Politechnika Wrocławska

Podstawy programowania W11OPA-SI0072G
Wstęp do programowania W11FTE-SI0141WL
Wstęp do programowania W11IKW-SI0080WL

rok akademicki 2023/24
semestr letni

Wykład 8

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220

Politechnika Wrocławska

Plan prezentacji

- Co to jest algorytm?
- Algorytmy sortowania:
  - Sortowanie przez wybór (selectsort)
  - Sortowanie bąbelkowe (bubblesort)
  - Sortowanie przez scalanie (mergesort)
  - Sortowanie szybkie (quicksort)

M. M. Sysło, *Algorytmy*, WSiP, 2002

Politechnika Wrocławska

Co to jest algorytm?

- Algorytm jest przepisem opisującym krok po kroku rozwiązanie problemu lub osiągnięcie jakiegoś celu
- Algorytmika to dziedzina zajmująca się algorytmami i ich właściwościami (dowodzenie poprawności, złożoność obliczeniowa i pamięciowa)

Politechnika Wrocławska

Zapis algorytmów

- Algorytm może być zapisywany na różne sposoby: językiem naturalnym, pseudokodem, schematem blokowym, językiem programowania

Politechnika Wrocławska

Znajdowanie elementu maksymalnego

- Dane wejściowe:
  - *n*-elementowa lista liczb
- Dane wyjściowe:
  - wartość największej liczby na liście

Politechnika Wrocławska

Znajdowanie elementu maksymalnego

01_element_maksymalny.py

```
1  #Program ilustruje działanie funkcji wbudowanej max()
2  #do znalezienia największego elementu na liście.
3
4  #Lista jest pomieszana funkcją shuffle z modułu random.
5
6  import random #import modułu - wykorzystywana funkcja shuffle
7
8  def main():
9      #utworzenie listy zawierającej liczby całkowite
10     l = list(range(8))
11     #pomieszanie kolejności liczb
12     random.shuffle(l)
13     print('Zawartość pomieszanej listy: ',l)
14     print('Element maksymalny znaleziony funkcją max():', max(l))
15
16 main()
```

Politechnika Wrocławska

Znajdowanie elementu maksymalnego

- Przypisz *maksimum* wartość początkowego elementu tablicy
- Dla kolejnych elementów tablicy:
  - Jeśli dany element jest większy od *maksimum*
    - Przypisz *maksimum* wartość danego elementu

Politechnika Wrocławska

Znajdowanie elementu maksymalnego

02_element_maksymalny.py

```python
#Program przedstawia funkcję my_max(),
#która znajduje największy element na liście.

#Lista jest pomieszana funkcją shuffle z modułu random.

import random #import modułu - wykorzystywana funkcja shuffle

def my_max(l):
    m = l[0]
    for item in l[1:]:
        if item > m:
            m = item
    return m

def main():
    #utworzenie listy zawierającej liczby całkowite
    l = list(range(8))
    #pomieszanie kolejności liczb
    random.shuffle(l)
    print('Zawartość pomieszanej listy: ',l)
    print('Element maksymalny znaleziony funkcją my_max():', my_max(l))

main()
```

Politechnika Wrocławska

Znajdowanie elementu maksymalnego

02_element_maksymalny.py

```
1  #Program przedstawia funkcję my_max(),
2  #która znajduje największy element na liście.
3
4  #Lista jest pomieszana funkcją shuffle z modułu random.
5
6  import random #import modułu - wyk
7
8  def my_max(l):
9      m = l[0]
10     for item in l[1:]:
11         if item > m:
12             m = item
13     return m
14
15 def main():
16     #utworzenie listy zawierającej
17     l = list(range(8))
18     #pomieszanie kolejności liczb
19     random.shuffle(l)
20     print('Zawartość pomieszanej listy: ',l)
21     print('Element maksymalny znaleziony funkcją my_max():', my_max(l))
22
23 main()
```

02_element_maksymalny.py

```
8  def my_max(l):
9      m = l[0]
10     for item in l[1:]:
11         if item > m:
12             m = item
13     return m
```

Politechnika Wrocławska

Znajdowanie elementu maksymalnego

03_element_maksymalny.py

```python
#Program przedstawia funkcję my_max(),
#która znajduje największy element na liście.
#Wykorzystuje indeks do przejścia przez elementy listy.

#Lista jest pomieszana funkcją shuffle z modułu random.

import random #import modułu - wykorzystywana funkcja shuffle

def my_max(l):
    m = l[0]
    n = len(l)
    for i in range(1,n):
        if l[i] > m:
            m = l[i]

    return m

def main():
    #utworzenie listy zawierającej liczby całkowite
    l = list(range(8))
    #pomieszanie kolejności liczb
    random.shuffle(l)
    print('Zawartość pomieszanej listy: ',l)
    print('Element maksymalny znaleziony funkcją my_max():', my_max(l))

main()
```

Politechnika Wrocławska

Znajdowanie elementu maksymalnego

03_element_maksymalny.py

```
#Program przedstawia funkcję my_max(),
#która znajduje największy element na liście.
#Wykorzystuje indeks do przejścia przez elementy listy.

#Lista jest pomieszana funkcją sh

import random #import modułu - wy

def my_max(l):
    m = l[0]
    n = len(l)
    for i in range(1,n):
        if l[i] > m:
            m = l[i]

    return m

def main():
    #utworzenie listy zawierające
    l = list(range(8))
    #pomieszanie kolejności liczb
    random.shuffle(l)
    print('Zawartość pomieszanej listy: ',l)
    print('Element maksymalny znaleziony funkcją my_max():', my_max(l))

main()
```

03_element_maksymalny.py

```
def my_max(l):
    m = l[0]
    n = len(l)
    for i in range(1,n):
        if l[i] > m:
            m = l[i]

    return m
```

Politechnika Wrocławska

Zamiana dwóch elementów na liście

04_sort.py

```python
#Moduł 04_sort zawiera funkcję swap(), która pozwala zamienić
#miejscami dwa elementy na liście.

#Moduł zawiera również funkcję main(), która demonstruje działanie
#funkcji swap().

def swap(l,left,right):
    # item     = l[left]
    # l[left]  = l[right]
    # l[right] = item
    l[left], l[right] = l[right], l[left]

def main():
    l = list(range(4))
    print(l)
    swap(l,1,2)
    print(l)

#Instrukcja warunkowa, która sprawdza, czy plik był
#uruchomiony jako skrypt, czy załadowany jako moduł.
if __name__ == "__main__":
    main()
```

Sortowanie bąbelkowe

- Jeżeli ciąg nie jest uporządkowany, to istnieją dwa sąsiednie elementy, które są w złej kolejności

Politechnika Wrocławska

Sortowanie bąbelkowe

3	5	1	2	8	4	7	6

3	1	5	2	8	4	7	6

3	1	2	5	8	4	7	6

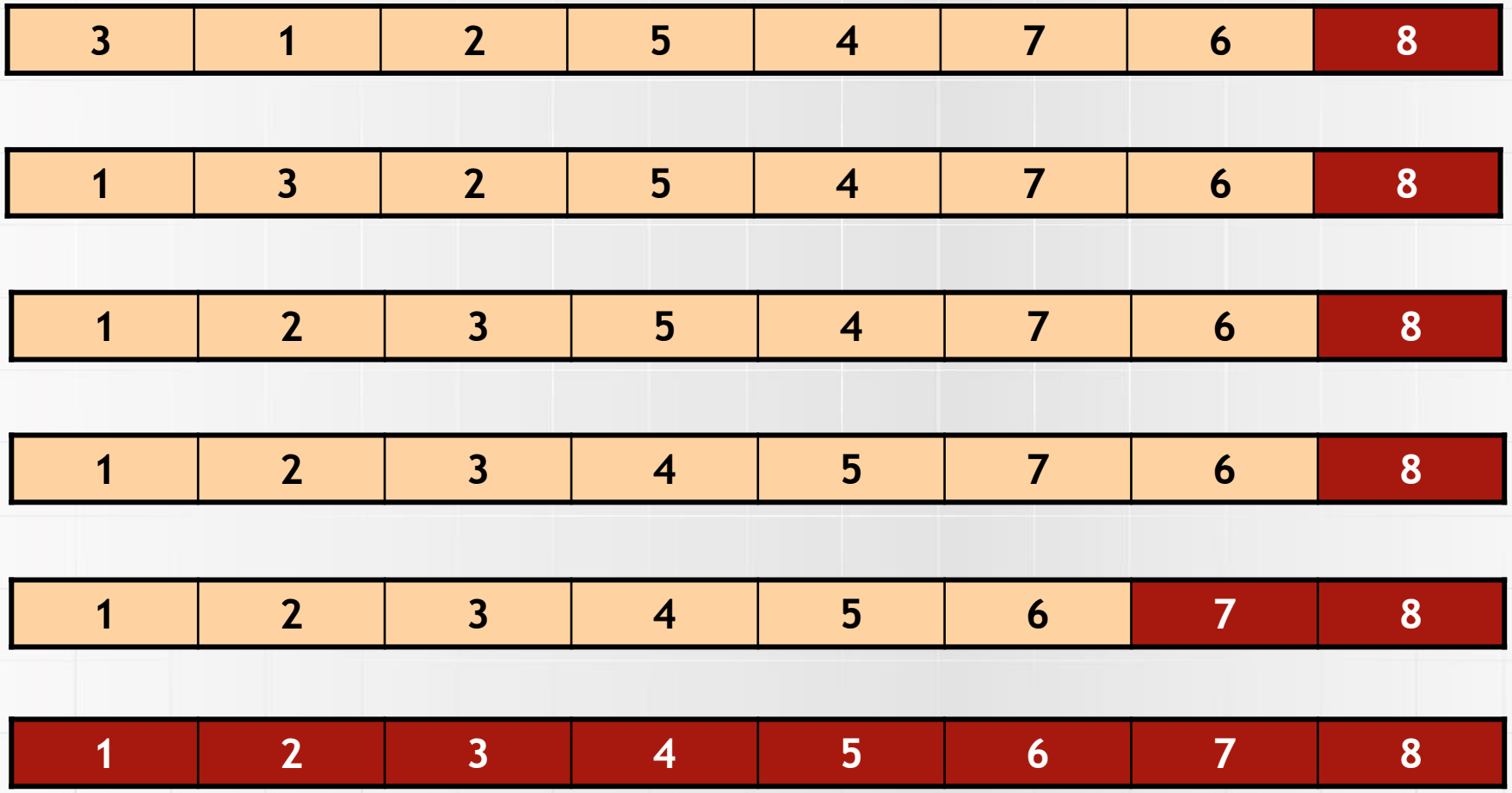
3	1	2	5	4	8	7	6

3	1	2	5	4	7	8	6

3	1	2	5	4	7	6	8

Politechnika Wrocławska

Sortowanie bąbelkowe



Politechnika Wrocławska

Sortowanie bąbelkowe

- Wykonaj *n*-1 przebiegów przez tablicę
  - Wykonaj *n*-*i*-1 porównań sąsiednich elementów (gdzie *i* to numer iteracji liczony od 0)
    - Jeśli elementy są nie po kolei to zamień je miejscami

Politechnika Wrocławska

Sortowanie bąbelkowe

sort.py

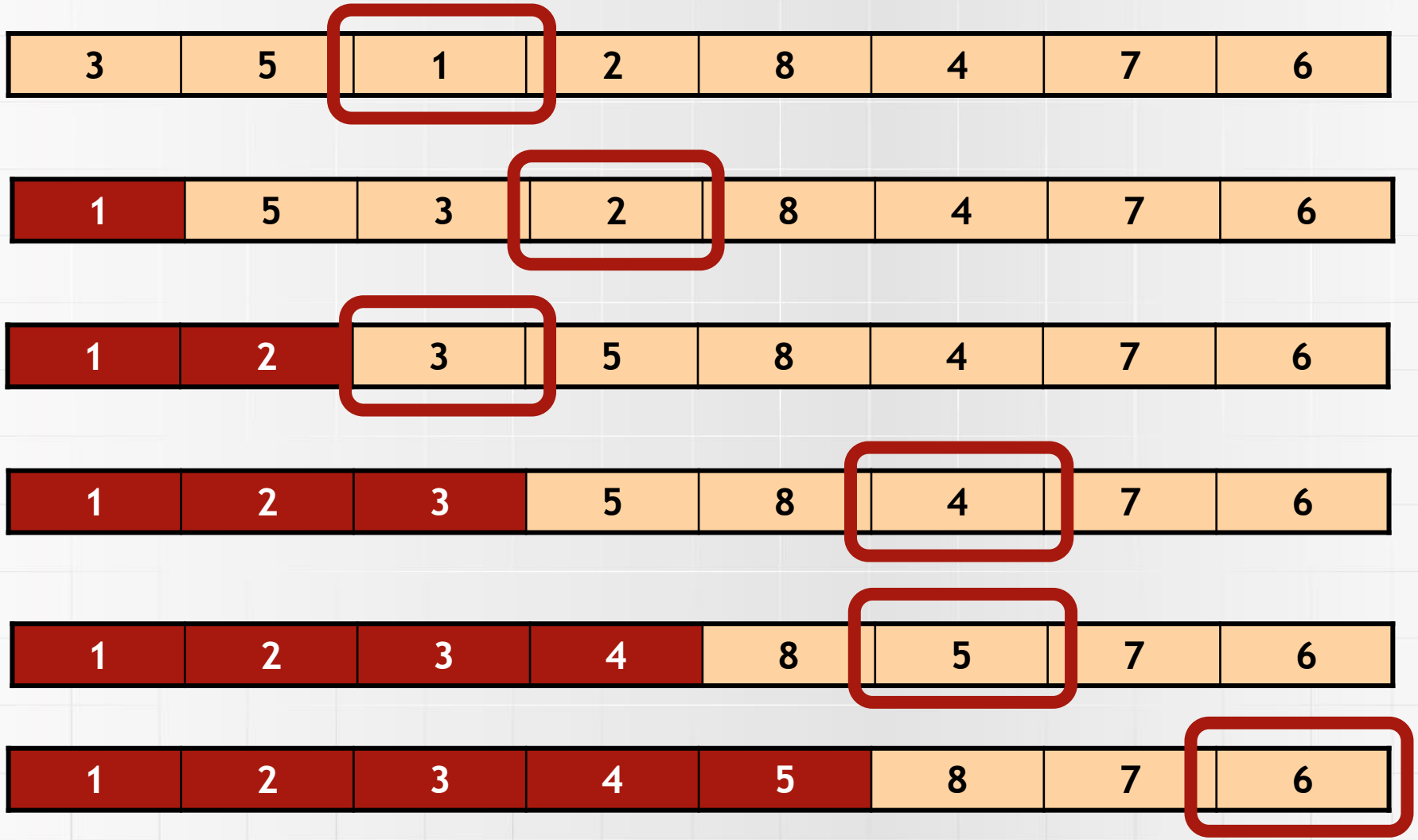
```
1  #Moduł sort.py zawiera funkcje implementujące
2  #różne algorytmy sortowania i funkcje pomocnicze.
3
4  import random #wykorzystywana funkcja shuffle()
5
6  def bubblesort(l):
7      n = len(l)
8      for i in range(n-1):
9          for j in range(n-i-1):
10             if l[j+1] < l[j]:
11                 swap(l,j,j+1)
```

Sortowanie przez wybór

- Wyszukujemy pozycję najmniejszego elementu i przestawiamy go na właściwą pozycję
- Następnie kontynuujemy dla pozostałej części tablicy

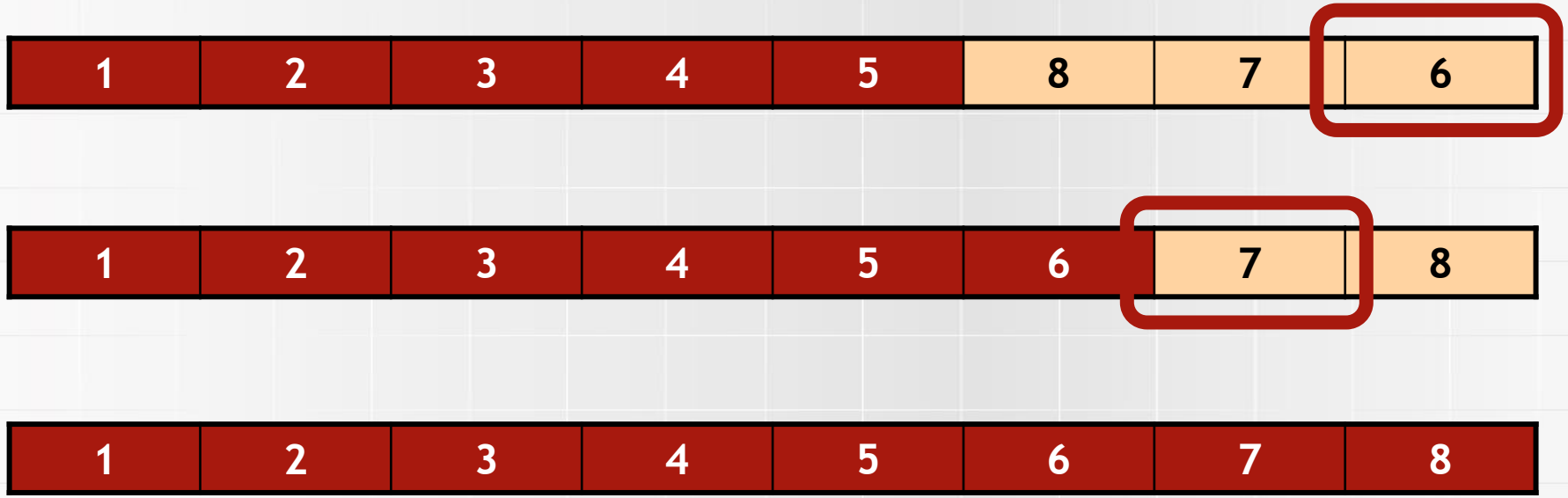
Politechnika Wrocławska

Sortowanie przez wybór



Politechnika Wrocławska

Sortowanie przez wybór



Sortowanie przez scalanie

- Podziel tablicę na dwie równe części
- Zastosuj sortowanie przez scalanie do każdej z nich oddzielnie
- Połącz posortowane podciągi w jeden ciąg posortowany

Politechnika Wrocławska

Sortowanie przez scalanie

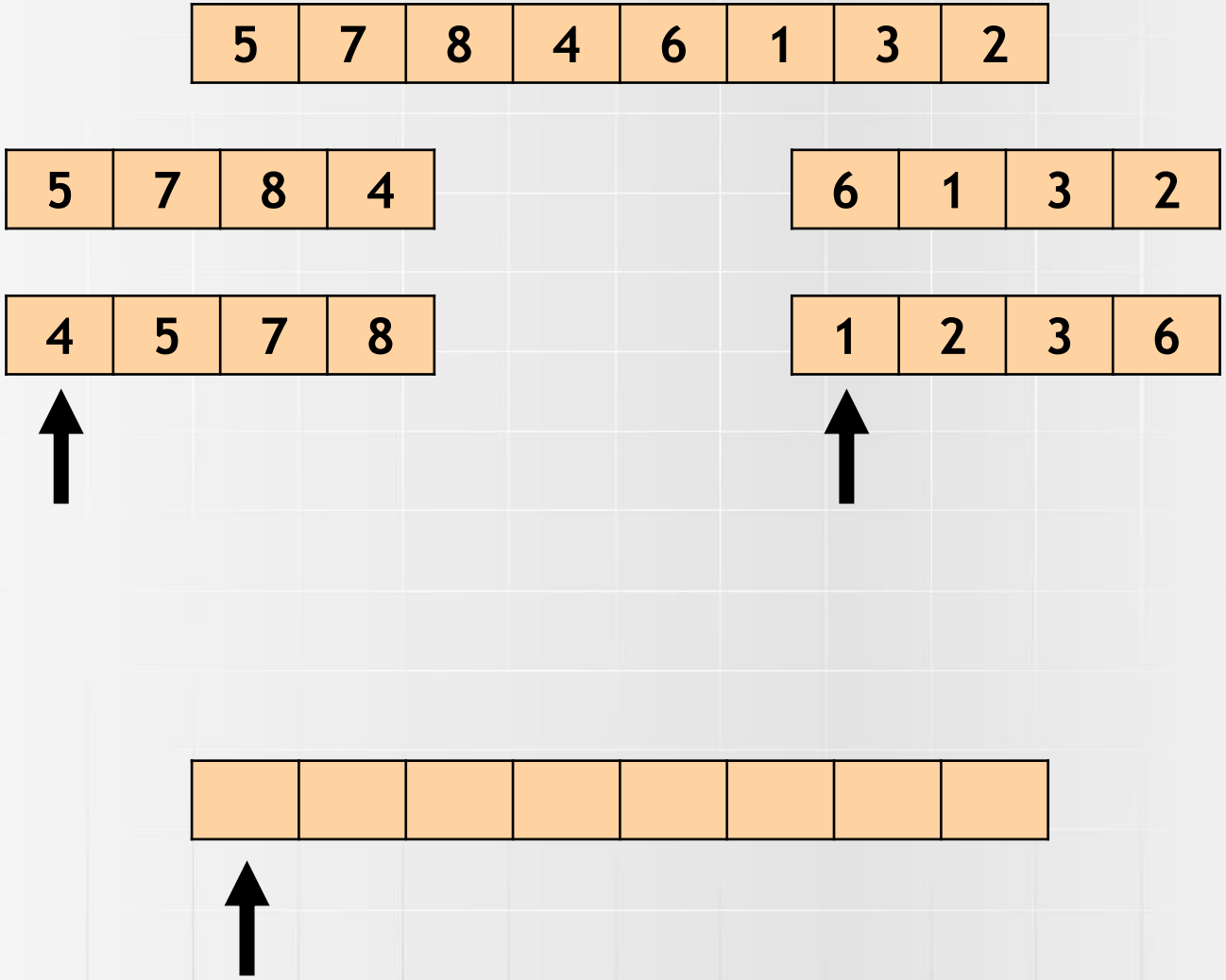
Scalanie

Procedura scalania dwóch ciągów a[0..n-1] i b[0..m-1] do ciągu c[0..n+m-1]

1. Ustaw indeksy na początek ciągów: i=0, j=0
2. Jeśli w ciągu a nie pozostało już nic do przetworzenia (i>=n), to dołącz pozostałe elementy z b co c i zakończ
3. Jeśli w ciągu b nie pozostało już nic do przetworzenia (j>=m), to dołącz pozostałe elementy z a co c i zakończ
4. Jeśli a[i] <= b[j] to dołącz a[i] do c i zwiększ i o 1, w p.p. dołącz b[j] do c i zwiększ j o 1.
5. Powtarzaj od 2.

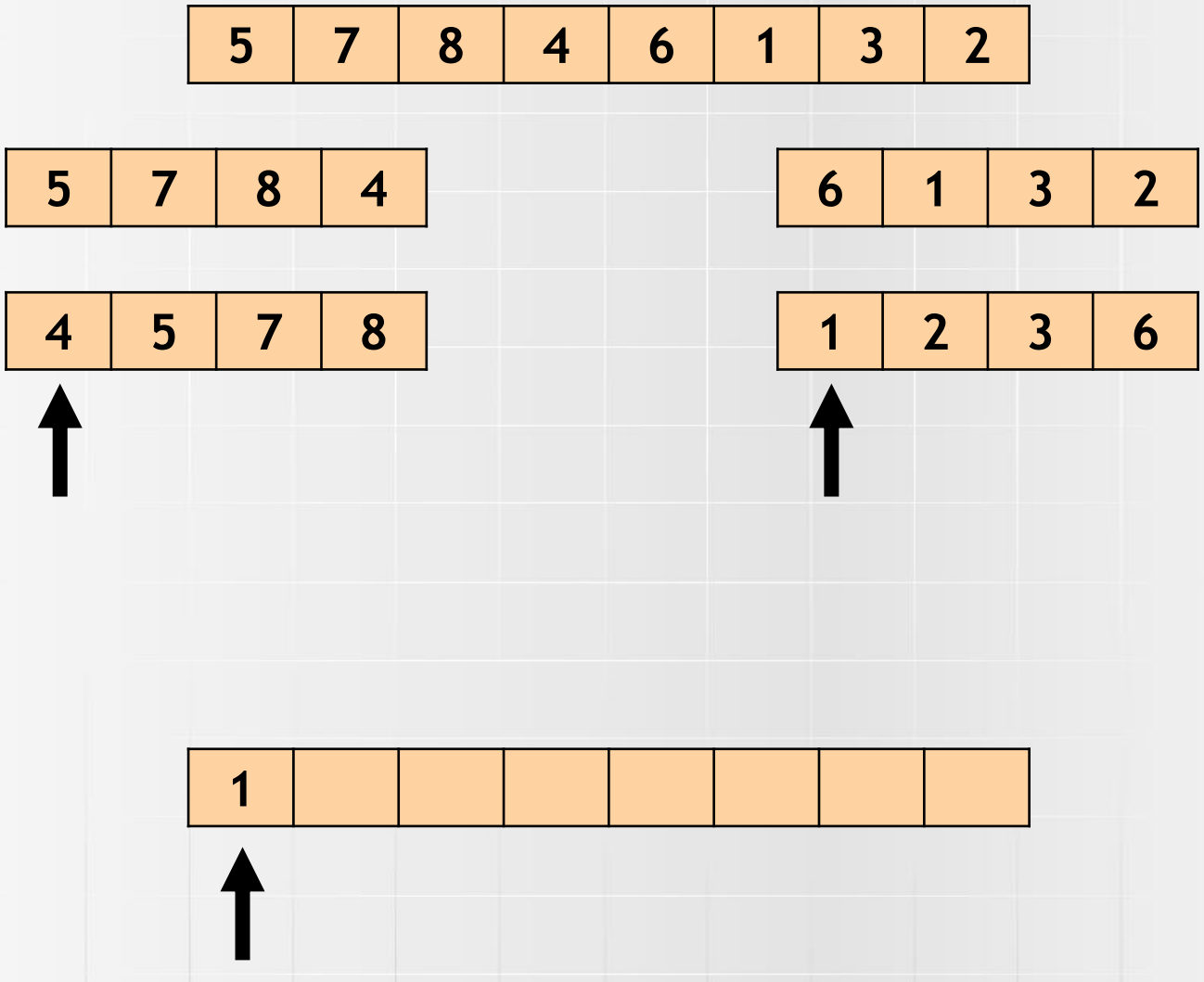
Politechnika Wrocławska

Sortowanie przez scalanie



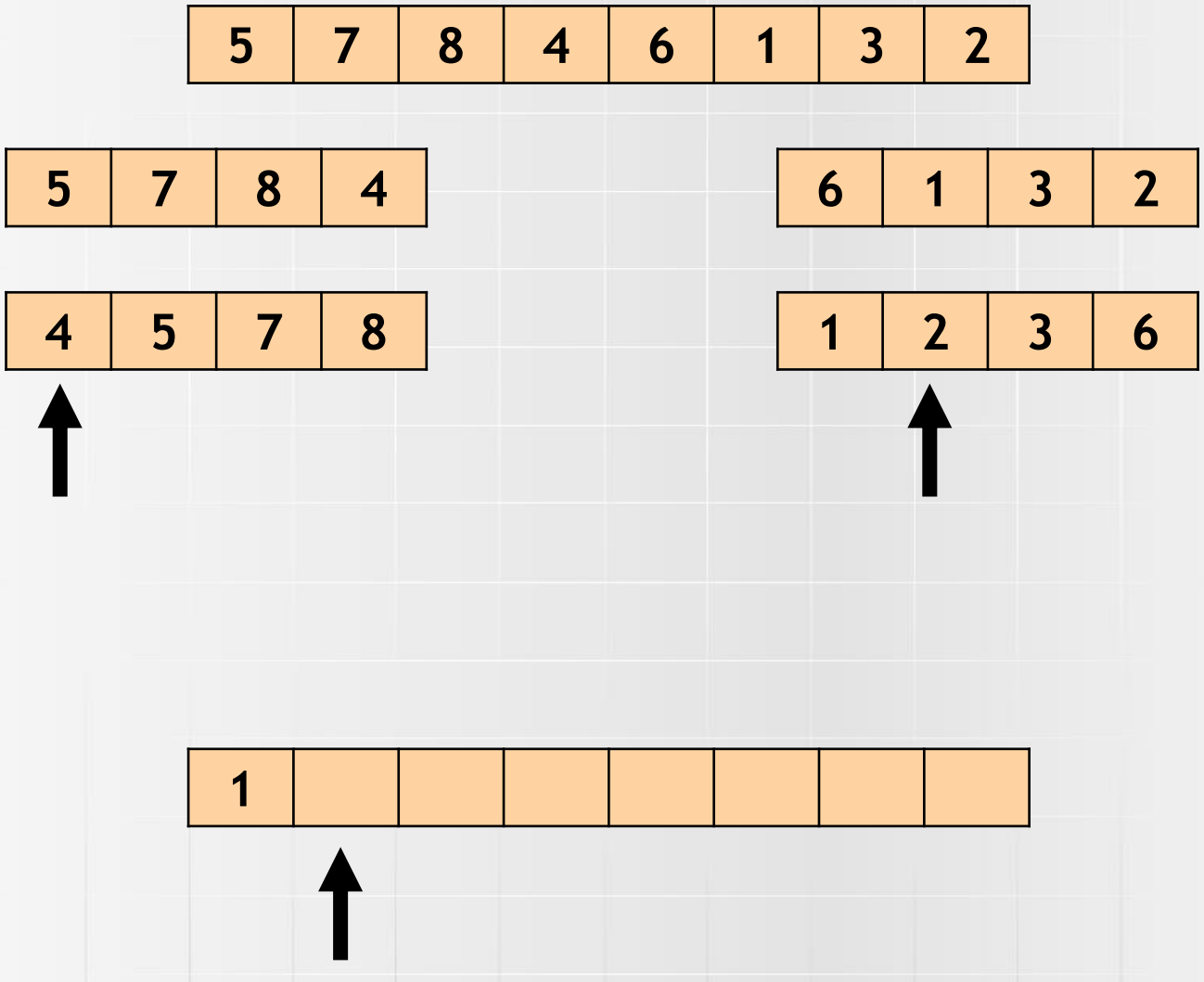
Politechnika Wrocławska

Sortowanie przez scalanie



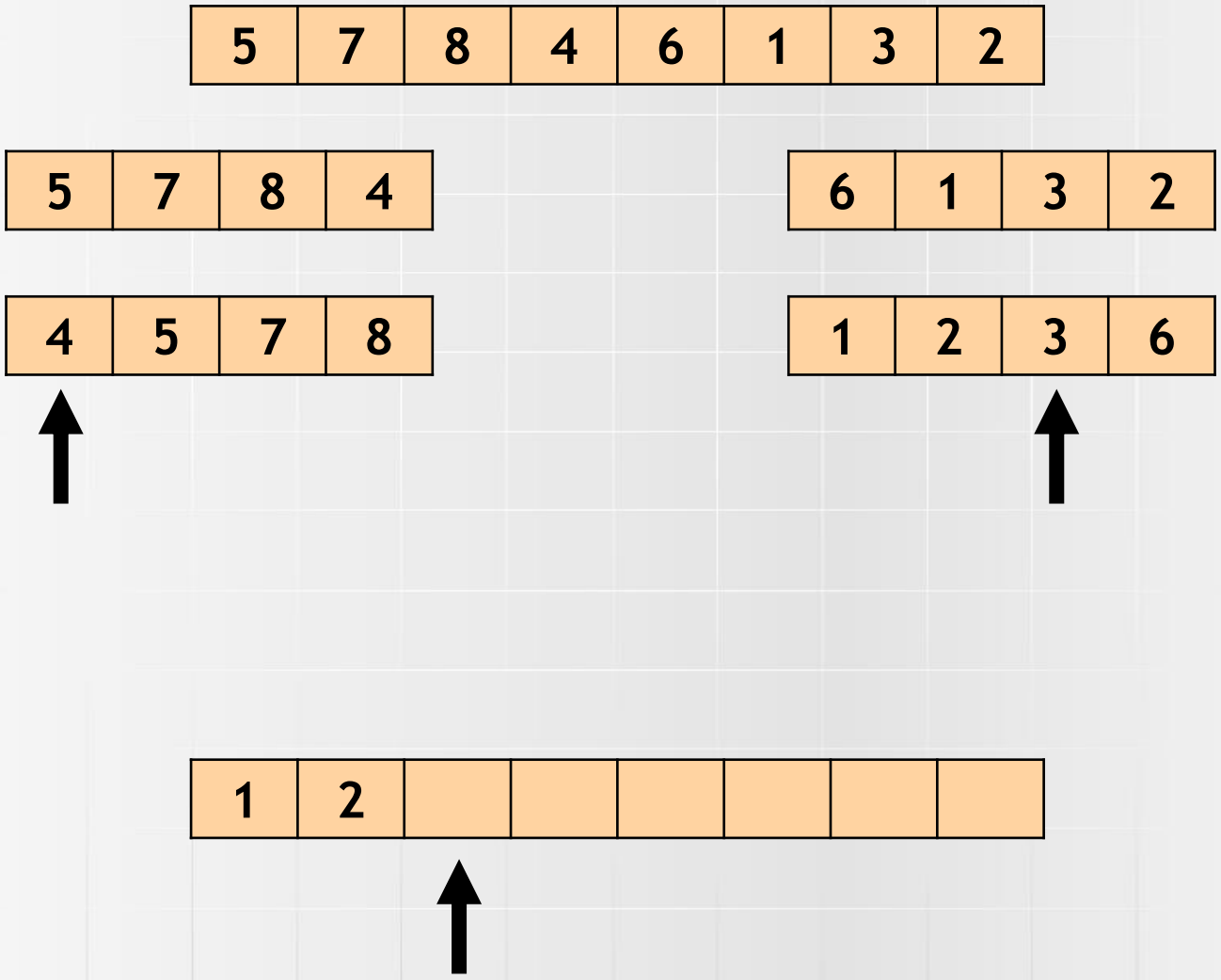
Politechnika Wrocławska

Sortowanie przez scalanie



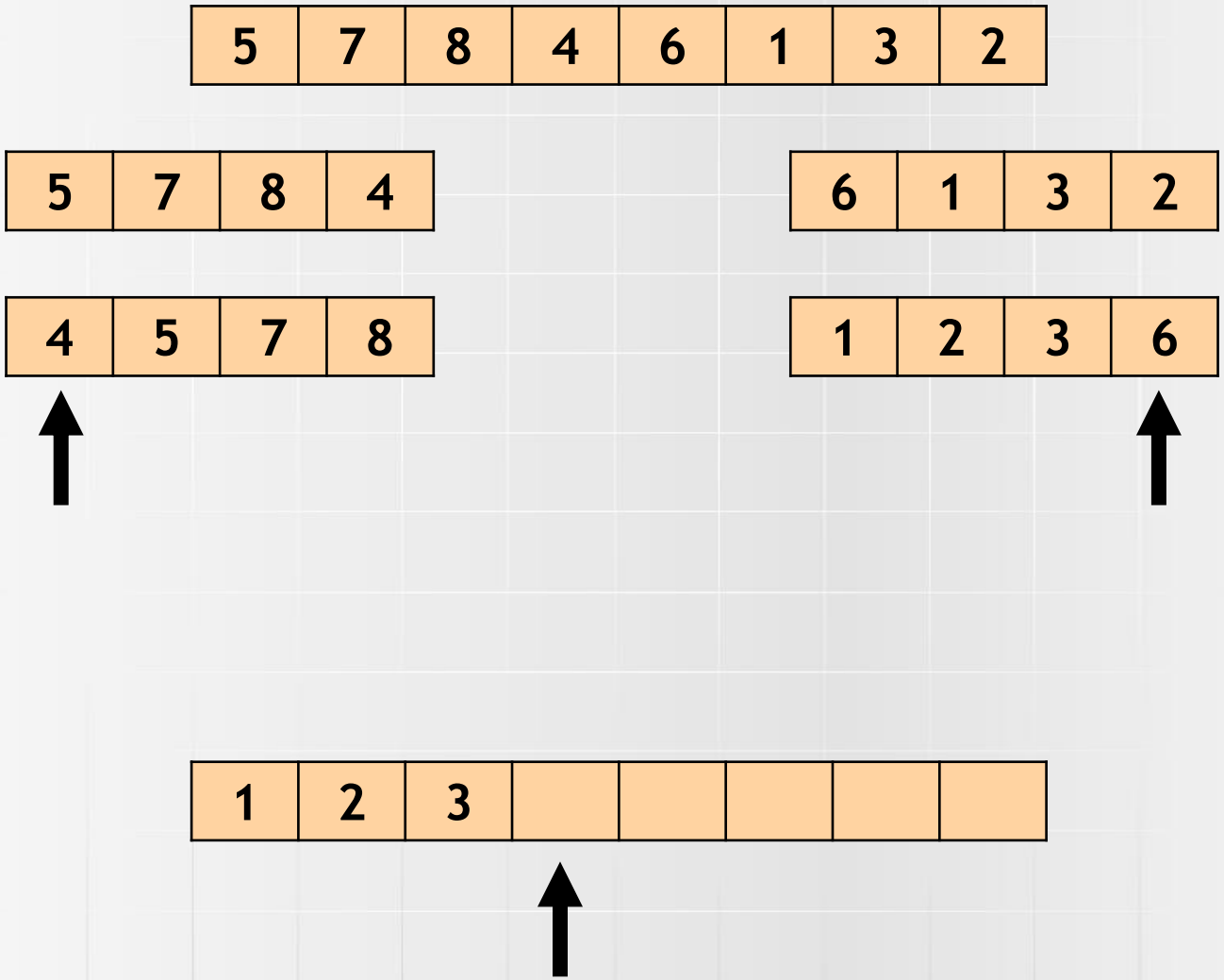
Politechnika Wrocławska

Sortowanie przez scalanie



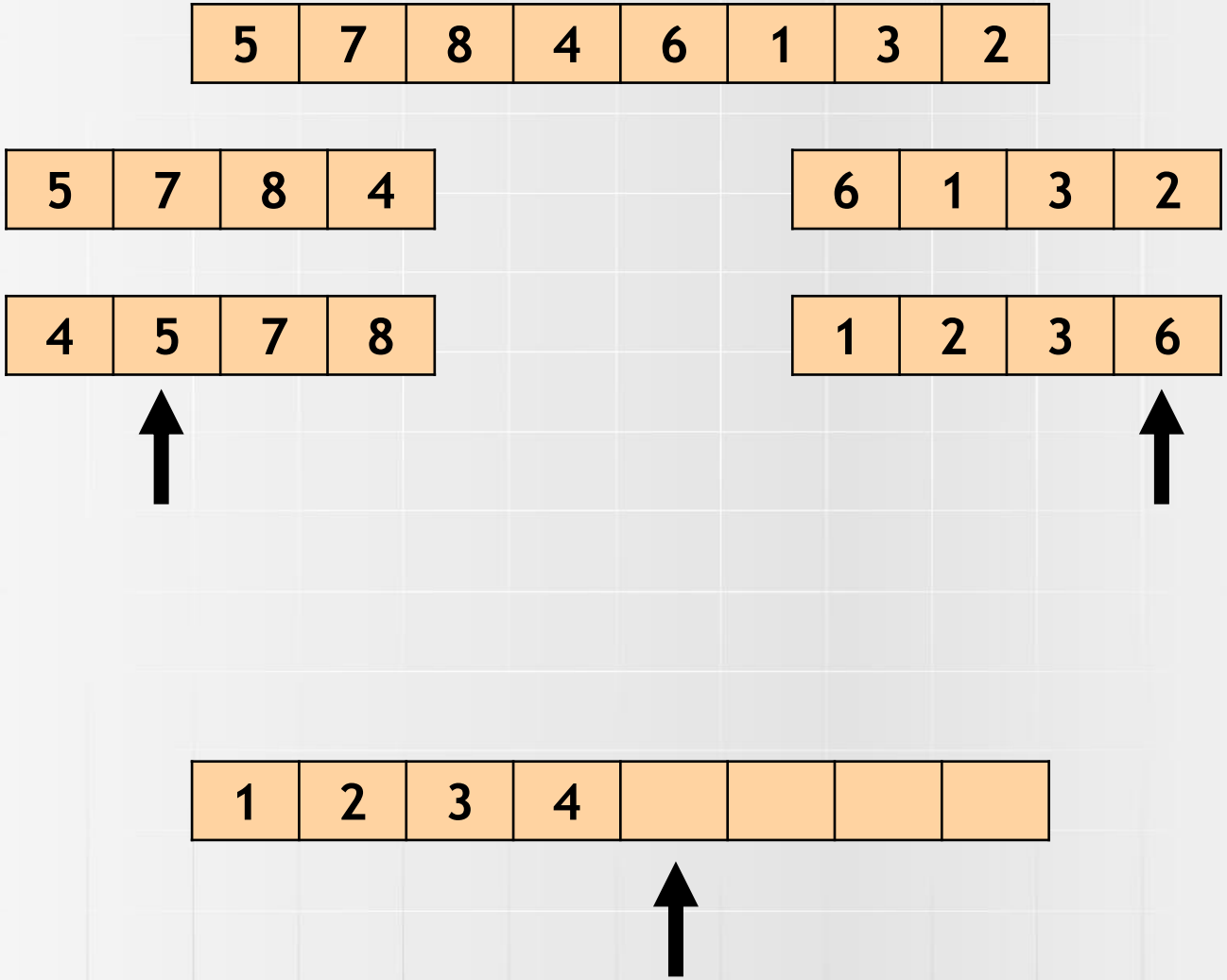
Politechnika Wrocławska

Sortowanie przez scalanie



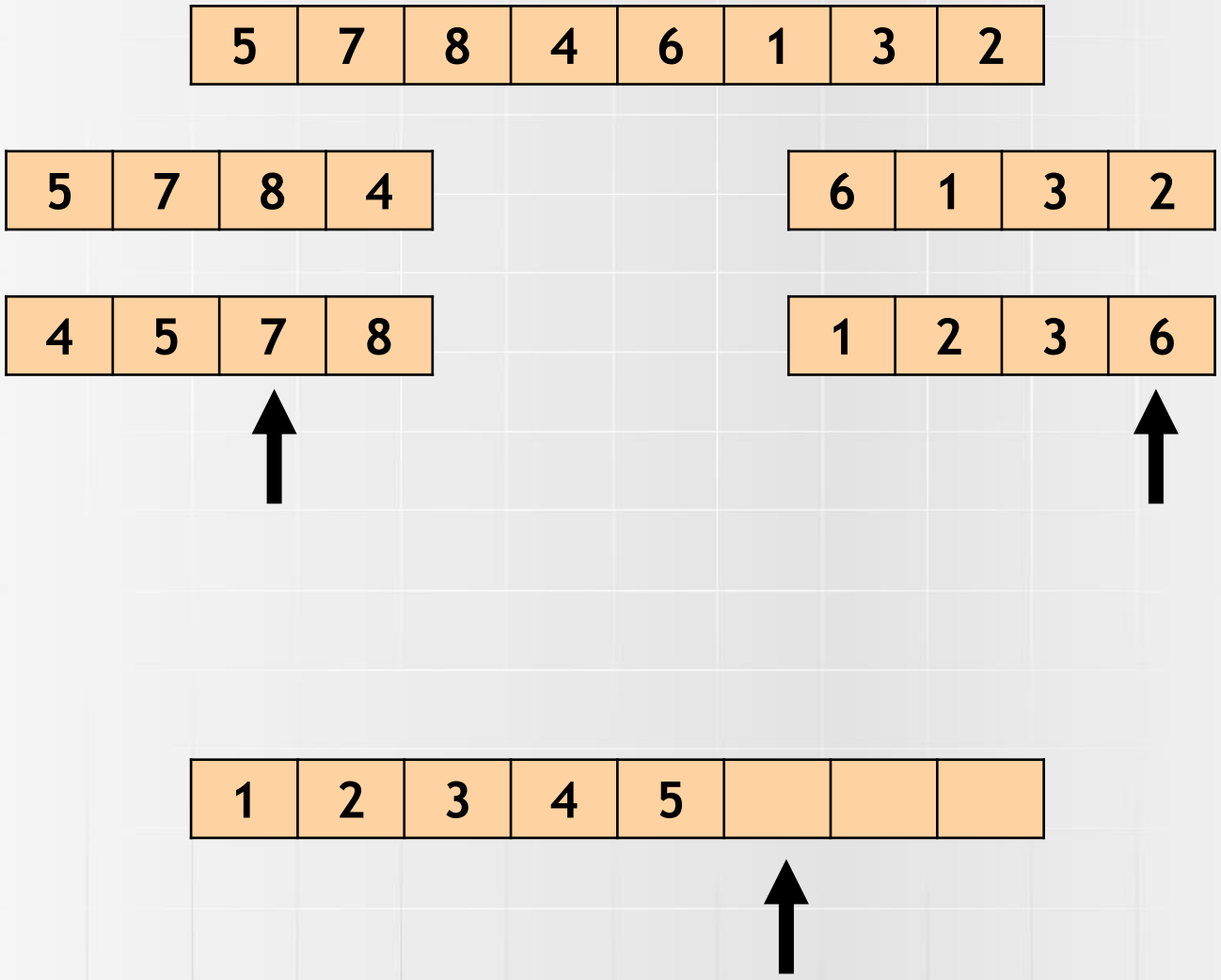
Politechnika Wrocławska

Sortowanie przez scalanie



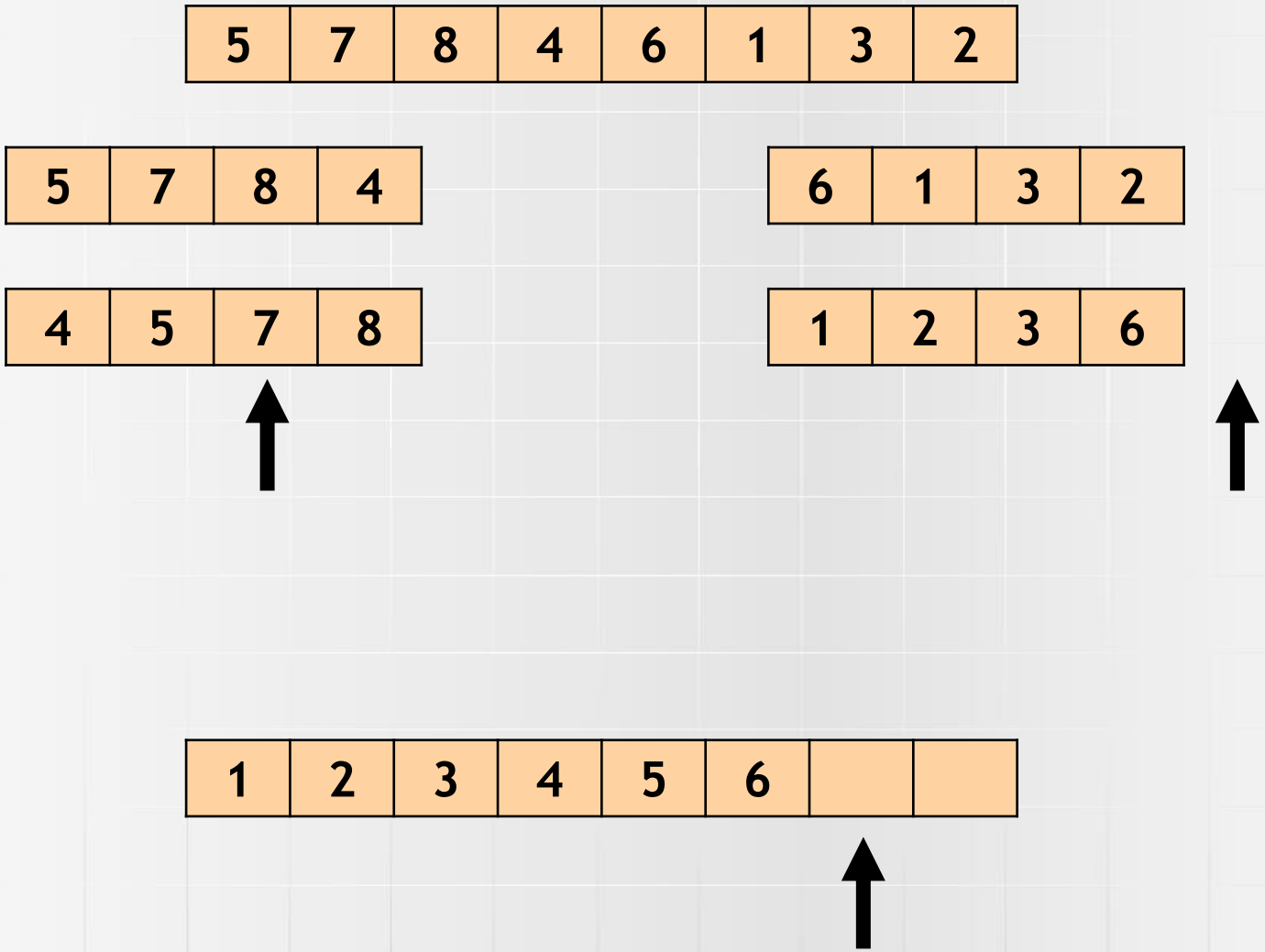
Politechnika Wrocławska

Sortowanie przez scalanie



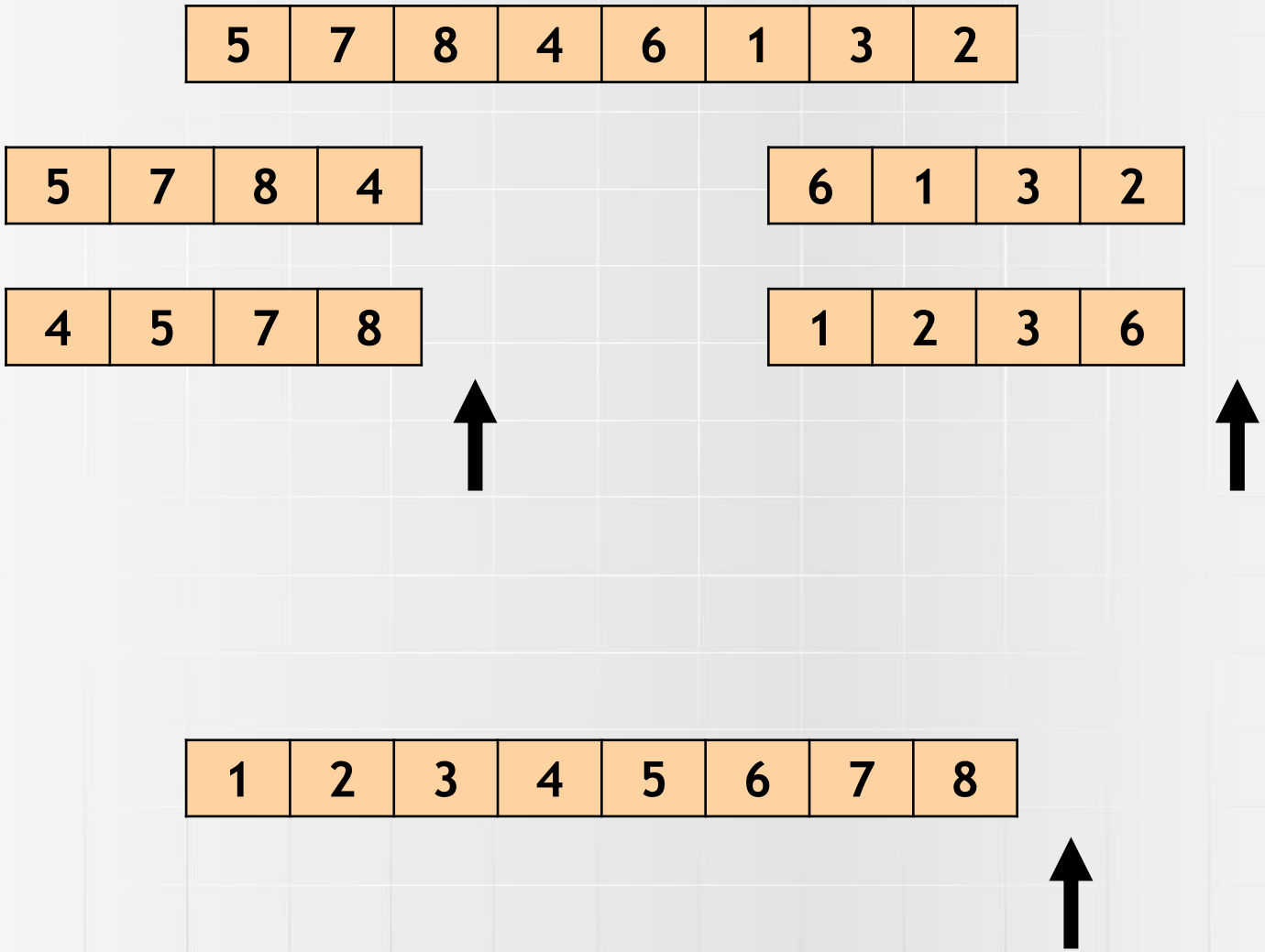
Politechnika Wrocławska

Sortowanie przez scalanie



Politechnika Wrocławska

Sortowanie przez scalanie



Politechnika Wrocławska

Sortowanie przez scalanie

Scalanie

Procedura scalania dwóch ciągów a[0..n-1] i b[0..m-1] do ciągu c[0..n+m-1]

1. Ustaw indeksy na początek ciągów: i=0, j=0
2. Jeśli w ciągu a nie pozostało już nic do przetworzenia (i>=n), to dołącz pozostałe elementy z b co c i zakończ
3. Jeśli w ciągu b nie pozostało już nic do przetworzenia (j>=m), to dołącz pozostałe elementy z a co c i zakończ
4. Jeśli a[i] <= b[j] to dołącz a[i] do c i zwiększ i o 1, w p.p. dołącz b[j] do c i zwiększ j o 1.
5. Powtarzaj od 2.

Sortowanie szybkie

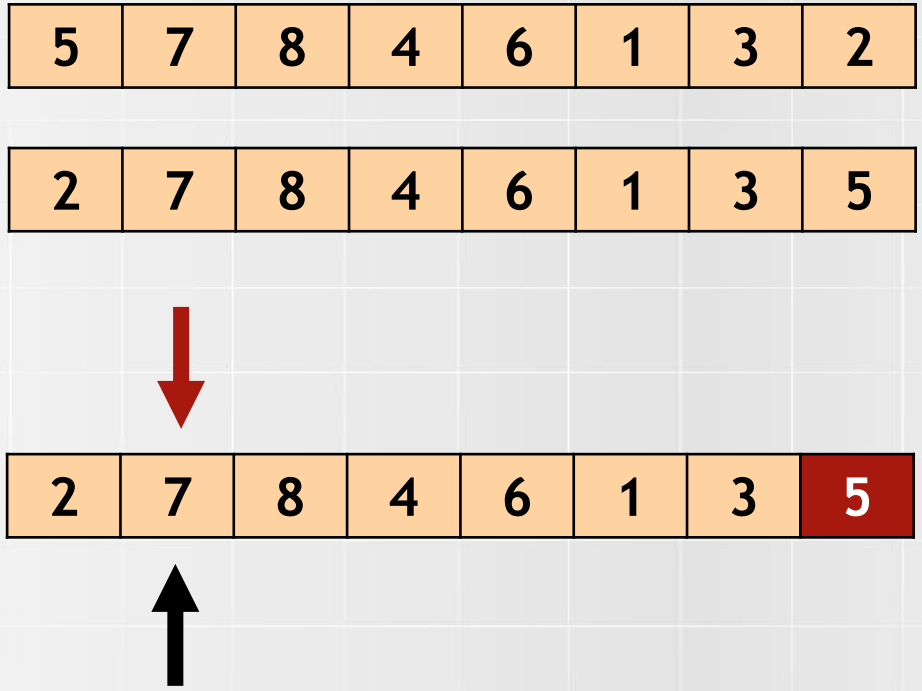
- Podziel tablicę na dwie części zawierające elemnty mniejsze i większe od wybranego elementu
- Zastosuj sortowanie szybkie do każdej z nich oddzielnie
- Połącz posortowane podciągi w jeden ciąg posortowany

Politechnika Wrocławska

Sortowanie szybkie

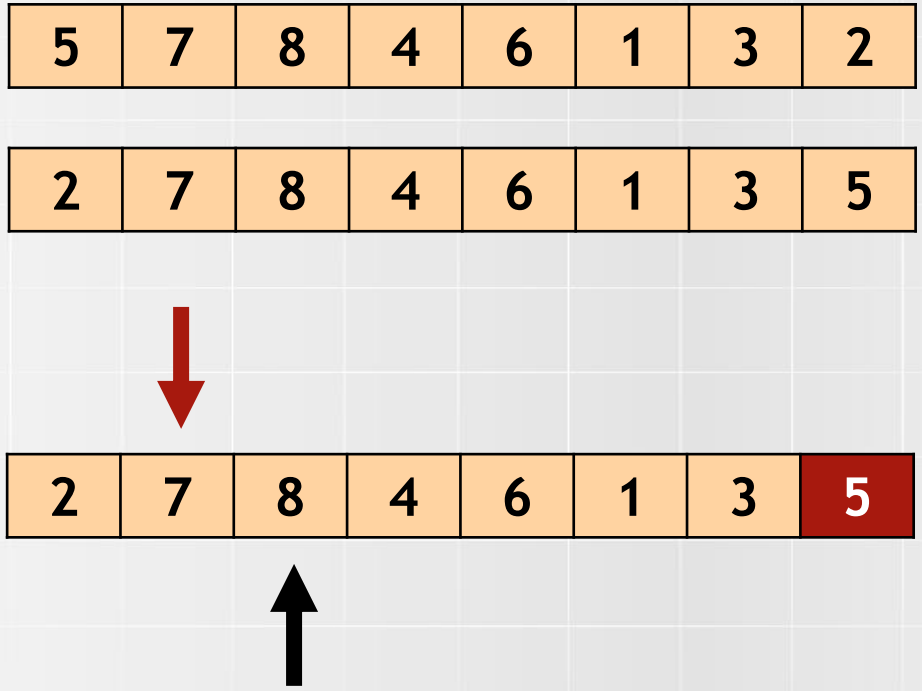
Politechnika Wrocławska

Sortowanie szybkie



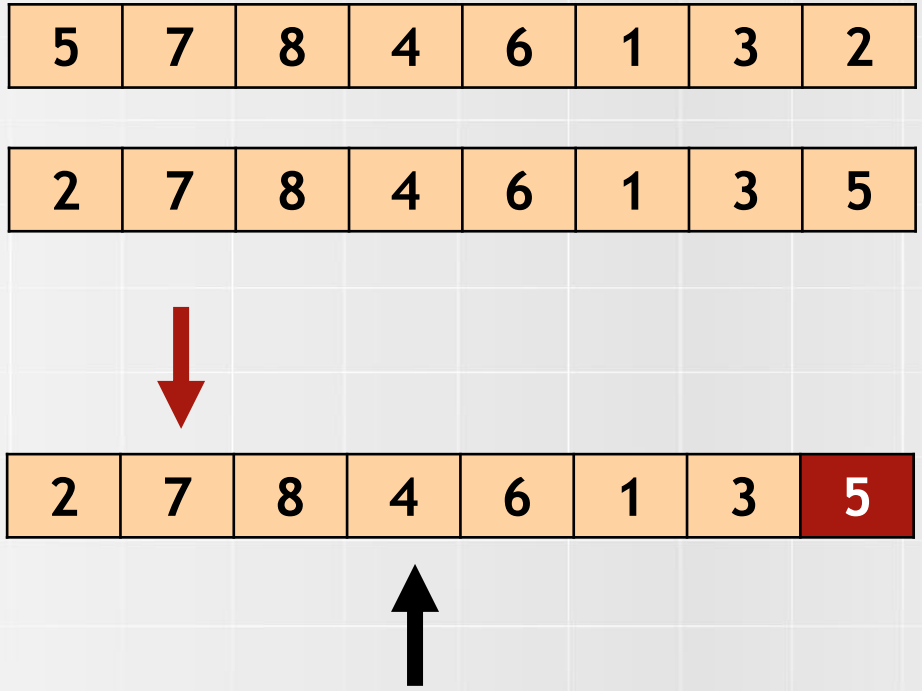
Politechnika Wrocławska

Sortowanie szybkie



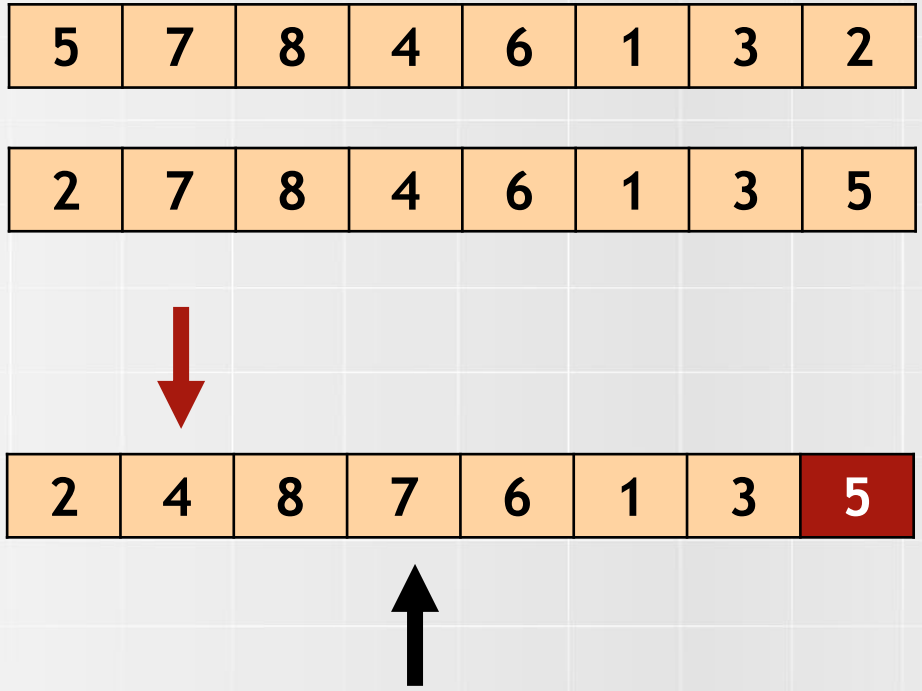
Politechnika Wrocławska

Sortowanie szybkie



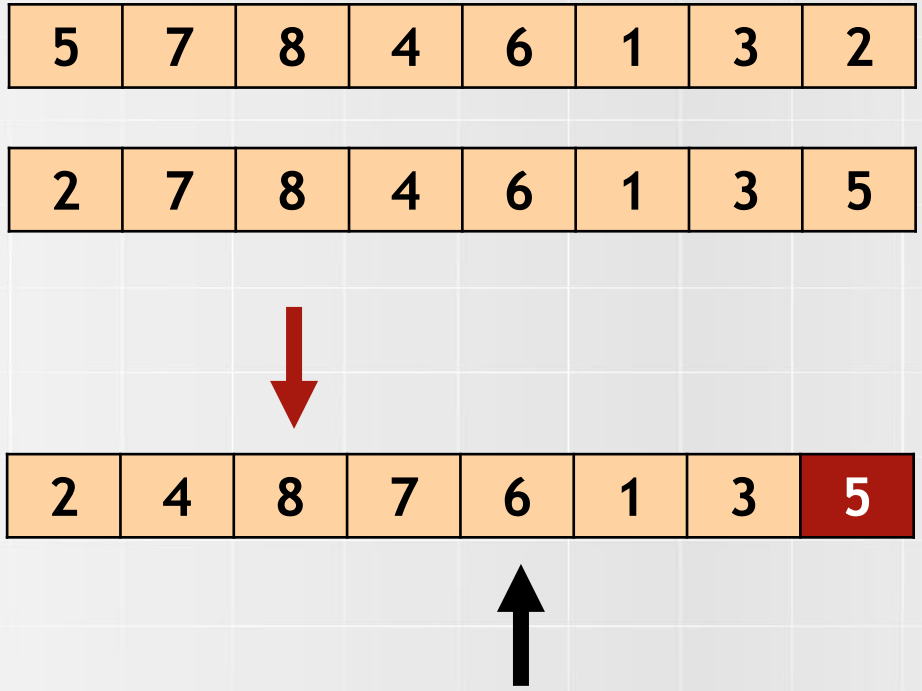
Politechnika Wrocławska

Sortowanie szybkie



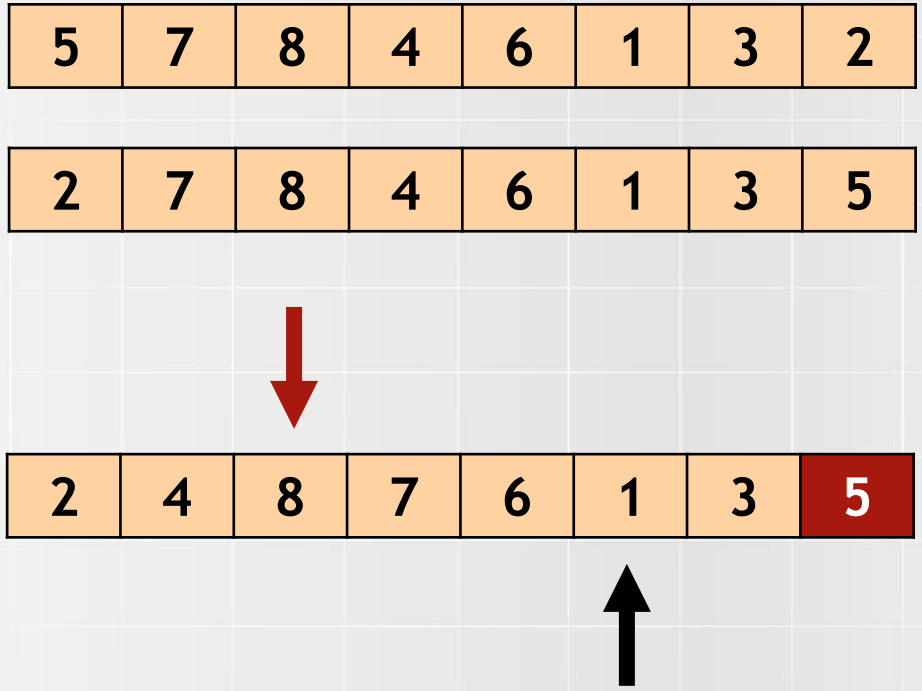
Politechnika Wrocławska

Sortowanie szybkie



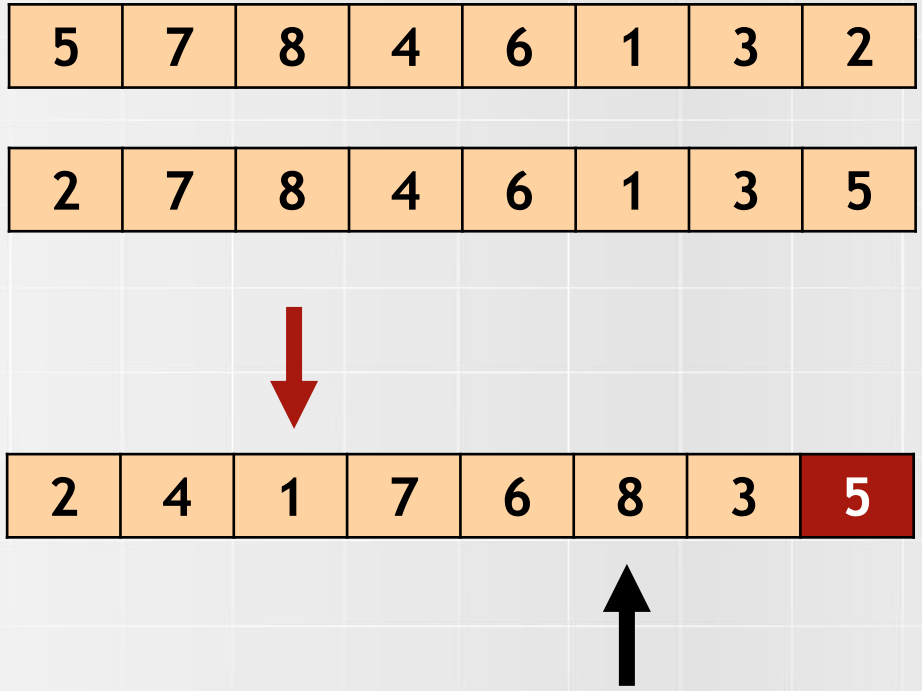
Politechnika Wrocławska

Sortowanie szybkie



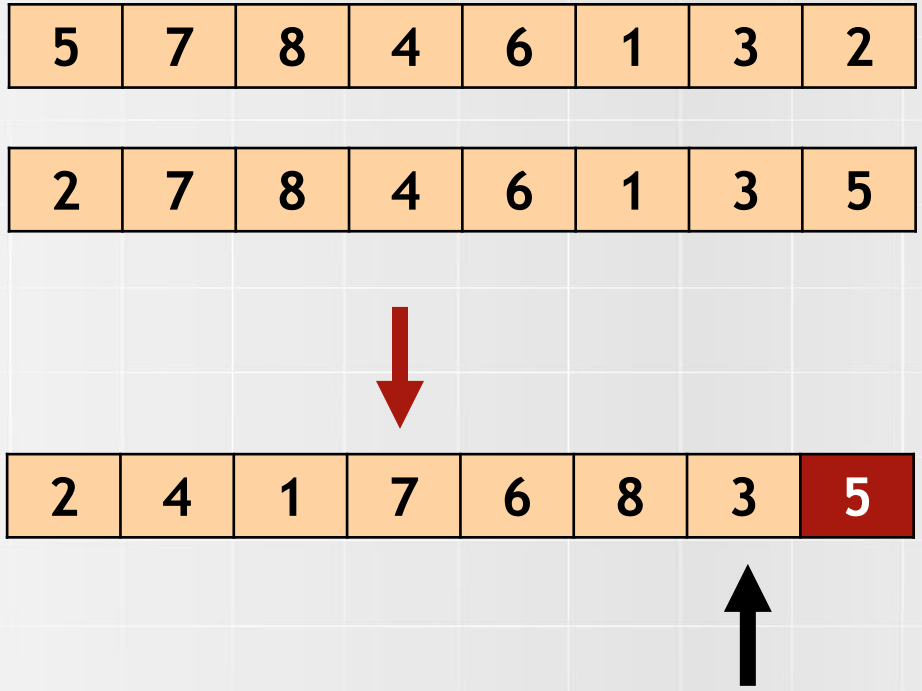
Politechnika Wrocławska

Sortowanie szybkie



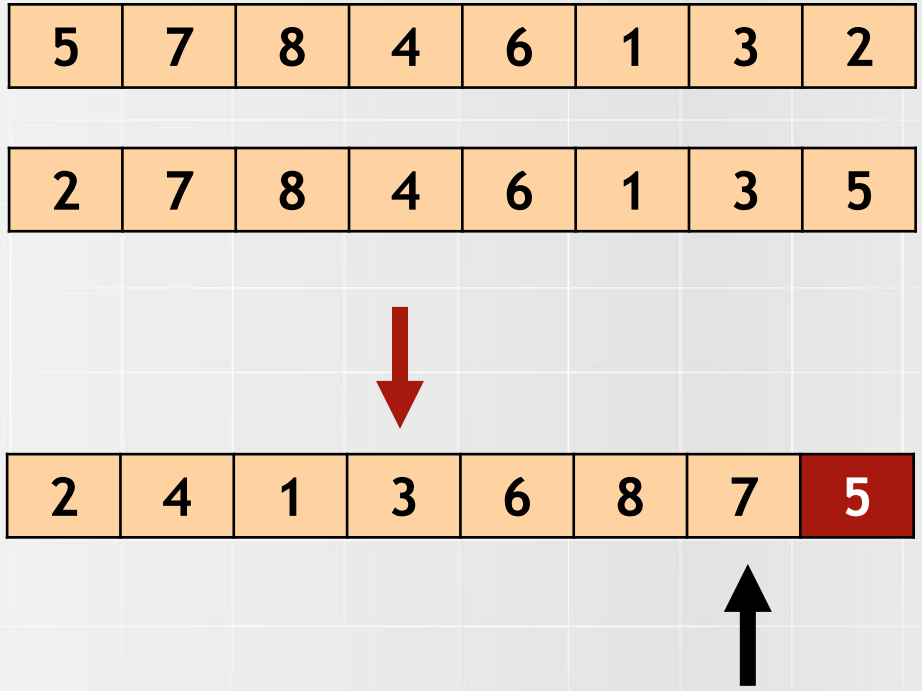
Politechnika Wrocławska

Sortowanie szybkie



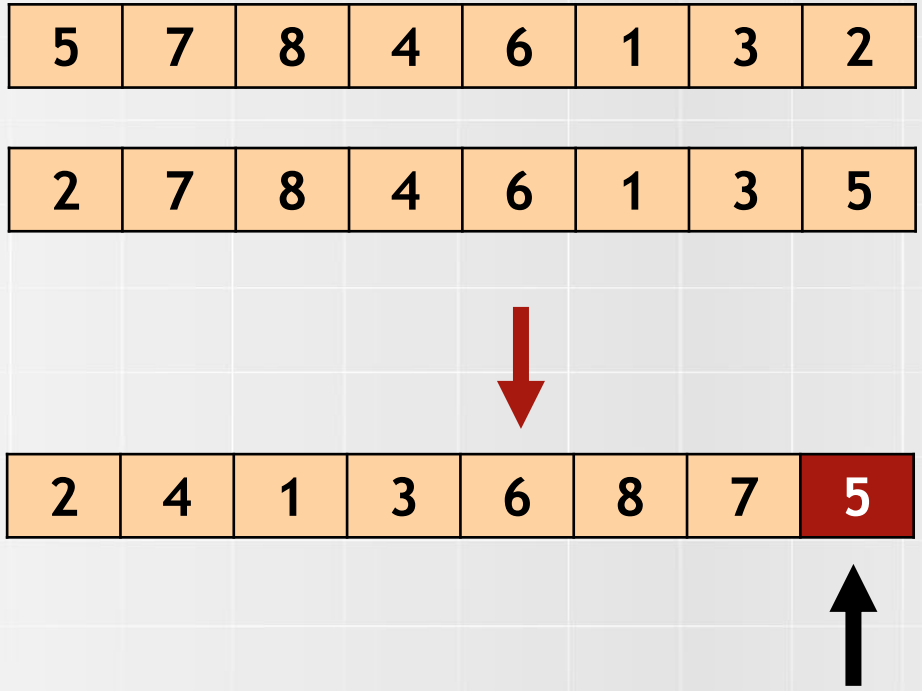
Politechnika Wrocławska

Sortowanie szybkie



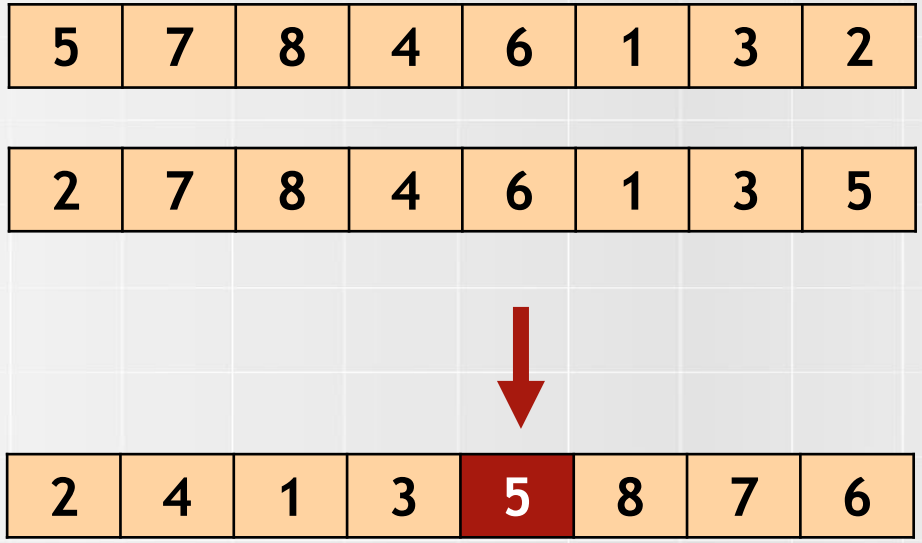
Politechnika Wrocławska

Sortowanie szybkie



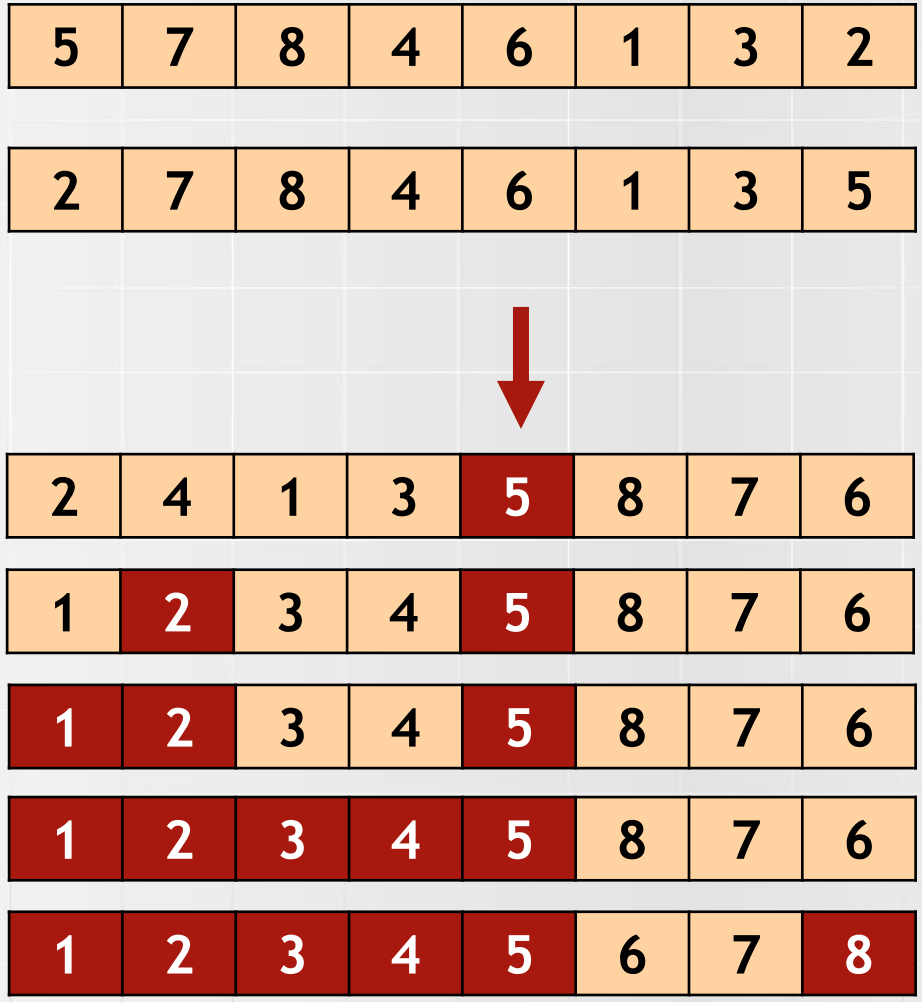
Politechnika Wrocławska

Sortowanie szybkie



Politechnika Wrocławska

Sortowanie szybkie



Politechnika Wrocławska

Podsumowanie

- Co to jest algorytm?
- Co to jest złożoność algorytmu?

- Algorytmy sortowania:
  - Sortowanie przez wybór (selectsort)
  - Sortowanie bąbelkowe (bubblesort)
  - Sortowanie przez scalanie (mergesort)
  - Sortowanie szybkie (quicksort)