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

OPIS SŁOWNY

Po wczytaniu danych wejściowych **a i b** porównaj wprowadzone liczby.
Jeśli **a < b**, to **min = a**. Wypisz wynik. Jeśli **a >= b**, to sprawdź czy **b < a**. Jeśli tak, to **min = b**. Wypisz wynik. W przeciwnym przypadku **min = a = b**. Wypisz wynik.

Algorytm wybierający liczbę mniejszą z podanych dwóch liczb

LISTA KROKÓW

Obliczanie sumy dwóch liczb

Dane wejściowe: dwie dowolne liczby a, b.
Dane wyjściowe: liczba suma, będąca sumą liczb a i b.

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

SCHEMAT BLOKOWY

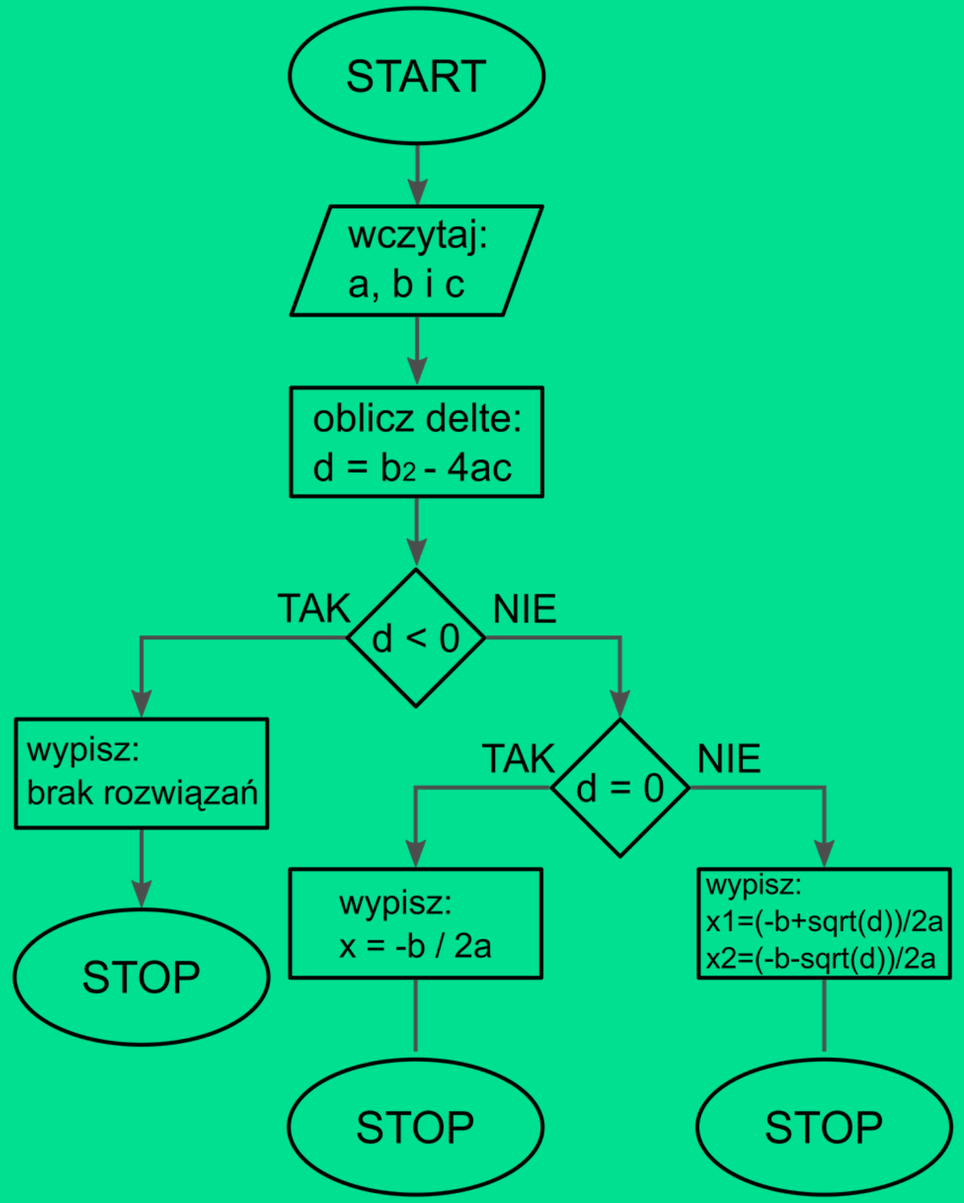
Uniwersalny schemat naprawy

Czy się rusza?

- nie
  - a powinno?
    - nie
      - więc nie ma problemu
    - tak

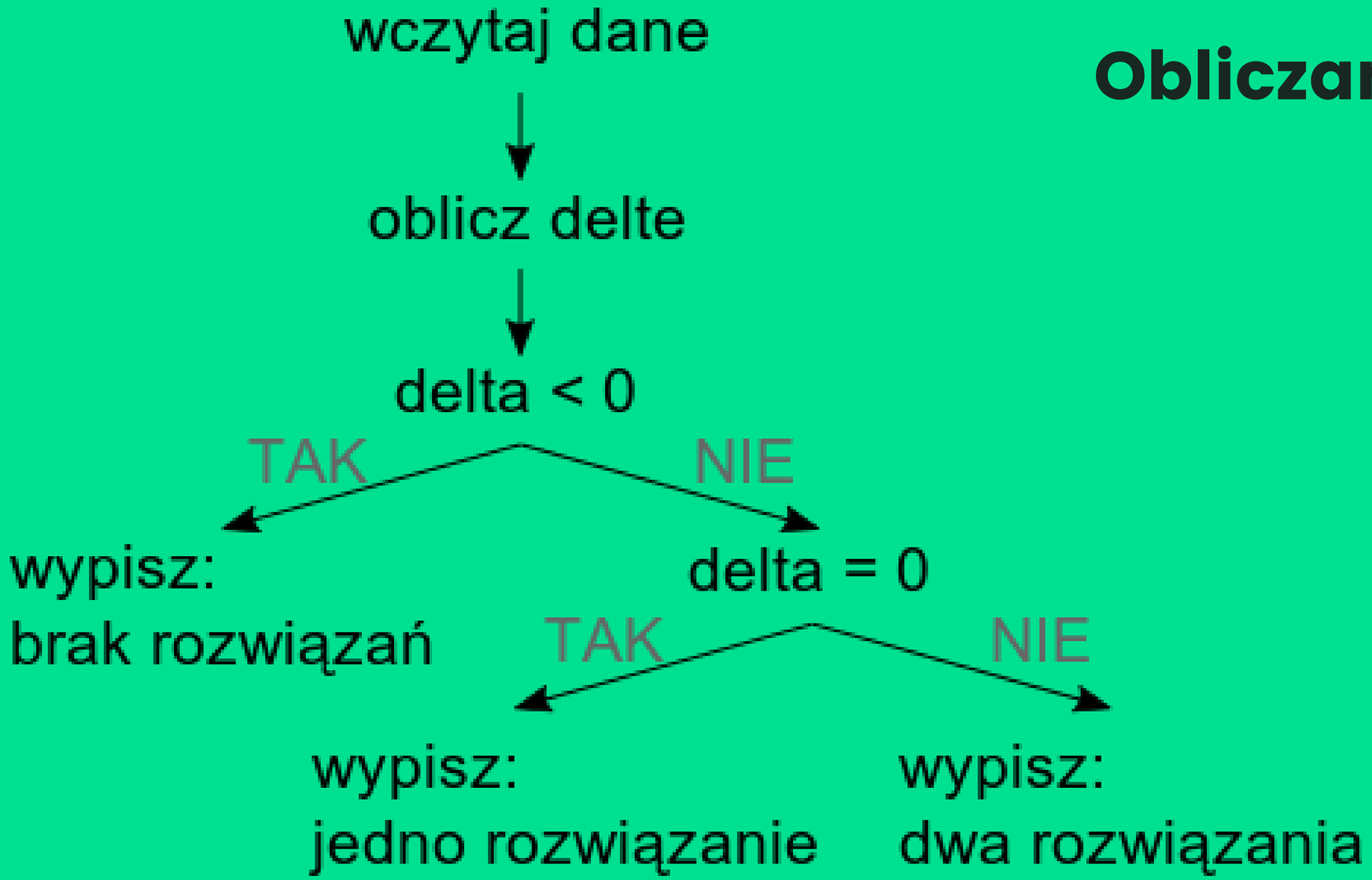
- tak
  - a powinno?
    - tak
      - więc nie ma problemu
    - nie

SCHEMAT BLOKOWY



Obliczanie delty

DRZEWO ALGORYTMU



Obliczanie delty

PSEUDOKOD

Sumowanie n liczb

```
Start
    Suma:=0
    Podaj(n)
    i:=0
    Dopóki i<n wykonuj:
        Wczytaj(a)
        Suma := Suma + a
        i := i + 1
    Wypisz(Suma)
Koniec
```

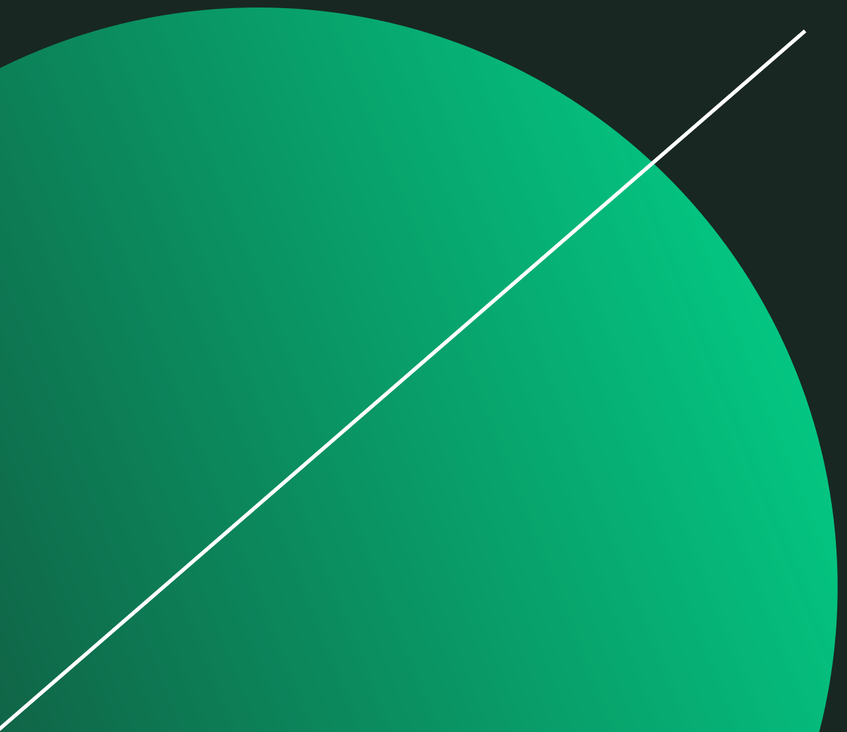
JĘZYK PROGRAMOWANIA

```
#include<iostream>
using namespace std;
int main ()
{
   int i, j,temp,pass=0;
   int a[10] = {10,2,0,14,43,25,18,1,5,45};
   cout <<"Input list ...\n";
   for(i = 0; i<10; i++) {
      cout <<a[i]<<"\t";
   }
cout<<endl;
for(i = 0; i<10; i++) {
   for(j = i+1; j<10; j++)
   {
      if(a[j] < a[i]) {
         temp = a[i];
         a[i] = a[j];
         a[j] = temp;
      }
   }
pass++;
}
cout <<"Sorted Element List ...\n";
for(i = 0; i<10; i++) {
   cout <<a[i]<<"\t";
}
cout<<"\nNumber of passes taken to sort the list:"<<pass<<endl;
return 0;
}
```

Source: softwaretestinghelp.com

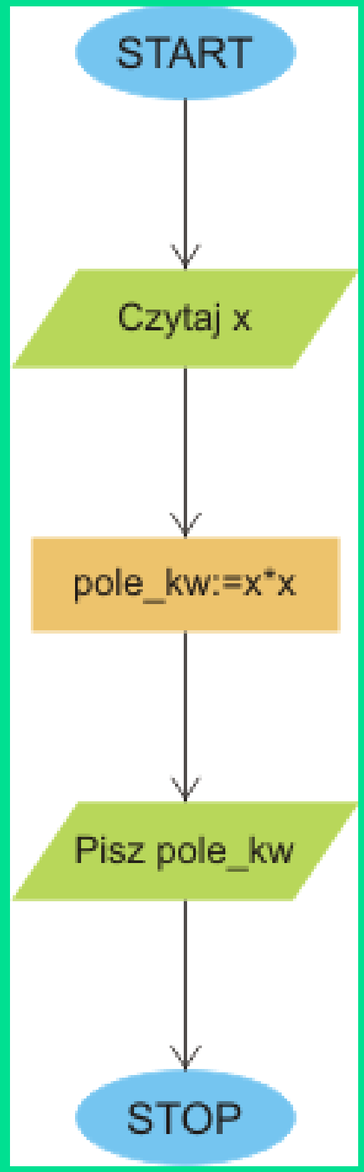
Przykładowe sortowanie bąblekowe w C++

Klasyfikacja algorytmów ze względu na sposób wykonywania operacji



SEKWENCYJNE

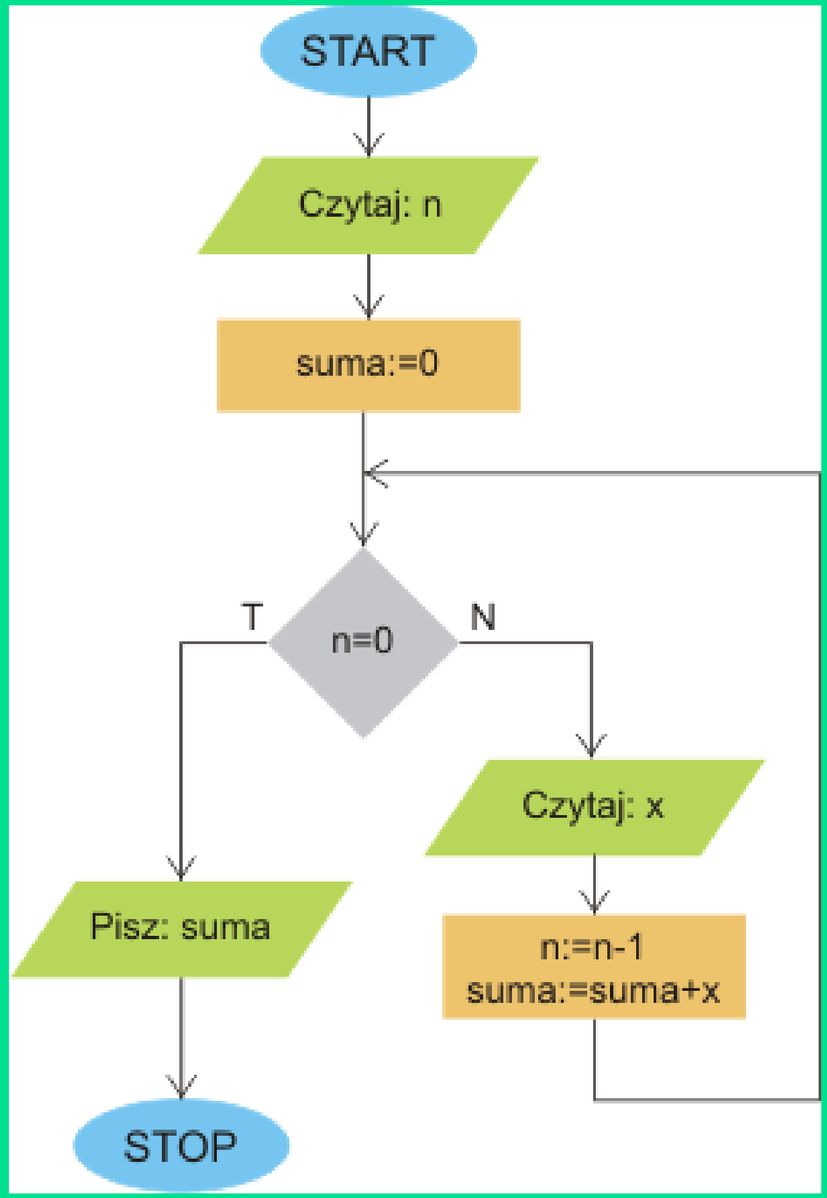
operacje w algorytmie wykonywane są w kolejności, w jakiej zostały opisane



Source

ITERACYJNE

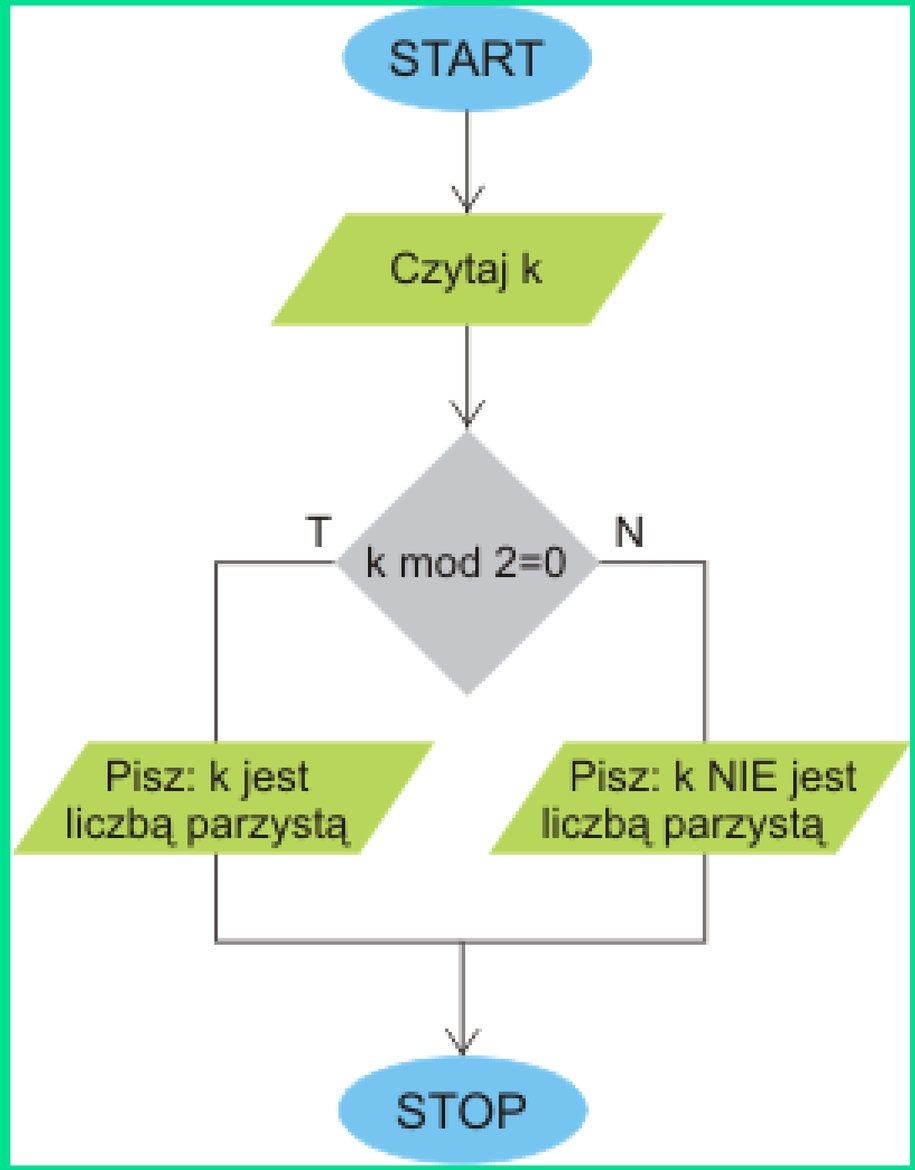
niektóre kroki są powtarzane, aż do spełnienia wymaganego warunku



Source

WARUNKOWE

występują w nich instrukcje warunkowe



Source

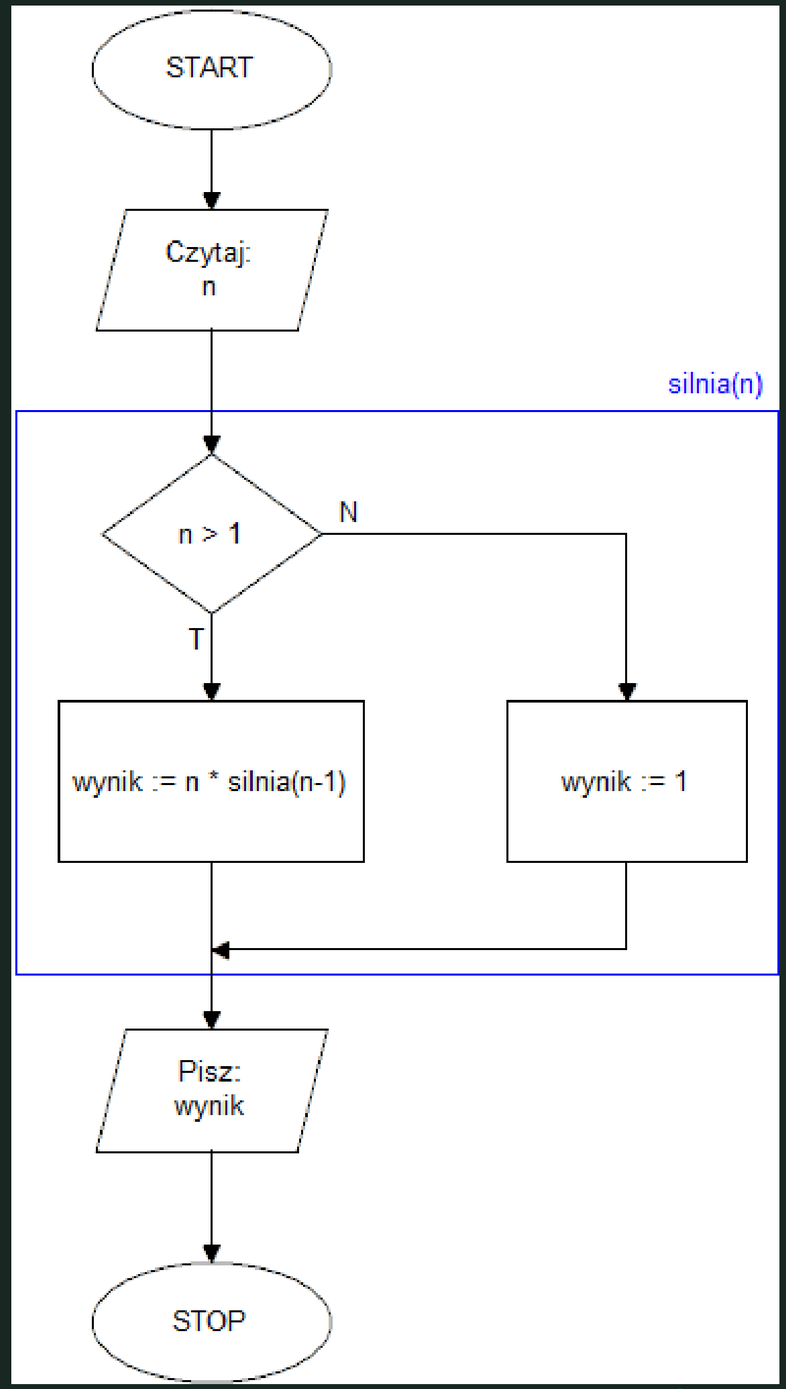
REKURENCYJNE

REKURENCYJNE

REKURENCYJNE

REKURENCYJNE

tworzona jest formuła odwołująca się do niej samej

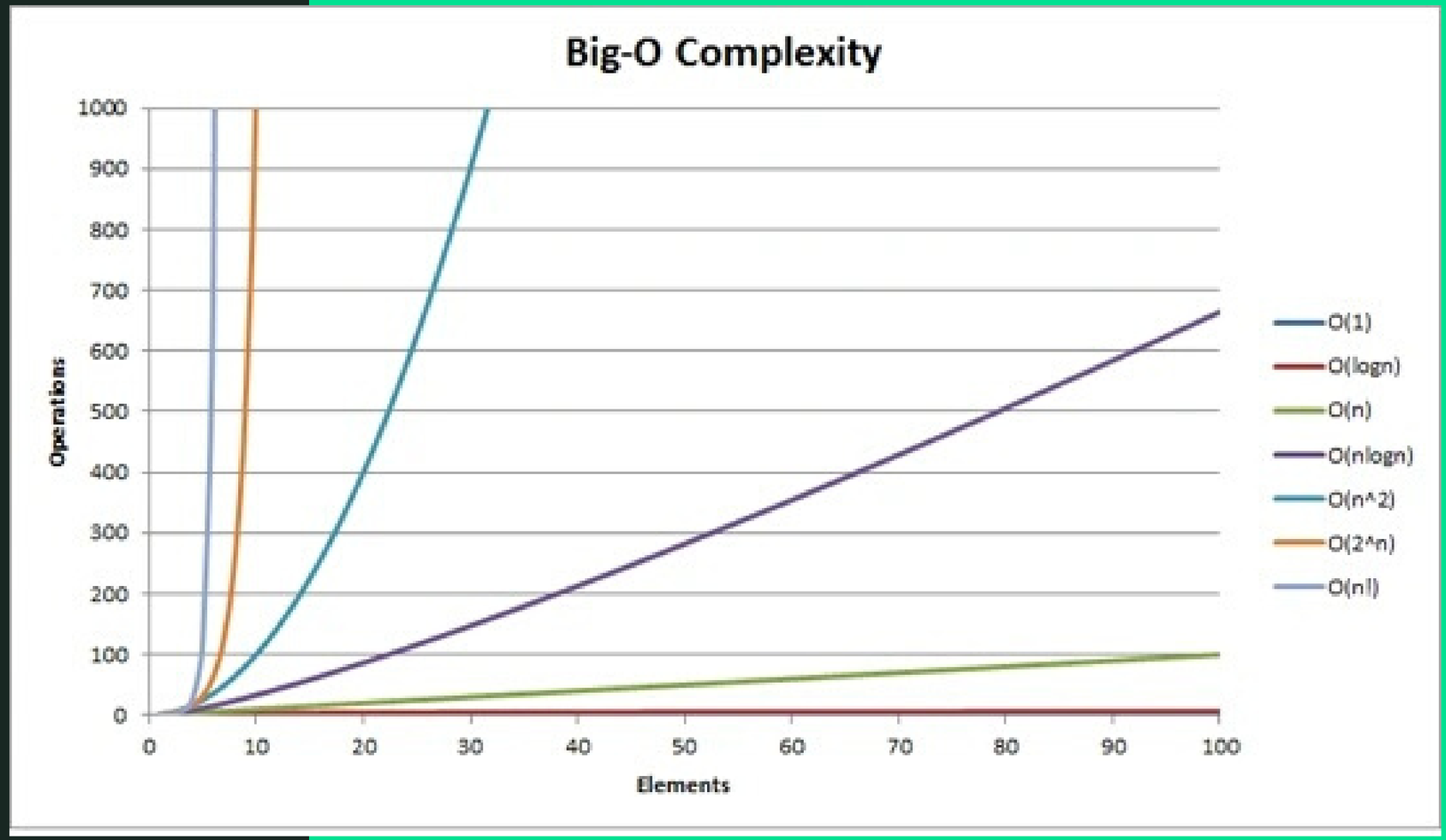


Source

Złożeność 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 podpoblemów i otrzymuje rozwiązanie problemu pierwotnego.

Symulacja działania algorytmu Quicksort

DZIĘKUJEMY ZA UWAGĘ