Technologie cyfrowe

Artur Kalinowski

Zakład Cząstek i Oddziaływań Fundamentalnych

Pasteura 5, pokój 4.15

Artur.Kalinowski@fuw.edu.pl

Semestr letni 2014/2015

Algorytmy

Skąd komputer wie co zrobić kiedy mu każemy wykonać jakieś zadanie?

Programy komputerowe → Algorytmy

Algorytm: złożony z kroków przepis na wykonanie jakieś czynności. W obszarze cyfrowym na wykonanie obliczeń lub operacji na danych.

Algorytmika: nauka o algorytmach.

Jaka jest miara „dobroci" algorytmu?

Kategorie algorytmów: które algorytmy są „szybkie", które „wolne"?

Metody czytelnego opisu algorytmów.

Wszystkie operacje wykonywane przez urządzenia cyfrowe realizują jakieś algorytmy.

Algorytmy

Zadanie algorytmiczne: charakteryzacja dopuszczalnych danych wejściowych, oraz pożądanego wyniku. To samo zadanie może być wykonane przy użyciu różnych algorytmów.

Dane wejściowe: specyfikacja zestawu informacji potrzebnych do wykonania konkretnego algorytmu, zwykle w postaci listy i rodzaju zmiennych wymaganych dla wykonania algorytmu

Zmienna: ”miejsce” na konkretne dane. Zmienna ma swoją nazwę, typ i wartość, np:

```
int x;
x = 3;
x = 2+5;
```

Wynik: specyfikacja zestawu informacji których oczekujemy po wykonaniu algorytmu.

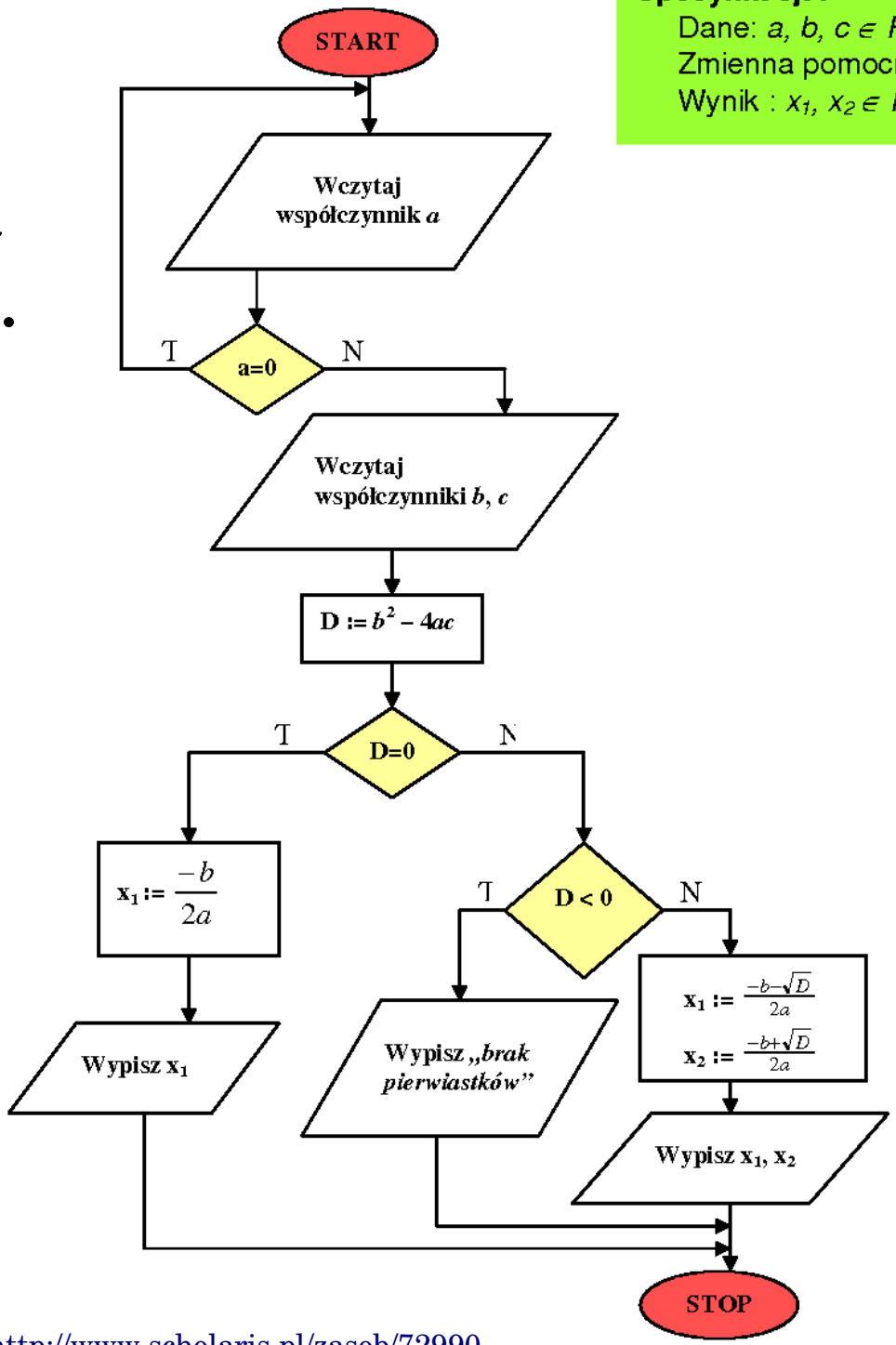
Algorytm rozwiązywania równania kwadratowego.

Specyfikacja:
Dane: $a, b, c \in R \quad a \neq 0$
Zmienna pomocnicza: $D \in R$
Wynik : $x_1, x_2 \in R$

- Dane wejściowe: trzy liczby (a,b,c) definiujące równanie: $a \bullet x^2 + b \bullet x + c = 0$

- Wynik: dwa rozwiązania równania.

Algorytm musi działać dla wszystkich dozwolonych wartości danych wejściowych.



http://www.scholaris.pl/zasob/72990

Struktury sterujące

Struktury sterujące: elementy algorytmu zawiadujące kolejnością wykonywania instrukcji.

”jeśli Q”: jeżeli zadanie logiczne, oznaczone symbolem Q jest prawdziwe, np: Q – „x>0”

bezpośrednie następstwo: “wykonaj A potem B”

wybór warunkowy: ”jeśli Q to wykonaj A, w przeciwnym razie wykonaj B”

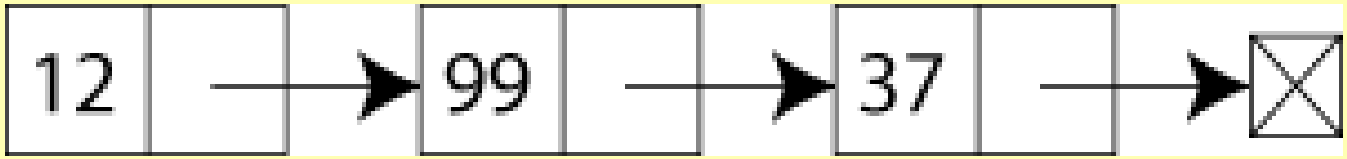
iteracja ograniczona: “wykonaj A dokładnie N razy”

iteracja warunkowa: “dopóki Q, wykonuj A”

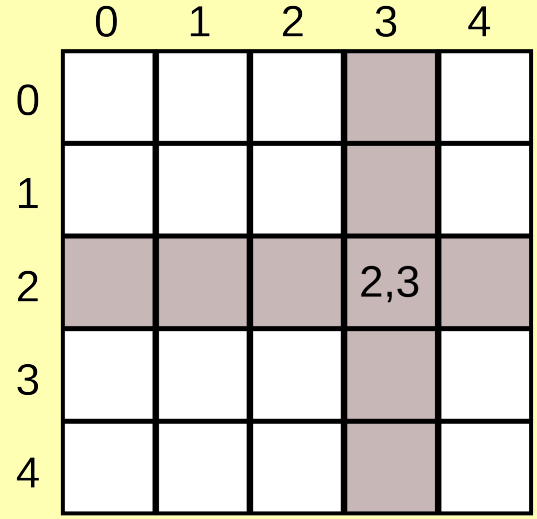
Struktury danych

Struktury danych: sposób organizacji informacji wewnątrz algorytmu

lista (macierz jednowymiarowa): zestaw podstawowych elementów, np. liczb. Elementy listy są uporządkowane. Można wskazać element pierwszy, następny, ostatni. Kolejne elementy są numerowane indeksem listy.

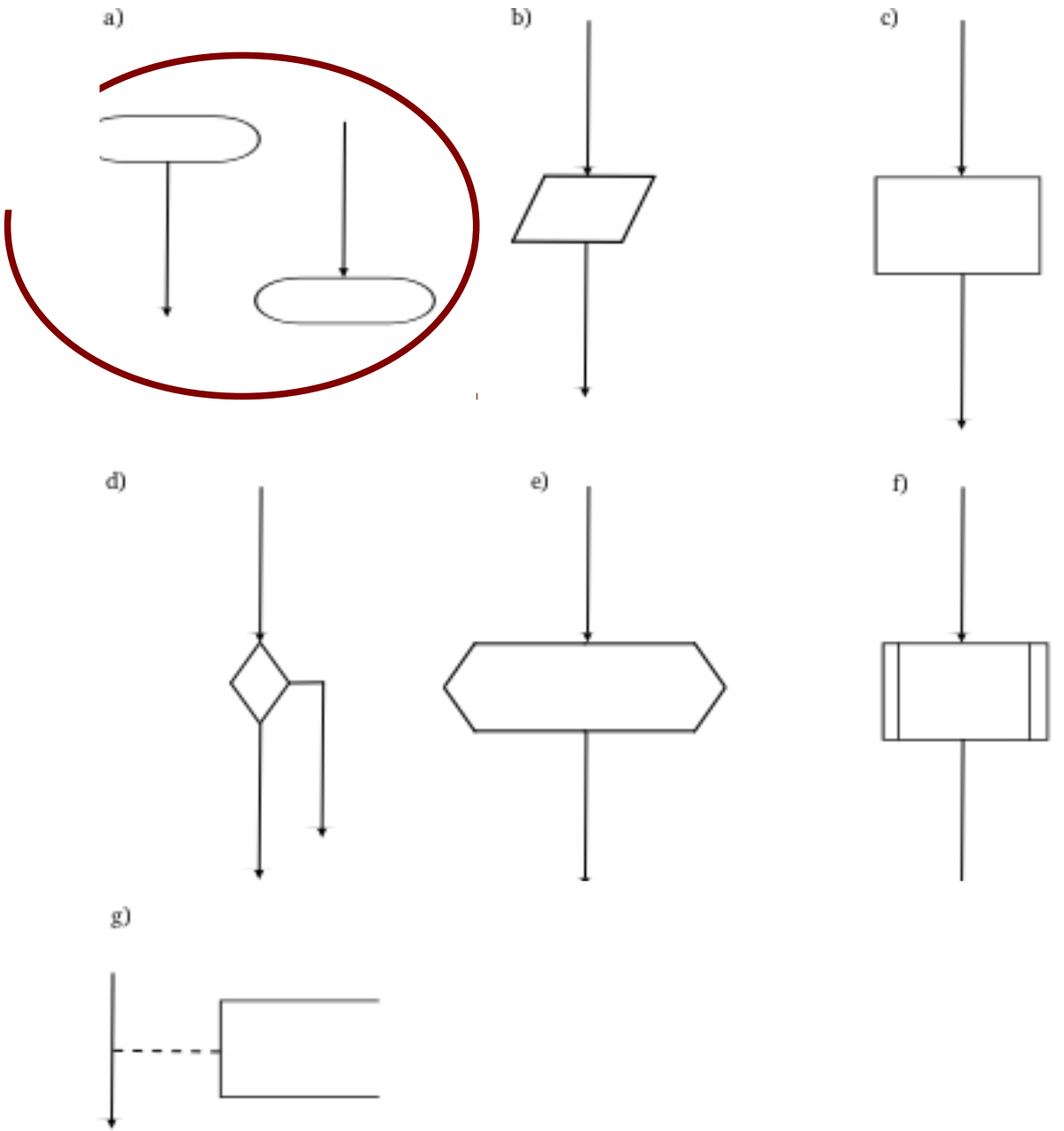


macierz (tablica): zestaw podstawowych elementów, np. liczb ułożonych w postaci wielowymiarowych struktur. Najczęstszy wariant: macierz dwuwymiarowa



Schematy blokowe

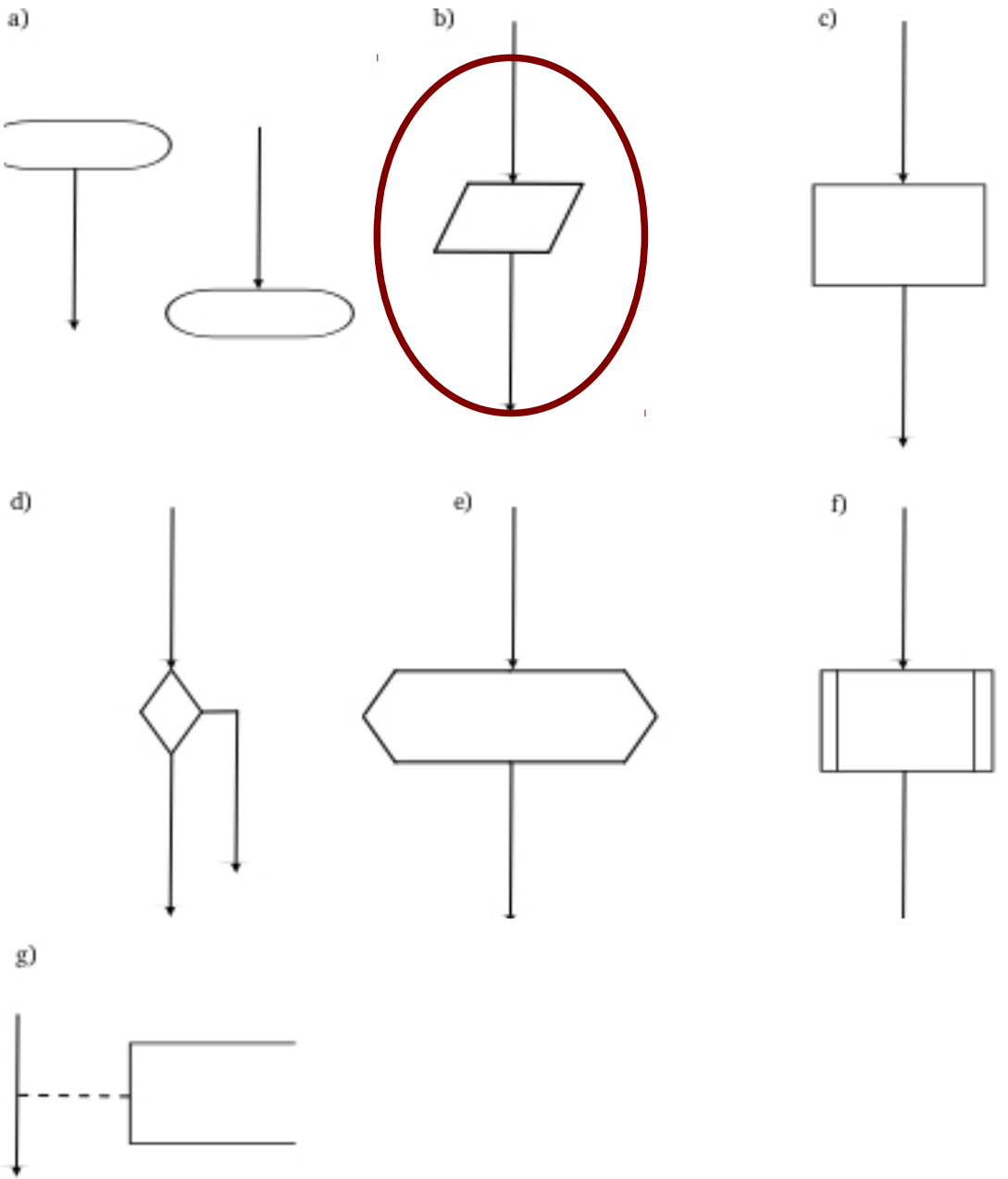
- blok graniczny: początek/koniec działania algorytmu



Schematy blokowe

- blok graniczny: początek/koniec działania algorytmu

- blok wejścia/wyjścia, np. “wczytaj liczbę z klawiatury”

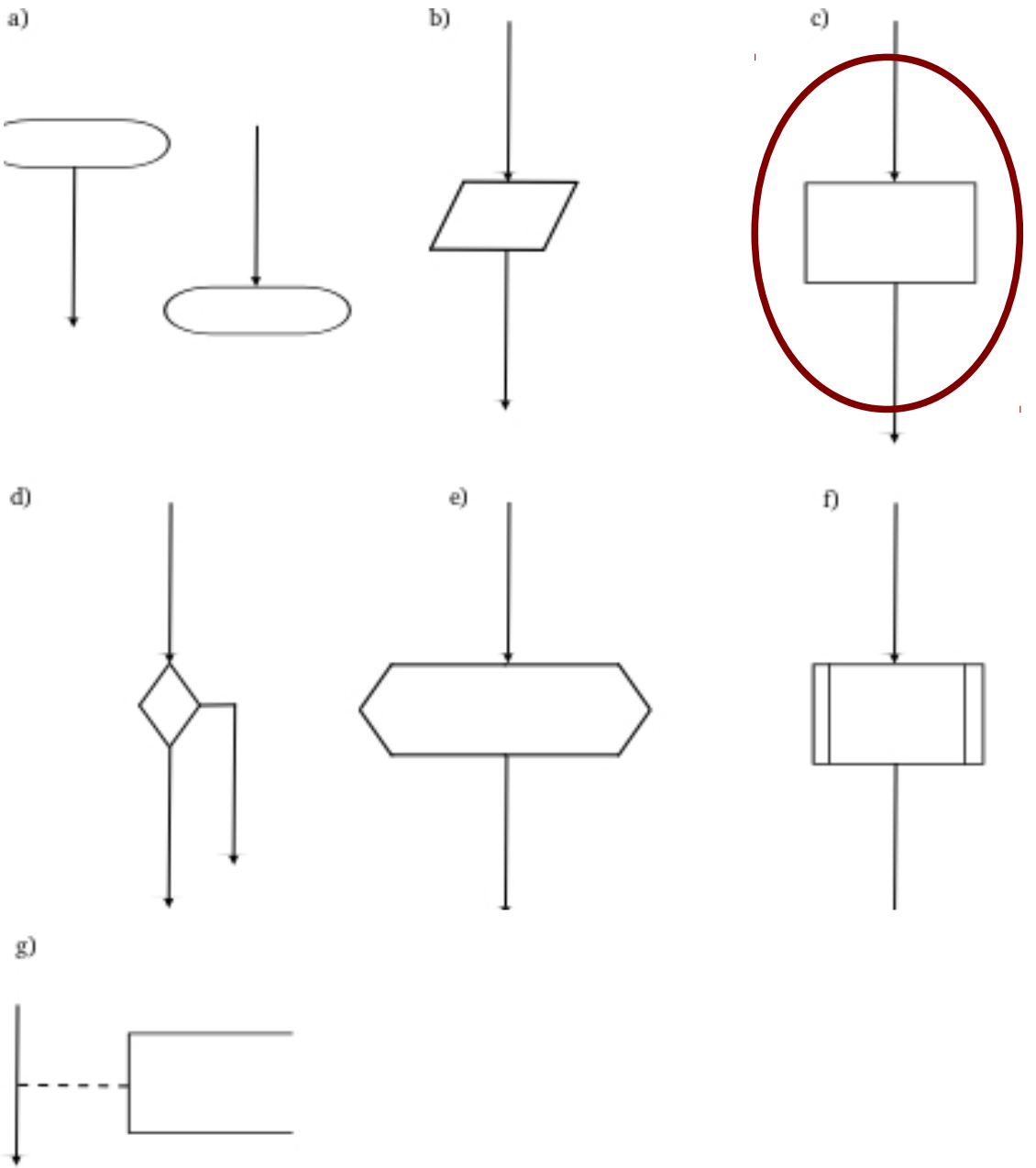


Schematy blokowe

- blok graniczny: początek/koniec działania algorytmu

- blok wejścia/wyjścia, np. “wczytaj liczbę z klawiatury”

- blok operacyjny, np. “dodaj 1 do zmiennej x”



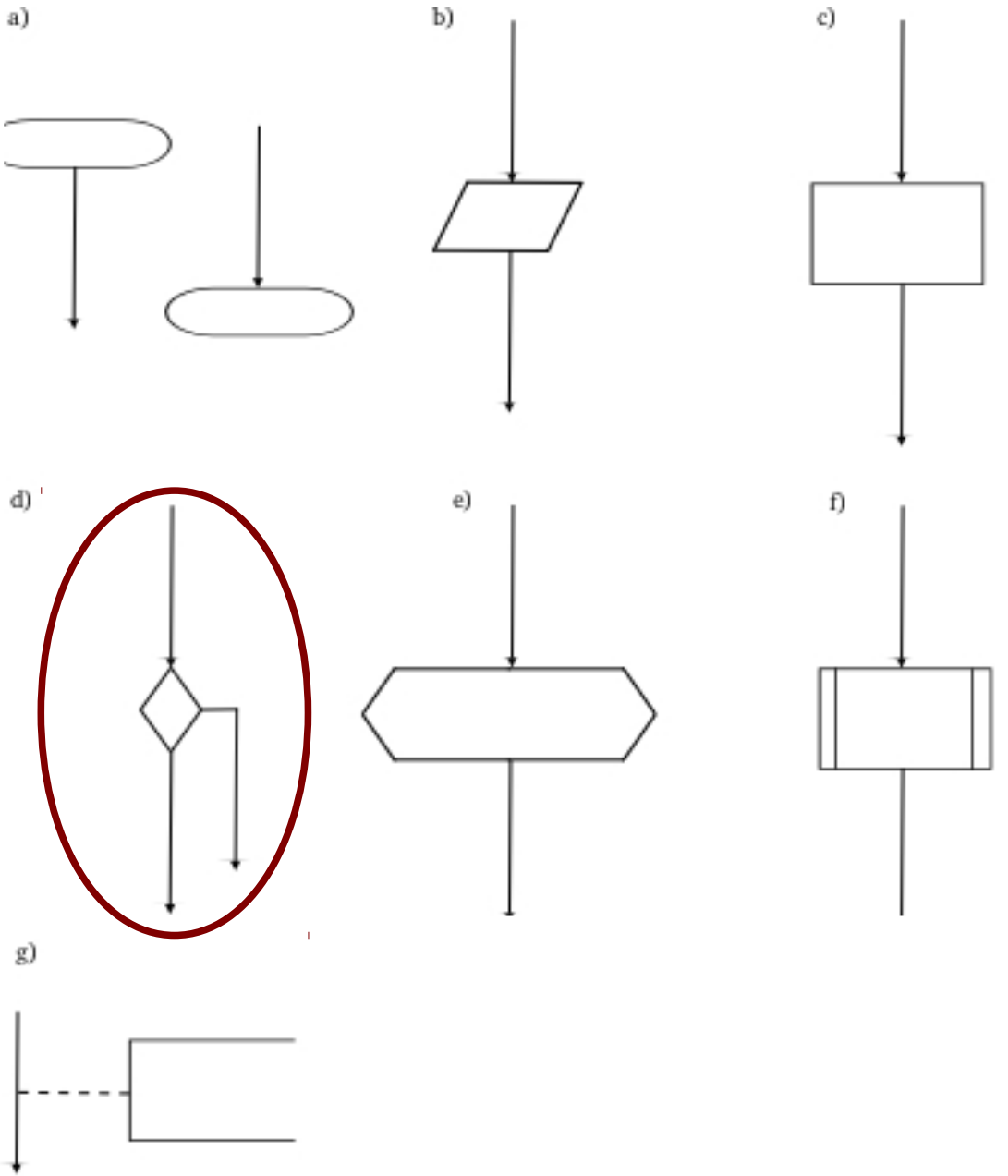
Schematy blokowe

- blok graniczny: początek/koniec działania algorytmu

- blok wejścia/wyjścia, np. “wczytaj liczbę z klawiatury”

- blok operacyjny, np. “dodaj 1 do zmiennej x”

- blok warunkowy, np. “jeśli x>3 to ..., jeśli nie to...”



Schematy blokowe

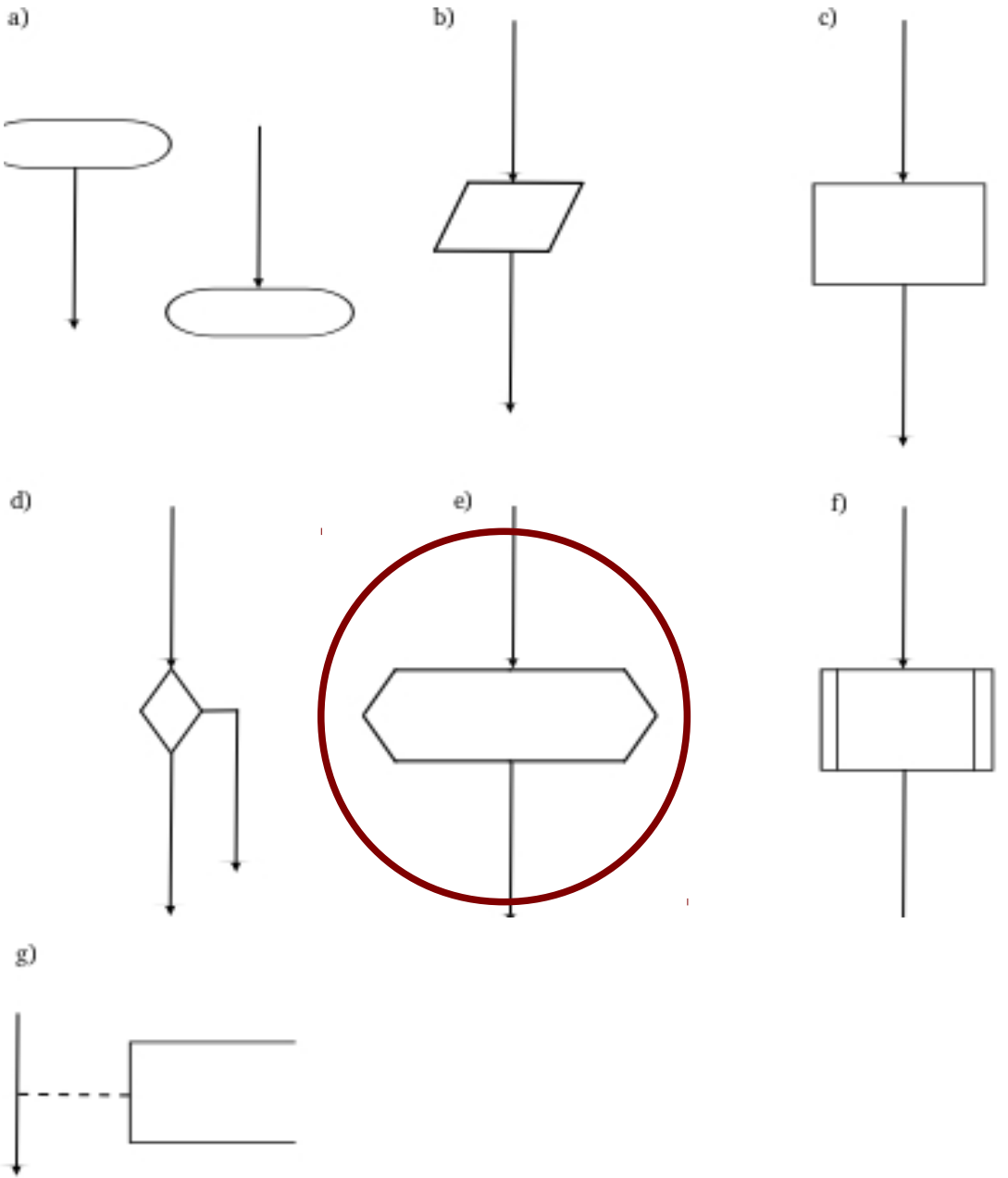
- blok graniczny: początek/koniec działania algorytmu

- blok wejścia/wyjścia, np. “wczytaj liczbę z klawiatury”

- blok operacyjny, np. “dodaj 1 do zmiennej x”

- blok warunkowy, np. “jeśli x>3 to ..., jeśli nie to...”

- blok wywołania podprogramu



Schematy blokowe

- blok graniczny: początek/koniec działania algorytmu

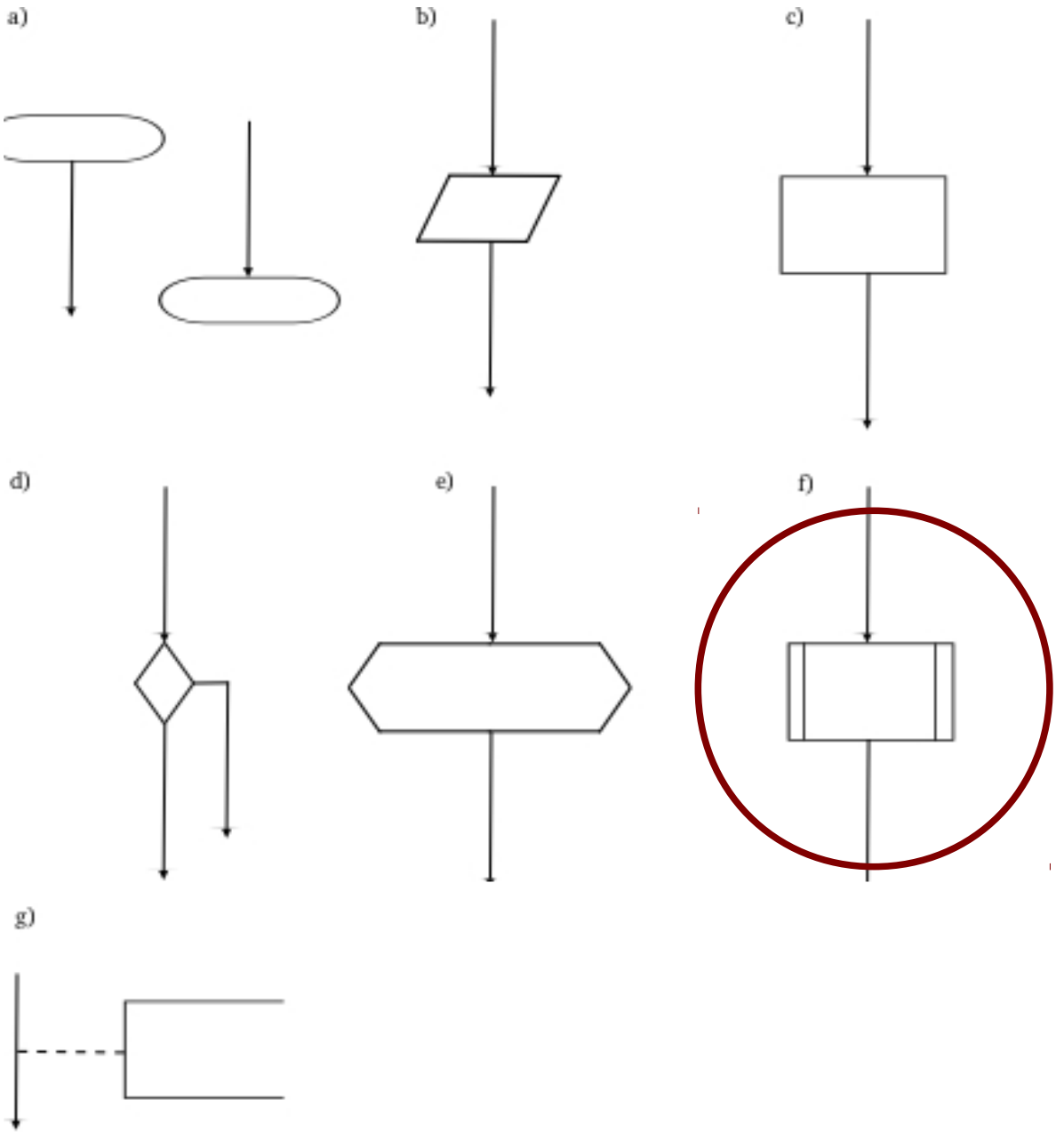
- blok wejścia/wyjścia, np. “wczytaj liczbę z klawiatury”

- blok operacyjny, np. “dodaj 1 do zmiennej x”

- blok warunkowy, np. “jeśli x>3 to ..., jeśli nie to...”

- blok wywołania podprogramu

- blok fragmentu



Schematy blokowe

- blok graniczny: początek/koniec działania algorytmu

- blok wejścia/wyjścia, np. “wczytaj liczbę z klawiatury”

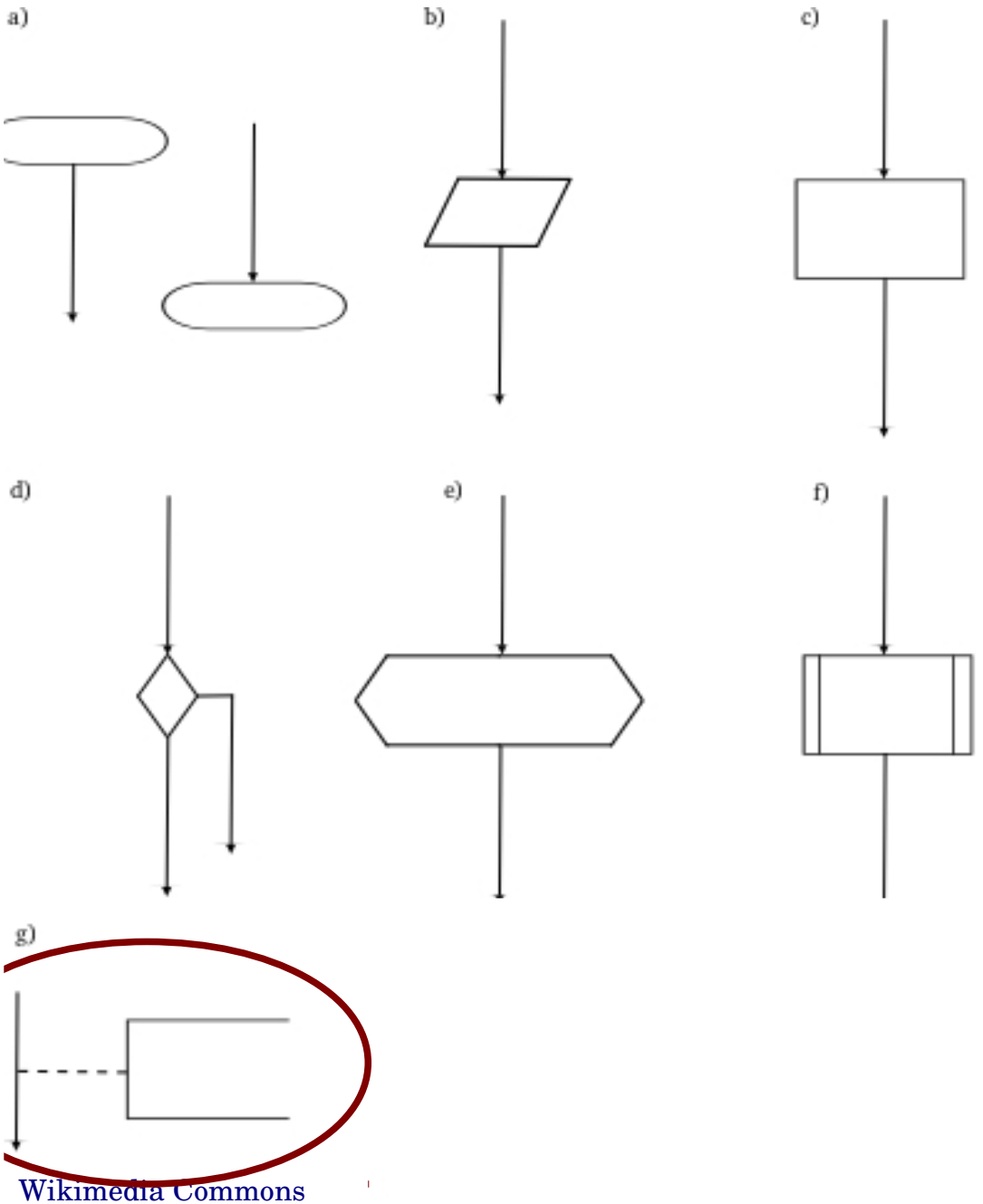
- blok operacyjny, np. “dodaj 1 do zmiennej x”

- blok warunkowy, np. “jeśli x>3 to ..., jeśli nie to...”

- blok wywołania podprogramu

- blok fragmentu

- blok komentarza

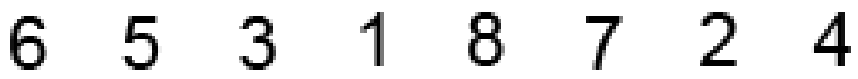


Zadanie algorytmiczne: sortowanie

dane wejściowe: nieuporządkowana lista elementów, które można porównywać ze sobą, tzn. stwierdzić który jest „większy/mniejszy (wcześniejszy/późniejszy)”

wynik: uporządkowana lista

Sortowanie bąbelkowe:

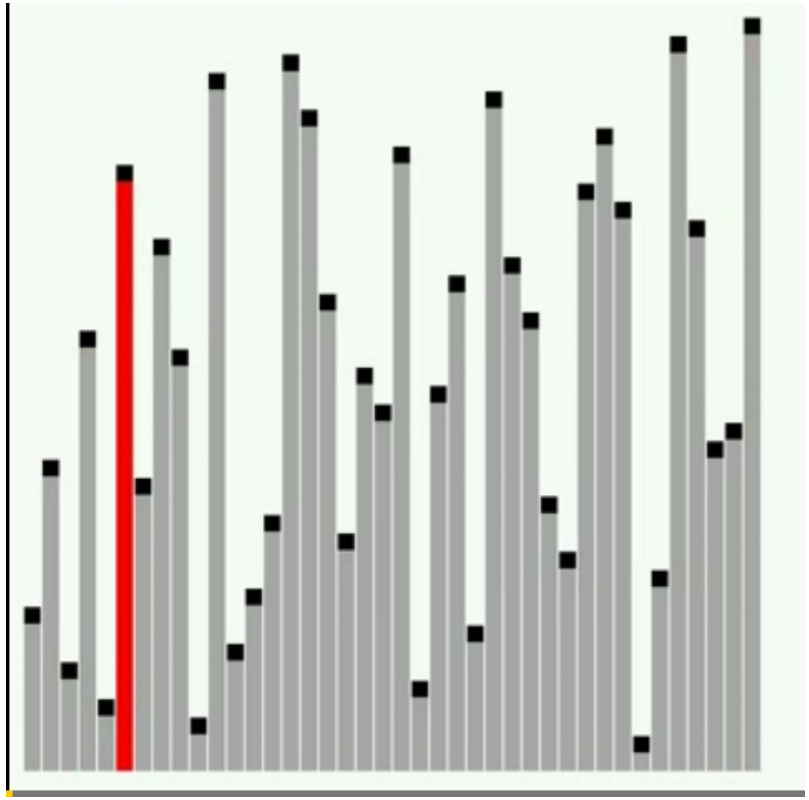


Zadanie algorytmiczne: sortowanie

dane wejściowe: nieuporządkowana lista elementów, które można porównywać ze sobą, tzn. stwierdzić który jest „większy/mniejszy (wcześniejszy/późniejszy)”

wynik: uporządkowana lista

Sortowanie bąbelkowe:



Zadanie algorytmiczne: sortowanie

dane wejściowe: nieuporządkowana lista elementów, które można porównywać ze sobą, tzn. stwierdzić który jest „większy/mniejszy (wcześniejszy/późniejszy)”

wynik: uporządkowana lista

Sortowanie listy [4, 2, 5, 1, 7]:

Pierwsza iteracja: $[\underbrace{4,2}_{4>2},5,1,7] \rightarrow [2,\underbrace{4,5}_{4<5},1,7] \rightarrow [2,4,\underbrace{5,1}_{5>1},7] \rightarrow [2,4,1,\underbrace{5,7}_{5<7}]$

Druga iteracja: $[\underbrace{2,4}_{2<4},1,5,7] \rightarrow [2,\underbrace{4,1}_{4>1},5,7] \rightarrow [2,1,\underbrace{4,5}_{4<5},7]$

Trzecia iteracja: $[\underbrace{2,1}_{2>1},4,5,7] \rightarrow [1,\underbrace{2,4}_{2<4},5,7]$

Czwarta iteracja: $[\underbrace{1,2}_{1<2},4,5,7]$

http://pl.wikipedia.org/wiki/Sortowanie_bąbelkowe

Zadanie algorytmiczne: sortowanie

Dane wejściowe: nieposortowana lista P o długości N

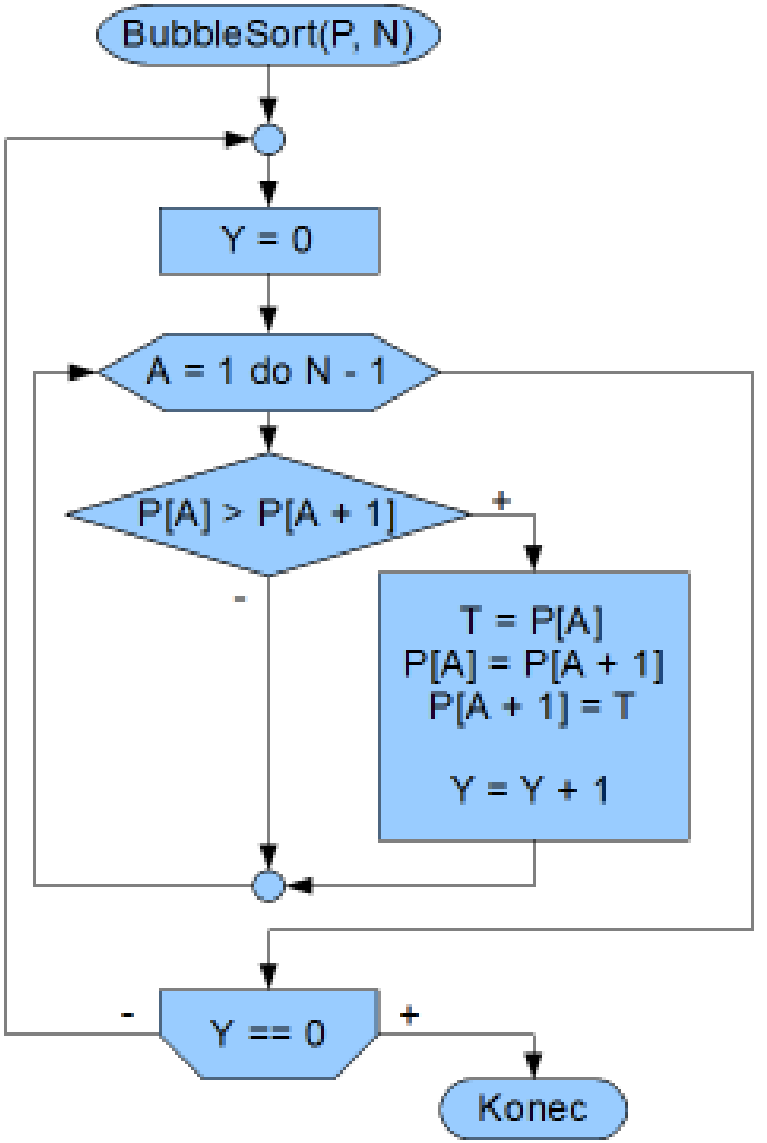
Wynik: posortowana lista P o długości N

Zmienne pomocnicze:

A – licznik iteracji

T – zmienna tymczasowa, przechowuje wartości zamienianych elementów

Y – zmienna pomocnicza, wskazujący czy były wykonane zamiany elementów.



http://programujte.com/clanek/2012011800-vyvojove-diagramy-selection-insert-a-bubble-sort-21-dil/

Zadanie algorytmiczne: sortowanie

Długość listy: N

Liczba iteracji: N-1

Liczba porównań w i-tej iteracji: N-i

Całkowita liczba operacji: (N-1)x(N-i)~ N^2

Pierwsza iteracja: $\underbrace{[4,2}_{4>2},5,1,7] \rightarrow [2,\underbrace{4,5}_{4<5},1,7] \rightarrow [2,4,\underbrace{5,1}_{5>1},7] \rightarrow [2,4,1,\underbrace{5,7}_{5<7}]$

Druga iteracja: $\underbrace{[2,4}_{2<4},1,5,7] \rightarrow [2,\underbrace{4,1}_{4>1},5,7] \rightarrow [2,1,\underbrace{4,5}_{4<5},7]$

Trzecia iteracja: $\underbrace{[2,1}_{2>1},4,5,7] \rightarrow [1,\underbrace{2,4}_{2<4},5,7]$

Czwarta iteracja: $\underbrace{[1,2}_{1<2},4,5,7]$

http://pl.wikipedia.org/wiki/Sortowanie_bąbelkowe

Zadanie algorytmiczne: sortowanie

Sortowanie przez scalanie (ang. merge sort): podziel listę na coraz to mniejsze elementy aż dojdziesz do list o długości 1.
Następnie połącz listy zachowując odpowiednią kolejność.

Liczba operacji porównania: N logN

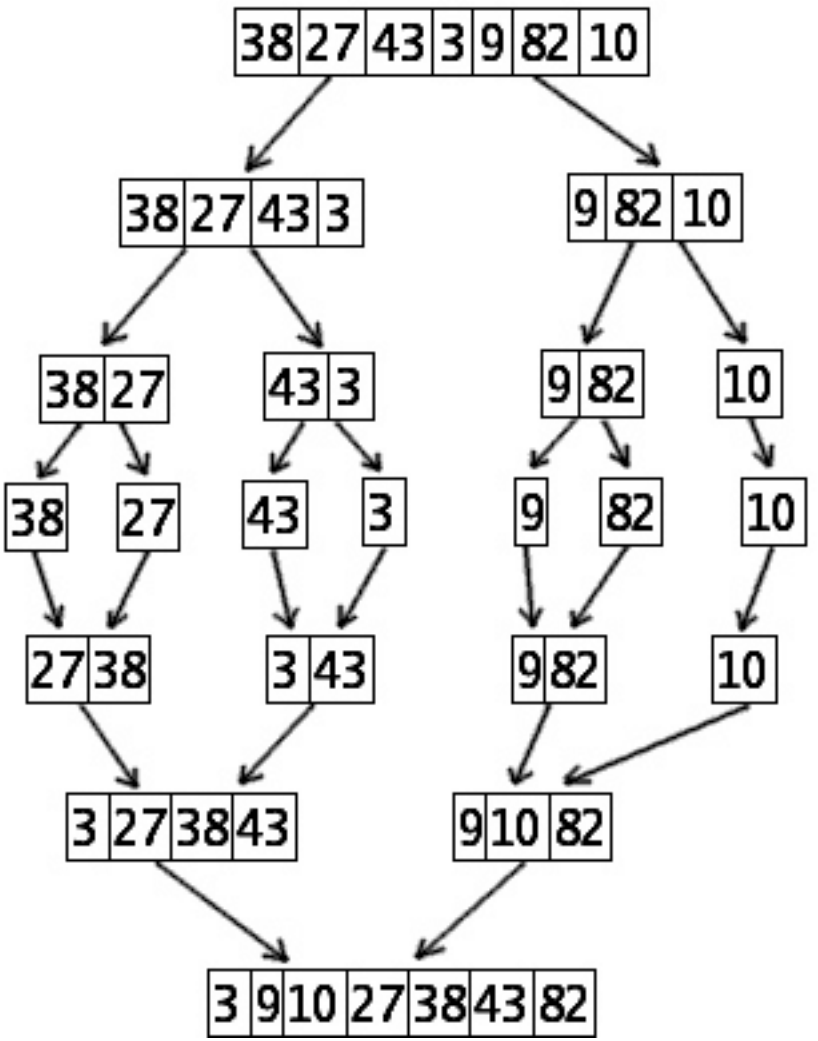
Dla N = 10000:

N^2 – 100 000 000

N $\log_{10}$N – 40 000

N^2/ N $\log_{10}$N = N / $\log_{10}$N

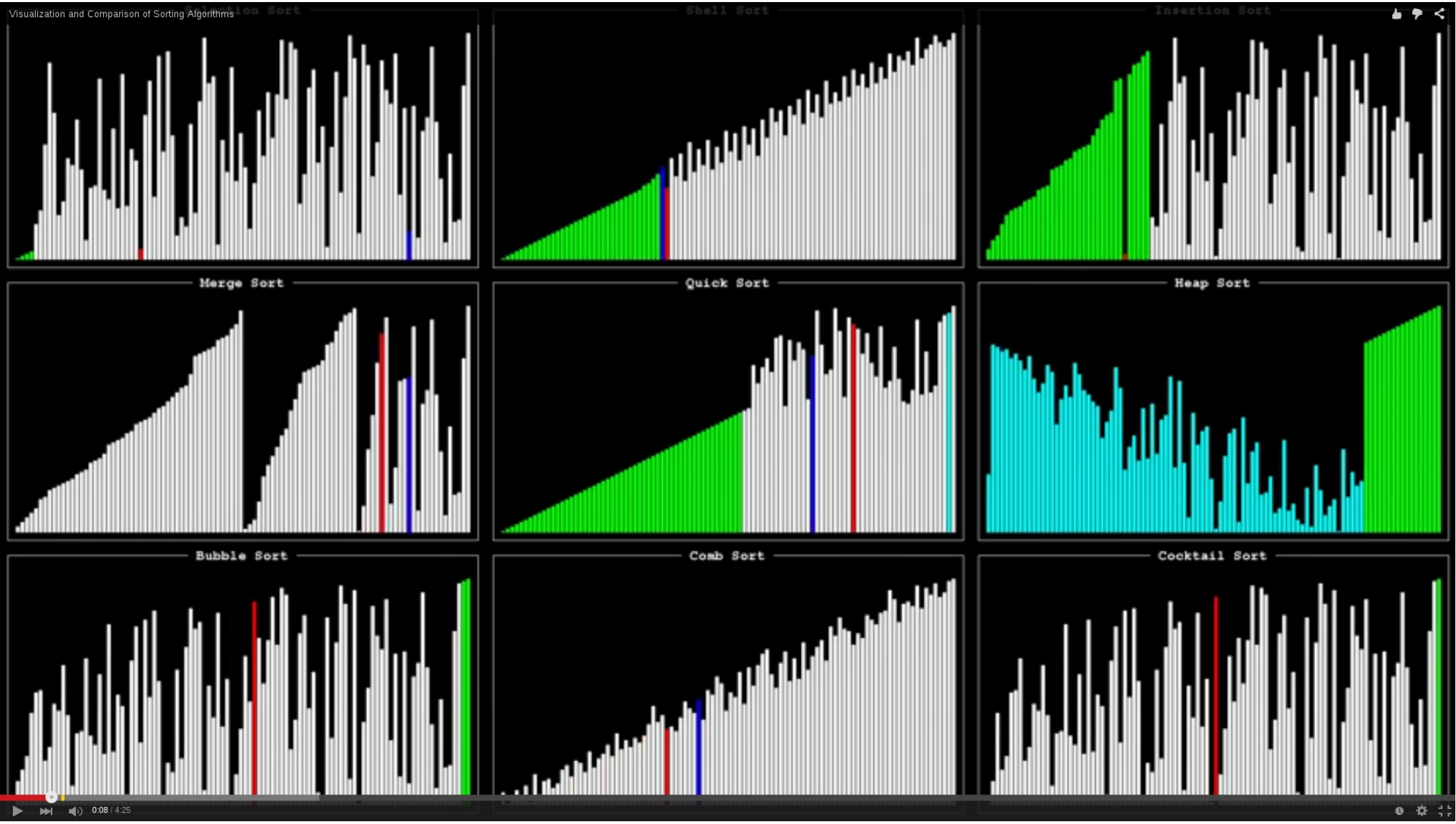
100 000 000/40 000 = 2500



Wikipedia Commons

Lista obecności

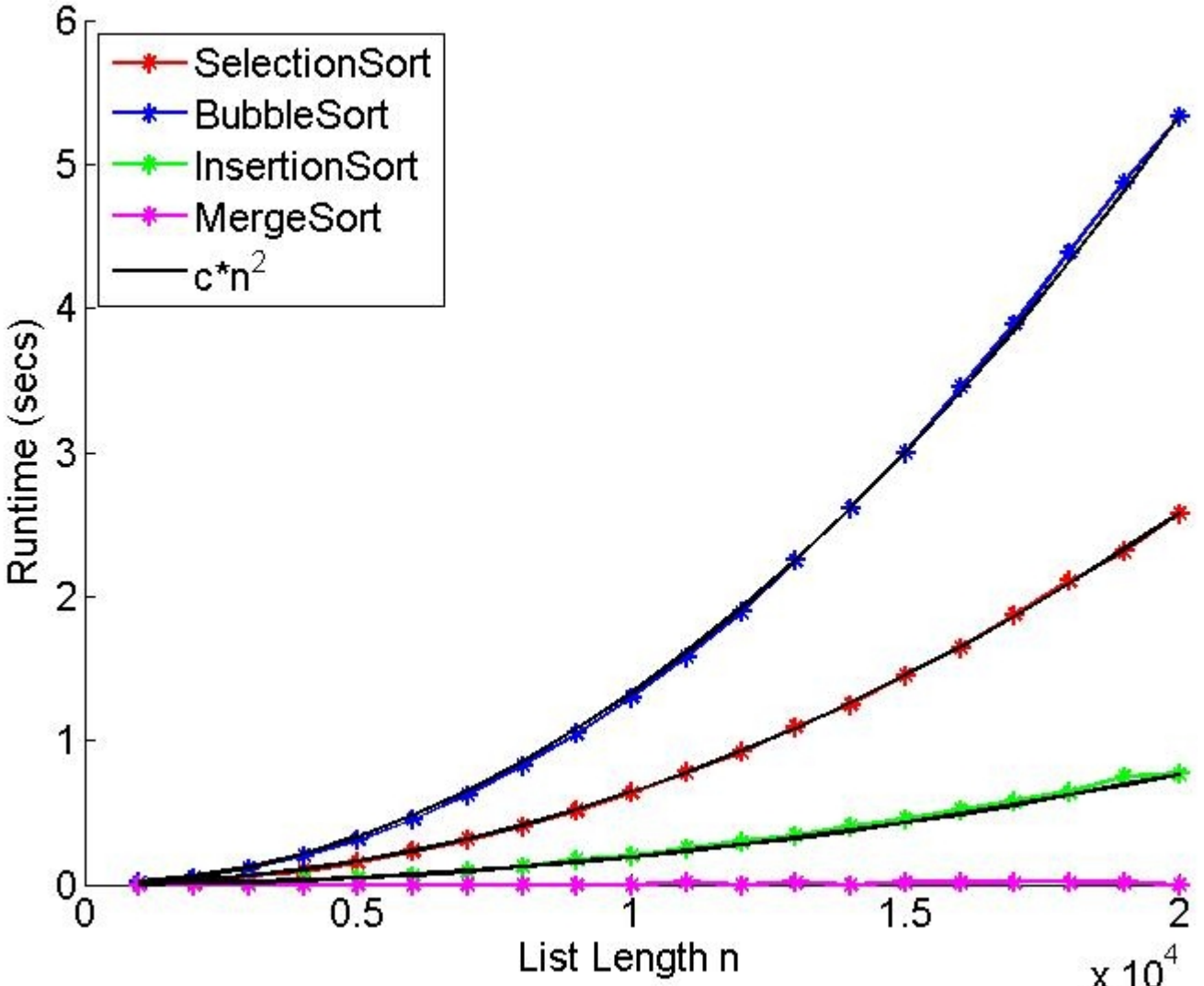
Zadanie algorytmiczne: sortowanie



Zadanie algorytmiczne: sortowanie

Złożoność obliczeniowa: ile operacji trzeba wykonać w czasie działania algorytmu w zależności od rozmiaru danych wejściowych.

algorytm sortowania bąbelkowego: złożoność obliczeniowa $O(N^2)$, co oznacza, że czas wykonywania algorytmu rośnie kwadratowo z długością listy. $O(N^2)$ to **złożoność wielomianowa**



http://www.cs.toronto.edu/~jepson/csc148/2007F/notes/sorting.html

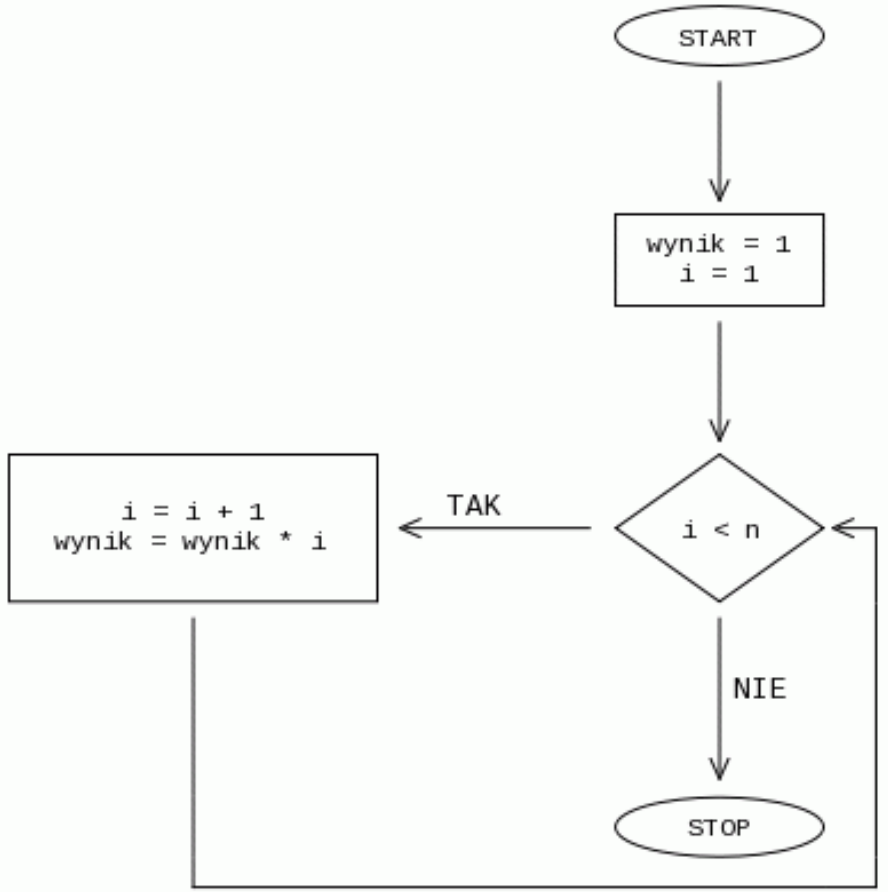
Zadanie algorytmiczne: silnia

dane wejściowe: liczba naturalna n

wynik: silnia liczby podanej na wejściu

silnia(0): 0, **silnia(1):** 1, **…,** **silnia(3):** 1·2·3=6

Algorytm iteracyjny:
mnóż kolejne liczby naturalne,
aż dojdziesz do liczby n



http://www.algorytm.org/algorytmy-arytmetyczne/silnia.html

Rekurencja

Rekurencja: odwołanie algorytmu do samego siebie.

Pink Floyd,
okładka albumu
Ummagumma

Zadanie algorytmiczne: silnia

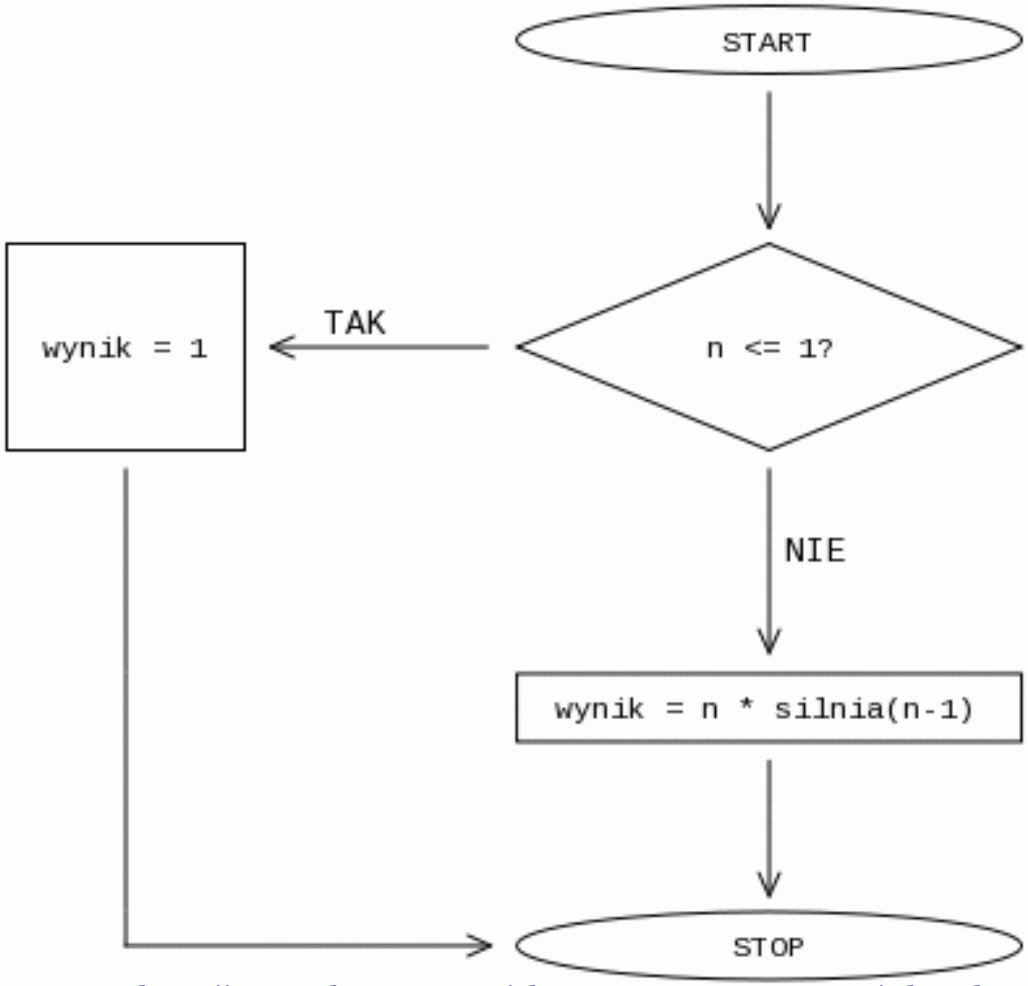
dane wejściowe: liczba naturalna

wynik: silnia liczby podanej na wejściu

silnia(0): 0, **silnia(1):** 1, **…,** **silnia(3):** 1·2·3=6

Algorytm rekurencyjny:
pomnóż n przez silnię (n-1)

silnia(3) = silnia(2) · 3



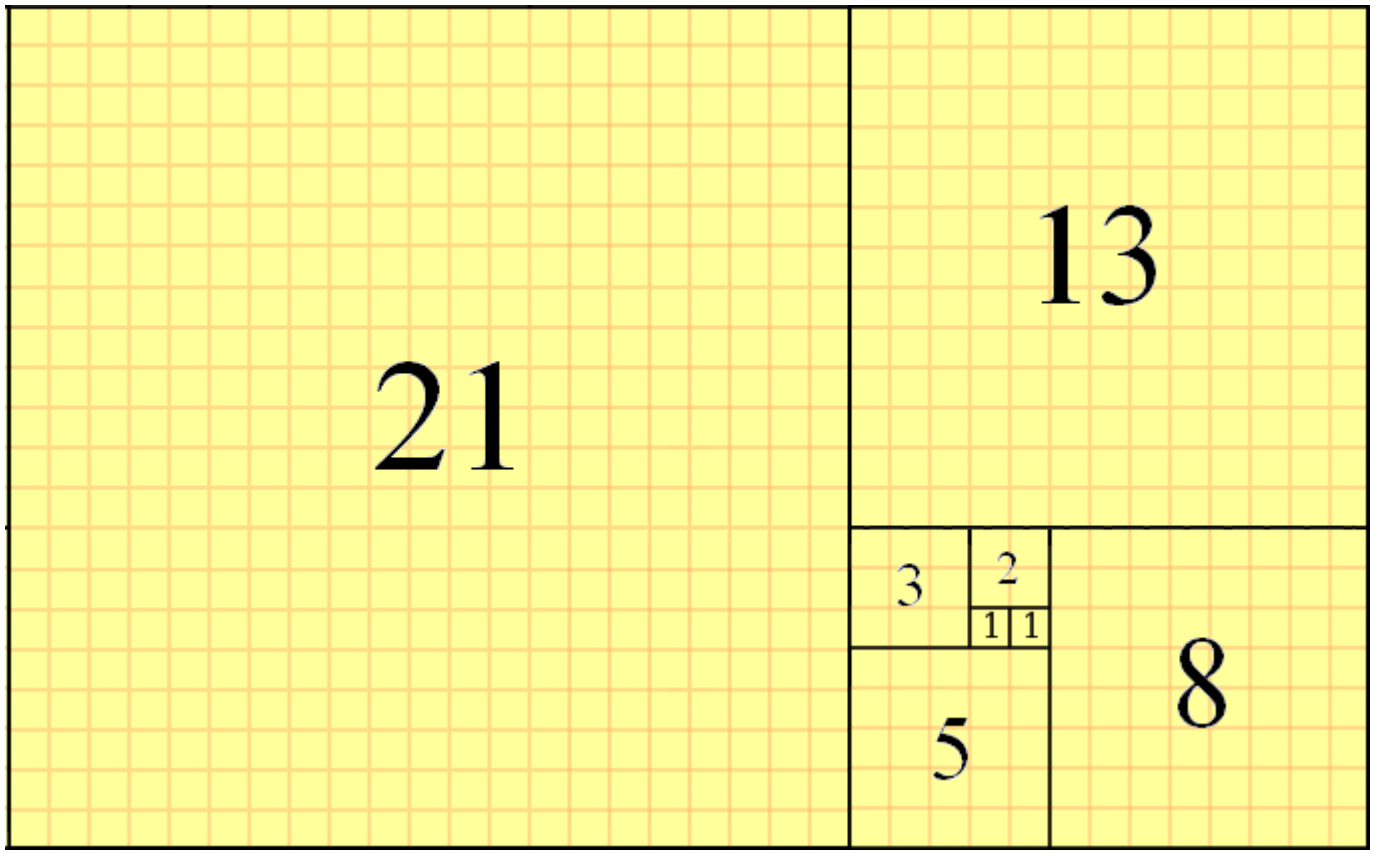
http://www.algorytm.org/algorytmy-arytmetyczne/silnia.html

Rekurencja

Ciąg Fibonnaciego: ciąg liczb, zadany rekurencyjnie:

$$f_0 = 0, f_1 = 1, f_n = f_{n-1} + f_{n-2}$$

0, 1, 1, 2, 3, 5

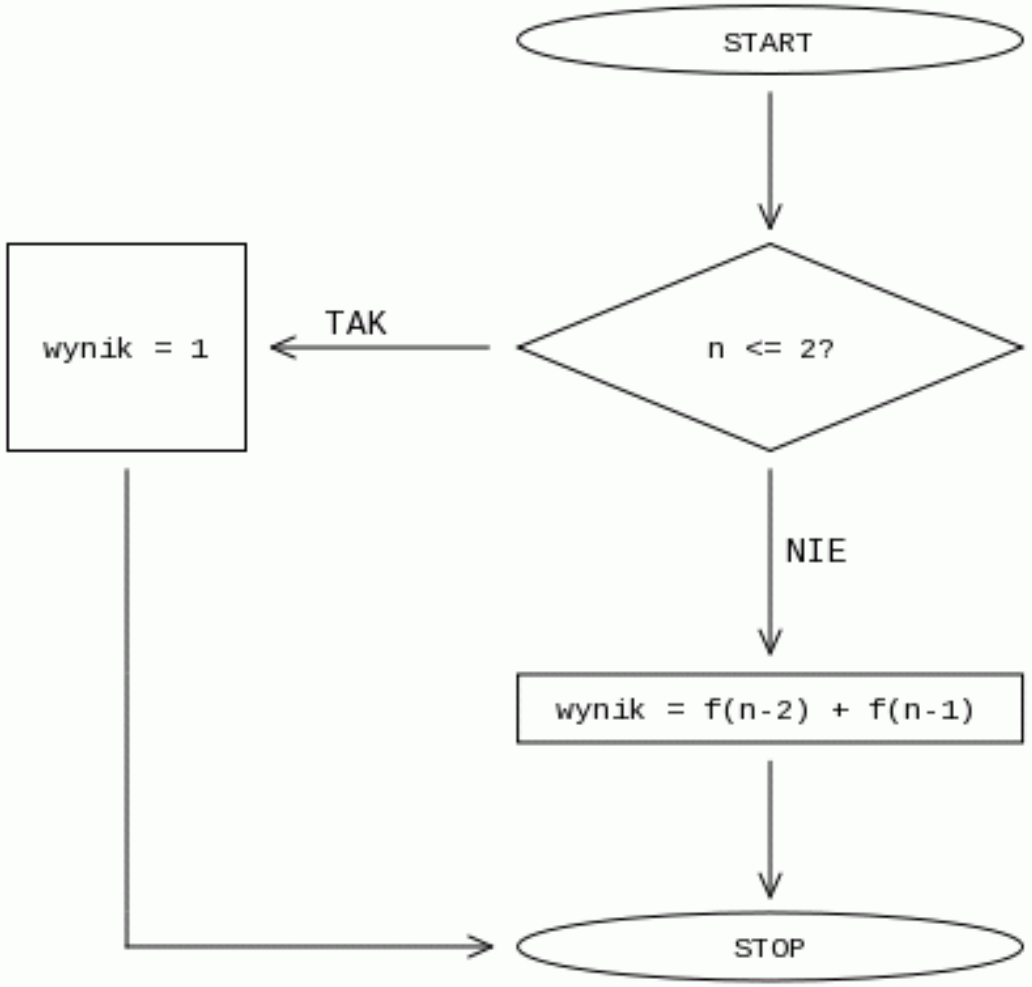


Ciąg Fibonacciego

Ciąg Fibonnaciego: ciąg liczb, zadany rekurencyjnie:

$$f_0 = 0, f_1 = 1, f_n = f_{n-1} + f_{n-2}$$

Złożoność obliczeniowa: ?



http://www.algorytm.org/algorytmy-arytmetyczne/ciag-fibonacciego.html

Ciąg Fibonacciego

Ciąg Fibonnaciego: ciąg liczb, zadany rekurencyjnie:

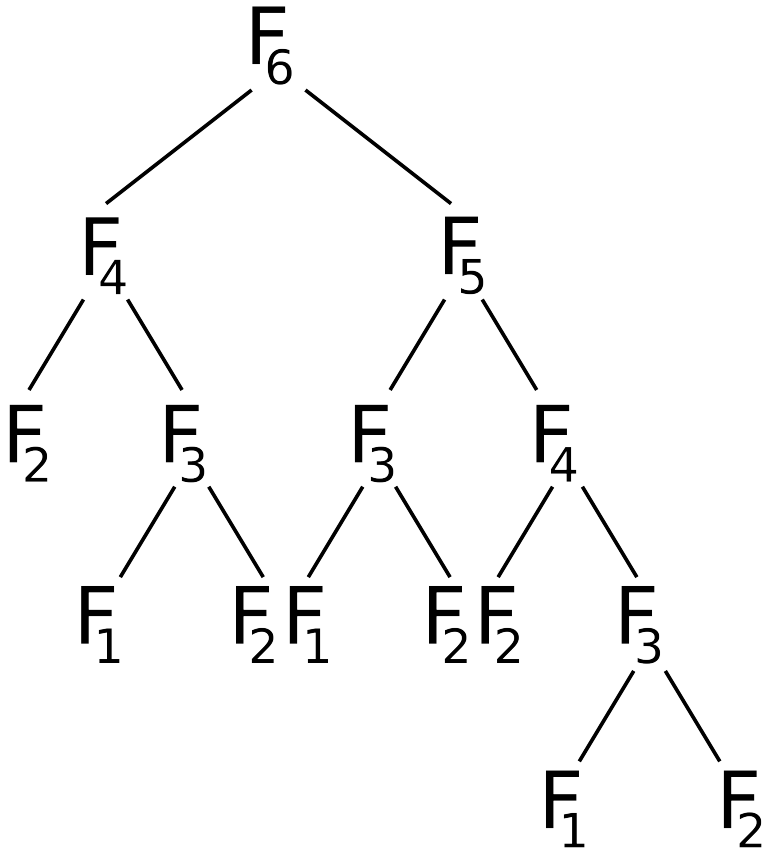
$$f_0 = 0, f_1 = 1, f_n = f_{n-1} + f_{n-2}$$

Złożoność obliczeniowa: $O(2^n)$

Lepszy wariant:

zapamiętywanie raz obliczonych wartości.

Złożoność obliczeniowa: O(n)



Wikipedia Commons

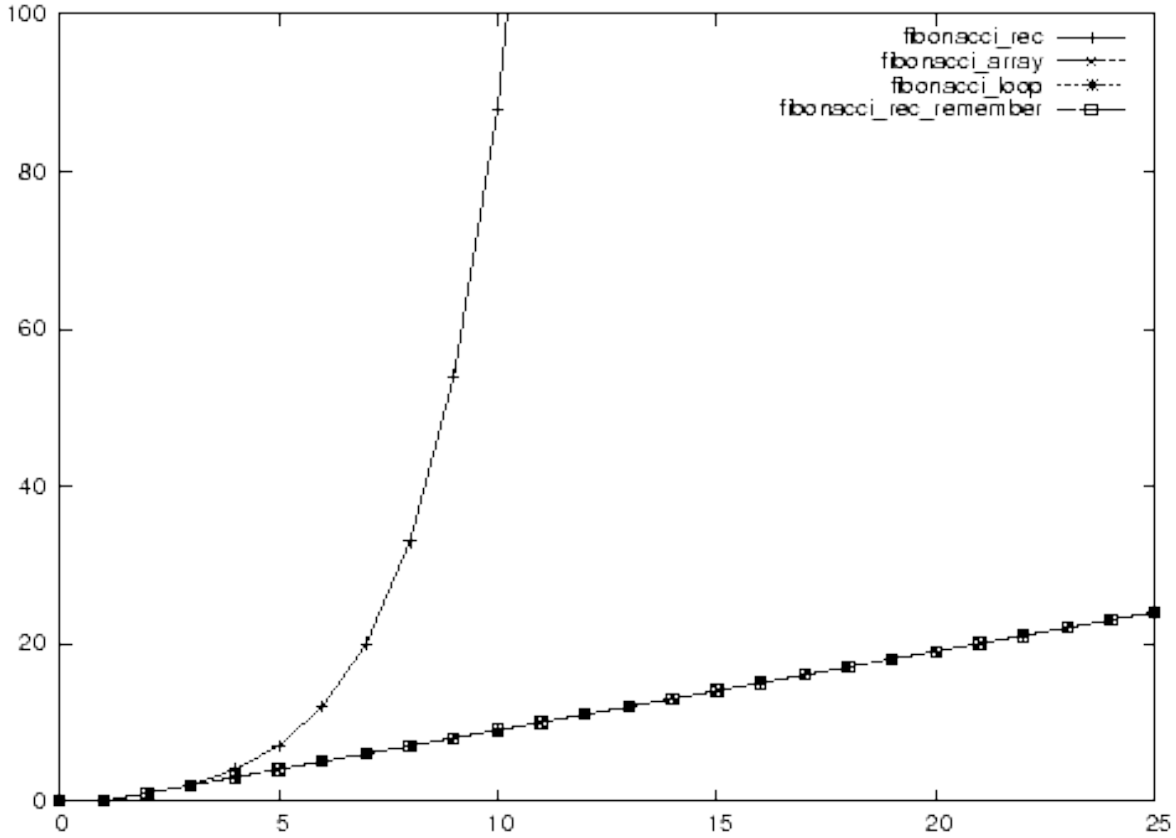
Ciąg Fibonacciego

Ciąg Fibonnaciego: ciąg liczb, zadany rekurencyjnie:

$$f_0 = 0, f_1 = 1, f_n = f_{n-1} + f_{n-2}$$

Złożoność obliczeniowa: $O(2^n)$

Lepszy wariant:
zapamiętywanie raz obliczonych wartości.
Złożoność obliczeniowa: O(n)



http://nicolas.thiery.name/macs358/Notes/2_Proofs/AnalysisOfAlgorithms.html