Politechnika Wrocławska

Programowanie proceduralne

INP001210WL

rok akademicki 2021/22

semestr letni

Wykład 4

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220

Politechnika Wrocławska

Plan prezentacji (1)

- Rozpakowywanie listy/krotki
- Operator * – lista argumentów o zmiennej długości

Politechnika Wrocławska

Plan prezentacji (2)

- 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

Rozpakowywanie listy/krotki

- Rozpakowywanie pozwala przypisać elementy krotki/listy do kilku zmiennych

```
>>> t = (1,2)
>>> t
(1, 2)
>>> a, b = t
>>> a
1
>>> b
2
```

Politechnika Wrocławska

Rozpakowywanie listy/krotki

- Liczba zmiennych po lewej stronie operatora przypisania powinna odpowiadać liczbie elementów krotki/listy

```
>>> t = tuple(range(5))
>>> t
(0, 1, 2, 3, 4)
>>> a, b = t
Traceback (most recent call last):
  File "<pyshell#45>", line 1, in <module>
    a, b = t
ValueError: too many values to unpack (expected 2)
```

Politechnika Wrocławska

Rozpakowywanie listy/krotki

- Jeśli nie znamy liczby argumentów możemy wykorzystać operator *

```
>>> t = tuple(range(5))
>>> t
(0, 1, 2, 3, 4)
>>> a, *b = t
>>> a
0
>>> b
[1, 2, 3, 4]
>>> type(a)
<class 'int'>
>>> type(b)
<class 'list'>
```

Politechnika Wrocławska

Rozpakowywanie listy/krotki

- Jeśli nie znamy liczby argumentów możemy wykorzystać operator *

```
>>> t = tuple(range(5))
>>> t
(0, 1, 2, 3, 4)
>>> first, *middle, last = t
>>> first
0
>>> middle
[1, 2, 3]
>>> last
4
```

Politechnika Wrocławska

Operator *

- Operator * (pakowania/rozpakowania) można wykorzystać do przekształcenia listy w kilka argumentów pozycyjnych

```
>>> list(range(2,5))
[2, 3, 4]
>>> list(range([2,5]))
Traceback (most recent call last):
  File "<pyshell#64>", line 1, in <module>
    list(range([2,5]))
TypeError: 'list' object cannot be interpreted as an integer
>>> list(range(*[2,5]))
[2, 3, 4]
```

Politechnika Wrocławska

Operator *

- Operator * można wykorzystać do zapakowania kilku argumentów pozycyjnych w krotkę

Politechnika Wrocławska

Operator *

variable_argument_list_01.py

File Edit Format Run Options Window Help

```
1 def main():
2     print('2 * 3 =',my_product1(2,3))
3
4 # funkcja oblicza iloczyn dwóch liczb
5 def my_product1(a,b):
6     return a*b
7
8 main()
9
```

Politechnika Wrocławska

Operator *

variable_argument_list_02.py

File Edit Format Run Options Window Help

```
1  def main():
2      print('2 * 3 =',my_product1([2,3]))
3      # podano tylko jeden argument!
4
5  def my_product1(a,b):
6      return a*b
7
8  main()
9
10 #Traceback (most recent call last):
11 #  File "...\variable_argument_list_02.py", line 7, in <module>
12 #    main()
13 #  File "...\variable_argument_list_02.py", line 2, in main
14 #    print('2 * 3 =',my_product1([2,3]))
15 #TypeError: my_product1() missing 1 required positional argument: 'b'
16
```

Politechnika Wrocławska

Operator *

variable_argument_list_03.py

File Edit Format Run Options Window Help

```
1  def main():
2      print('2 * 3 =',my_product1(2,3))
3      print('2 * 3 =',my_product2([2,3]))
4      print('2 * 3 * 4 =',my_product2([2,3,4]))
5
6  def my_product1(a,b):
7      return a*b
8
9  # funkcja oblicza iloczyn liczb z listy
10 def my_product2(l):
11     result = 1
12     for x in l:
13         result *= x
14     return result
15
16 main()
```

Politechnika Wrocławska

Operator *

variable_argument_list_04.py

File Edit Format Run Options Window Help

```
1  def main():
2      print('2 * 3 =',my_product1(2,3))
3      print('2 * 3 =',my_product2([2,3]))
4      print('2 * 3 * 4 =',my_product2([2,3,4]))
5      print('2 * 3 * 4 =',my_product2(2,3,4))
6      # podano więcej niż jeden argument
7
8  def my_product1(a,b):
9      return a*b
10
11 def my_product2(l):
12     result = 1
13     for x in l:
14         result *= x
15     return result
16
17 main()
18
19 #2 * 3 = 6
20 #2 * 3 = 6
21 #2 * 3 * 4 = 24
22 #Traceback (most recent call last):
23 #  File "...\variable_argument_list_04.py", line 16, in <module>
24 #    main()
25 #  File "...\variable_argument_list_04.py", line 5, in main
26 #    print('2 * 3 * 4 =',my_product2(2,3,4))
27 #TypeError: my_product2() takes 1 positional argument but 3 were given
```

Politechnika Wrocławska

Operator *

variable_argument_list_05.py

File Edit Format Run Options Window Help

```
1  def main():
2      print('2 * 3 =',my_product1(2,3))
3      print('2 * 3 =',my_product2([2,3]))
4      print('2 * 3 * 4 =',my_product2([2,3,4]))
5      print('2 * 3 * 4 =',my_product3(2,3,4))
6
7  def my_product1(a,b):
8      return a*b
9
10 def my_product2(l):
11     result = 1
12     for x in l:
13         result *= x
14     return result
15
16 # funkcja przyjmuje wiele argumentów
17 # i pakuje je w krotkę
18 def my_product3(*l):
19     #print(type(l))
20     result = 1
21     for x in l:
22         result *= x
23     return result
24
25 main()
```

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

```
1  #Moduł 04_sort zawiera funkcję swap(), która pozwala zamienić
2  #miejscami dwa elementy na liście.
3
4  #Moduł zawiera również funkcję main(), która demonstruje działanie
5  #funkcji swap().
6
7  def swap(l,left,right):
8      # item     = l[left]
9      # l[left]  = l[right]
10     # l[right] = item
11     l[left], l[right] = l[right], l[left]
12
13 def main():
14     l = list(range(4))
15     print(l)
16     swap(l,1,2)
17     print(l)
18
19 #Instrukcja warunkowa, która sprawdza, czy plik był
20 #uruchomiony jako skrypt, czy załadowany jako moduł.
21 if __name__ == "__main__":
22     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

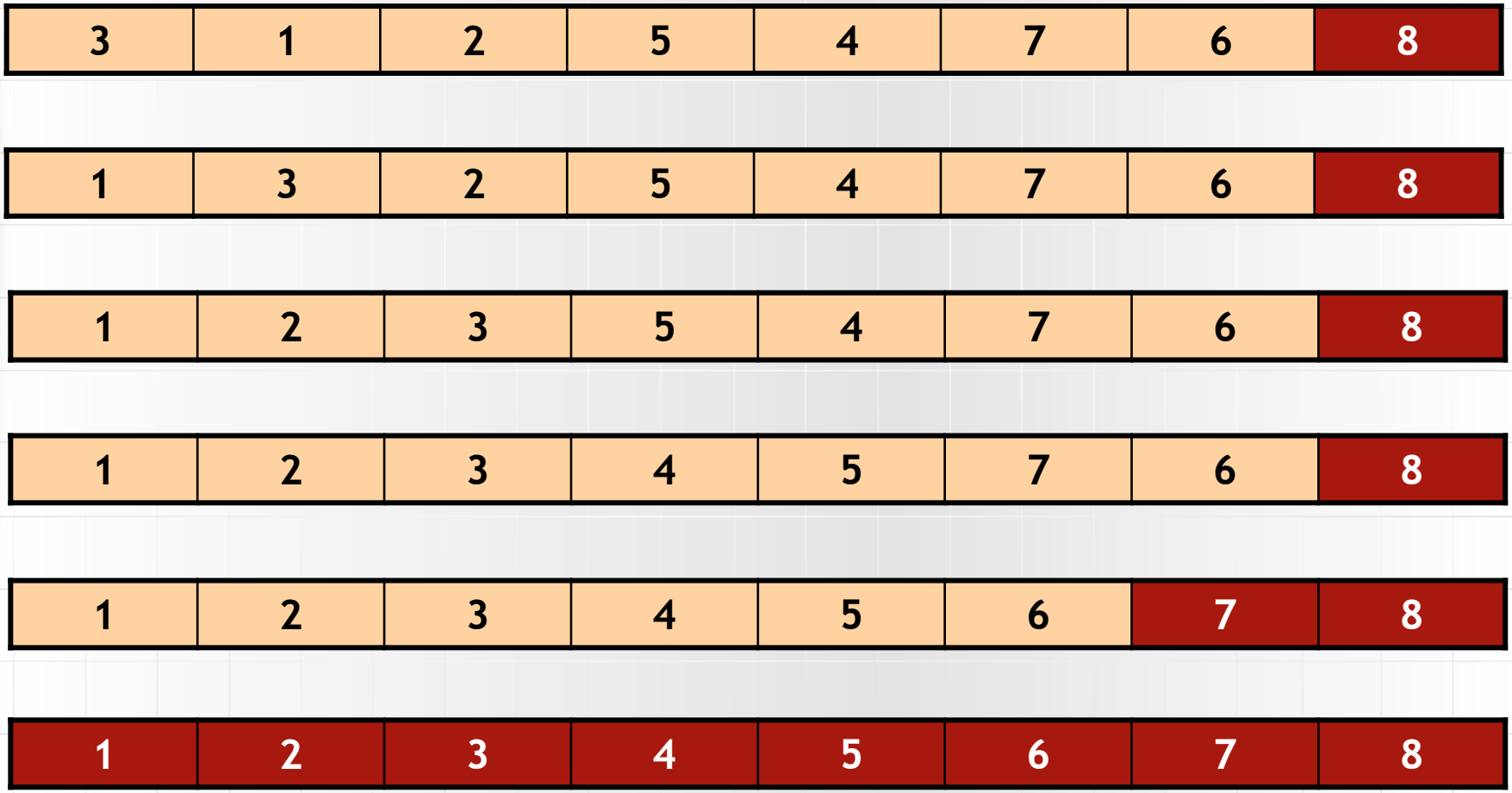
3	1	2	5	4	8	7	6

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)
```

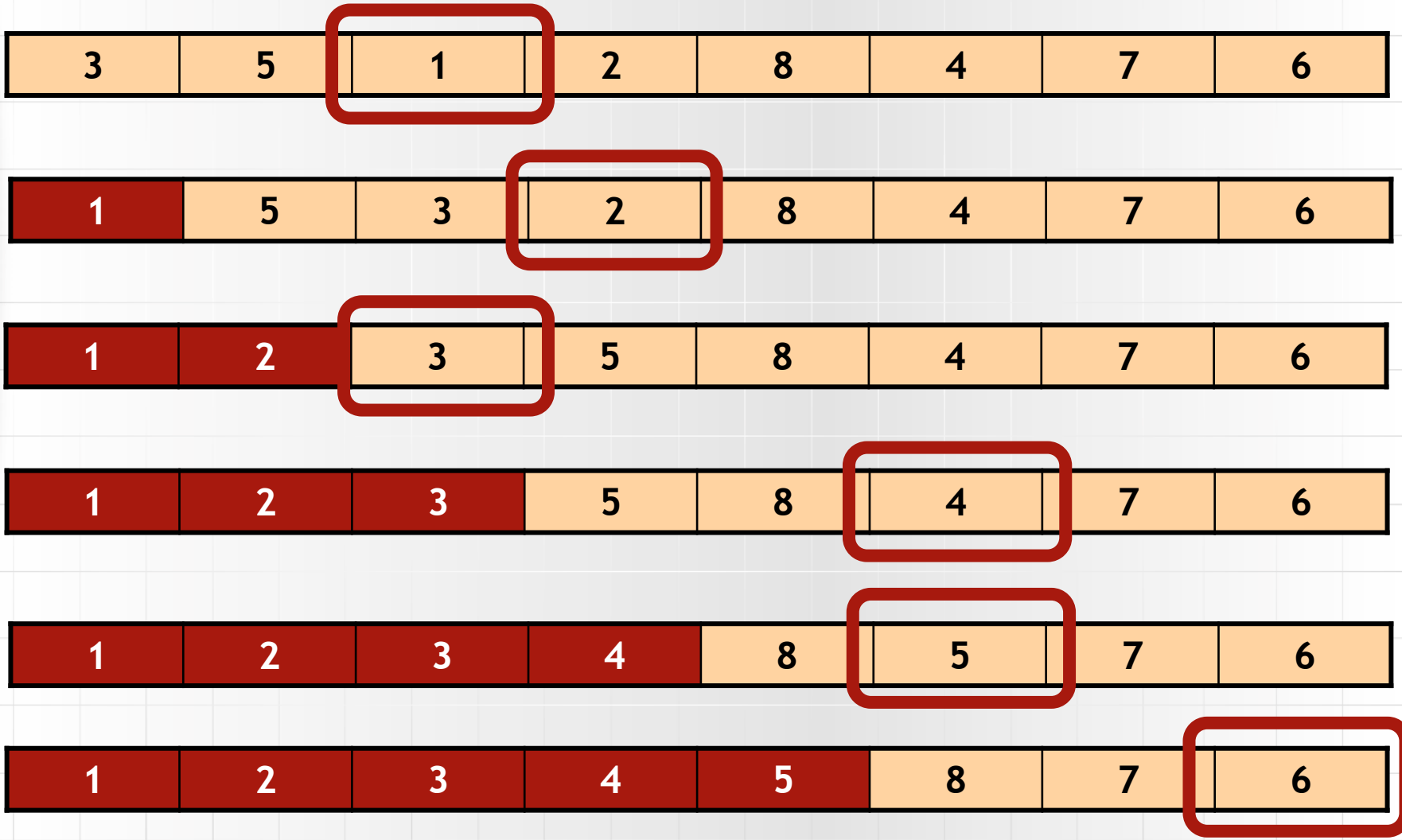
Politechnika Wrocławska

Sortowanie przez wybór

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

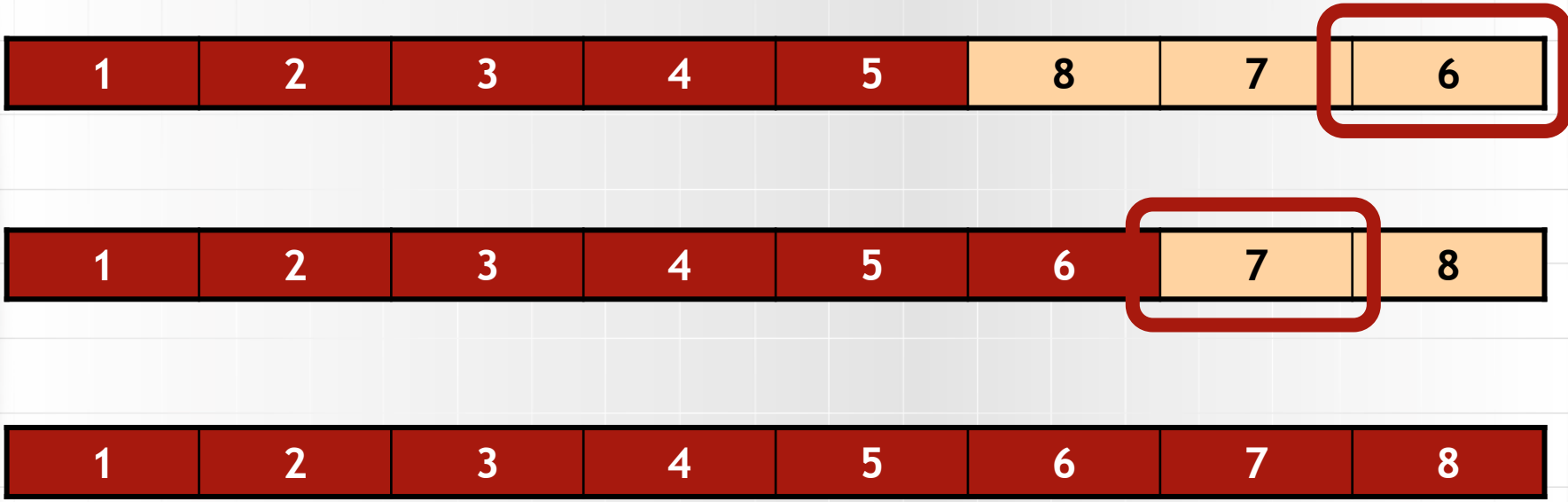
Politechnika Wrocławska

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

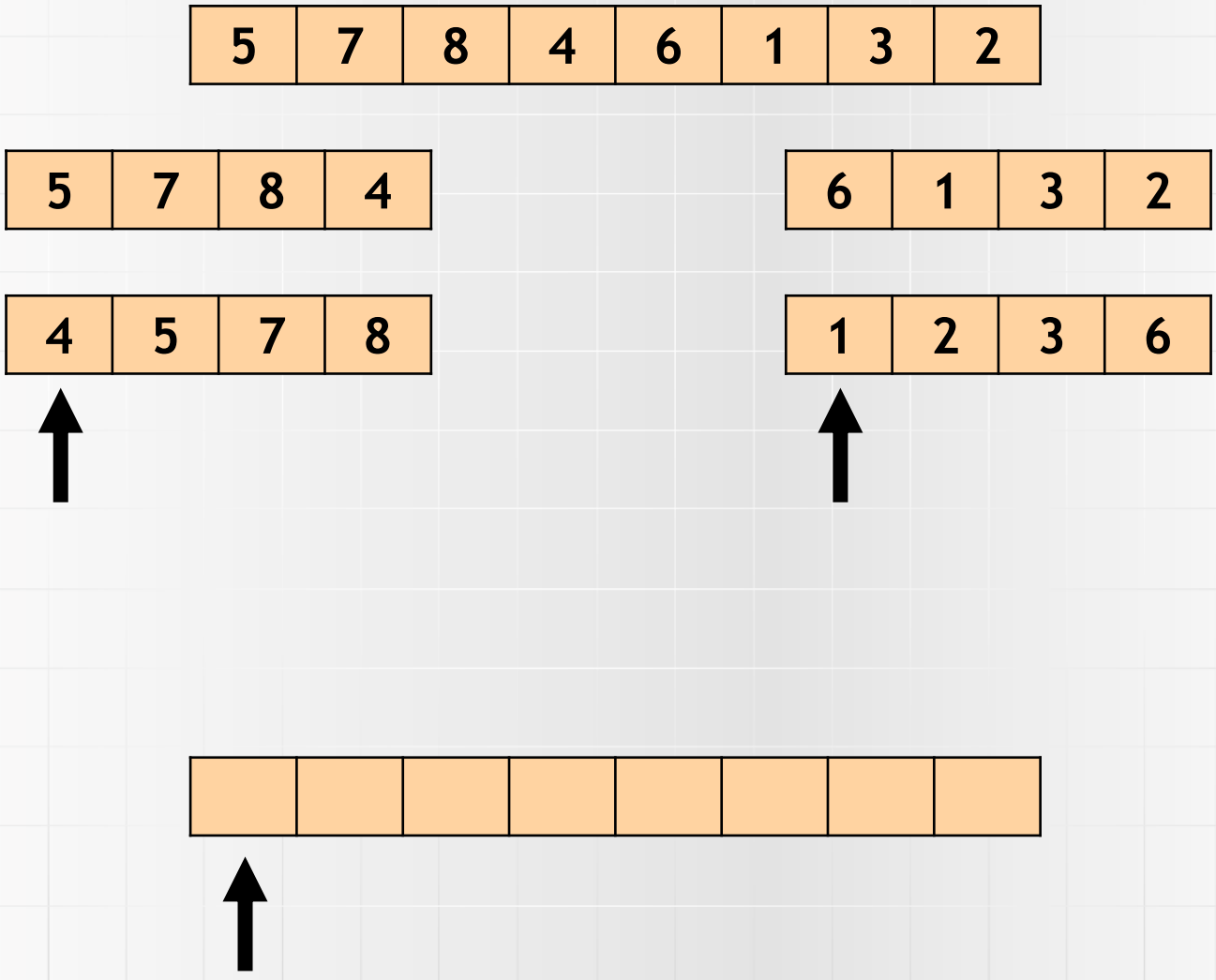
Scalanie

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.

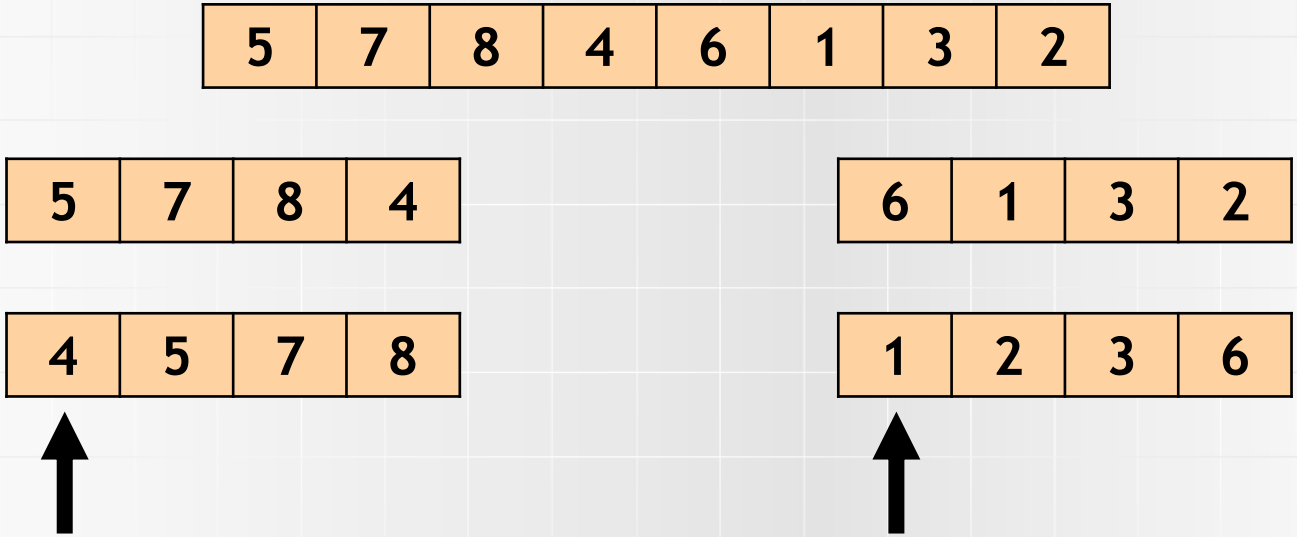
Politechnika Wrocławska

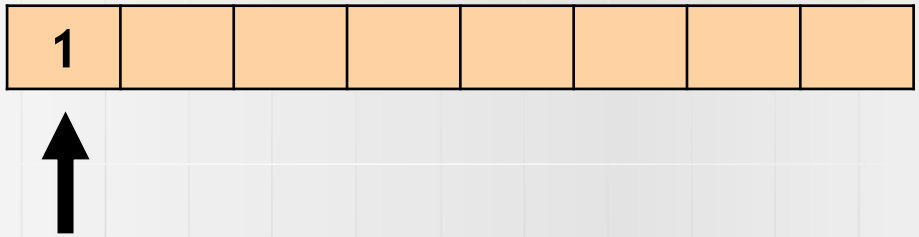
Sortowanie przez scalanie



Politechnika Wrocławska

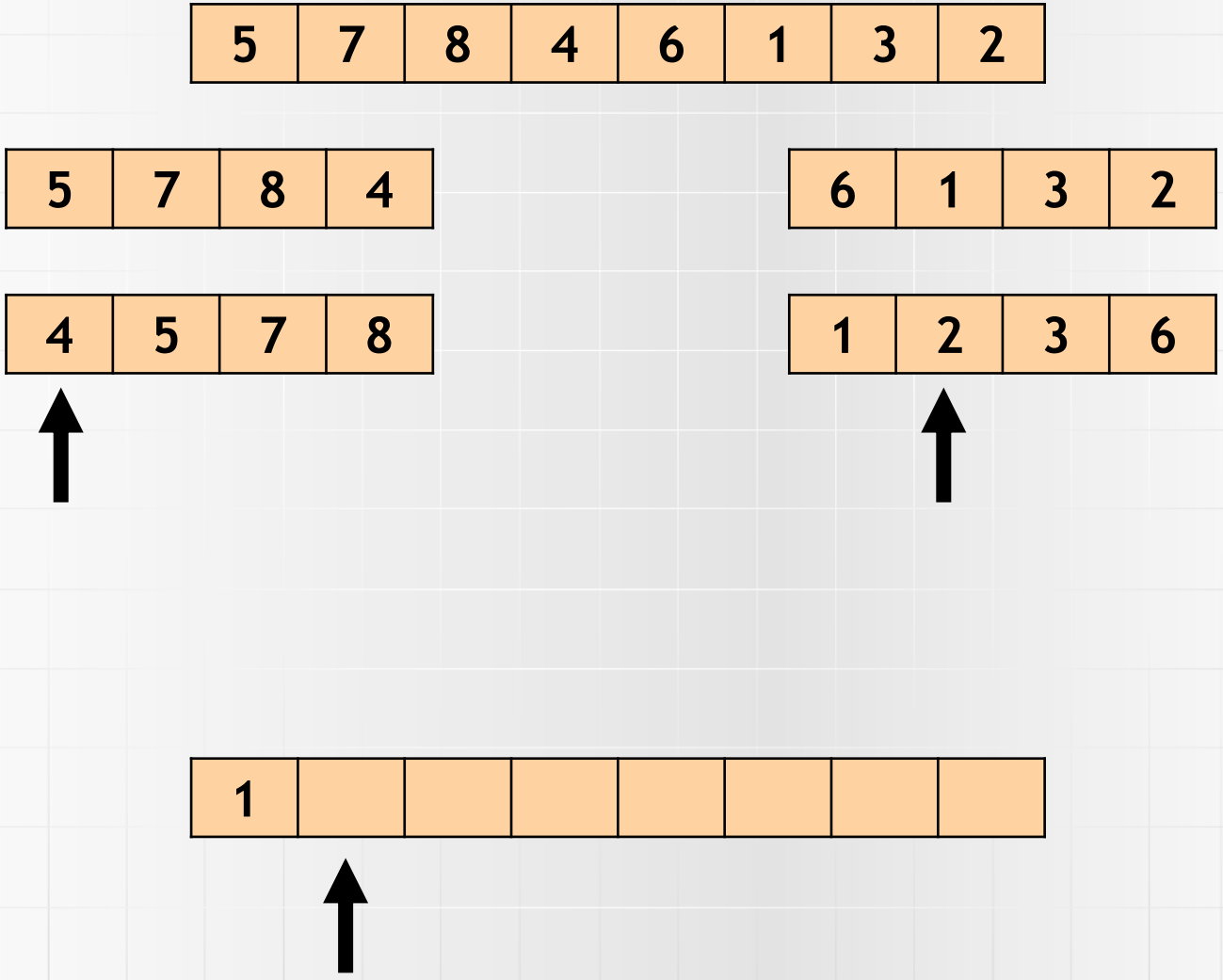
Sortowanie przez scalanie





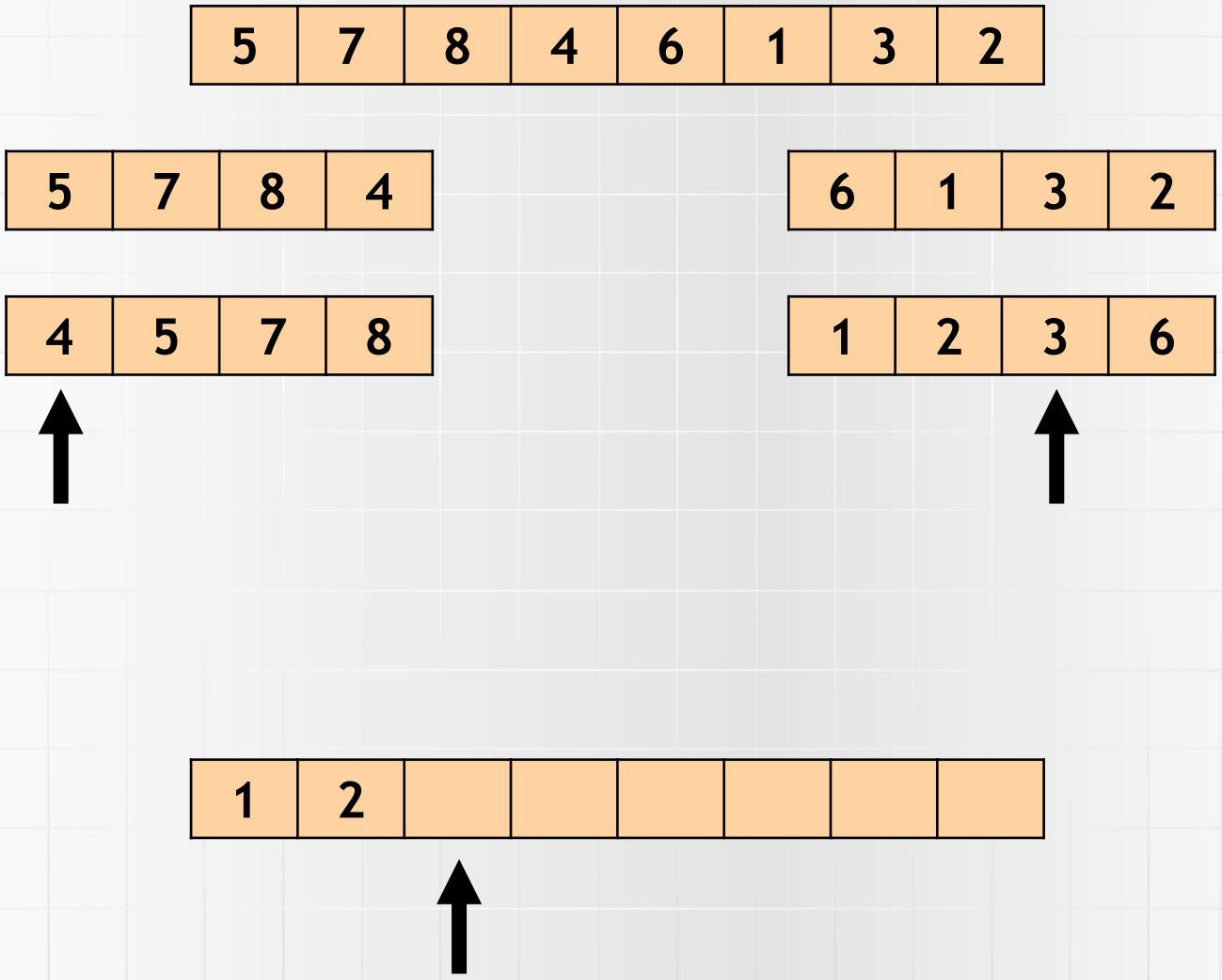
Politechnika Wrocławska

Sortowanie przez scalanie



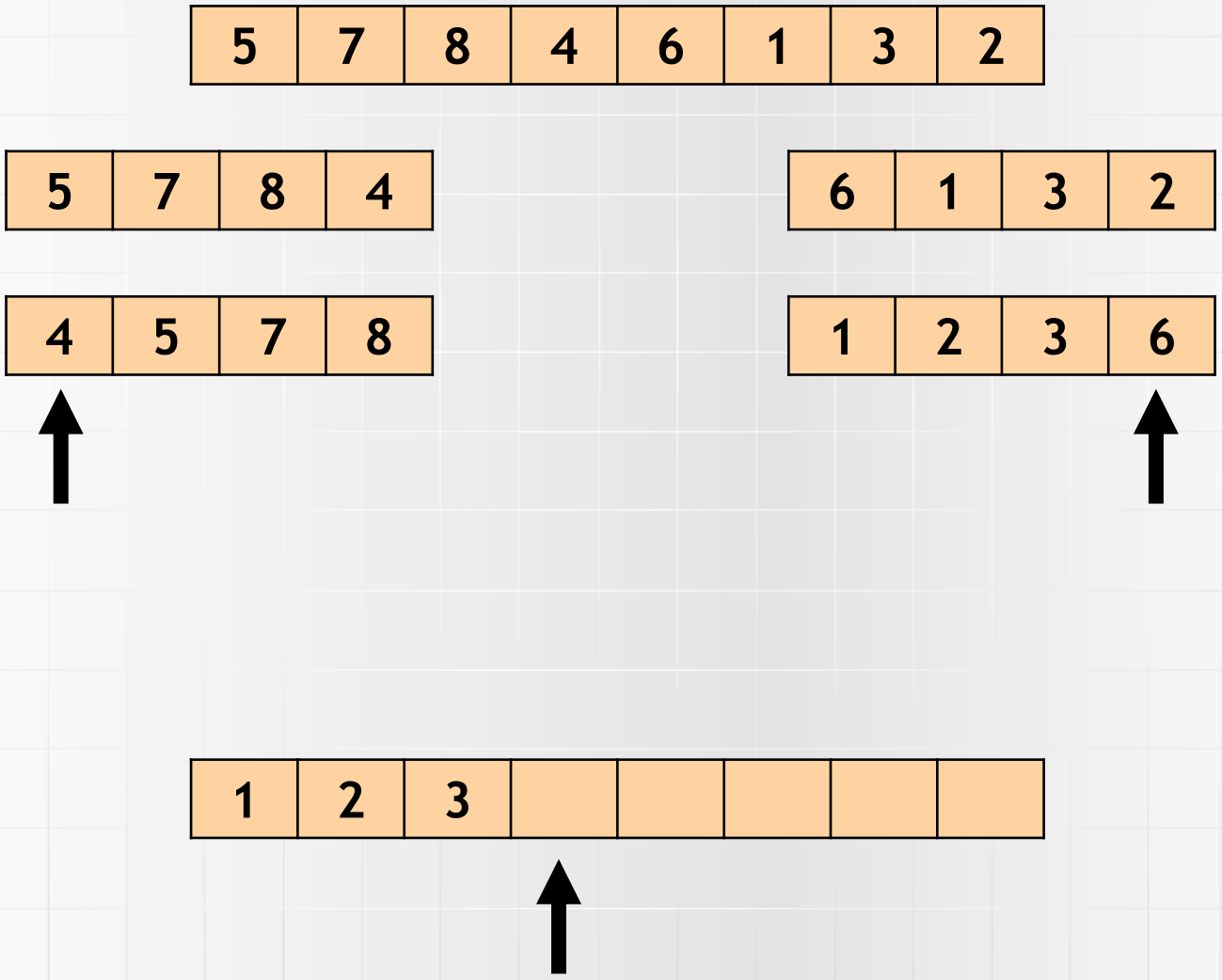
Politechnika Wrocławska

Sortowanie przez scalanie



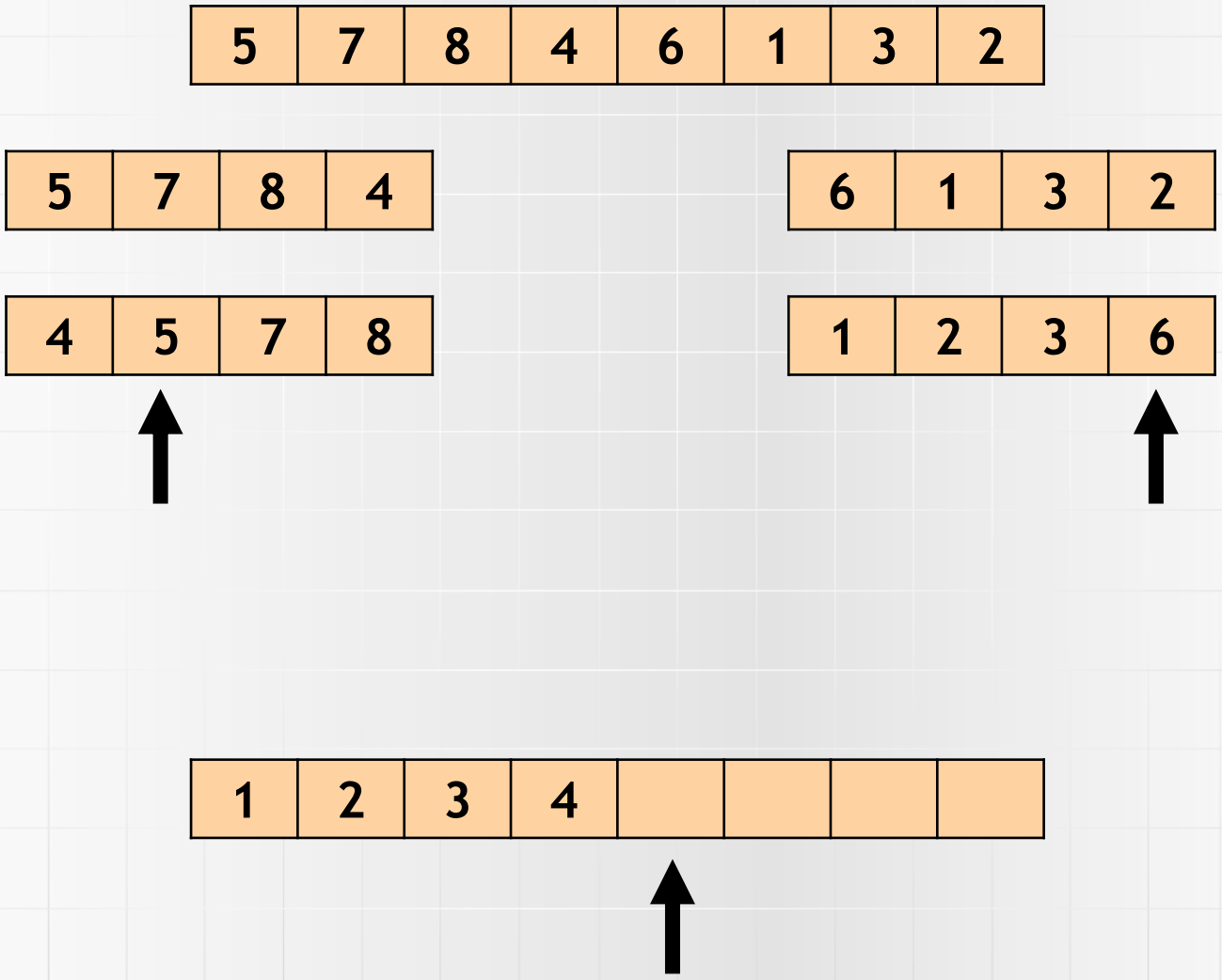
Politechnika Wrocławska

Sortowanie przez scalanie



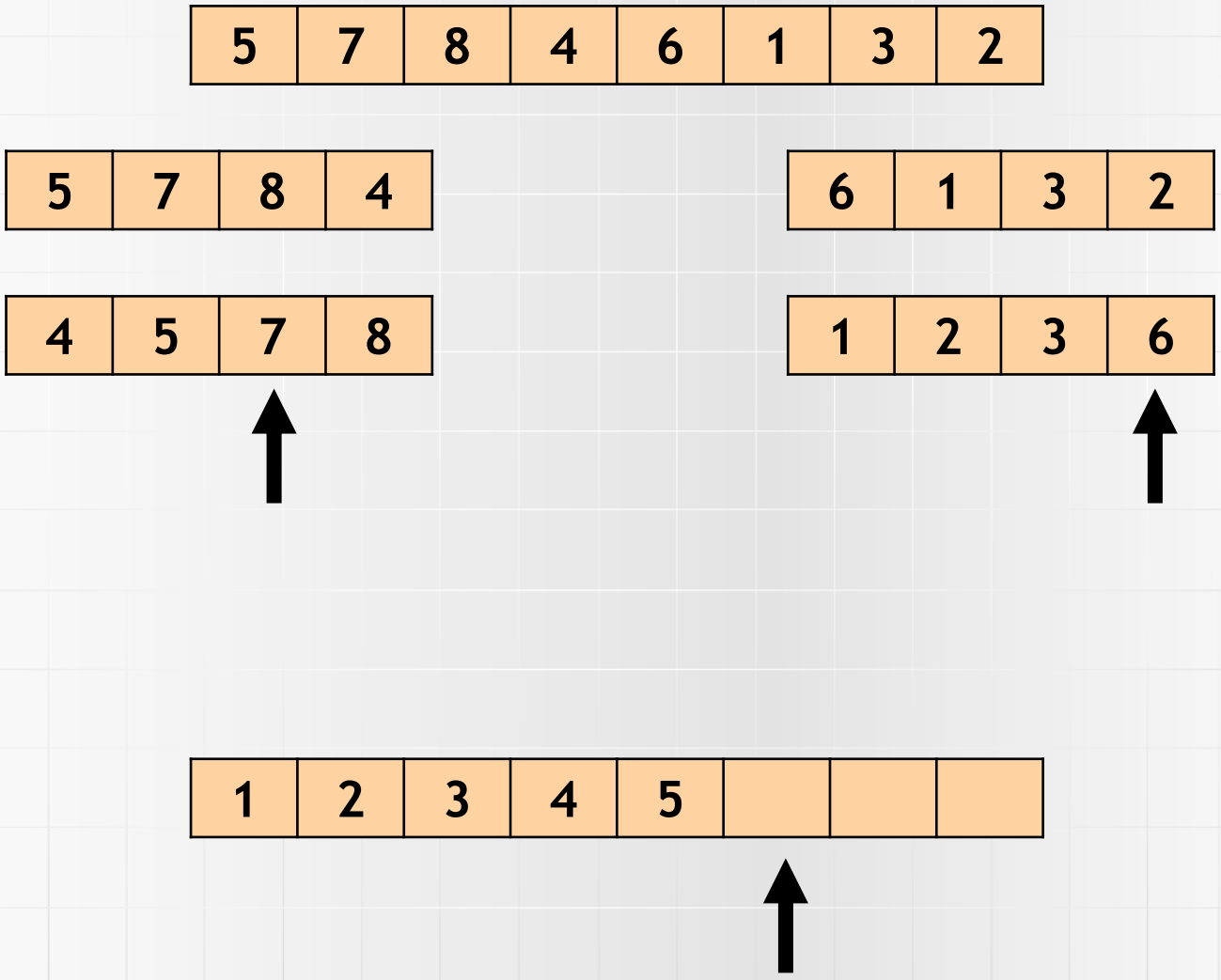
Politechnika Wrocławska

Sortowanie przez scalanie



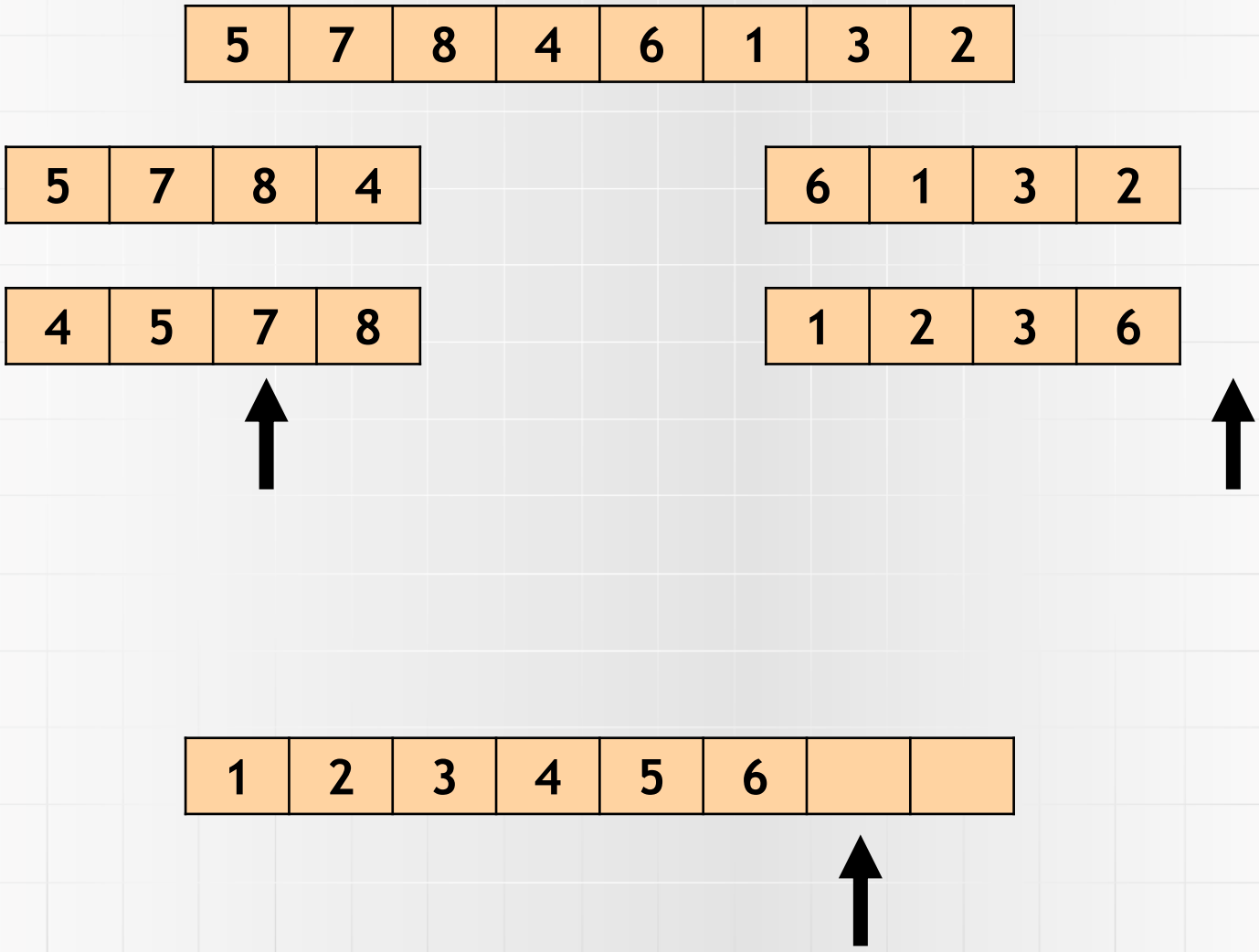
Politechnika Wrocławska

Sortowanie przez scalanie



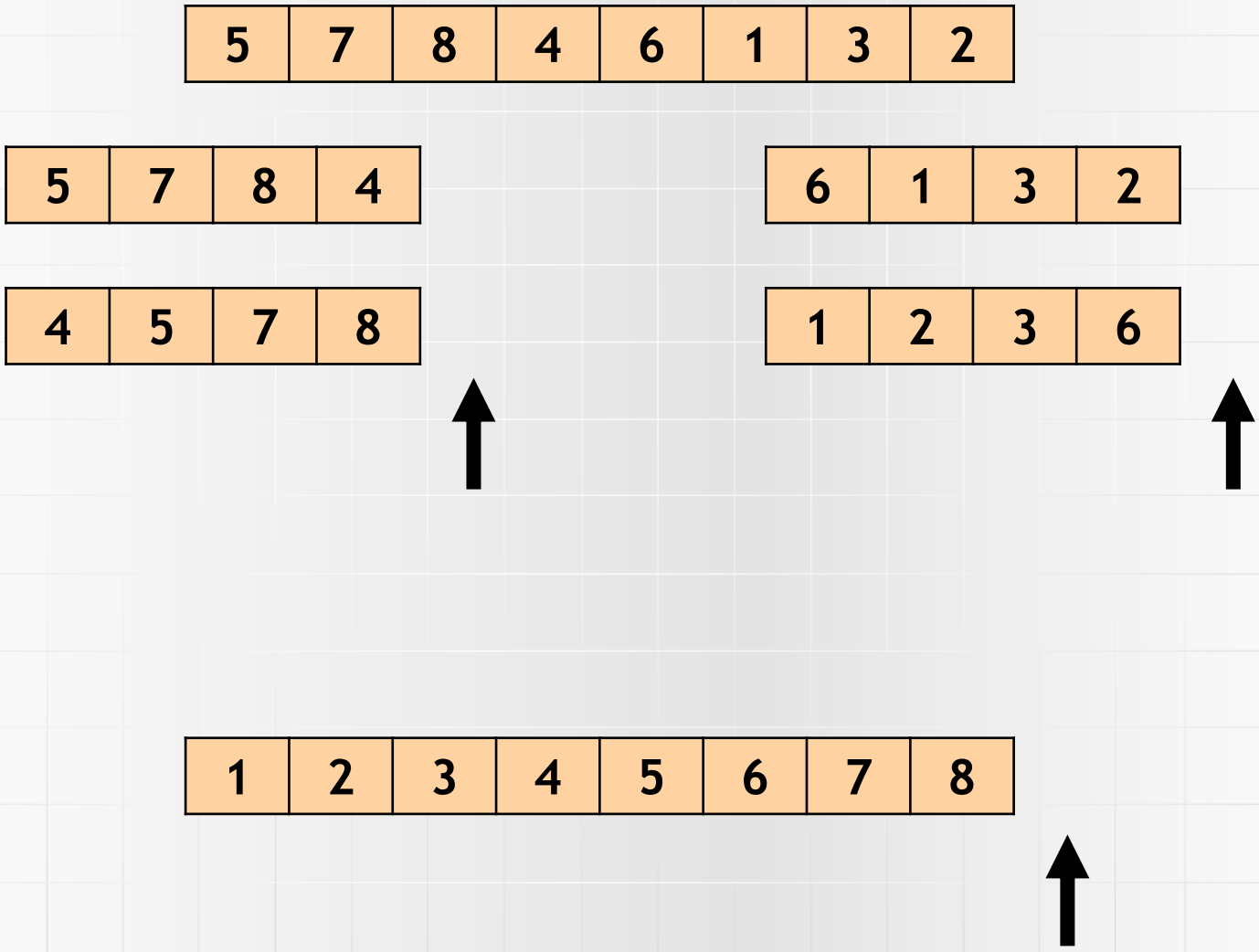
Politechnika Wrocławska

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.

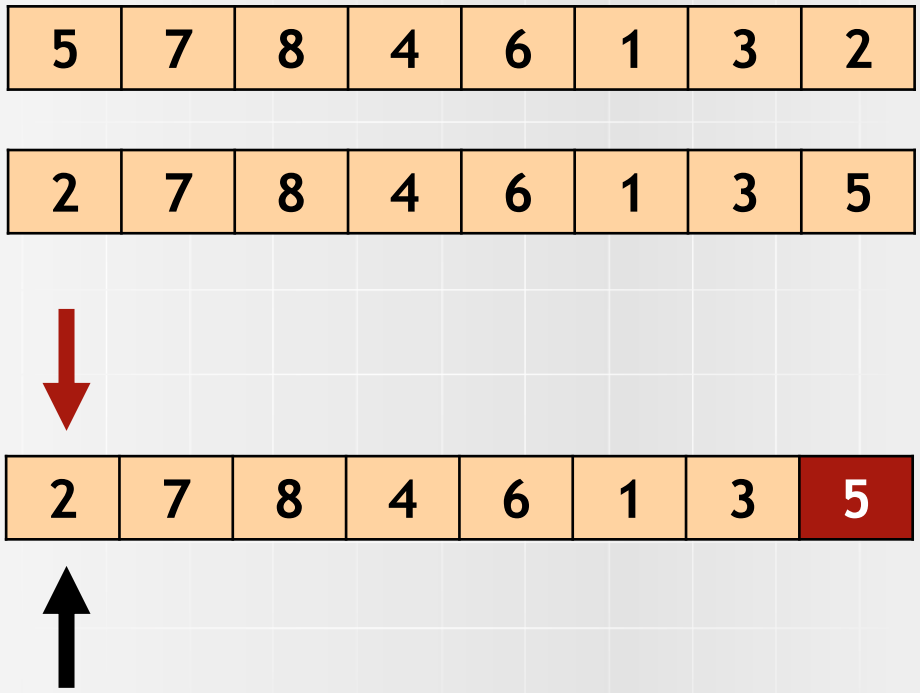
Politechnika Wrocławska

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

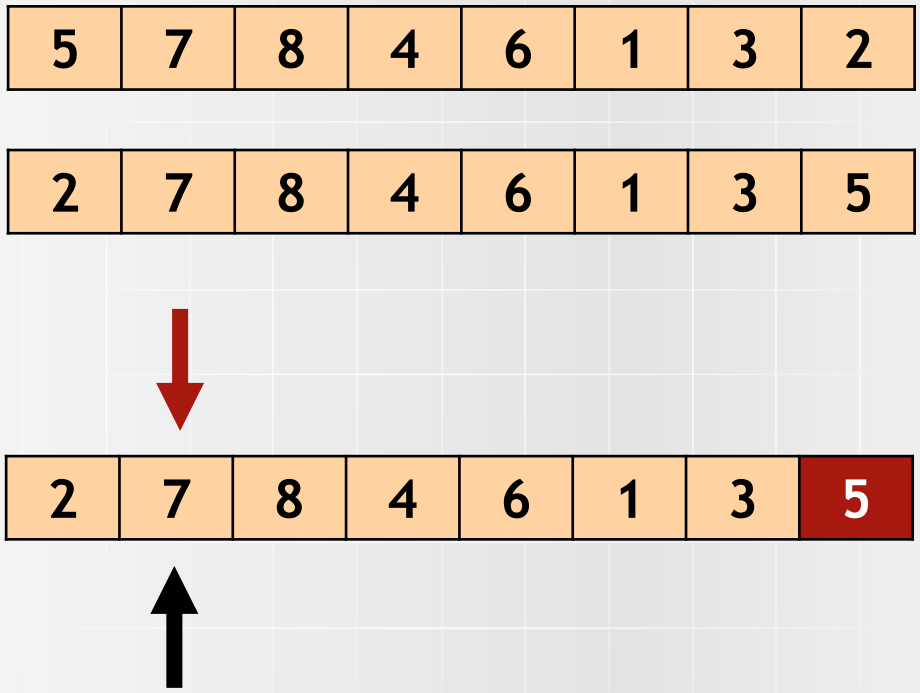
Sortowanie szybkie



Politechnika Wrocławska

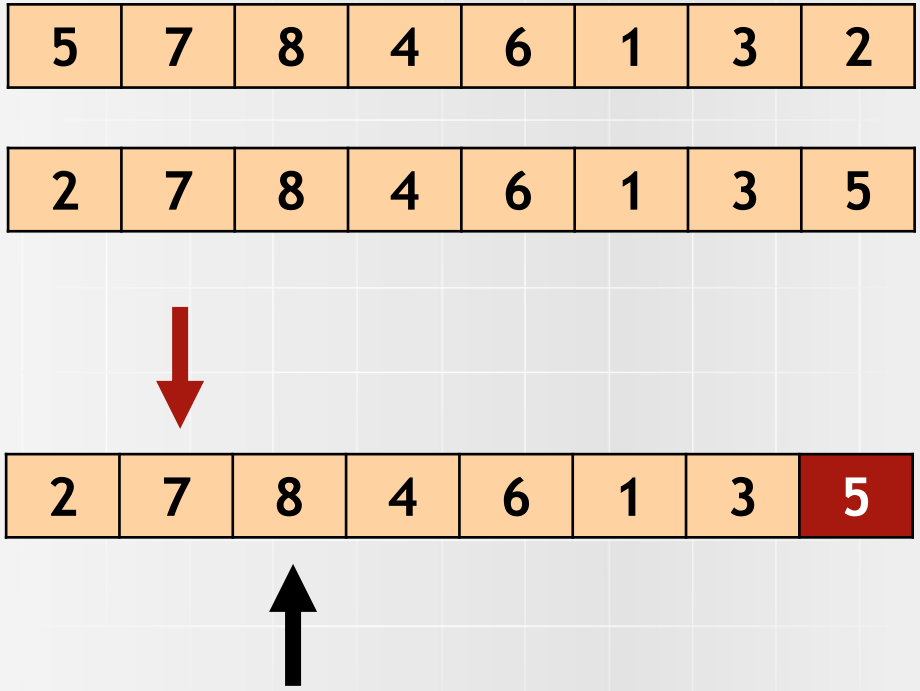
Algorytmy sortowania

Sortowanie szybkie



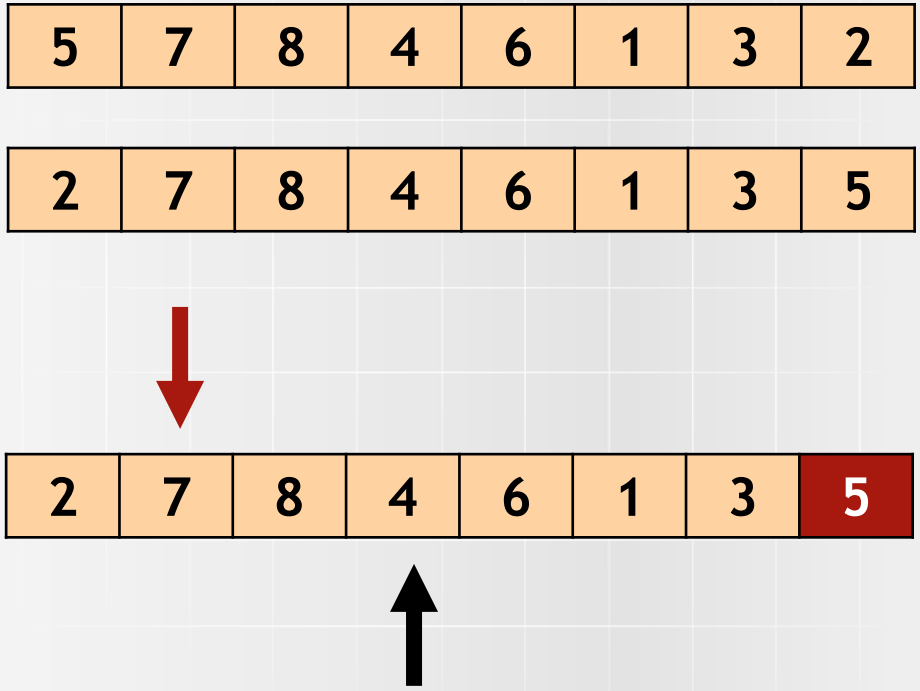
Algorytmy sortowania

Sortowanie szybkie



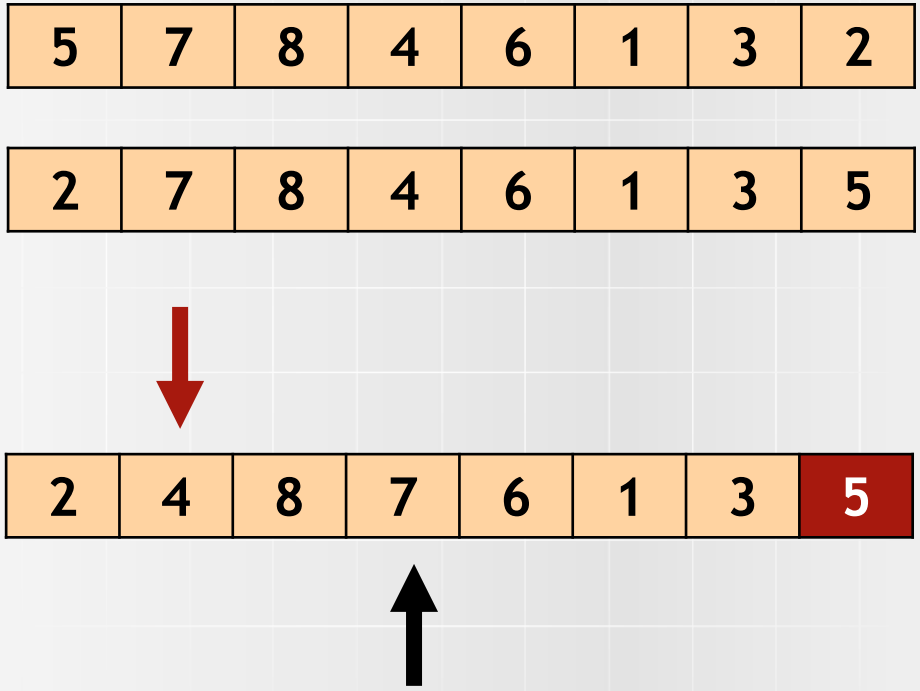
Algorytmy sortowania

Sortowanie szybkie



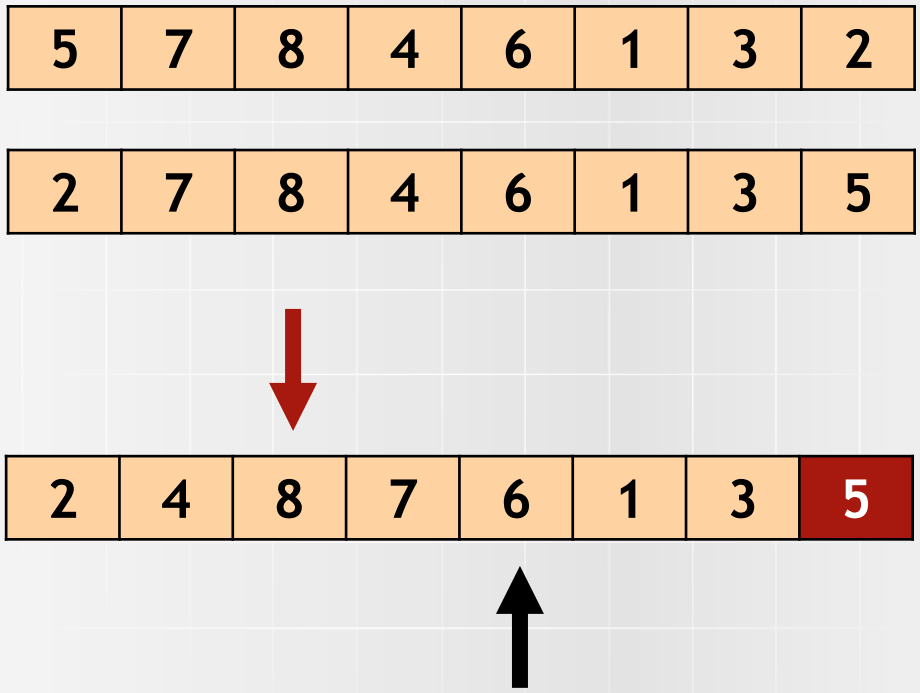
Algorytmy sortowania

Sortowanie szybkie



Algorytmy sortowania

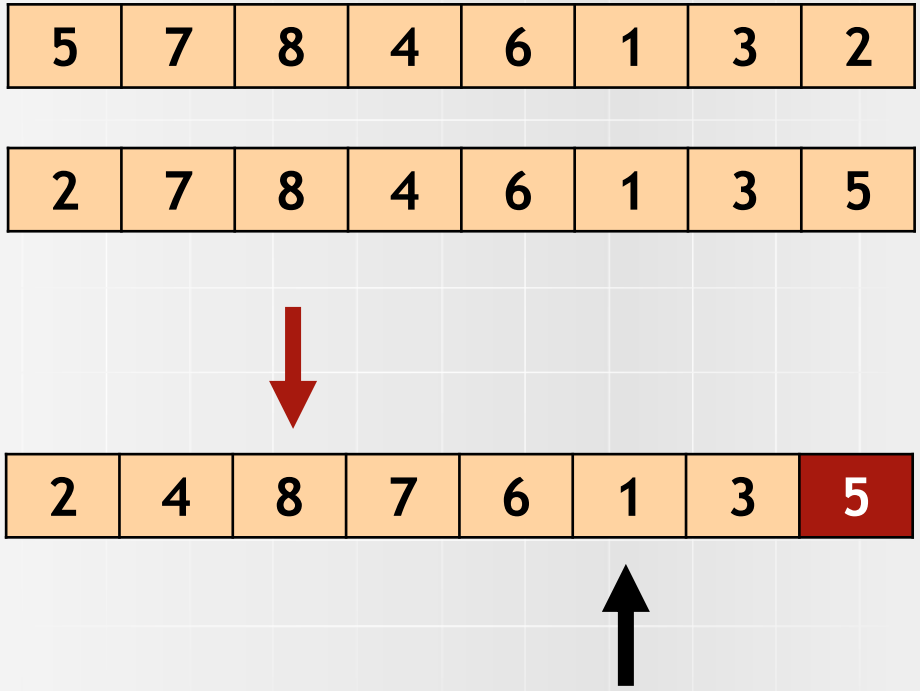
Sortowanie szybkie



Politechnika Wrocławska

Algorytmy sortowania

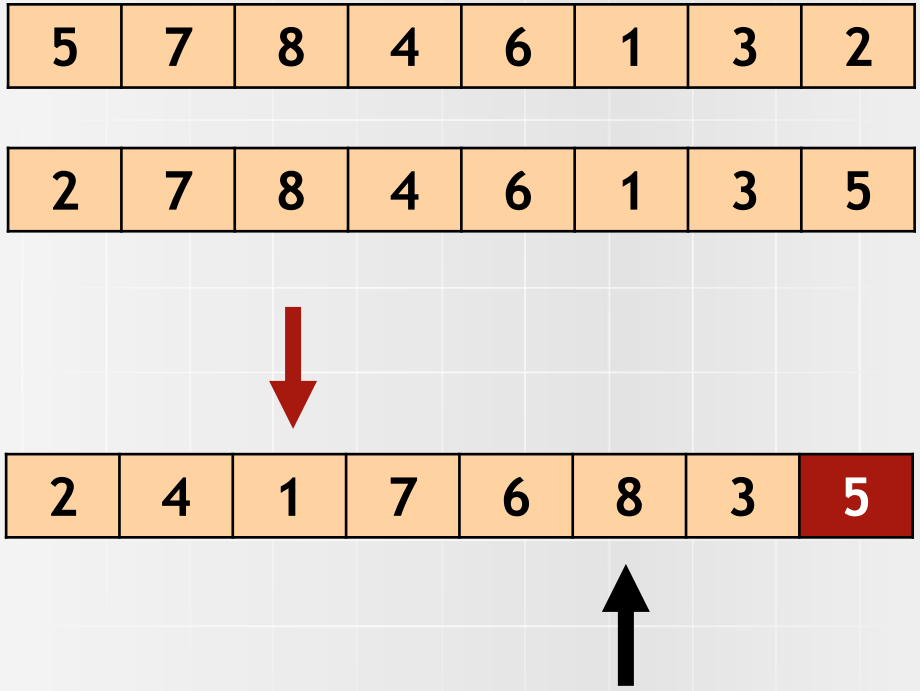
Sortowanie szybkie



Politechnika Wrocławska

Algorytmy sortowania

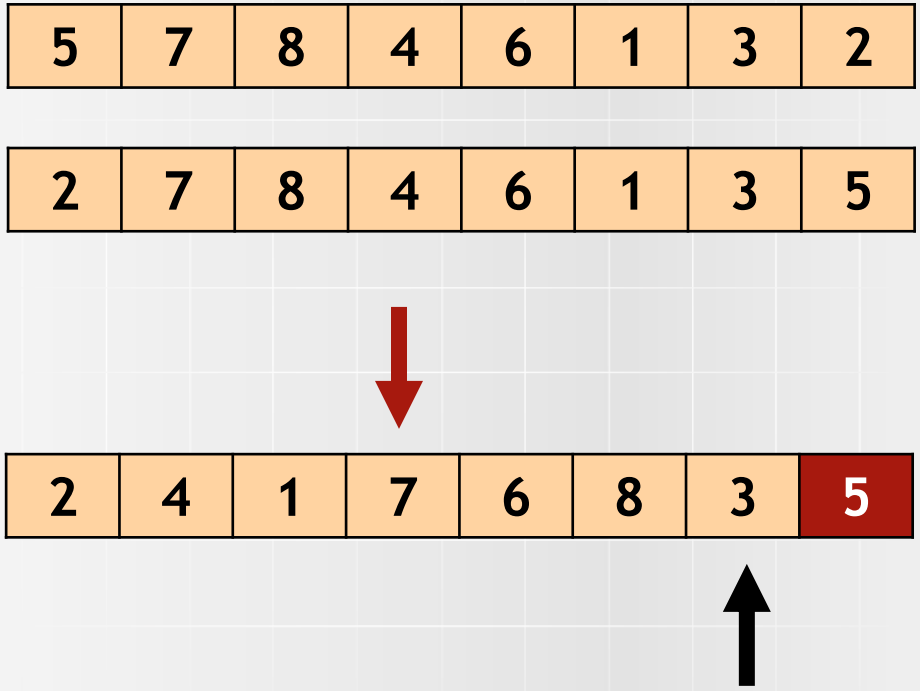
Sortowanie szybkie



Politechnika Wrocławska

Algorytmy sortowania

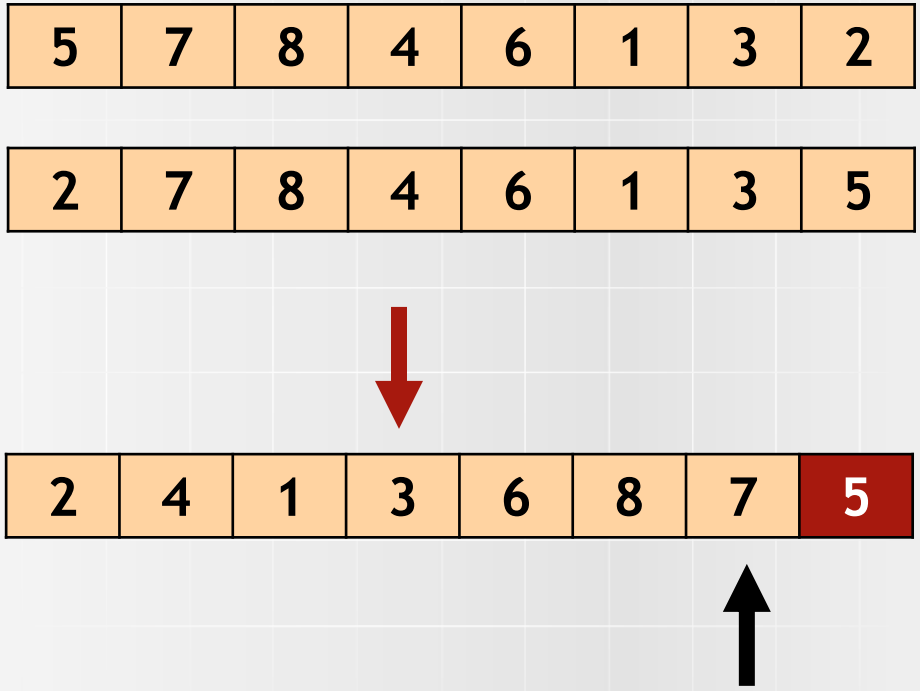
Sortowanie szybkie



Politechnika Wrocławska

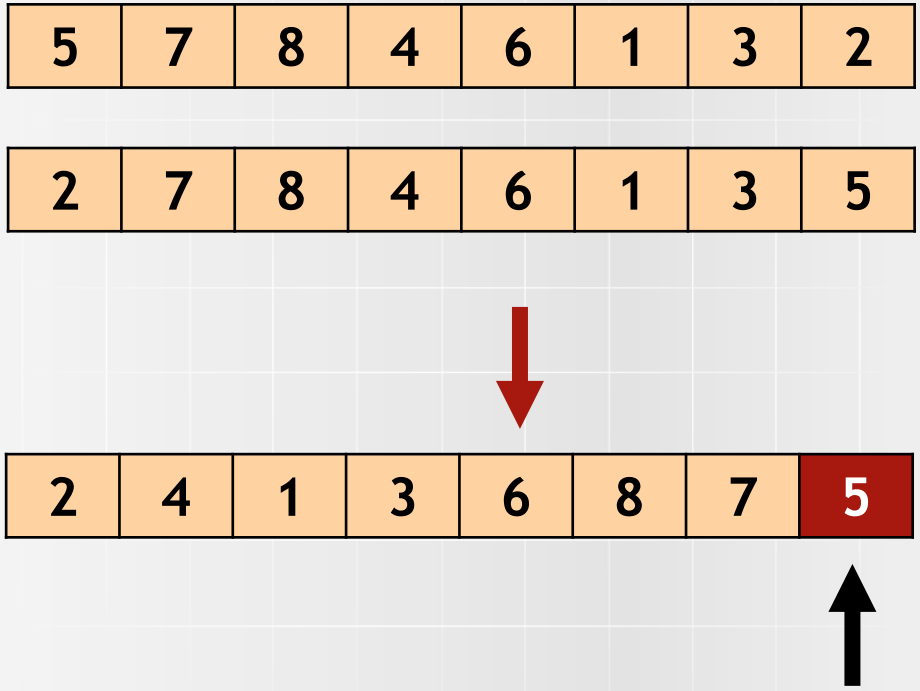
Algorytmy sortowania

Sortowanie szybkie



Algorytmy sortowania

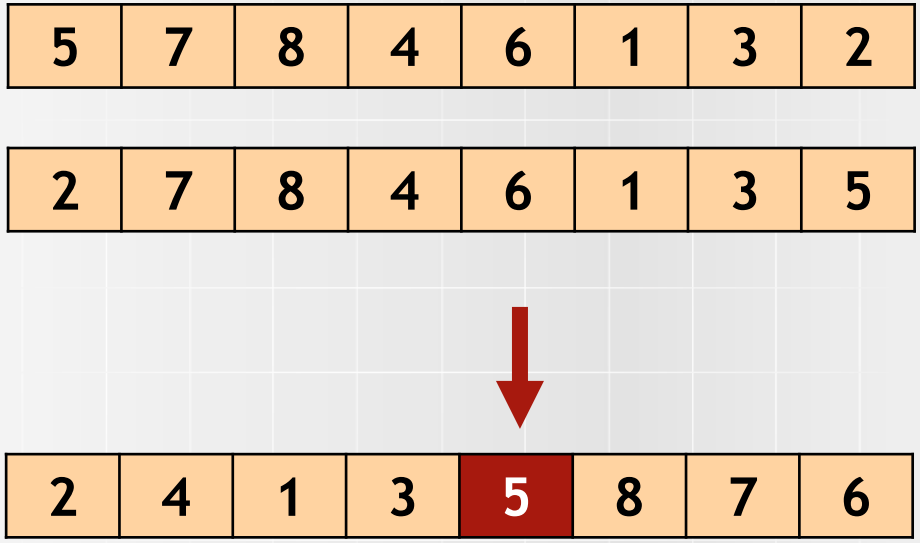
Sortowanie szybkie



Politechnika Wrocławska

Algorytmy sortowania

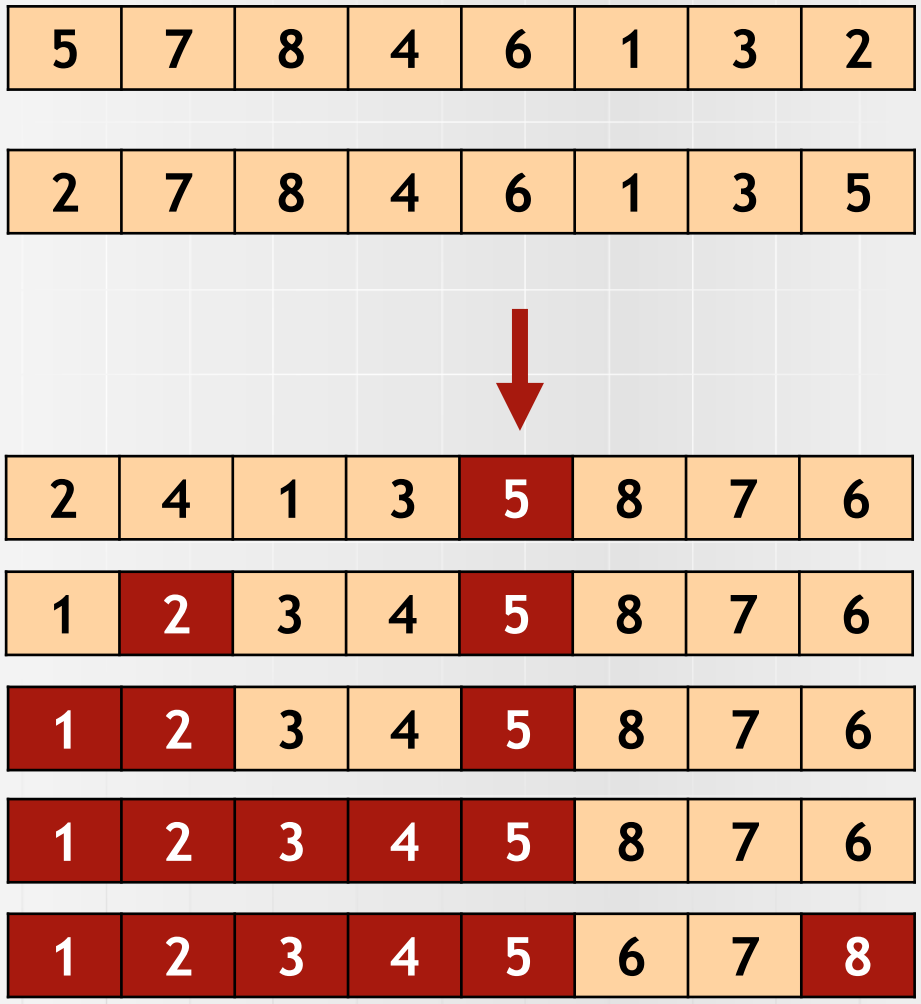
Sortowanie szybkie



Politechnika Wrocławska

Algorytmy sortowania

Sortowanie szybkie



Politechnika Wrocławska

Absolutne minimum

- Pakowanie/rozpakowanie listy/krotki

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