Politechnika Wrocławska

Programowanie proceduralne

INP001210WL

rok akademicki 2020/21

semestr letni

Wykład 4

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

Politechnika Wrocławska

Zapis algorytmów

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

Politechnika Wrocławska

Znajdowanie elementu maksymalnego na liście

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

Znajdowanie elementu maksymalnego na liście

01_element_maksymalny.py

File Edit Format Run Options Window Help

```
#Program ilustruje działanie funkcji wbudowanej max()
#do znalezienia największego elementu na liście.

#Lista jest pomieszana funkcją shuffle z modułu random.

import random #import modułu - wykorzystywana funkcja shuffle

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 max():', max(l))

main()
```

Znajdowanie elementu maksymalnego na liście

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

Znajdowanie elementu maksymalnego na liście

02_element_maksymalny.py

File Edit Format Run Options Window Help

```
#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
```

Znajdowanie elementu maksymalnego na liście

03_element_maksymalny.py

File Edit Format Run Options Window Help

```
#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
```

Zamiana miejscami dwóch elementów na liście

04_sort.py

File Edit Format Run Options Window Help

```
#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

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()
```

Politechnika Wrocławska

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

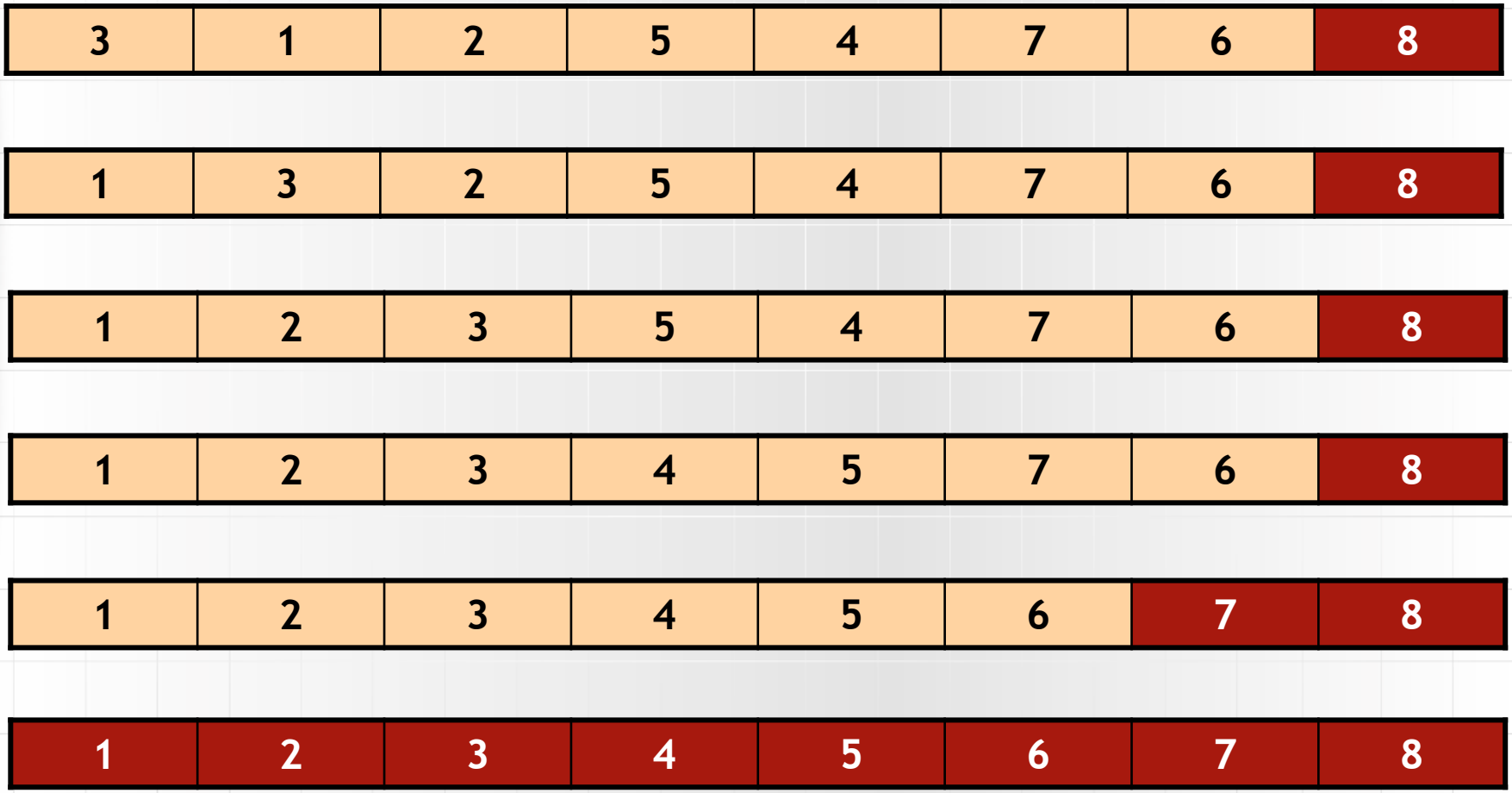
3	1	2	5	4	8	7	6

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

File Edit Format Run Options Window Help

```
#Moduł sort.py zawiera funkcje implementujące
#różne algorytmy sortowania i funkcje pomocnicze.

import random #wykorzystywana funkcja shuffle()

def bubblesort(l):
    n = len(l)
    for i in range(n):
        for j in range(n-i-1):
            if l[j+1] < l[j]:
                swap(l,j,j+1)
```

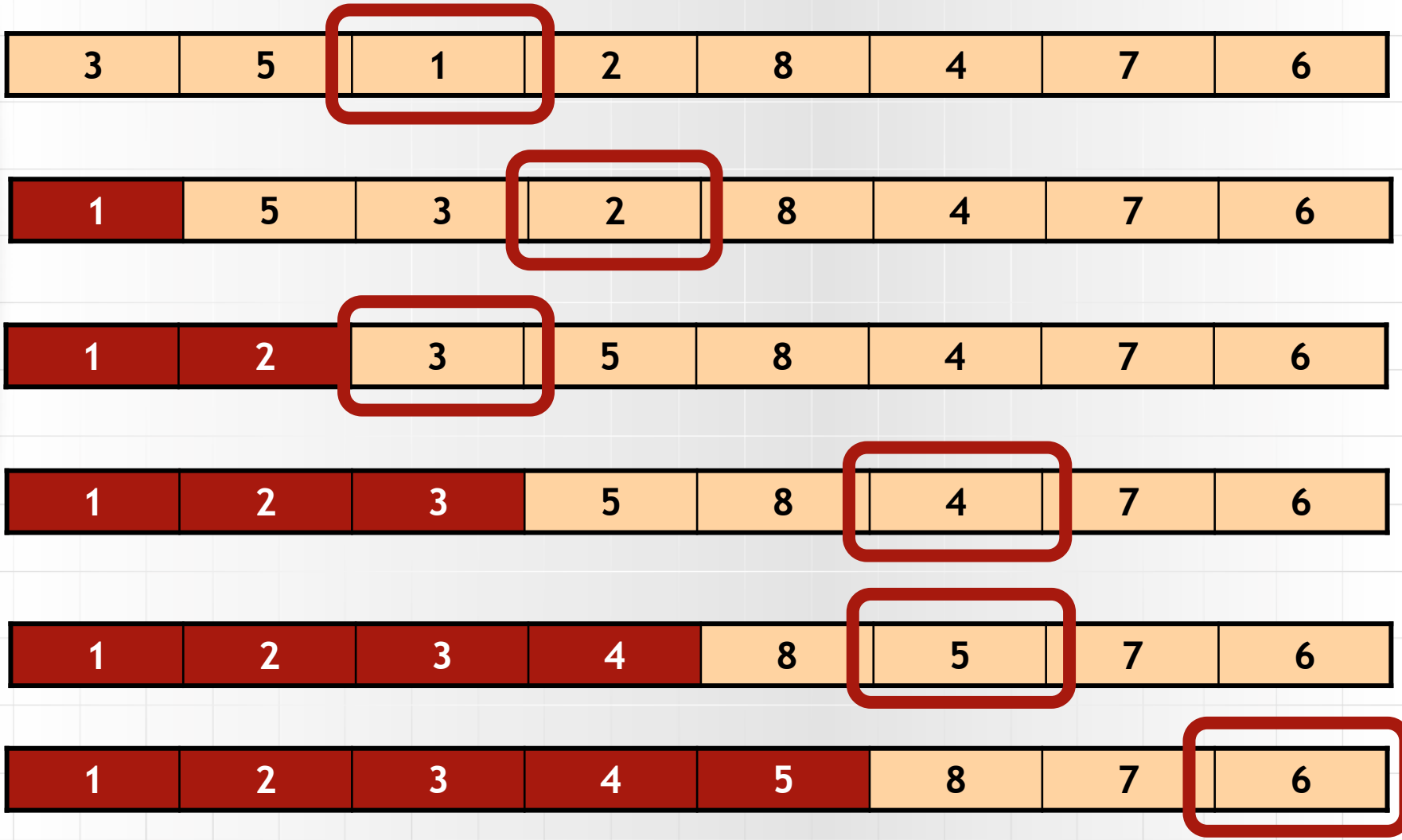
Politechnika Wrocławska

Sortowanie przez wybór

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

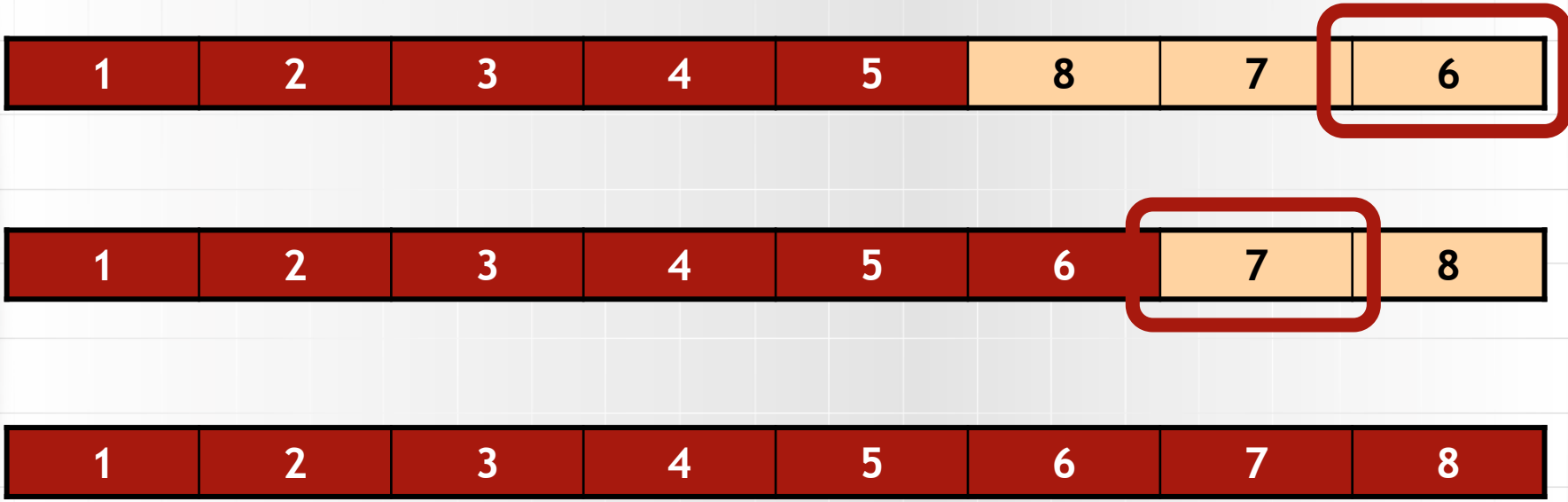
Politechnika Wrocławska

Sortowanie przez wybór



Politechnika Wrocławska

Sortowanie przez wybór



Politechnika Wrocławska

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

Sortowanie przez scalanie

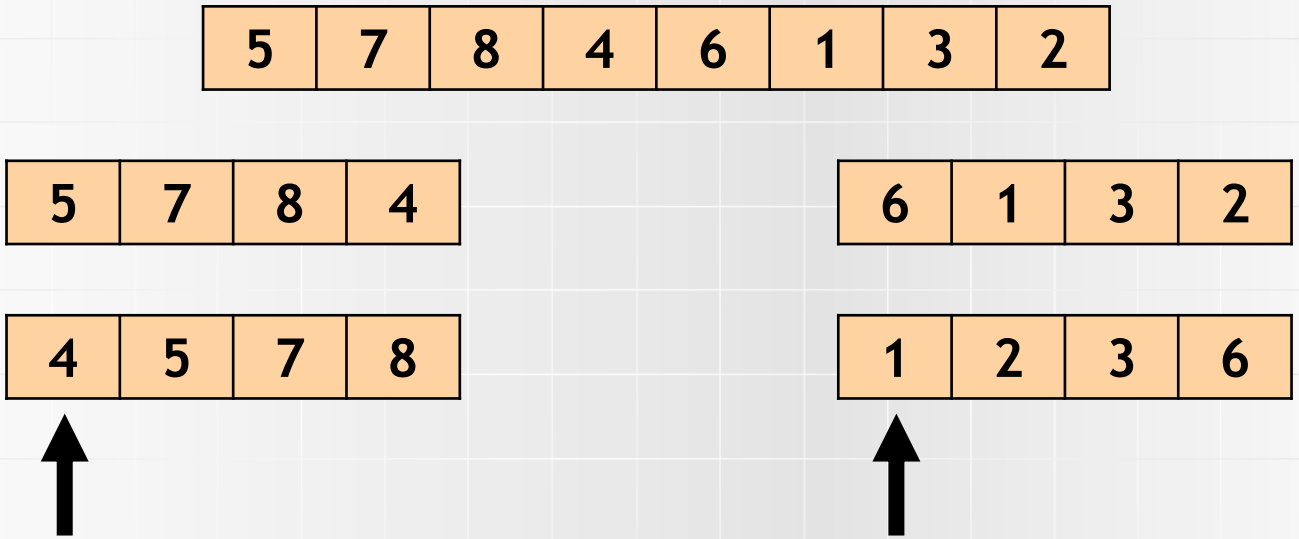
Scalanie

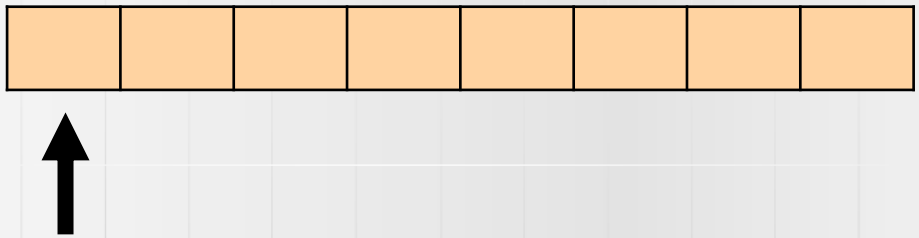
Procedura scalania dwóch ciągów a[0..n-1] i b[0..m-1] do ciągu c[0..n+m-1]

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

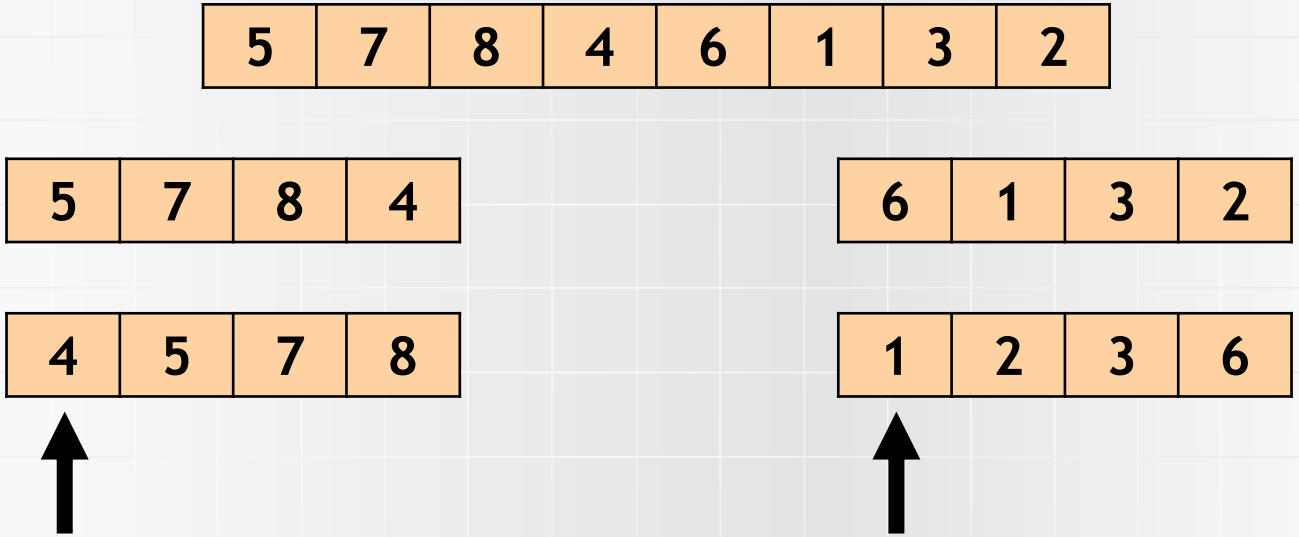
Sortowanie przez scalanie

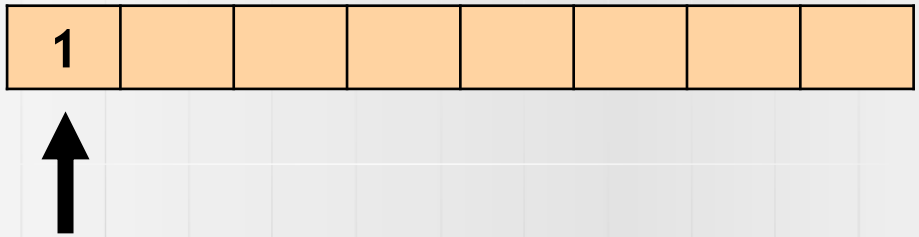




Politechnika Wrocławska

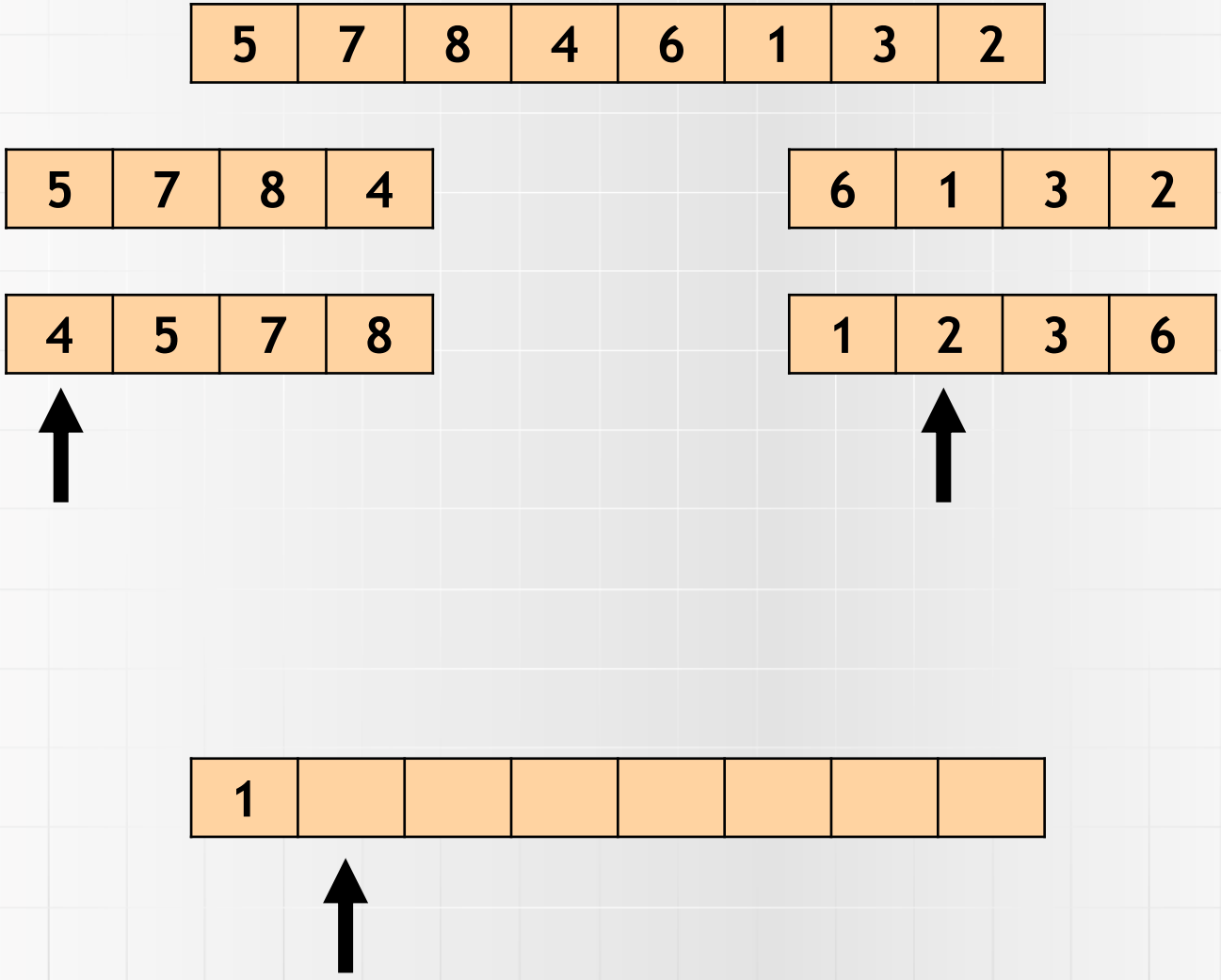
Sortowanie przez scalanie





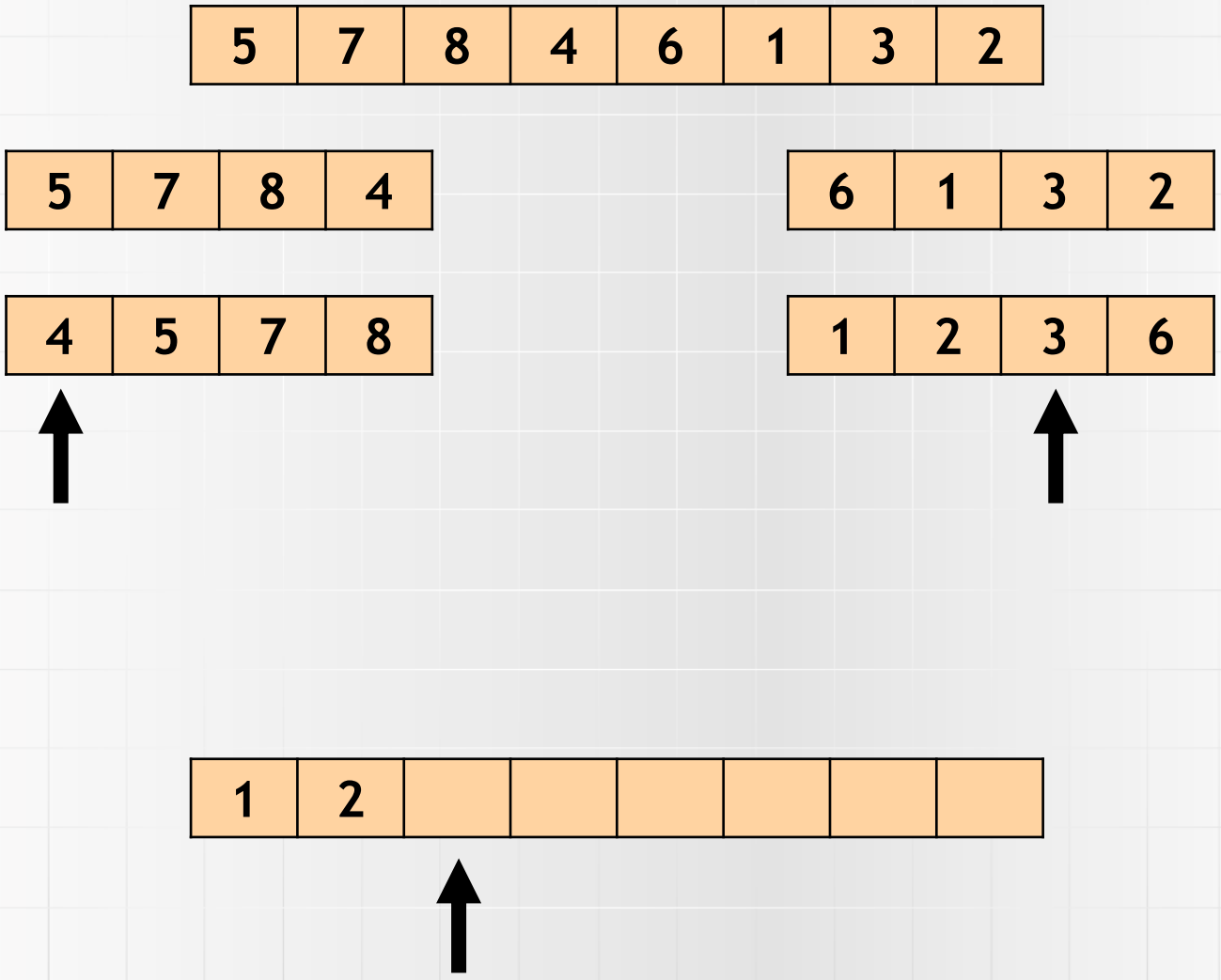
Politechnika Wrocławska

Sortowanie przez scalanie



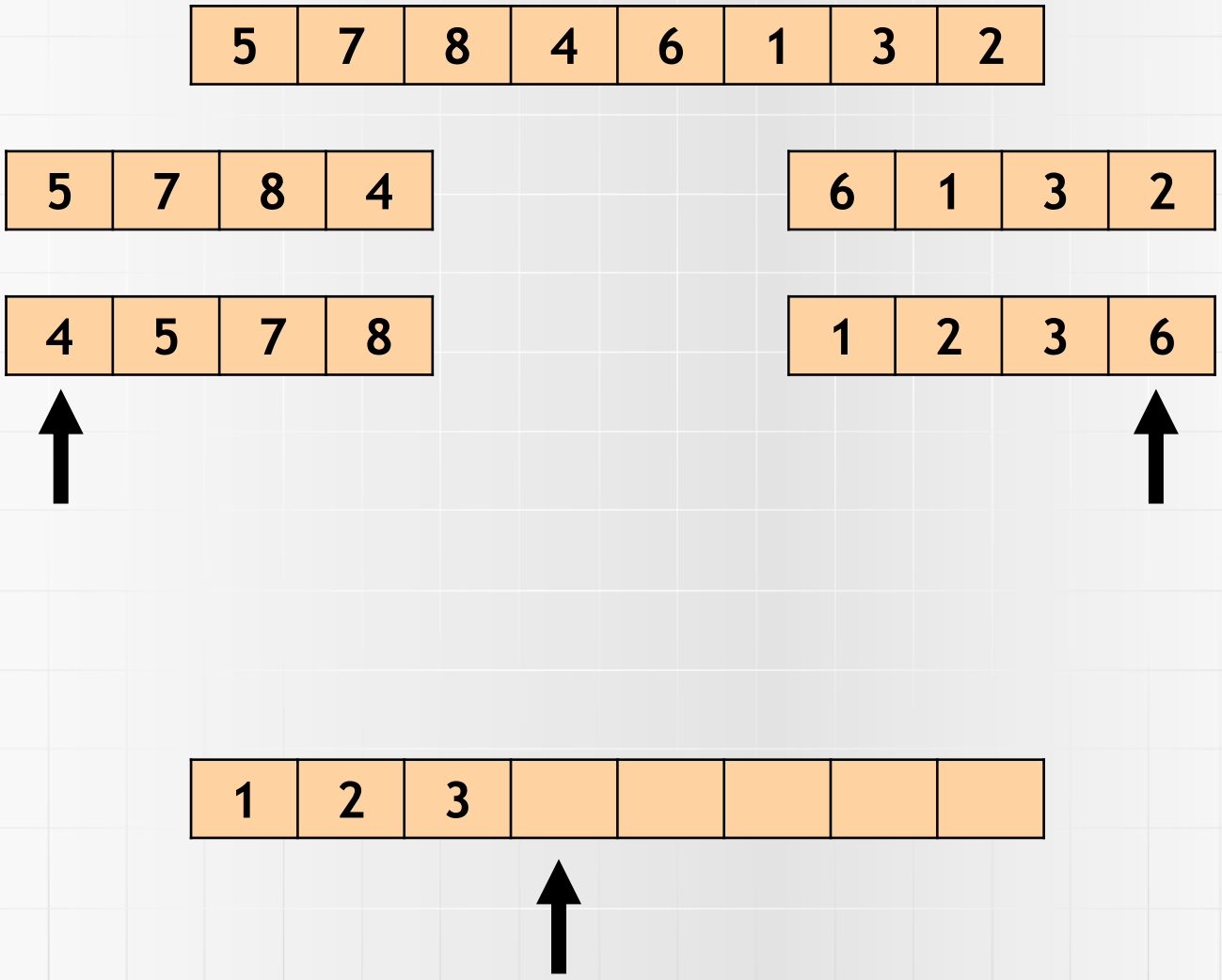
Politechnika Wrocławska

Sortowanie przez scalanie



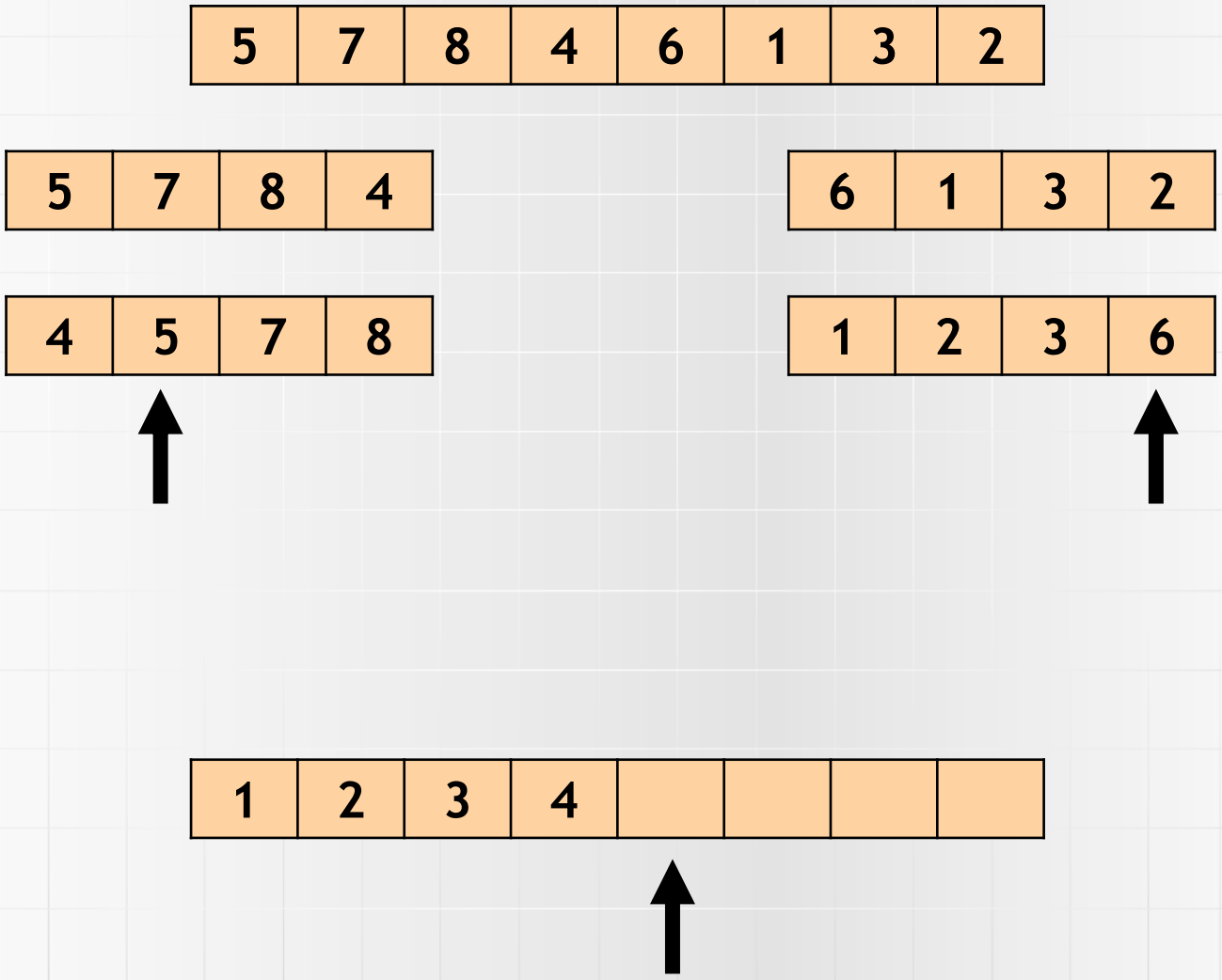
Politechnika Wrocławska

Sortowanie przez scalanie



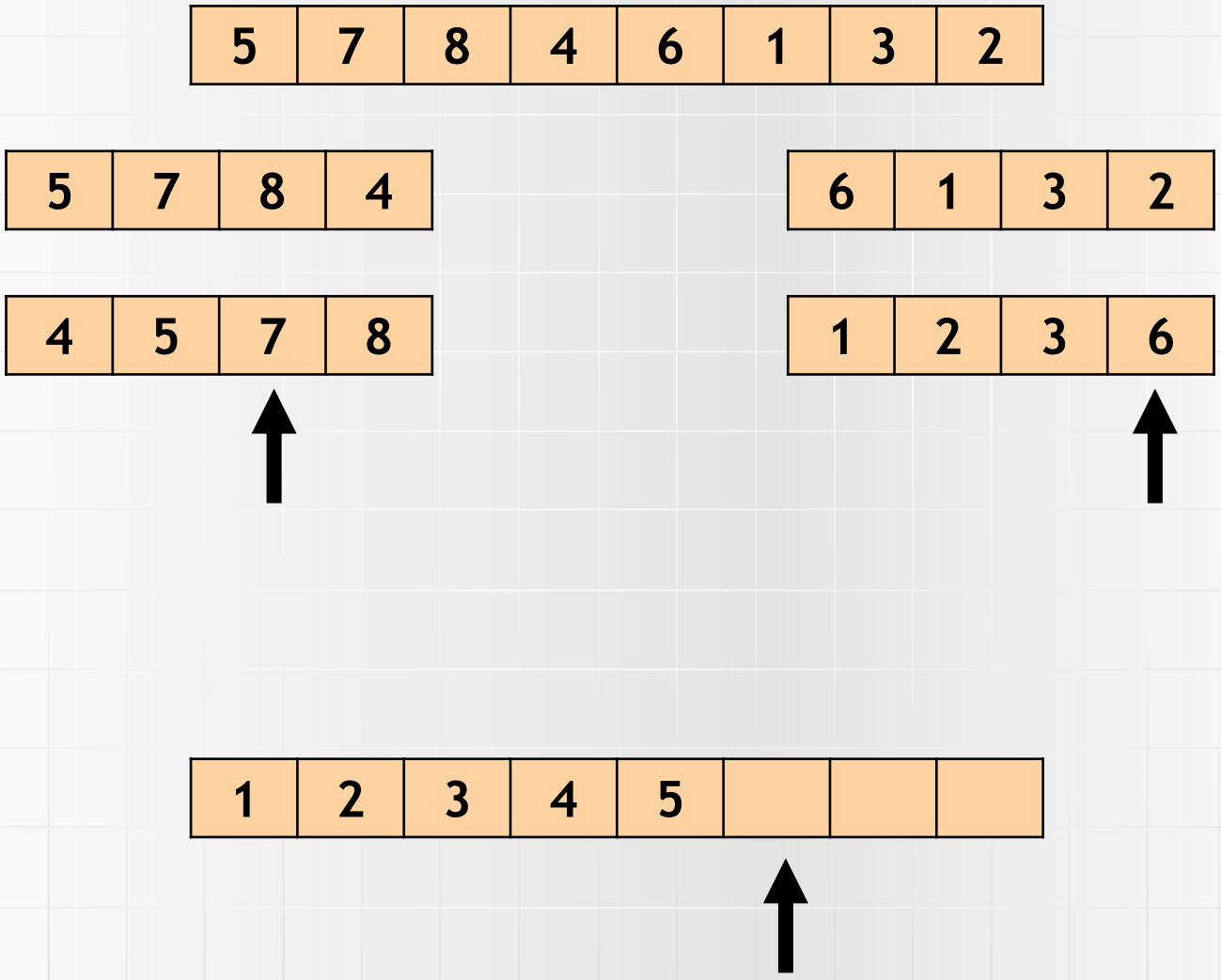
Politechnika Wrocławska

Sortowanie przez scalanie



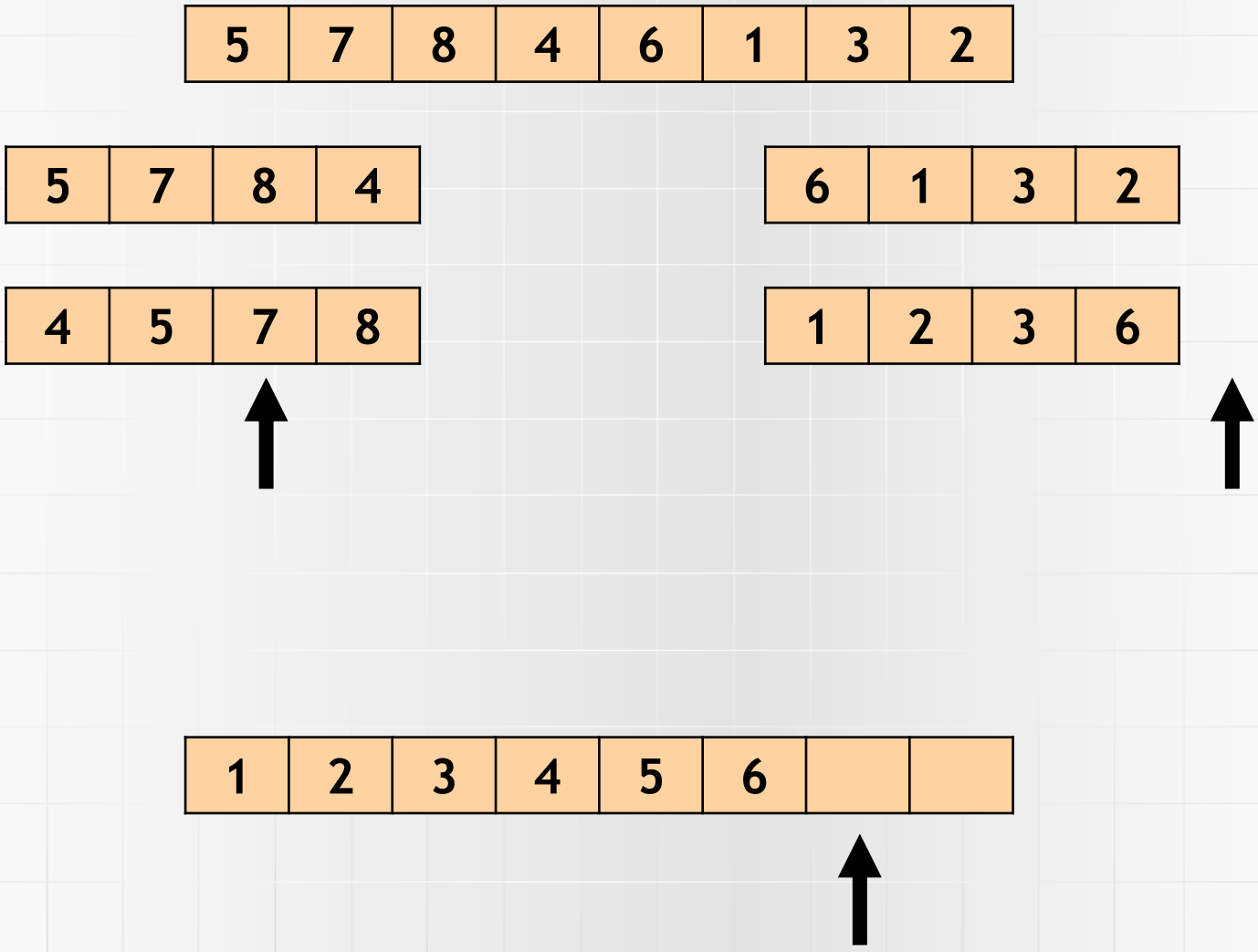
Politechnika Wrocławska

Sortowanie przez scalanie



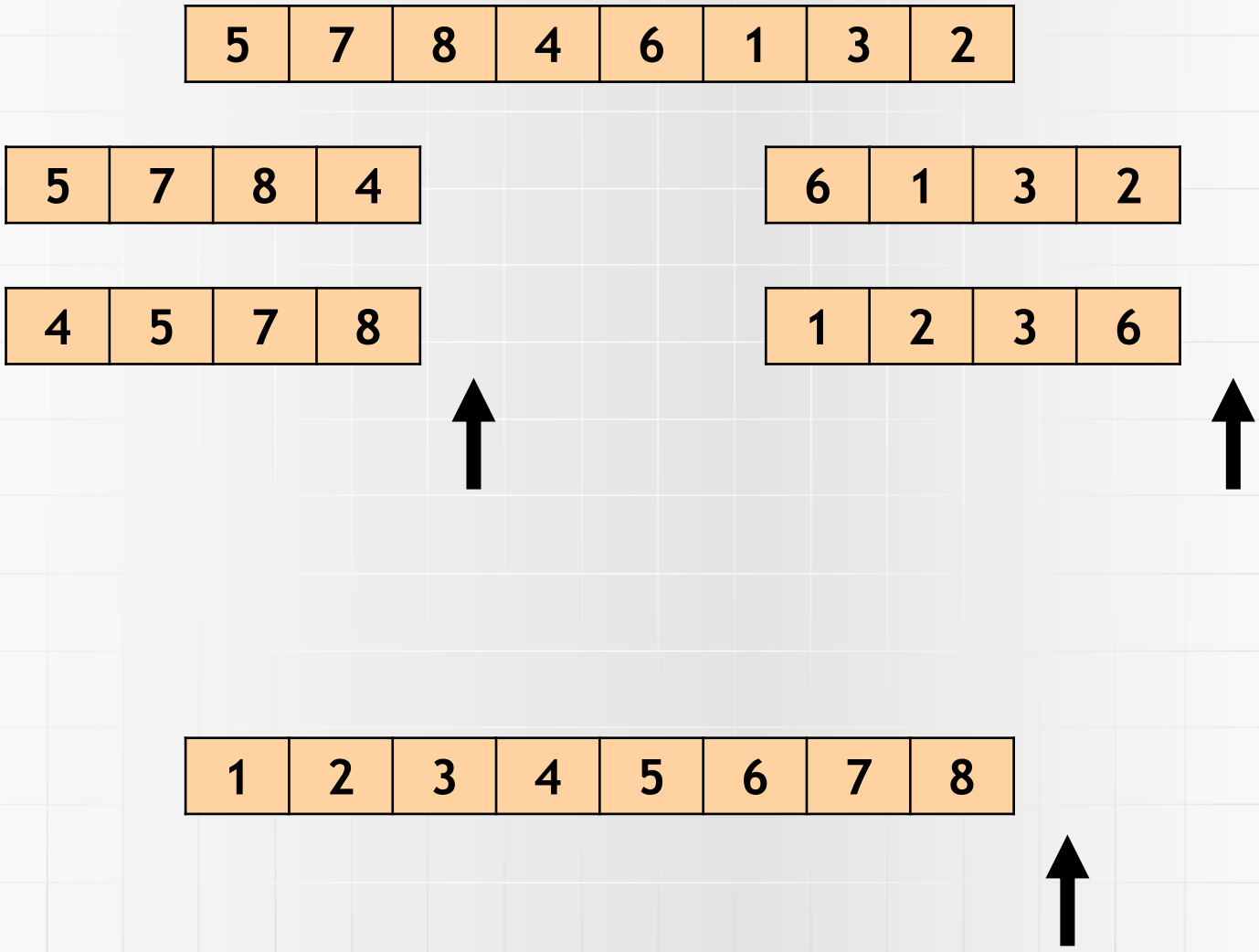
Politechnika Wrocławska

Sortowanie przez scalanie



Algorytmy sortowania

Sortowanie przez scalanie



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.

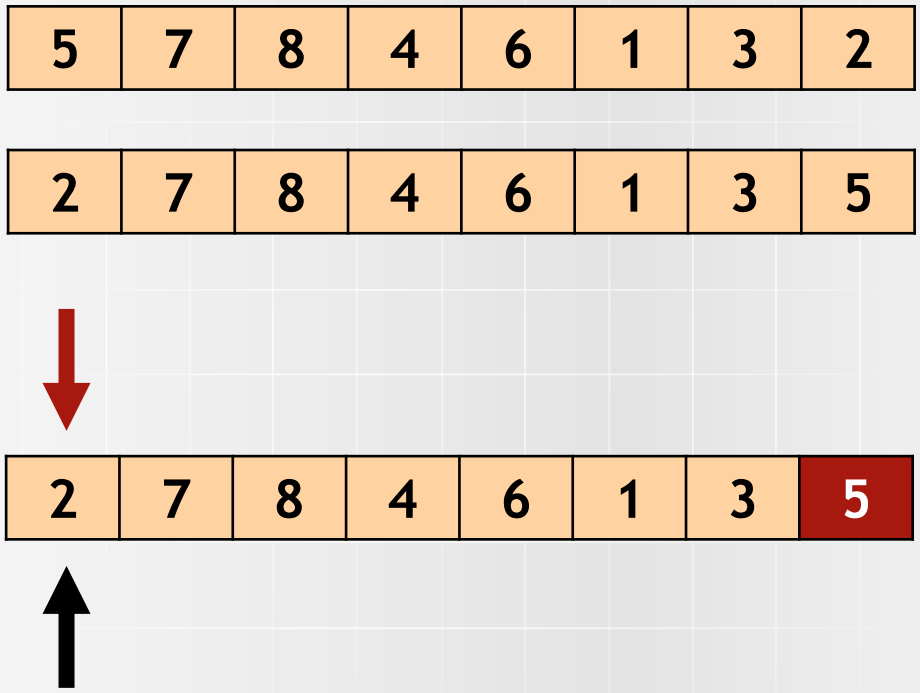
Politechnika Wrocławska

Sortowanie szybkie

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

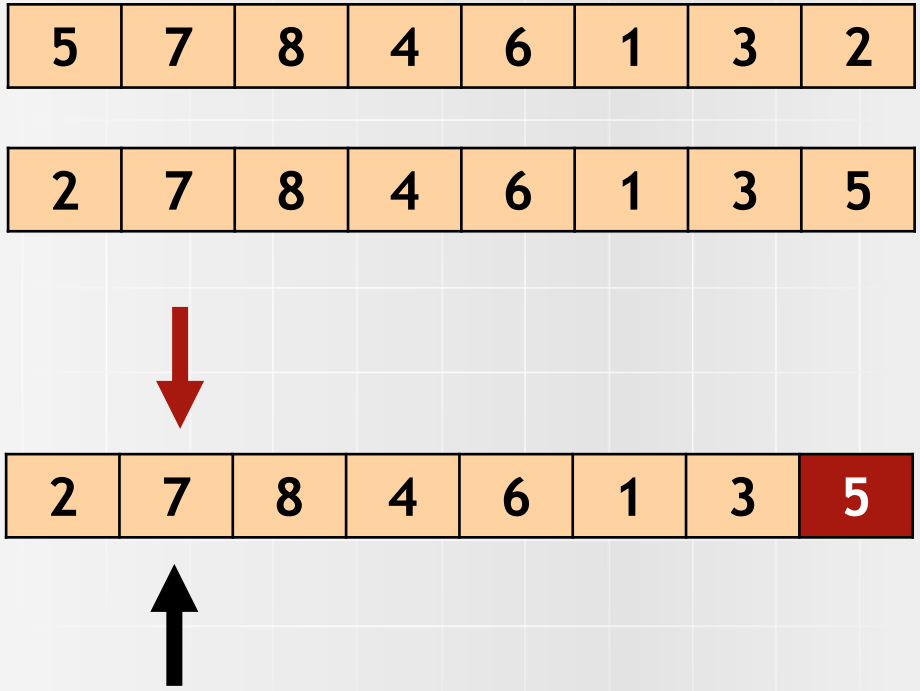
Algorytmy sortowania

Sortowanie szybkie



Algorytmy sortowania

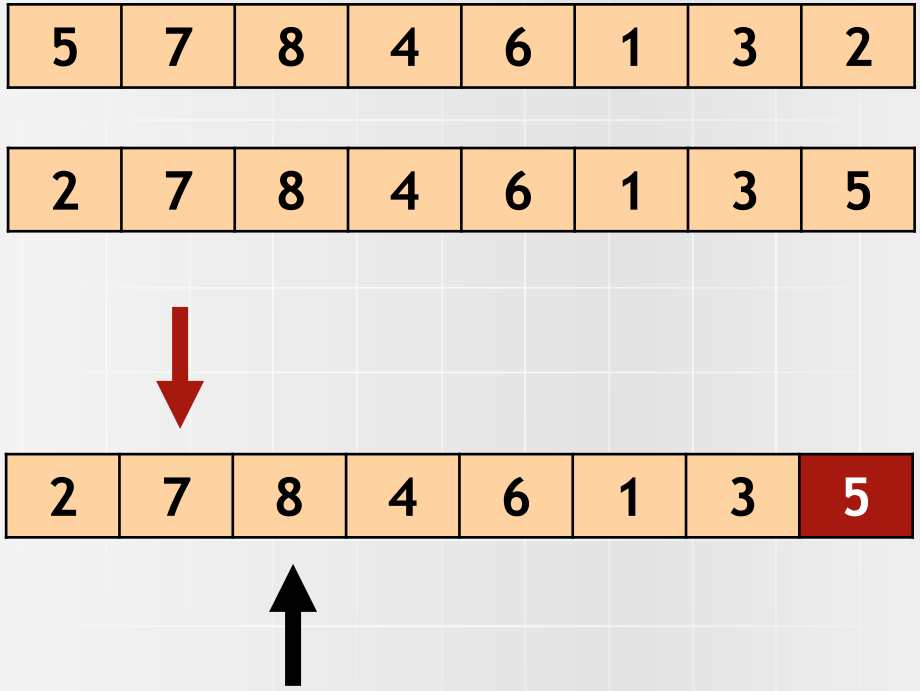
Sortowanie szybkie



Politechnika Wrocławska

Algorytmy sortowania

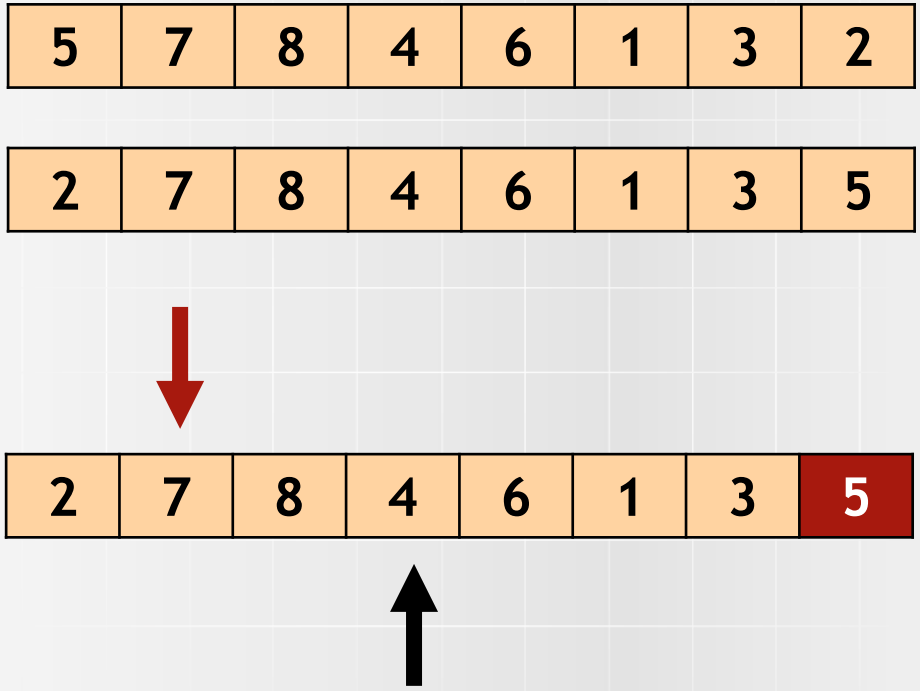
Sortowanie szybkie



Politechnika Wrocławska

Algorytmy sortowania

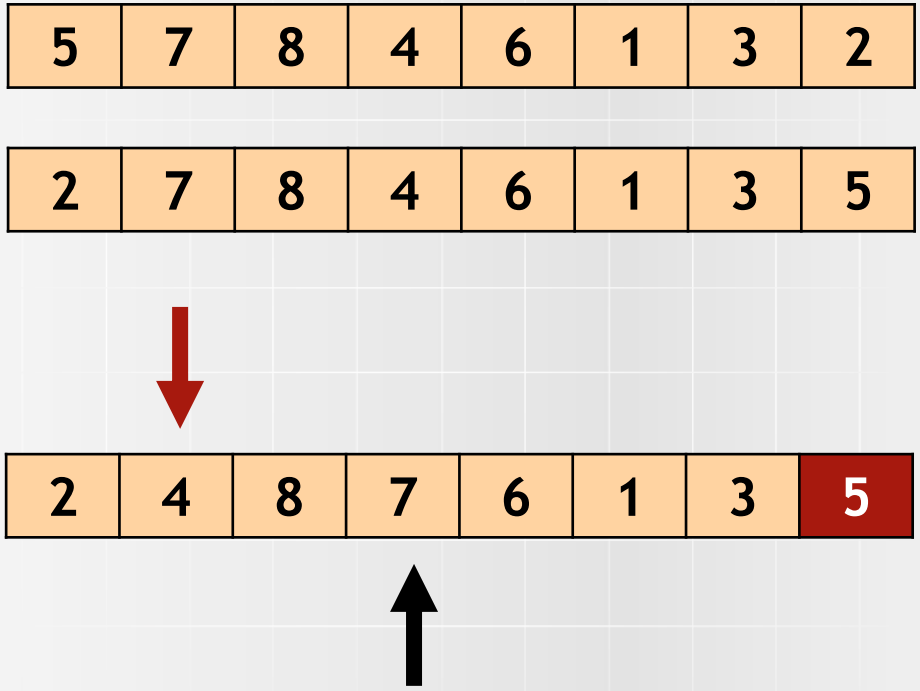
Sortowanie szybkie



Politechnika Wrocławska

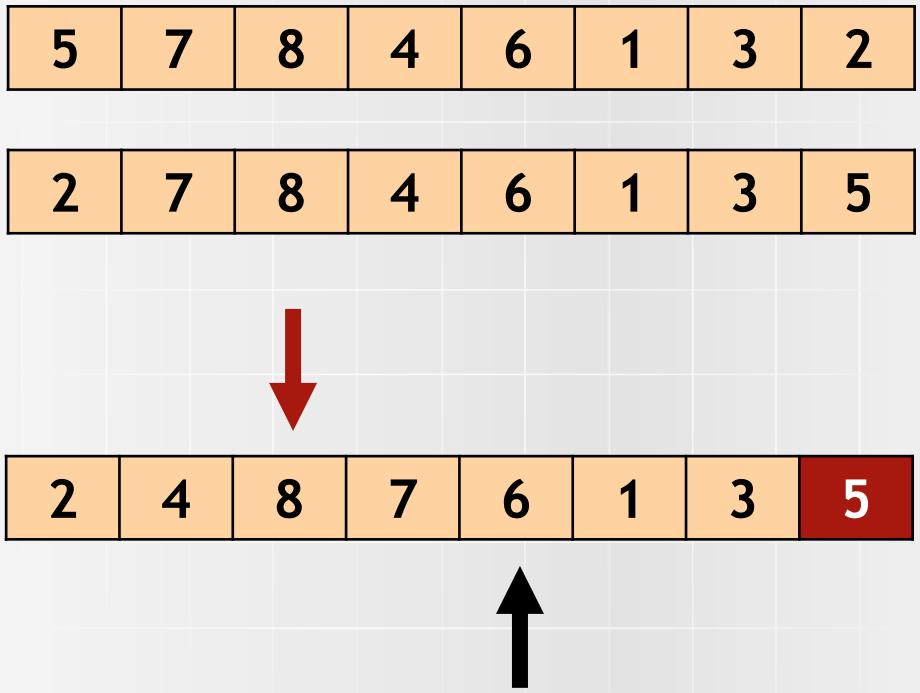
Algorytmy sortowania

Sortowanie szybkie



Algorytmy sortowania

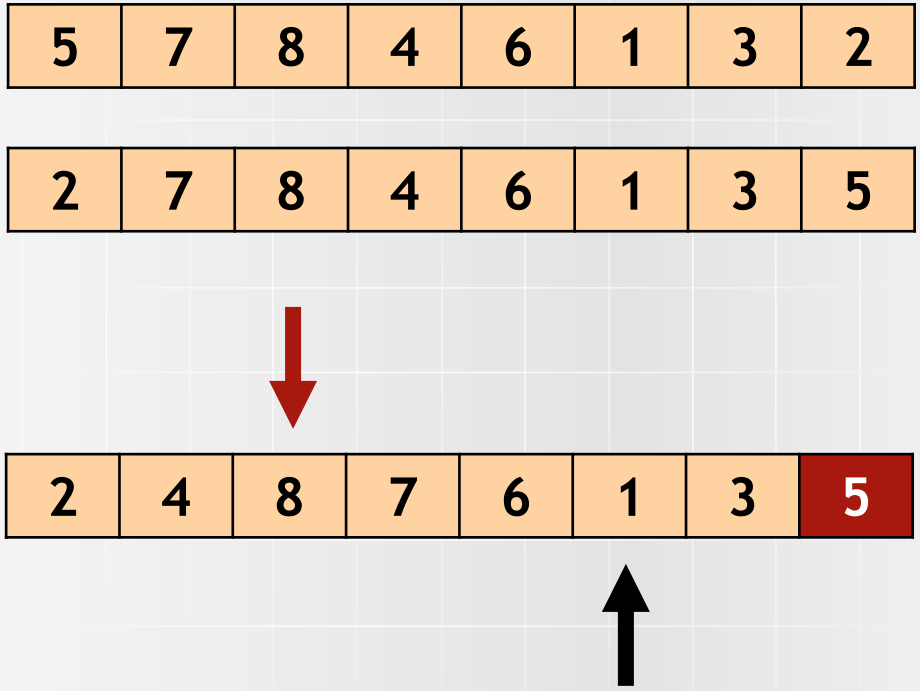
Sortowanie szybkie



Politechnika Wrocławska

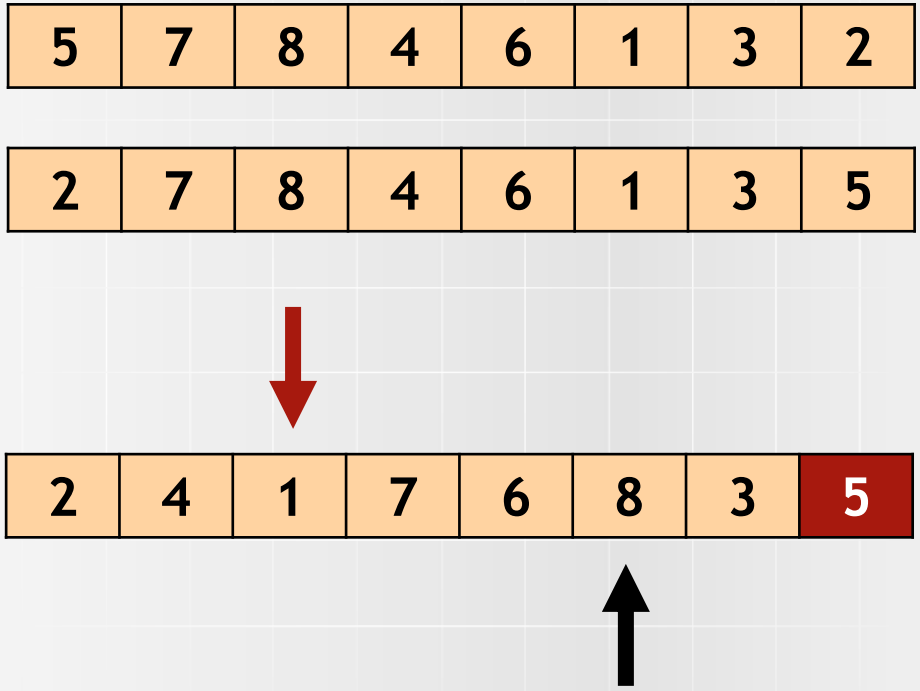
Algorytmy sortowania

Sortowanie szybkie



Algorytmy sortowania

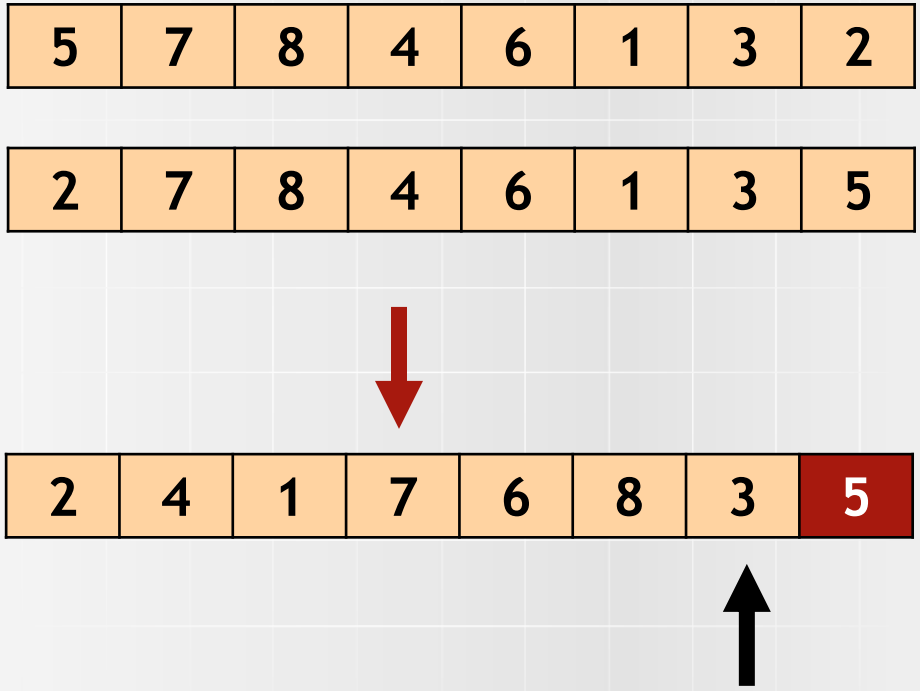
Sortowanie szybkie



Politechnika Wrocławska

Algorytmy sortowania

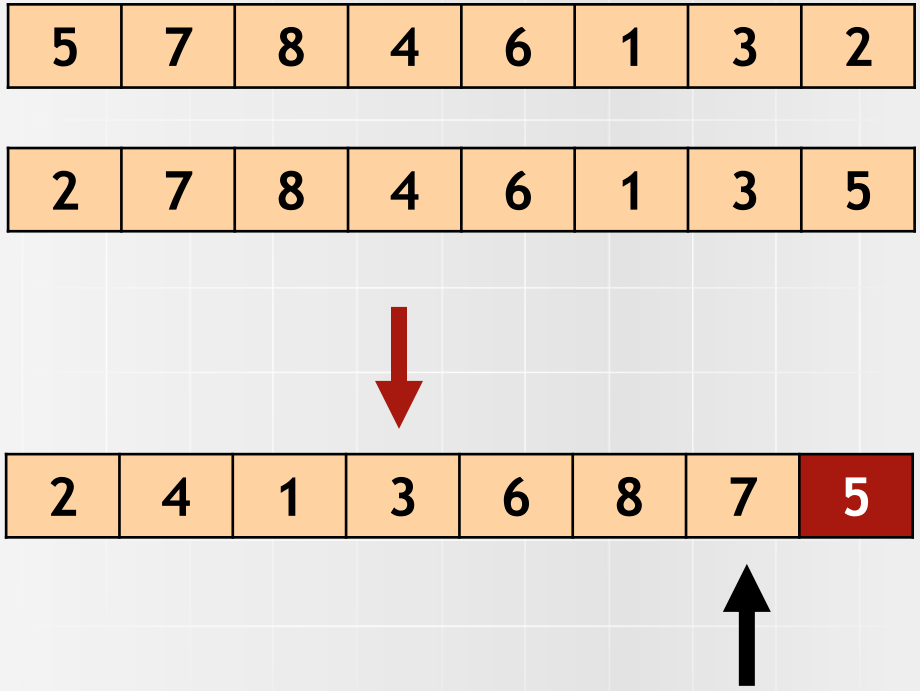
Sortowanie szybkie



Politechnika Wrocławska

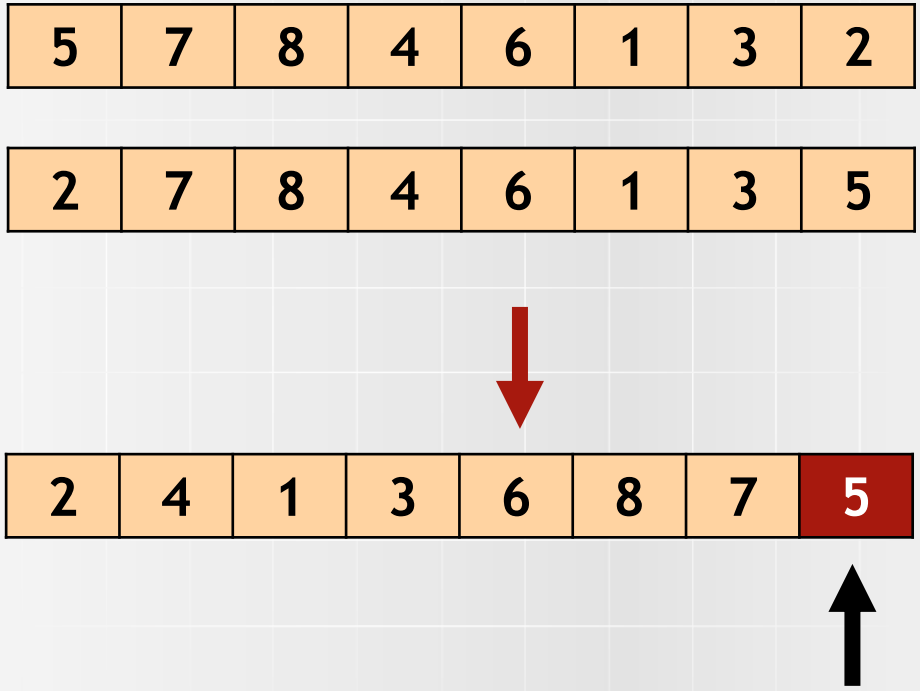
Algorytmy sortowania

Sortowanie szybkie



Algorytmy sortowania

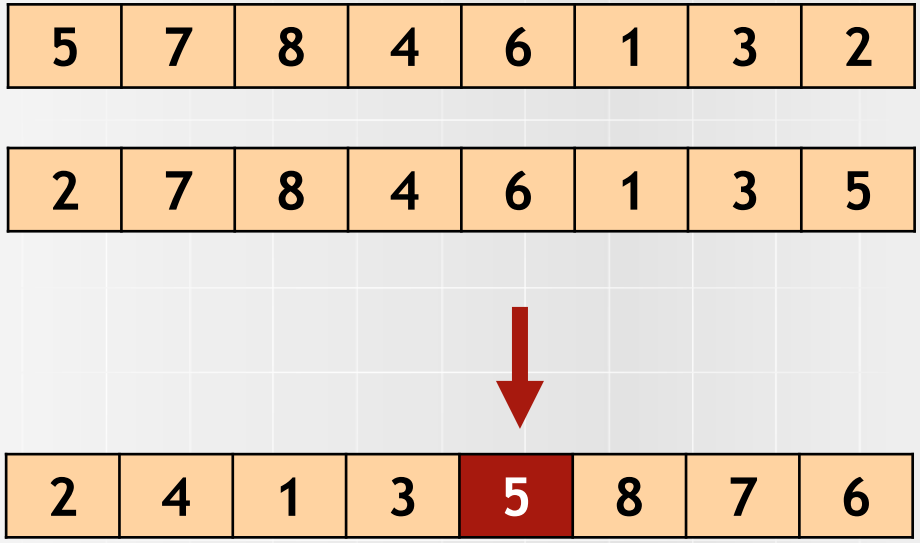
Sortowanie szybkie



Politechnika Wrocławska

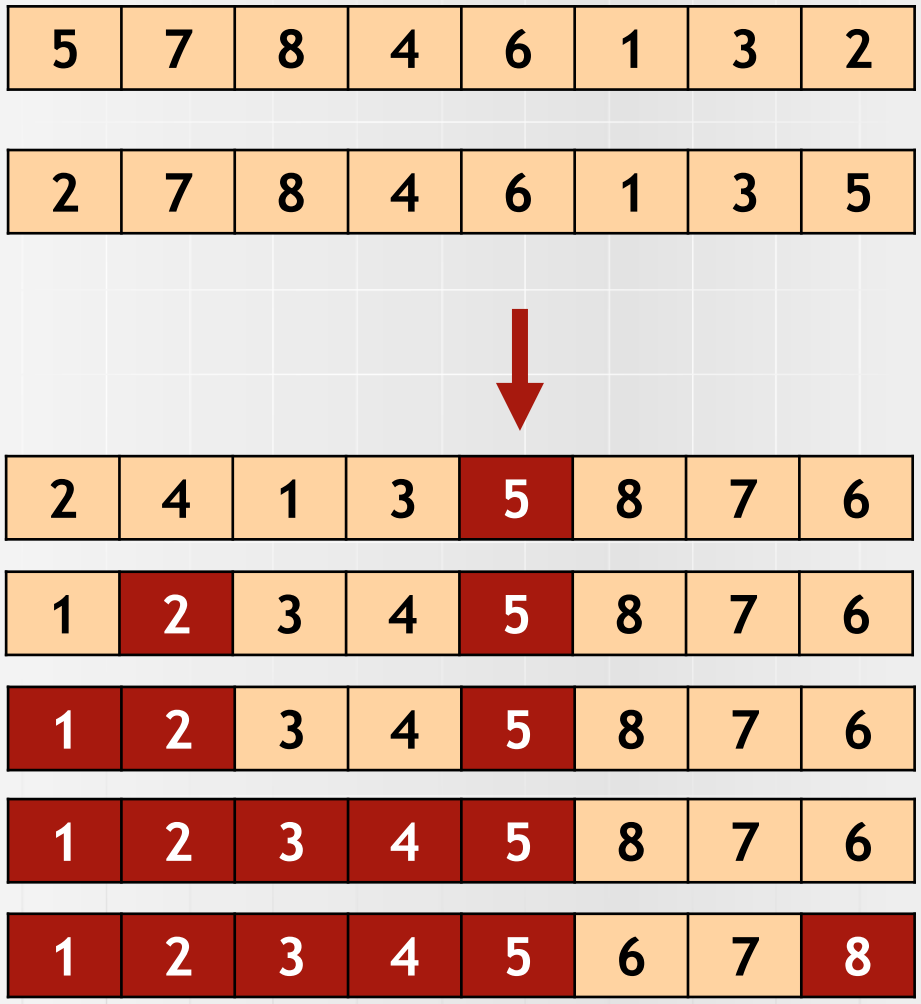
Algorytmy sortowania

Sortowanie szybkie



Algorytmy sortowania

Sortowanie szybkie



Politechnika Wrocławska

Absolutne minimum

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