

Czym są algorytmy?

CECHY, SPOSOBY PRZEDSTAWIANIA, PRZYKŁADY

Paweł Musioł, Dominik Wolniaczyk

DEFINICJA

ALGORYTM

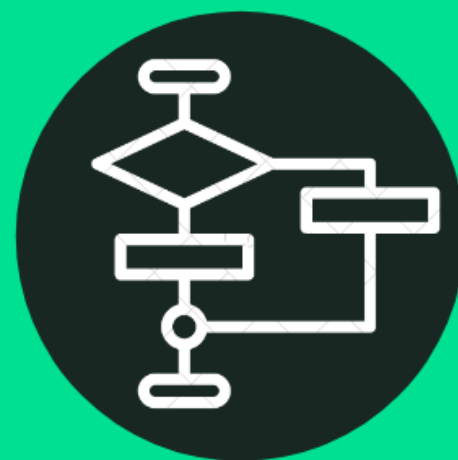
to zestaw ściśle określonych czynności prowadzących do wykonania danego zadania

Cechy algorytmu



Poprawność

zwraca prawidłowy
wynik



Jednoznaczność

dla tych samych danych
wejściowych zwraca
te same wyniki



Skończoność

liczba kroków jest
skończona



Efektywność

rozwiązuje problem
w możliwie najmniejszej
liczbie kroków

Sposoby reprezentacji algorytmów

Z PRZYKŁADAMI

OPIŚ SŁOWNY

Po wczytaniu danych wejściowych **a** i **b** porównaj wprowadzone liczby.

Jeśli **$a < b$** , to **$\text{min} = a$** . Wypisz wynik. Jeśli **$a \geq b$** , to sprawdź czy **$b < a$** . Jeśli tak, to **$\text{min} = b$** . Wypisz wynik. W przeciwnym przypadku **$\text{min} = a = b$** . Wypisz wynik.

**Algorytm wybierający
liczbę mniejszą
z podanych dwóch liczb**

LISTA KROKÓW

Dane wejściowe: dwie dowolne liczby a , b .

Dane wyjściowe: liczba suma, będąca sumą liczb a i b .

**Obliczanie sumy
dwóch liczb**

1. Rozpocznij wykonywanie algorytmu (**Start**)
2. Wczytaj wartości liczb **a i b**
3. **$\text{suma} := a + b$**
4. Wypisz **suma**
5. Zakończ działanie algorytmu (**Koniec**).

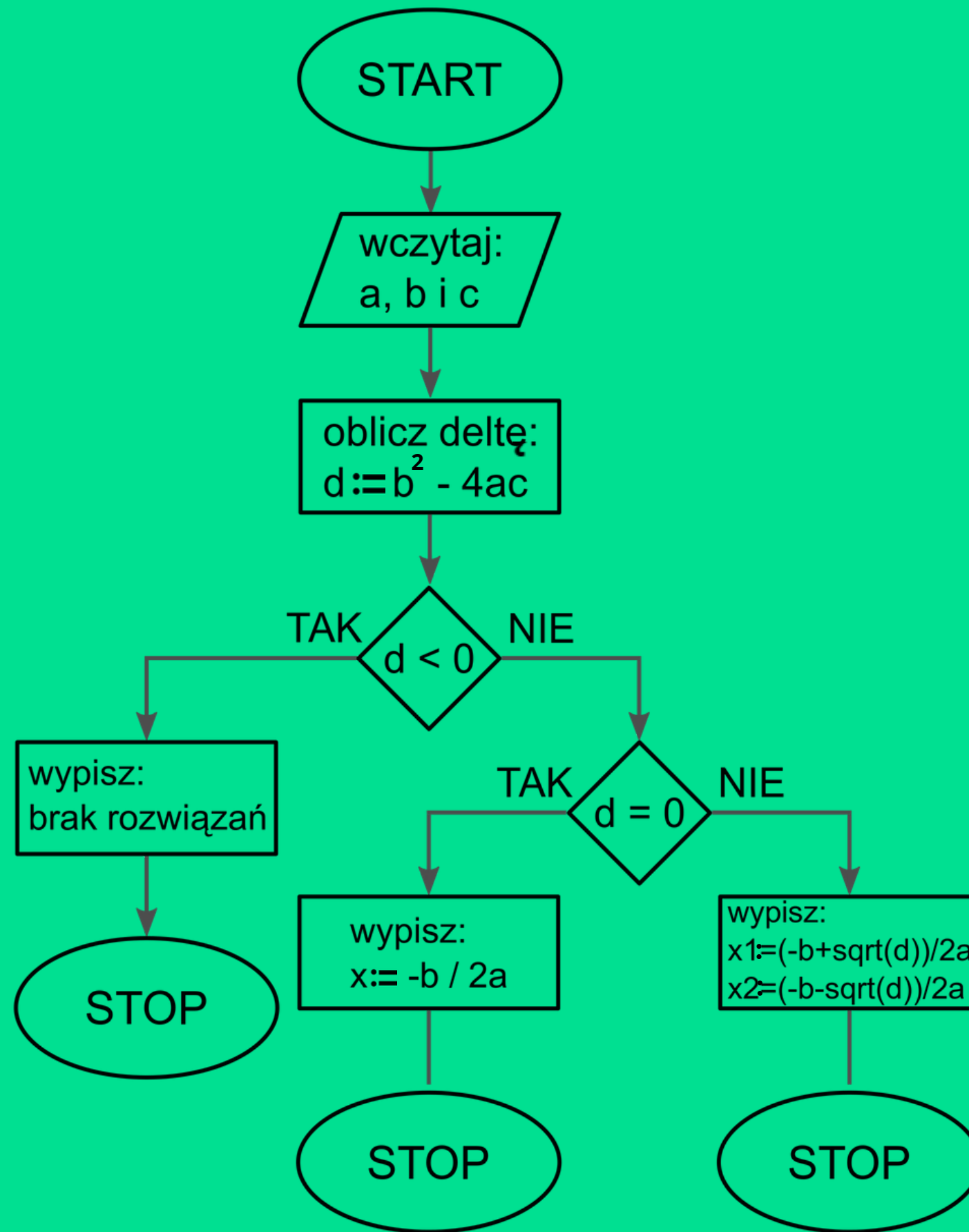
SCHEMAT BLOKOWY

Uniwersalny schemat naprawy

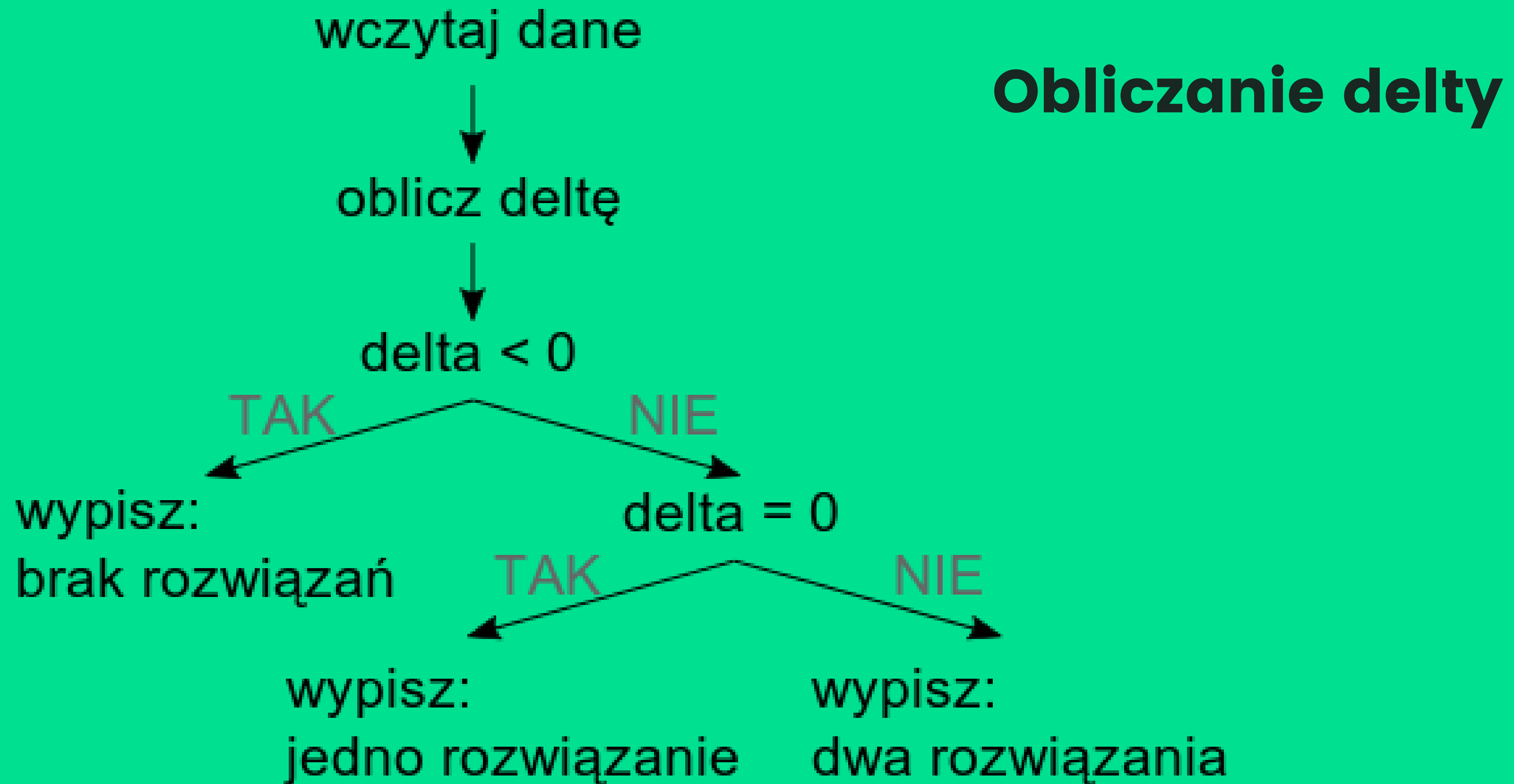


SCHEMAT BLOKOWY

Obliczanie delty



DRZEWO ALGORYTMU



PSEUDOKOD

Start

Suma:=0

Podaj(n)

i:=0

Dopóki $i < n$ wykonuj:

 Wczytaj(a)

 Suma := Suma + a

 i := i + 1

Wypisz(Suma)

Koniec

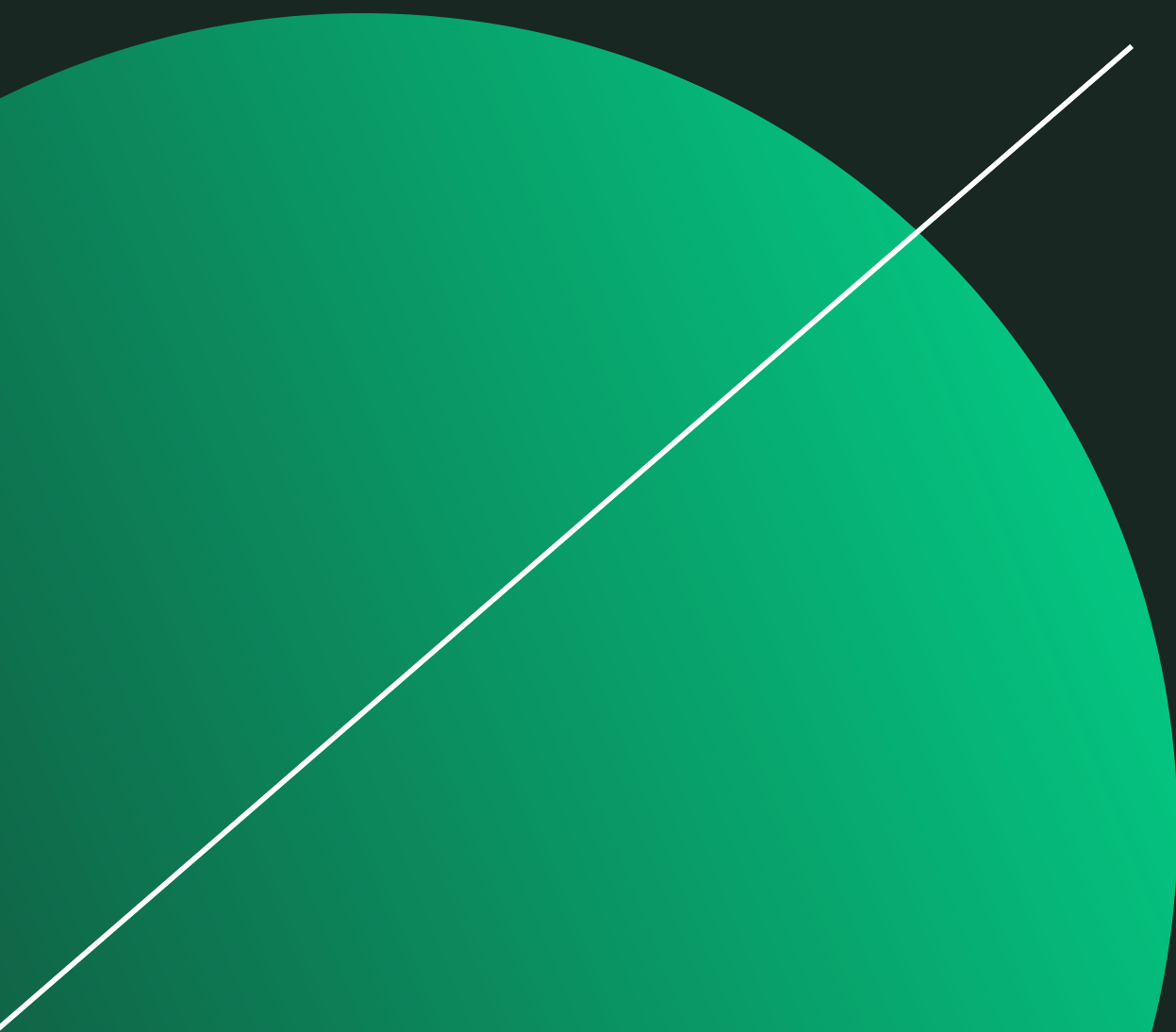
Sumowanie n liczb

JĘZYK PROGRAMOWANIA

Przykładowe sortowanie bąblekowe w C++

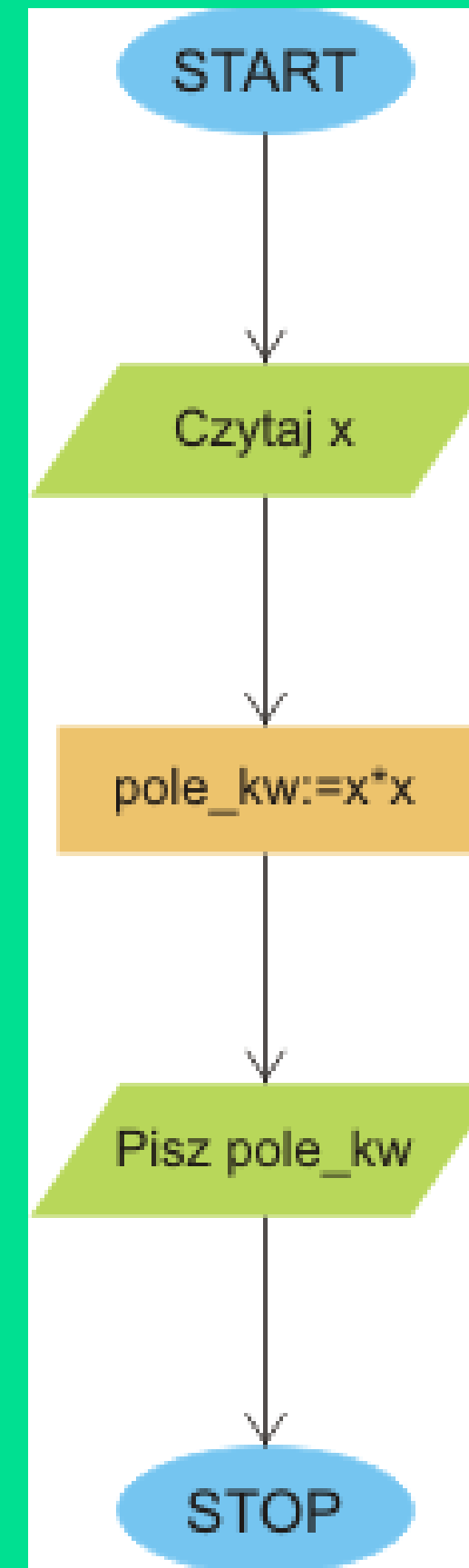
```
1  #include<iostream>
2  using namespace std;
3  int main ()
4  {
5      int i, j,temp,pass=0;
6      int a[10] = {10,2,0,14,43,25,18,1,5,45};
7      cout <<"Input list ...\n";
8      for(i = 0; i<10; i++) {
9          cout <<a[i]<<"\t";
10     }
11     cout<<endl;
12     for(i = 0; i<10; i++) {
13         for(j = i+1; j<10; j++)
14         {
15             if(a[j] < a[i]) {
16                 temp = a[i];
17                 a[i] = a[j];
18                 a[j] = temp;
19             }
20         }
21         pass++;
22     }
23     cout <<"Sorted Element List ...\n";
24     for(i = 0; i<10; i++) {
25         cout <<a[i]<<"\t";
26     }
27     cout<<"\nNumber of passes taken to sort the list:"<<pass<<endl;
28     return 0;
29 }
```

Klasyfikacja algorytmów ze względu na sposób wykonywania operacji



SEKWENCYJNE

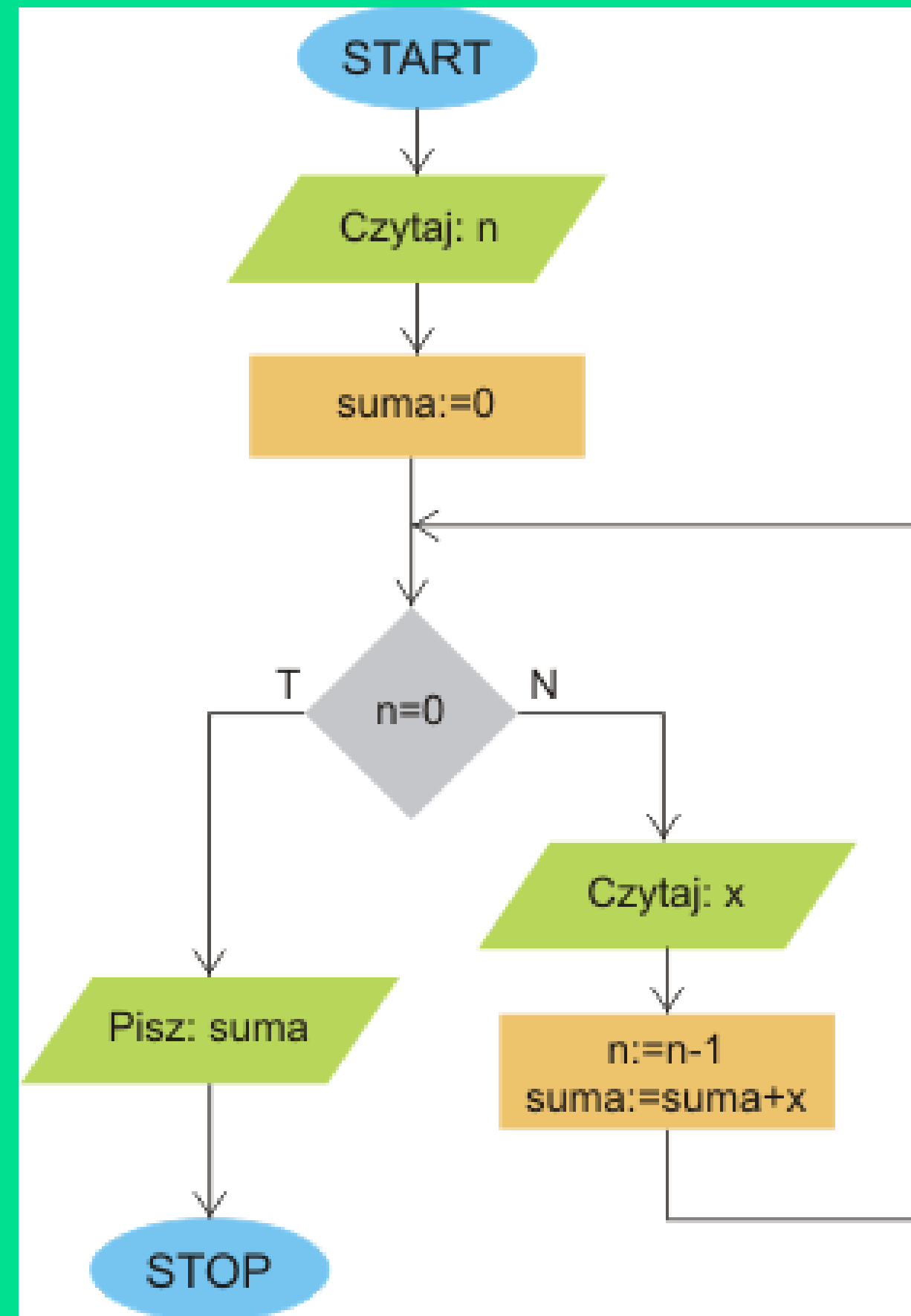
operacje w algorytmie wykonywane są w kolejności, w jakiej zostały opisane



Source

ITERACYJNE

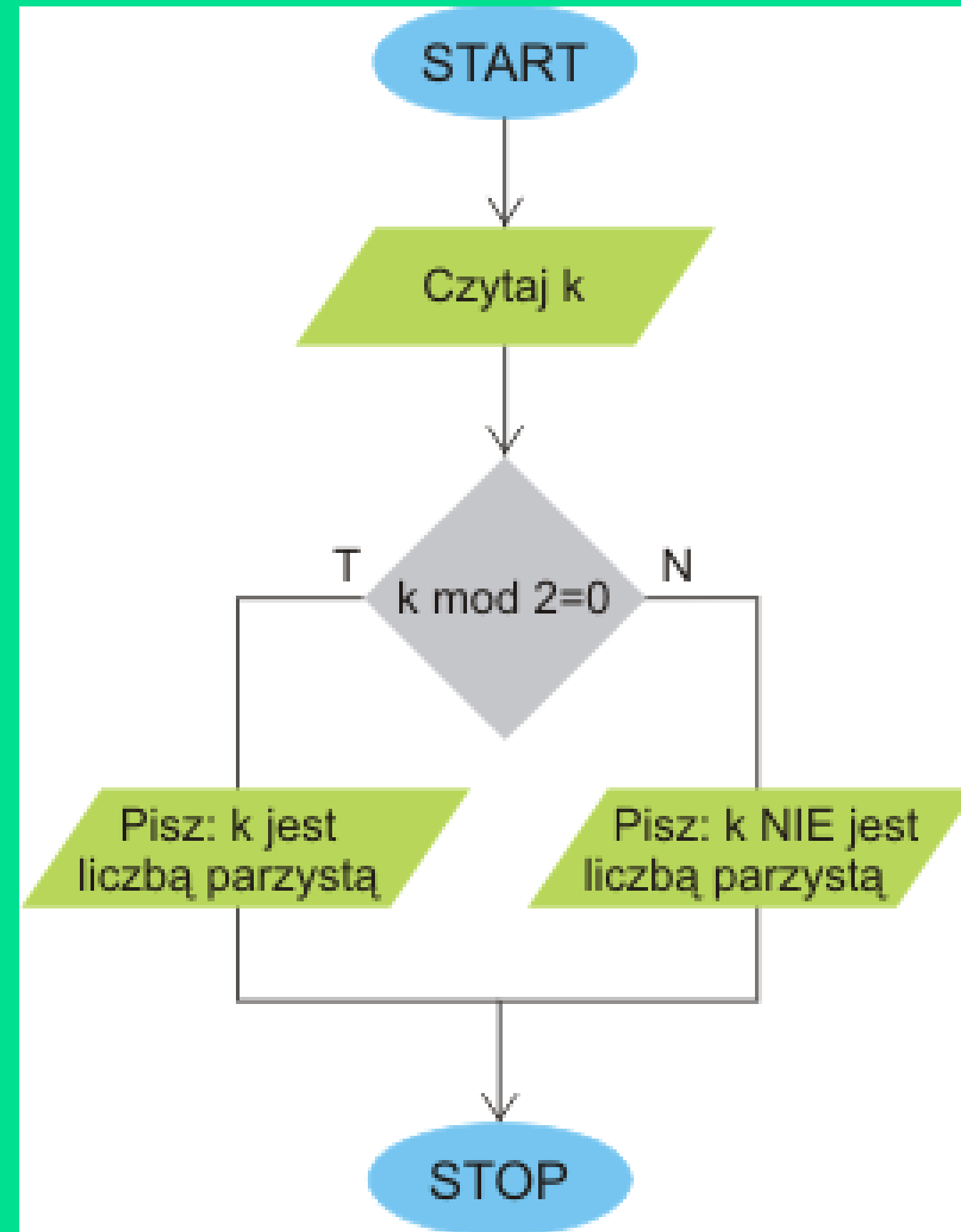
niektóre kroki są powtarzane, aż do spełnienia wymaganego warunku



Source

WARUNKOWE

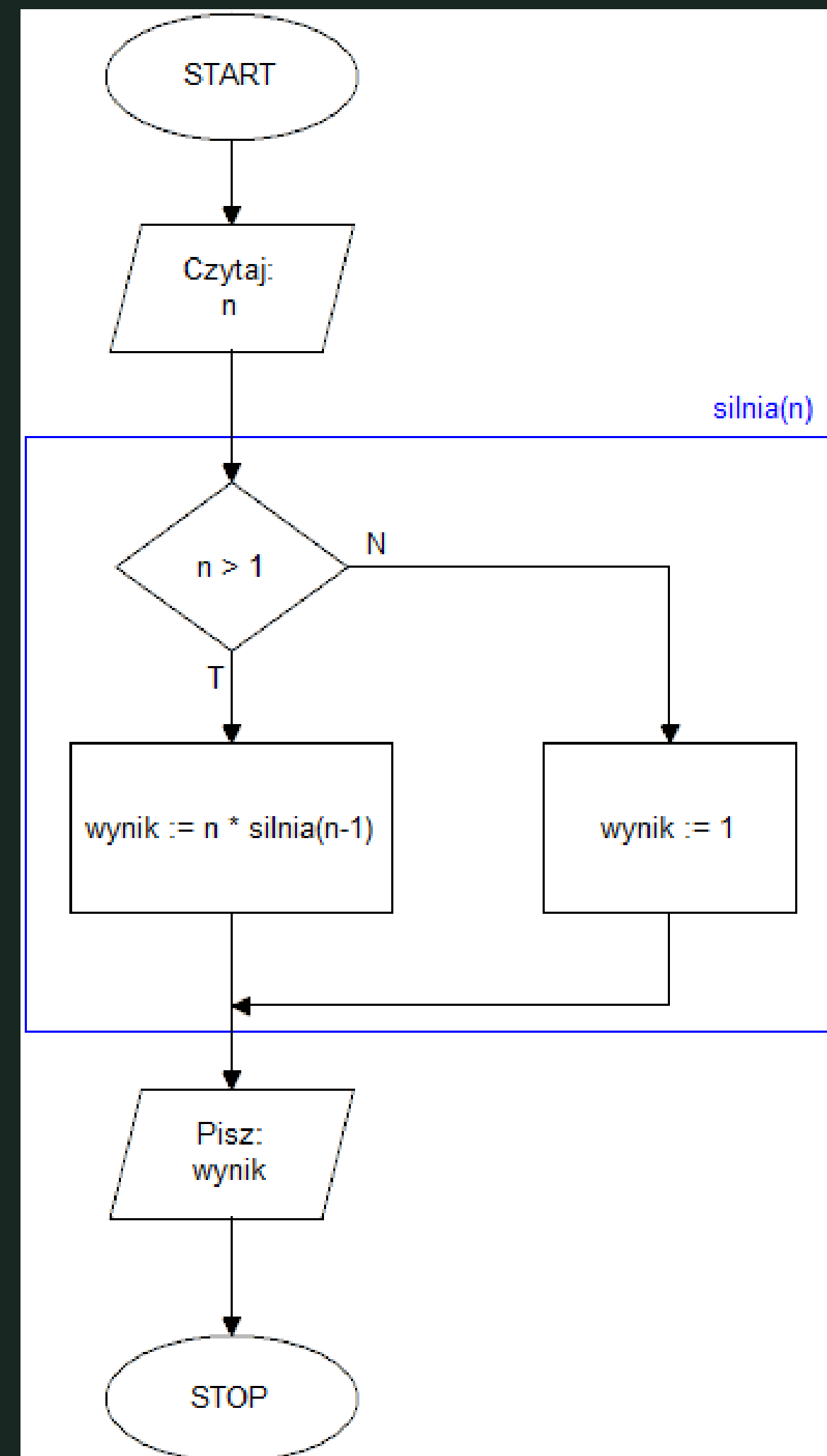
występują w nich
instrukcje warunkowe



Source

REKURENCYJNE
REKURENCYJNE
REKURENCYJNE
REKURENCYJNE

tworzona jest formuła
odwołująca się do niej
samej

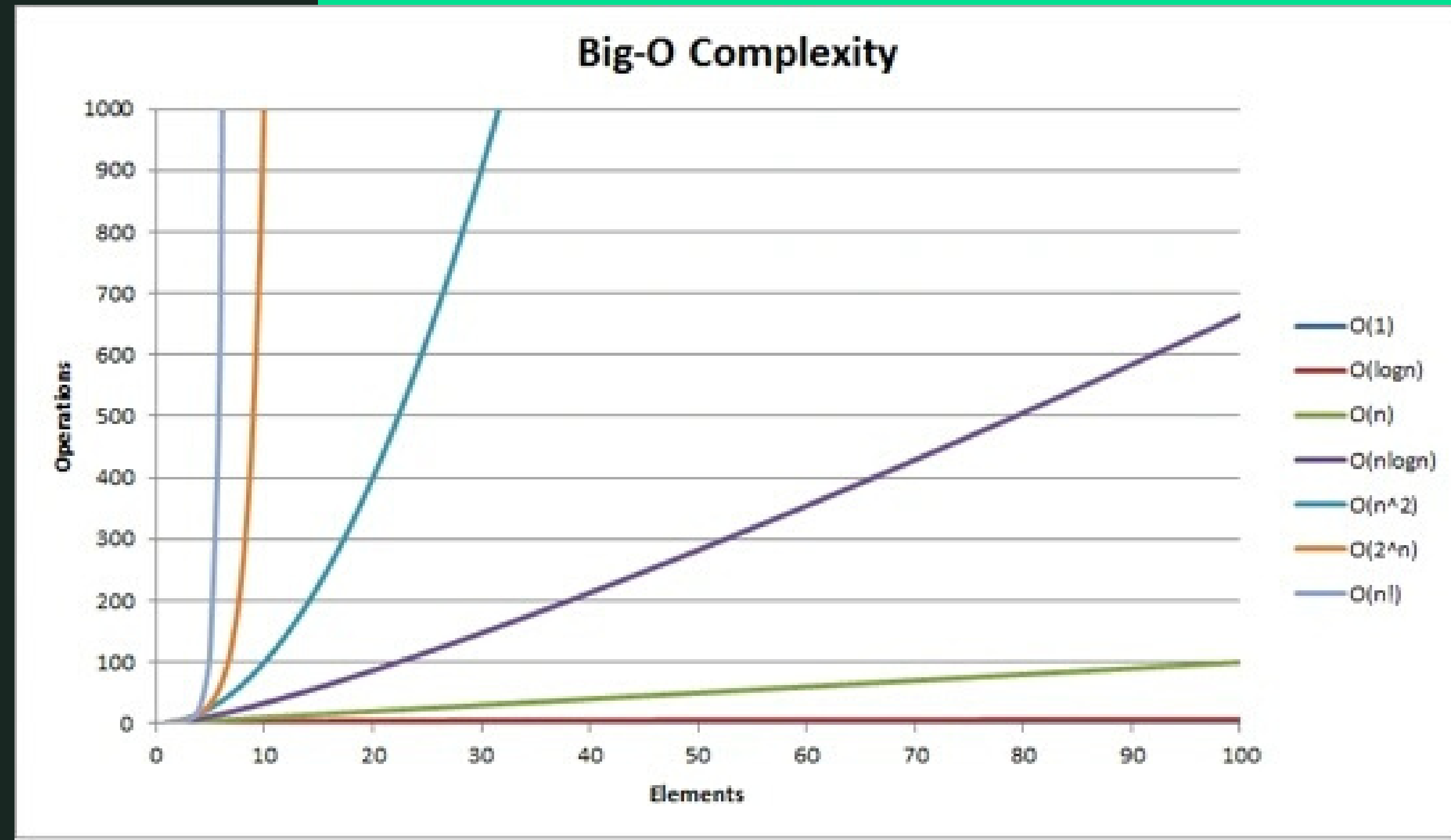




Złożoność obliczeniowa algorytmów

Złożoność czasowa, złożoność pamięciowa

Złożoność czasowa



Source

Czyli liczba wykonywanych operacji
w zależności od danych wejściowych.
Zapisywana w notacji dużego O

Złożoność pamięciowa

Określa wielkość pamięci operacyjnej komputera, która jest potrzebna do przechowywania danych wejściowych, pośrednich oraz wyników obliczeń.



Algorytm Quicksort



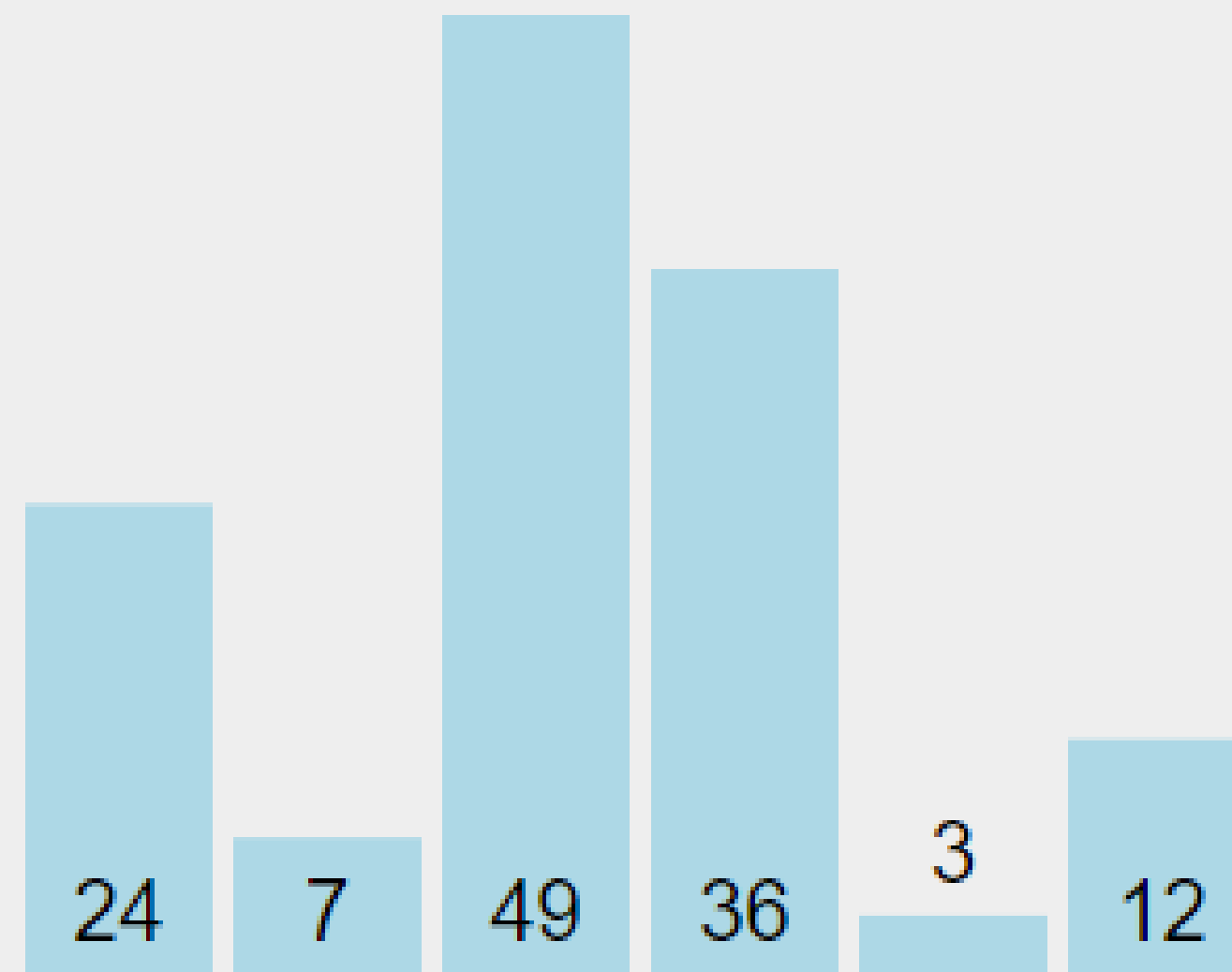
Quicksort stosuje technikę **divide and conquer** (dziel i zwyciężaj).

Dzieli pierwotny problem na podobne mniejsze podproblemy, które rozwiązuje rekurencyjnie. Następnie łączy ze sobą rozwiązania podproblemów i otrzymuje rozwiązanie problemu pierwotnego.

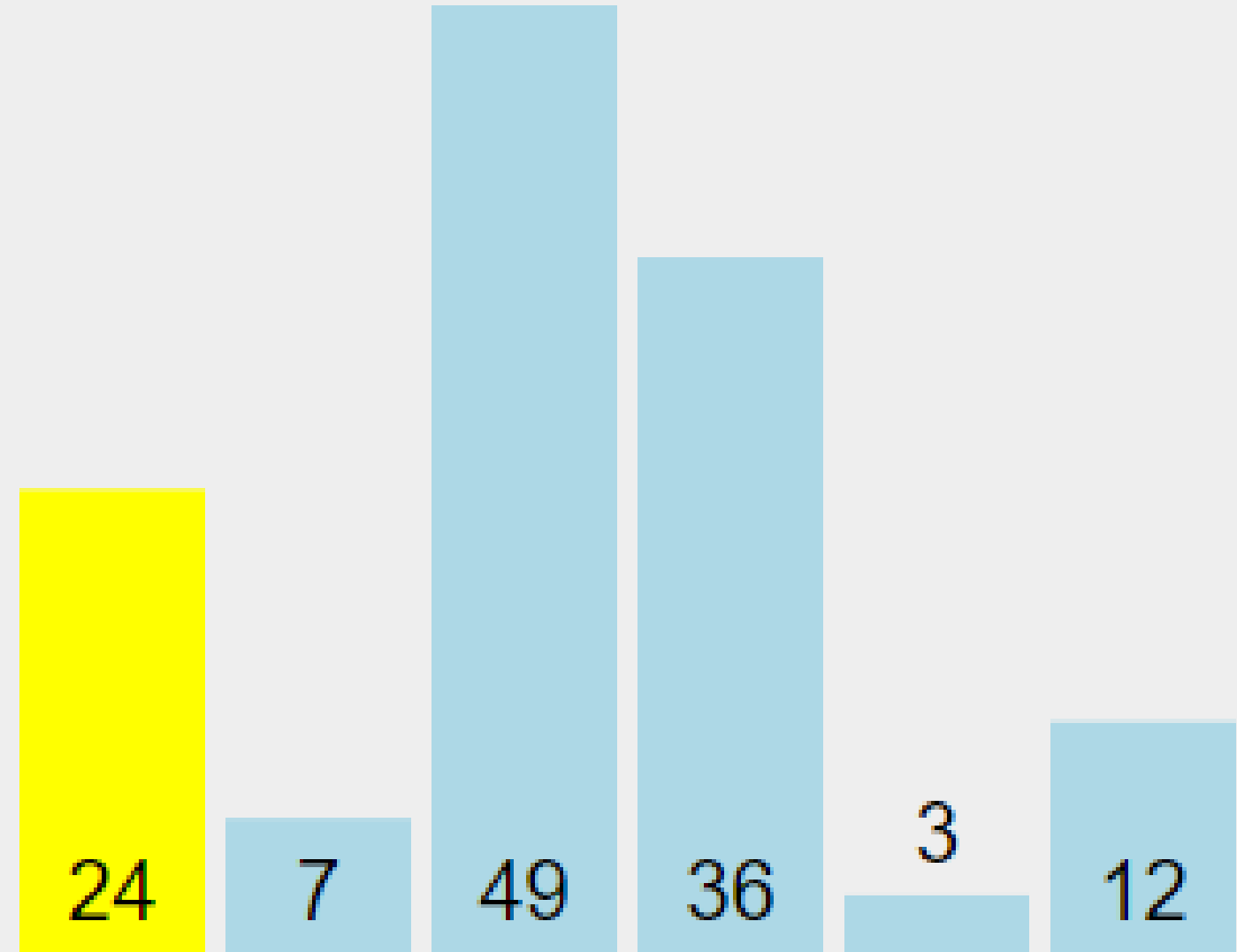
Symulacja działania algorytmu Quicksort



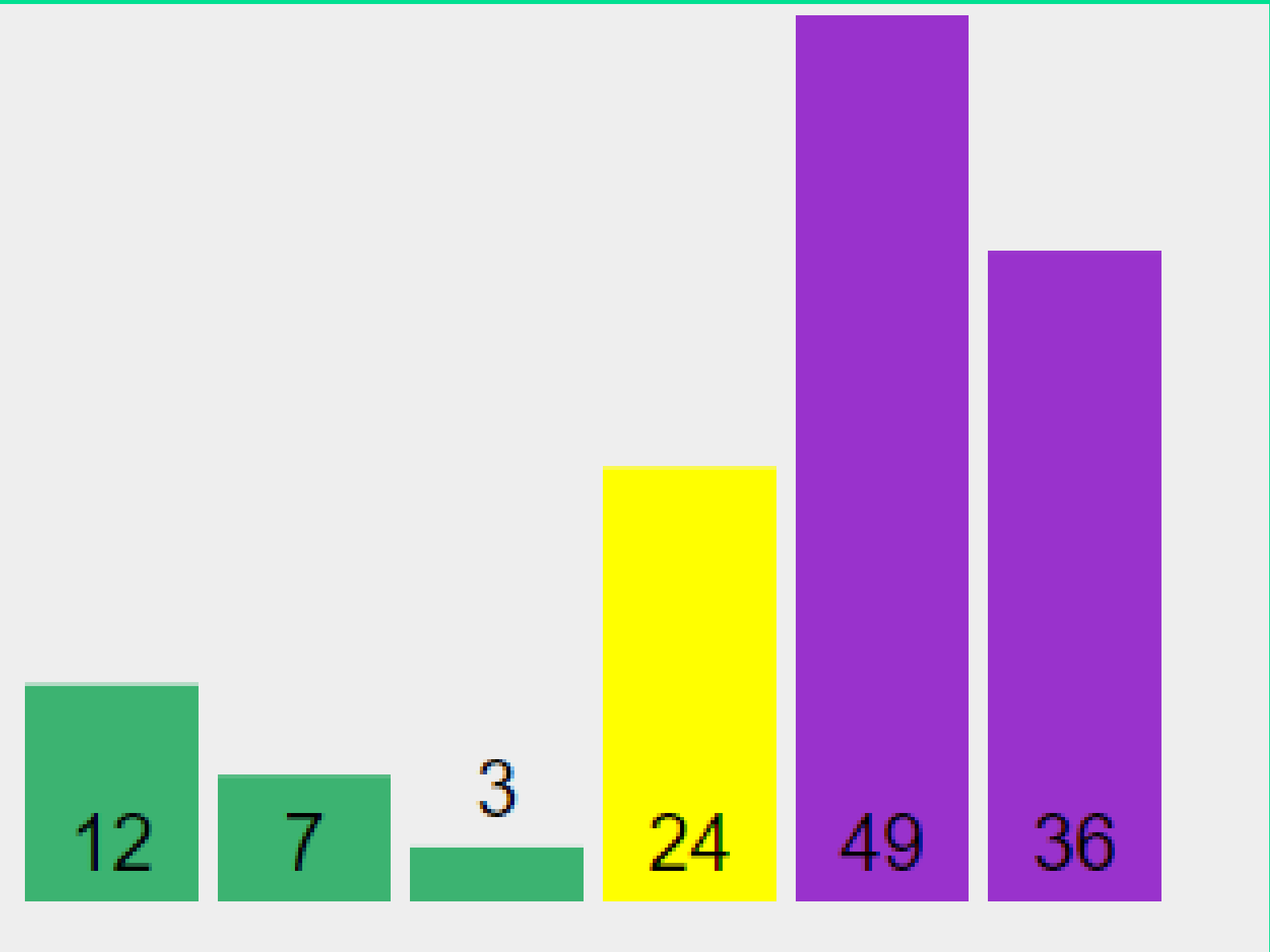
Sortowanie
przeprowadzimy na
tych przykładowych
liczbach.



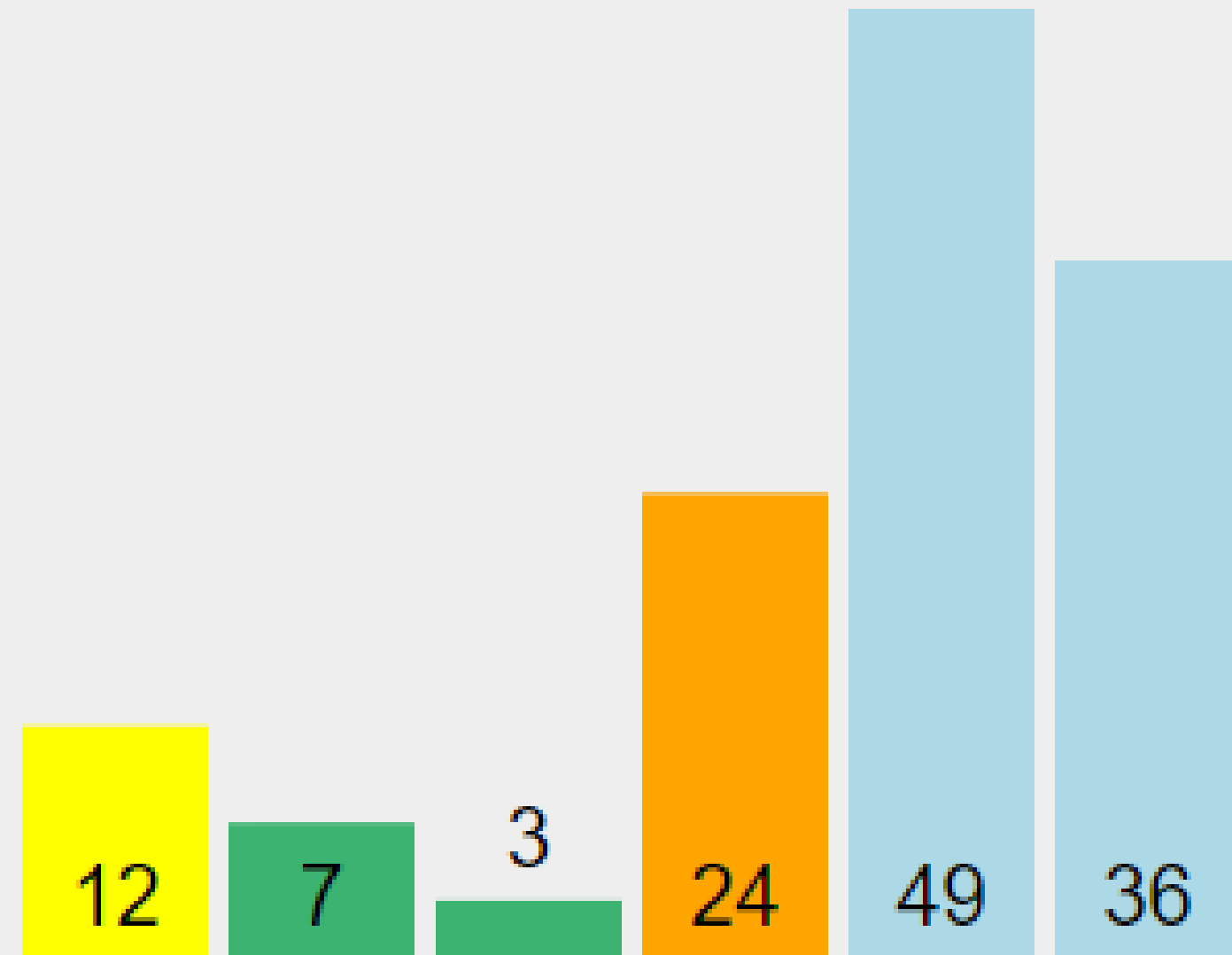
Algorytm najpierw wybiera
element rozdzielający.
Zazwyczaj jest to element
najbardziej na prawo
lub lewo.



Następnie sprawdza, czy pozostałe elementy są **mniejsze**, czy **większe** od elementu rozdzielającego. Mniejsze ustawia po jego lewej stronie, a większe po prawej. Wykonał w ten sposób partycjonowanie. Otrzymanymi w ten sposób grupami liczb zajmie się w kolejnych krokach.

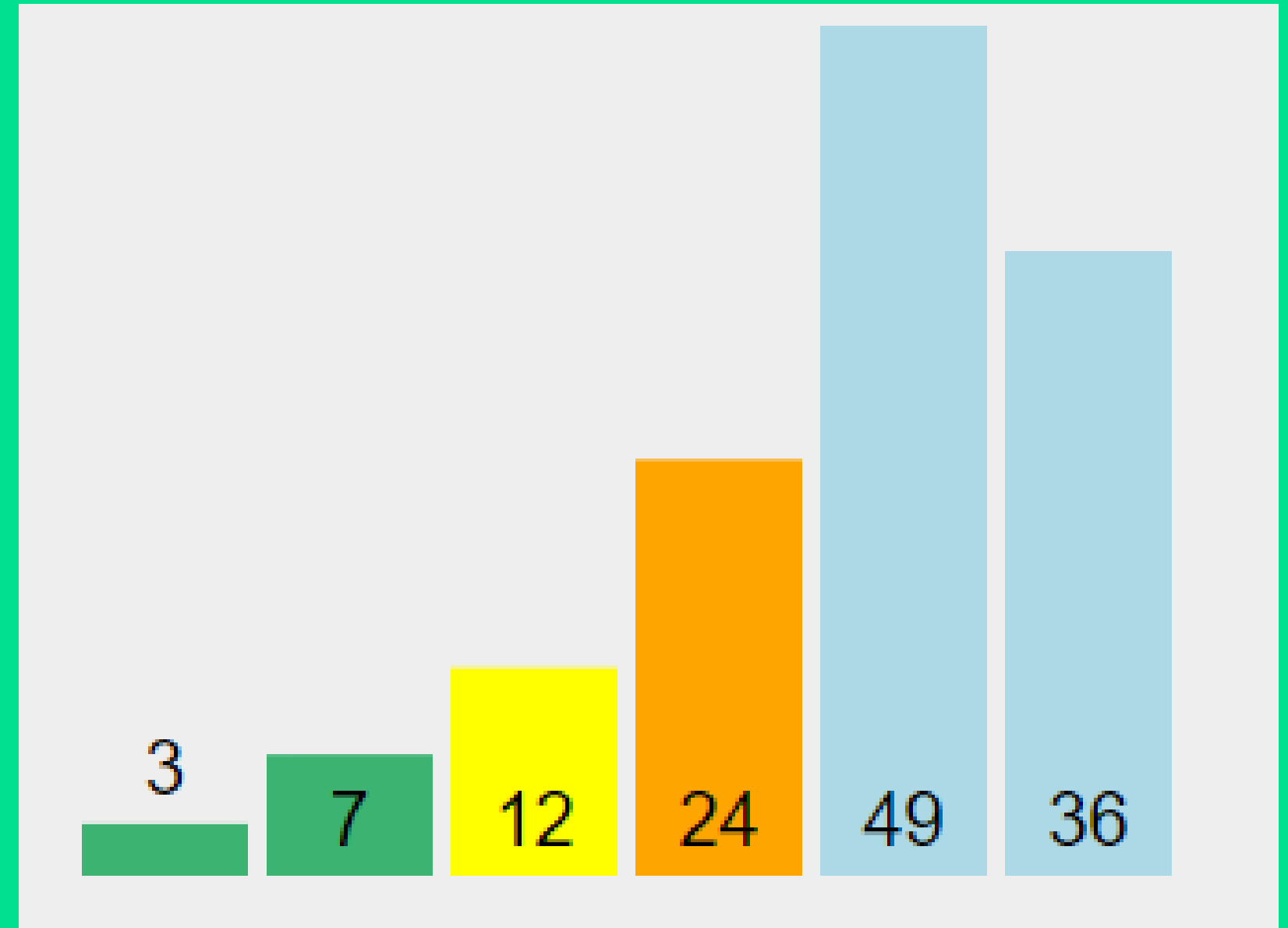
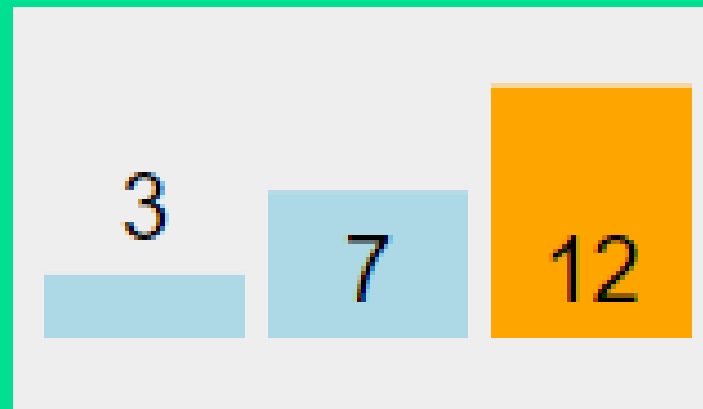


Z grupą liczb na lewo od 24 postępuje tak samo, jak wcześniej ze wszystkimi liczbami. wybiera spośród nich **element rozdzielający**, a potem sprawdza czy pozostałe liczby są od niego mniejsze, czy większe.

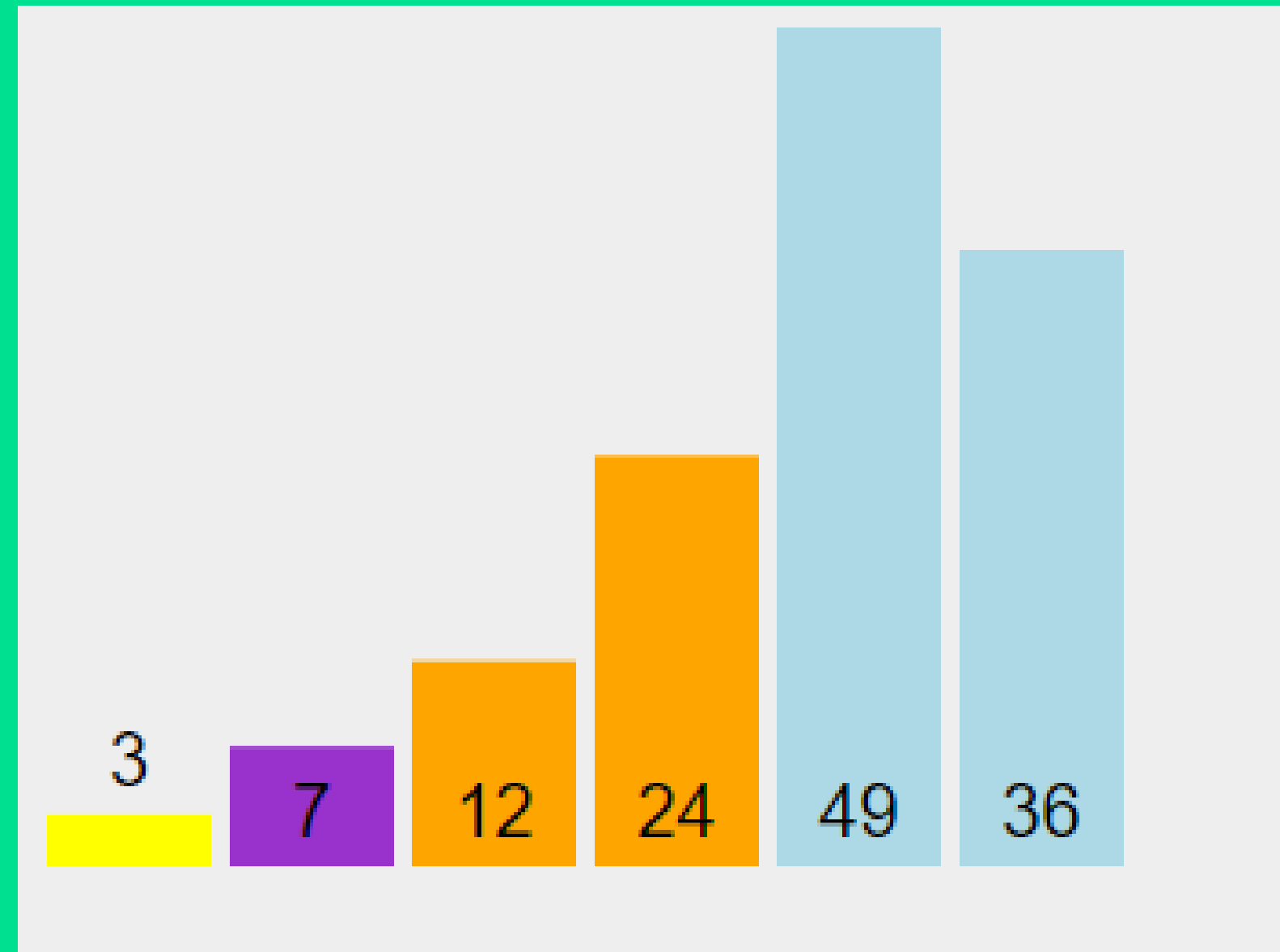


W kolejnym kroku ustawia
liczby mniejsze od 12 po lewo.

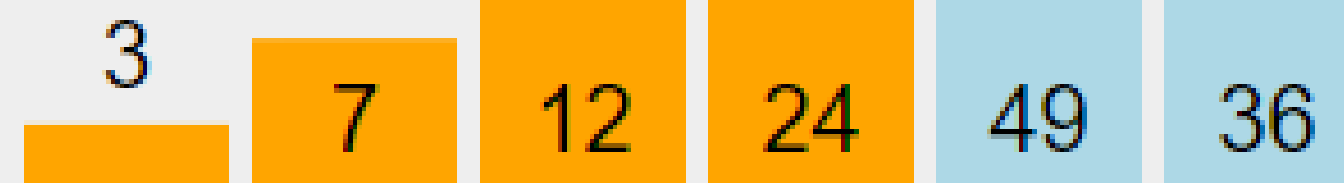
Teraz zajmie się grupą liczb
mniejszych od 12.



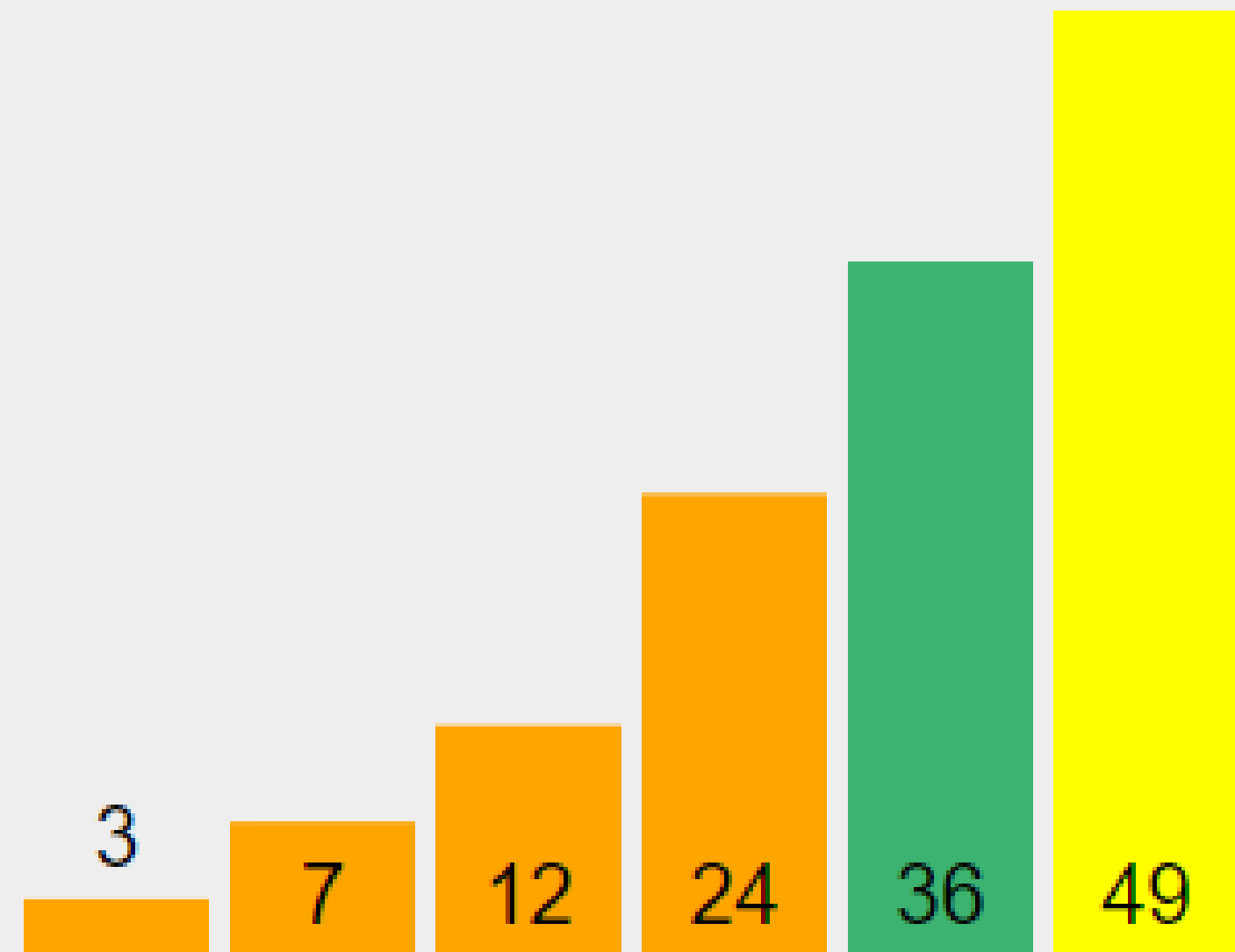
Elementem rozdzielającym zostaje 3, teraz wystarczy sprawdzić czy pozostałe liczby (tylko 7) są mniejsze czy większe od 3 i ustawić je odpowiednio po lewo lub prawo od 3.



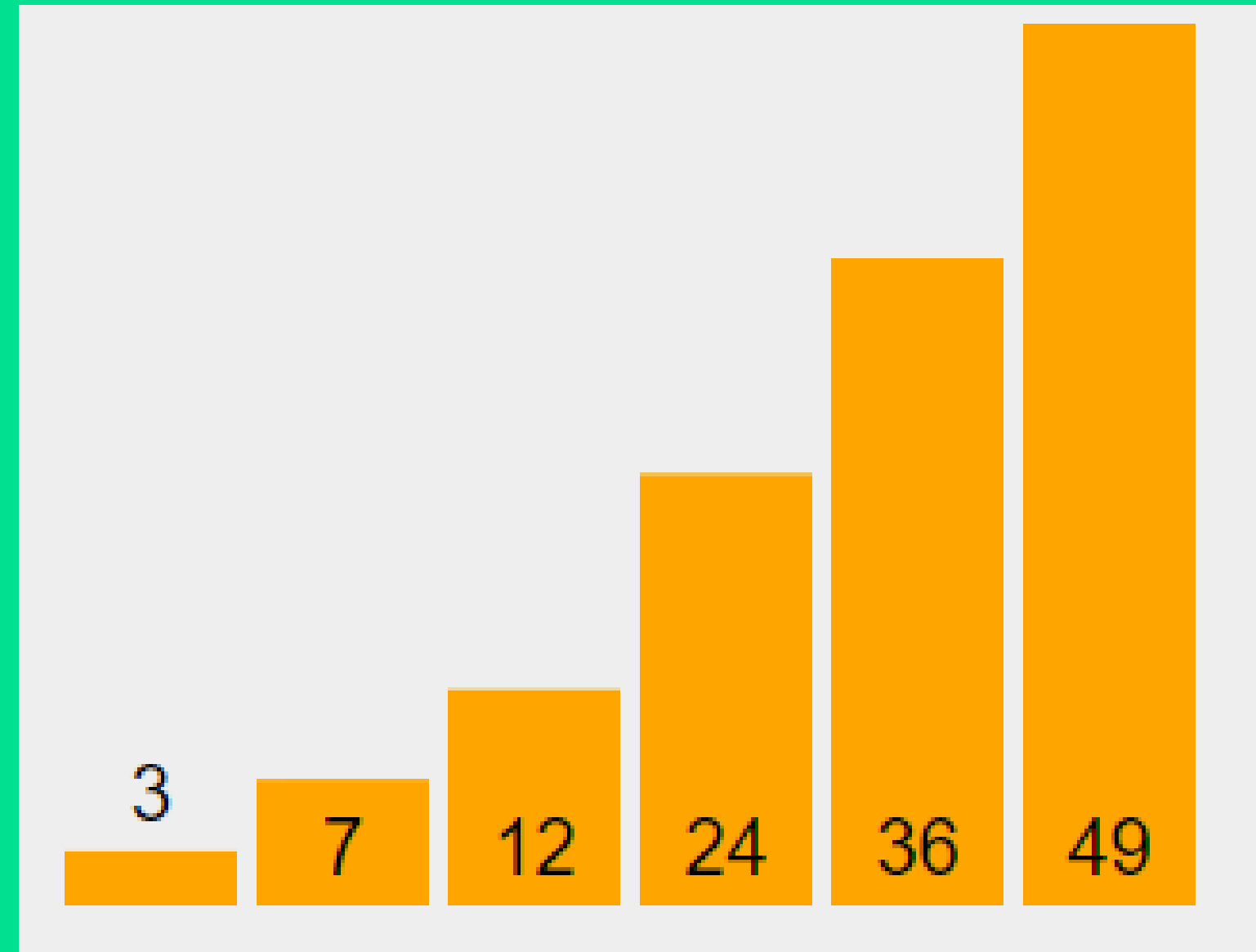
Do posortowania została
ostatnia grupa - liczby
większe od 24.

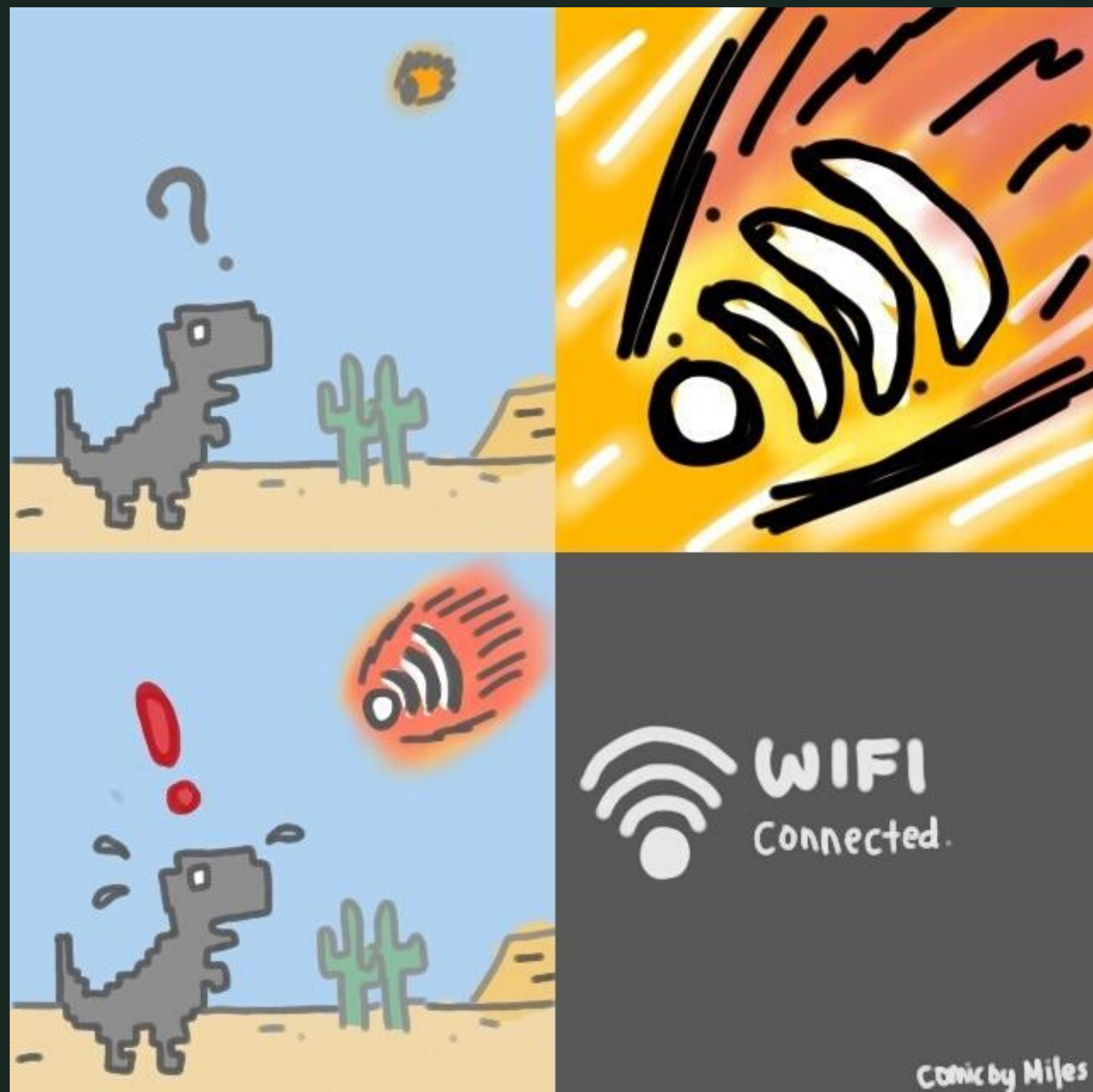


Obiera 49 jako element rozdzielający, sprawdza czy 36 jest mniejsze czy większe od 49 i ustawia 36 po odpowiedniej stronie.



Po wykonaniu tego kroku
liczby są posortowane.





**DZIĘKUJEMY ZA
UWAGĘ**