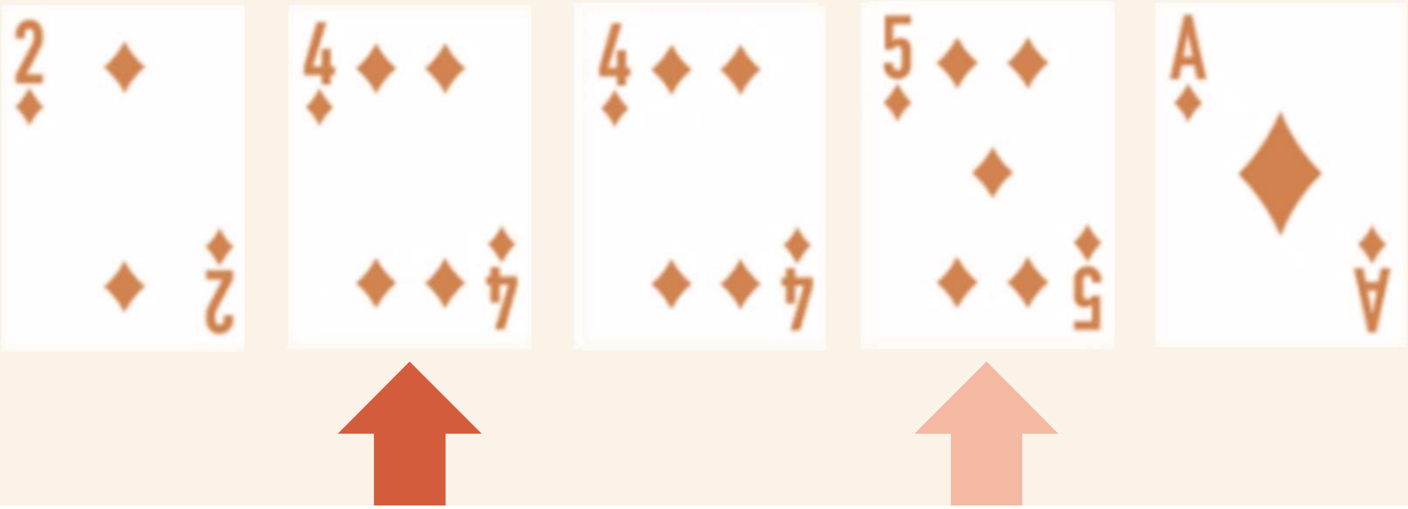
```
INSERTION SORT (L)

for i=1 to n - 1
    y = L[i]
    j = i - 1
    found = false
    while j >= 0 and found = false
        if y < L[j]
            L[j+1] = L[j]
            j = j - 1
        else
            found = true
    L[j+1] = y
```

i = 3

y = 3

j = 0

found = false



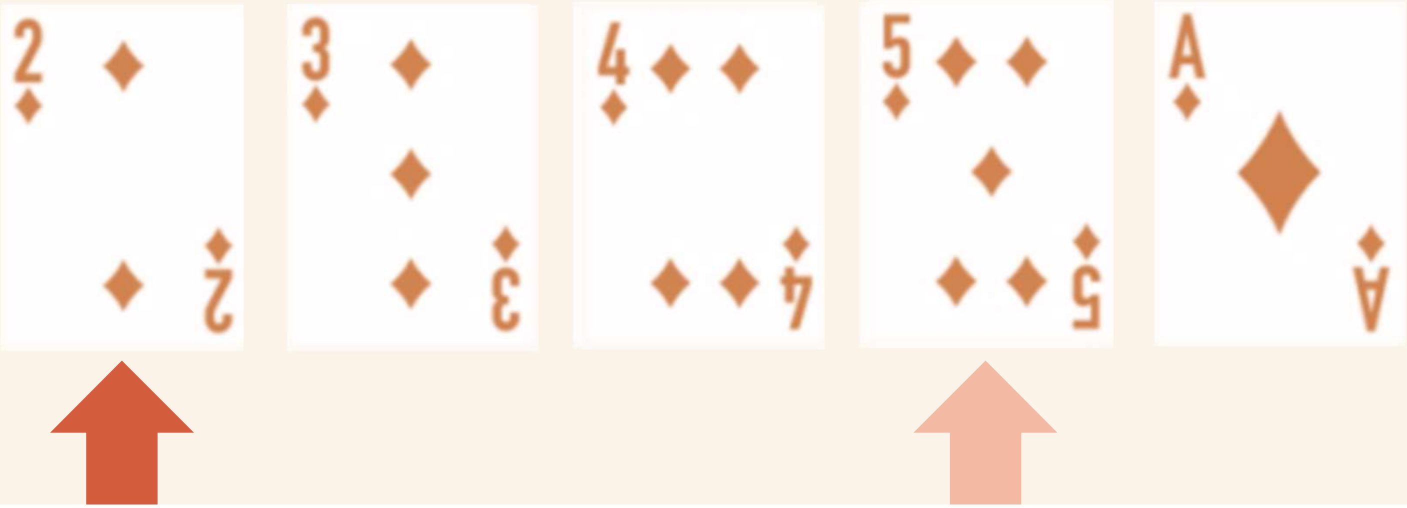
```
INSERTION SORT (L)

for i=1 to n - 1
    y = L[i]
    j = i - 1
    found = false
    while j >= 0 and found = false
        if y < L[j]
            L[j+1] = L[j]
            j = j - 1
        else
            found = true
    L[j+1] = y
```

i = 3

y = 3

j = 0

found = false true



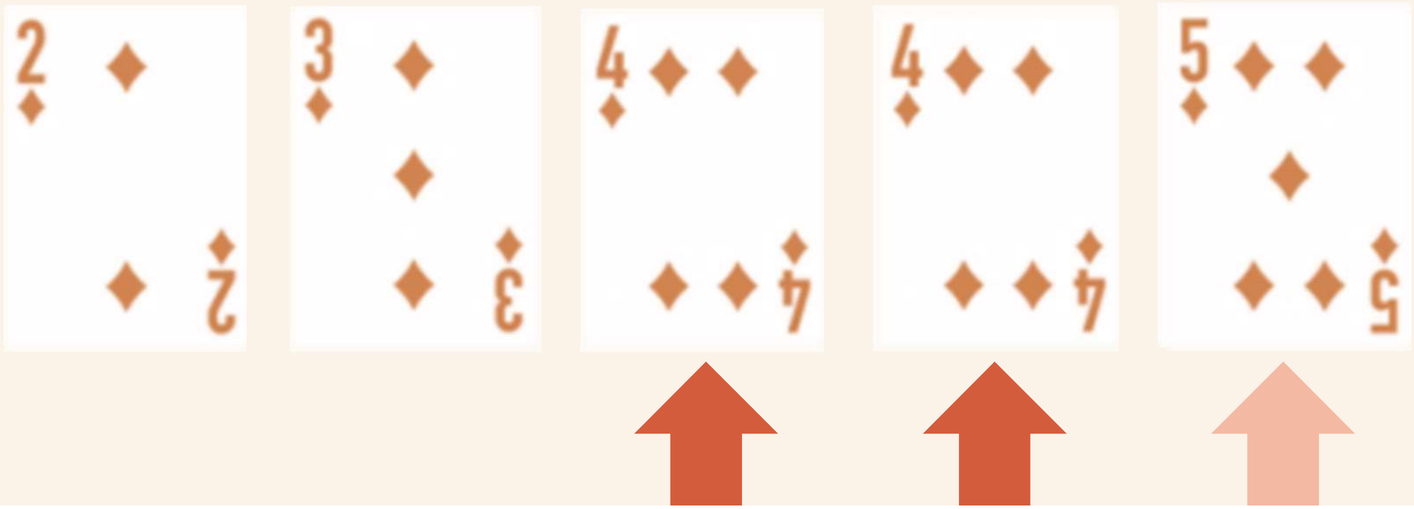
```
INSERTION SORT (L)

for i=1 to n - 1
    y = L[i]
    j = i - 1
    found = false
    while j >= 0 and found = false
        if y < L[j]
            L[j+1] = L[j]
            j = j - 1
        else
            found = true
    L[j+1] = y
```

i = 4

y = 1

j = 3

found = false



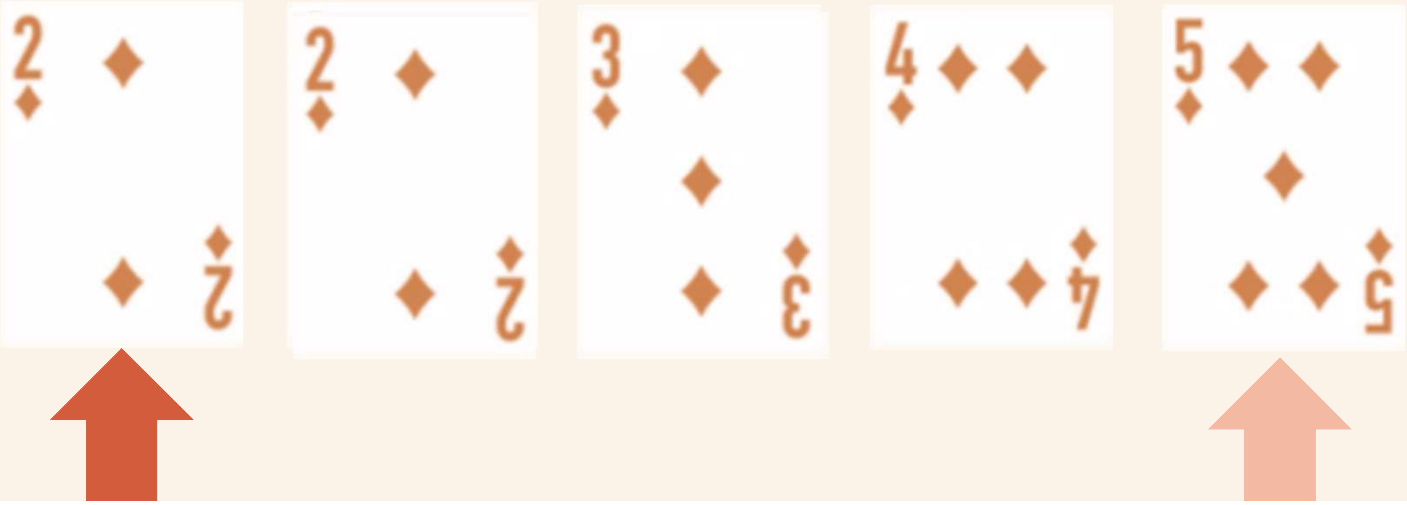
```
INSERTION SORT (L)

for i=1 to n - 1
    y = L[i]
    j = i - 1
    found = false
    while j >= 0 and found = false
        if y < L[j]
            L[j+1] = L[j]
            j = j - 1
        else
            found = true
    L[j+1] = y
```

i = 4

y = 1

j = 0

found = false



```
INSERTION SORT (L)

for i=1 to n - 1
    y = L[i]
    j = i - 1
    found = false
    while j >= 0 and found = false
        if y < L[j]
            L[j+1] = L[j]
            j = j - 1
        else
            found = true
    L[j+1] = y
```

i = 4

y = 1

j =-1

found = false