Podstawy programowania

Rafał Spręga
rsprega@o2.pl

Plan realizowanych zagadnień

1. Algorytm, zapis algorytmów
2. Złożoność obliczeniowa, klasy algorytmów
3. Program komputerowy, języki programowania
4. Metody programowania
5. Rekurencja, Iteracja
6. Struktury danych (listy, tablice, kolejki, drzewa binarne)
7. Algorytmy sortowania
8. Algorytmy przeszukiwania
9. Algorytmy numeryczne
10. Algorytmy heurystyczne
11. Algorytmy kryptograficzne i kompresji danych
12. Algorytmy sztucznej inteligencji

Zaliczenia

- Egzamin
- Zaliczenie laboratoriów
- Zaliczenie projektu

Literatura

- Dawid Harel, **Rzecz o istocie informatyki**, WNT 1992
- Piotr Wróblewski, **Algorytmy, struktury danych i techniki programowania**, Helion2003
- T.H. Cormen, **Wprowadzenie do algorytmów**, WNT 1999
- Serwis Internetowy: http://www.algorytm.org

Pojęcie algorytmu. Trochę historii

- Pierwsze opisy, które później nazwano algorytmami, dotyczyły rozwiązań zadań matematycznych.

- Pomiędzy 400 a 300 rokiem p.n.e. grecki matematyk i filozof Euklides, wymyślił pierwszy znany nam nietrywialny algorytm, czyli przepis na realizację zadania. Był to algorytm znajdowania największego wspólnego dzielnika dwóch dodatnich liczb całkowitych.

Pojęcie algorytmu. Trochę historii

- Słowo algorytm pochodzi od nazwiska matematyka arabskiego, który żył na przełomie VIII i IX wieku naszej ery.

- Muhammad ibn Musa al-Chorezmi zasłużył się stworzeniem kilku dzieł z dziedziny matematyki, w których opisał dużą ilość reguł matematycznych (w tym dodawania, odejmowania, mnożenia i dzielenia zwykłych liczb dziesiętnych). Opis tych procedur był na tyle precyzyjny i formalny, jak na tamte czasy, że właśnie od jego nazwiska pochodzi słowo algorytm.

Pojęcie algorytmu.

- Algorytm - dokładny przepis podający sposób rozwiązania określonego problemu w postaci skończonej liczby uporządkowanych operacji.

- Algorytm – (inf.) ściśle określony ciąg kroków obliczeniowych, prowadzący do przekształcenia danych wejściowych w wyjściowe.

Cechy algorytmu

1. Musi posiadać określony stan początkowy, czyli operację od której zaczyna się jego realizacja.

2. Ilość operacji potrzebnych do zakończenia pracy musi być skończona - warunek dyskretności (skończoności).

3. Musi dać się zastosować do rozwiązywania całej klasy zagadnień, a nie jednego konkretnego zadania - warunek uniwersalności.

4. Interpretacja poszczególnych etapów wykonania musi być jednoznaczna - warunek jednoznaczności.

5. Cel musi być osiągnięty w akceptowalnym czasie - warunek efektywności.

6. Musi posiadać wyróżniony koniec.

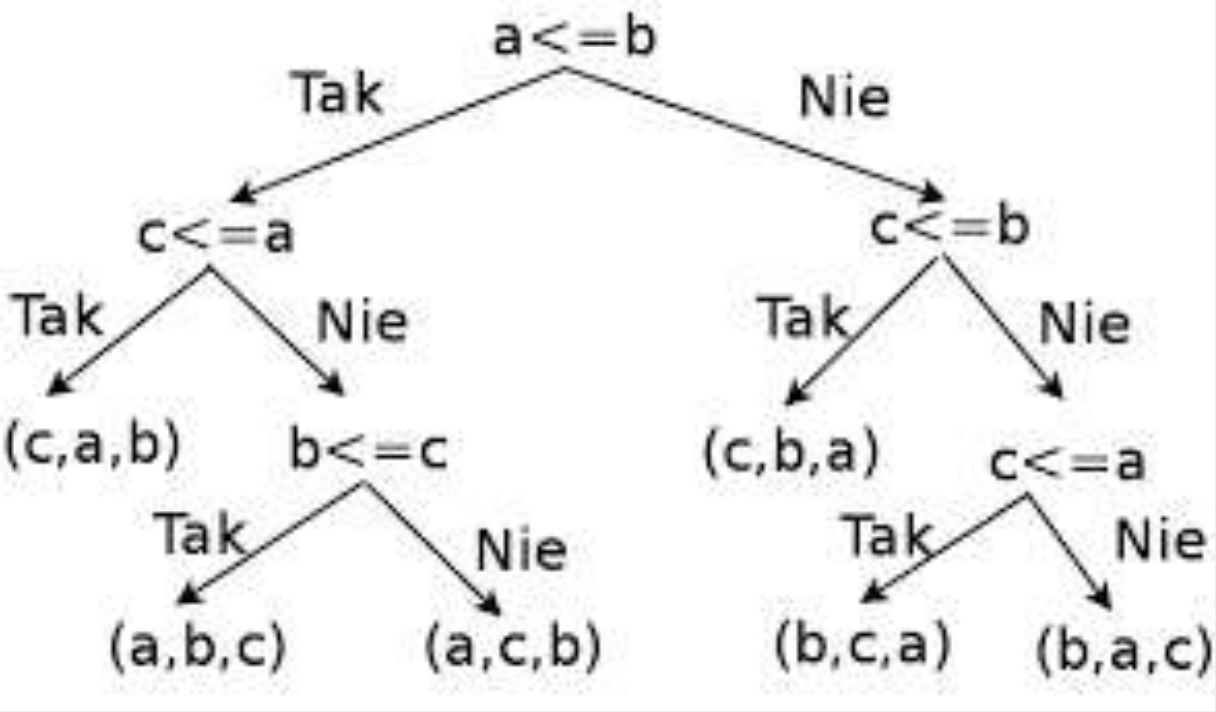
Sposoby zapisu algorytmów

I. Opis słowny za pomocą języka naturalnego:

Przykład:

1. *Dana jest liczba naturalna **n***
2. *Jeśli **n** jest równe 1 to zakończ*
3. *Jeśli **n** jest parzyste to za n przyjmij **n** / 2, w przeciwnym przypadku za **n** przyjmij 3***n** + 1*
4. *Przejdź do punktu 1*

II. Drzewo algorytmu



Schemat blokowy

- Schemat blokowy (sieć działań) – narzędzie służące do przedstawienia kolejnych czynności w projektowanym algorytmie.

- Jest to diagram, na którym procedura, system lub program komputerowy są reprezentowane przez opisane figury geometryczne połączone wektorami zgodnie z kolejnością wykonywania czynności wynikających z przyjętego algorytmu rozwiązania zadania.

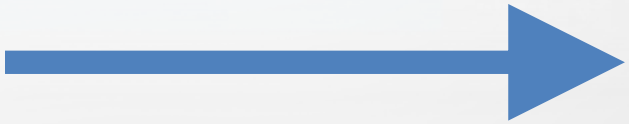
Schemat blokowy

Schemat blokowy zapewnia:

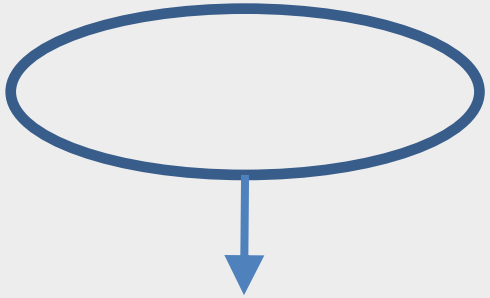
- elastyczność zapisów
- łatwą kontrolę poprawności algorytmu.
- Schematy blokowe pozwalają na prostą zamianę instrukcji na instrukcje programu komputerowego.
- Schemat blokowy pozwala dostrzec istotne etapy algorytmu i logiczne zależności między nimi.

Elementy schematu blokowego

- **strzałka (wektor)** – wskazuje jednoznacznie powiązania i ich kierunek

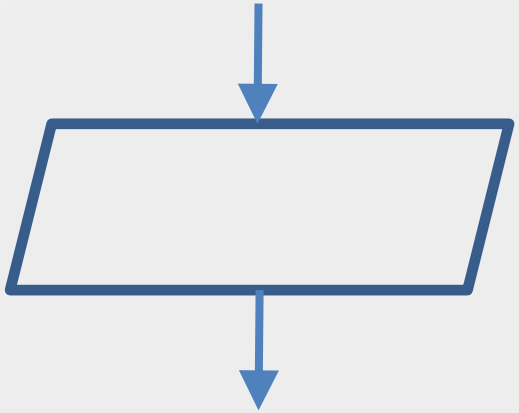


- **blok graniczny** – oznacza początek, koniec, przerwanie lub wstrzymanie wykonywania działania, np. blok startu programu.



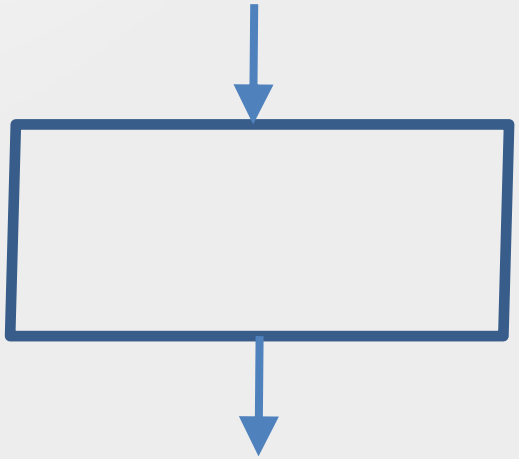
Elementy schematu blokowego

- **blok wejścia-wyjścia** – przedstawia czynność wprowadzania danych do programu i przyporządkowania ich zmiennym dla późniejszego wykorzystania, jak i wyprowadzenia wyników obliczeń, np. ***czytaj*** z, ***pisz*** z+10.



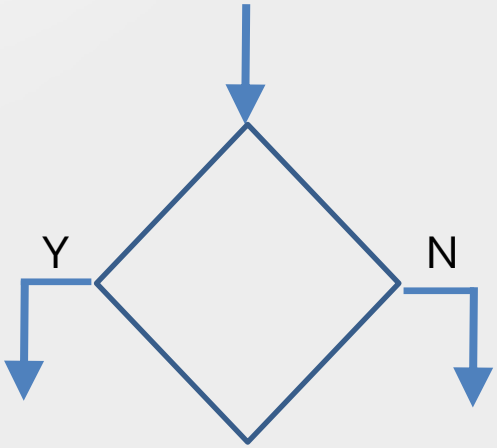
Elementy schematu blokowego

- **blok operacyjny** – oznacza wykonanie operacji, w efekcie której zmienią się wartości, postać lub miejsce zapisu danych, np. z: = z + 1



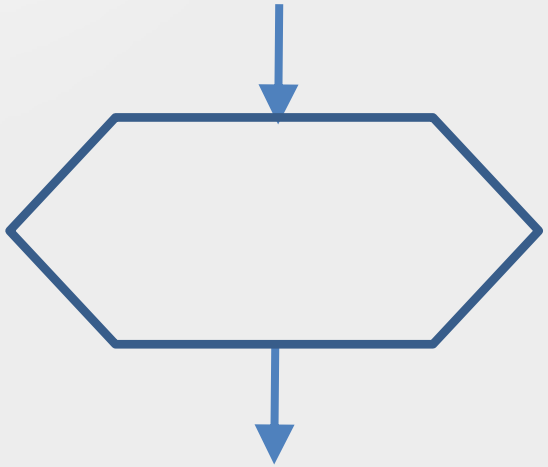
Elementy schematu blokowego

- **blok decyzyjny (warunkowy)**– przedstawia wybór jednego z dwóch wariantów wykonywania programu na podstawie sprawdzenia warunku wpisanego w ów blok, np. a = b.



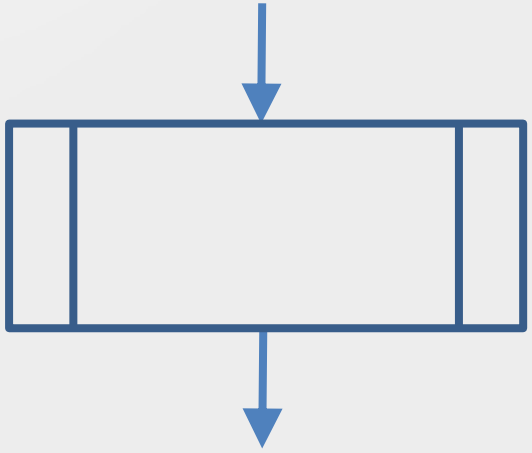
Elementy schematu blokowego

- **blok wywołania programu -** oznacza zmianę wykonywanej czynności na skutek wywołania podprogramu.



Elementy schematu blokowego

- **blok fragmentu** – przedstawia część programu zdefiniowanego odrębnie, np. wybraną procedurę sortowania.



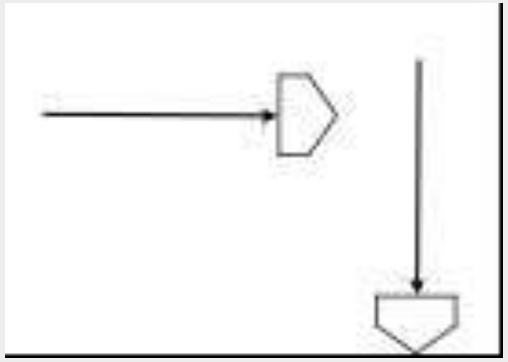
Elementy schematu blokowego

- **blok komentarza** – pozwala wprowadzać komentarze wyjaśniające poszczególne części schematu, co ułatwia zrozumienie go czytającemu, np. wprowadzenie danych.

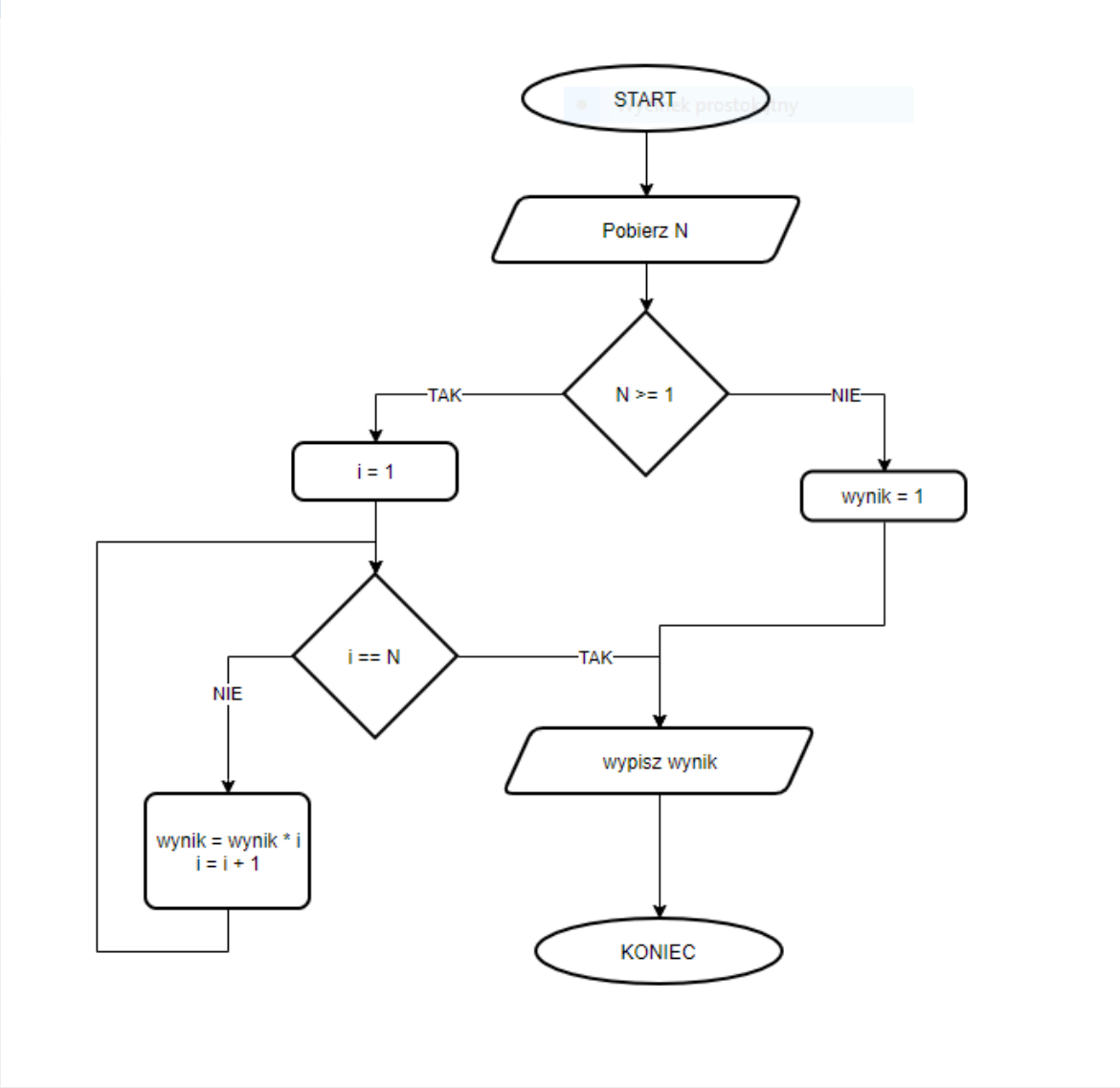
Elementy schematu blokowego

- **łącznik wewnętrzny**– służy do łączenia odrębnych części schematu znajdujących się na tej samej stronie, powiązane ze sobą łączniki oznaczone są tym samym napisem, np. A1, 7..

- **łącznik zewnętrzny** – służy do łączenia odrębnych części schematu znajdujących się na rożnych stronach; powinien być opisany jak łącznik wewnętrzny i zewnętrzny; poza tym powinien zawierać numer strony, do której się odwołuje, np. 2, 4.3, B5.



Schemat blokowy - przykład



Schemat blokowy - przykład

1. Dane są trzy liczby całkowite.
   Napisać schemat blokowy wyznaczający maksymalną z nich.

2. Dany jest ciąg liczb o nieznanej długości.
   Ostatnia liczba w ciągu równa się zero.
   Napisać schemat blokowy obliczający sumę elementów tego ciągu.

Schemat blokowy – przykład cd

3. Dane są dwie liczby całkowite dodatnie **a** i **n**:. Napisać schemat blokowy wyznaczający $\mathbf{a}^{\mathbf{n}}$.

4. Dany jest ciąg liczb o nieznanej długości. Ostatnia liczba w ciągu równa się zero. Napisać schemat blokowy wyznaczający największą z liczb.

Pseudo-język

- Opis słowny za pomocą pseudo-języka. Zaletą tego podejścia jest bardzo łatwa implementacja algorytmu za pomocą konkretnie wybranego, istniejącego języka programowania.
- Wada - mniejsza przejrzystość zapisu.

Przykład:

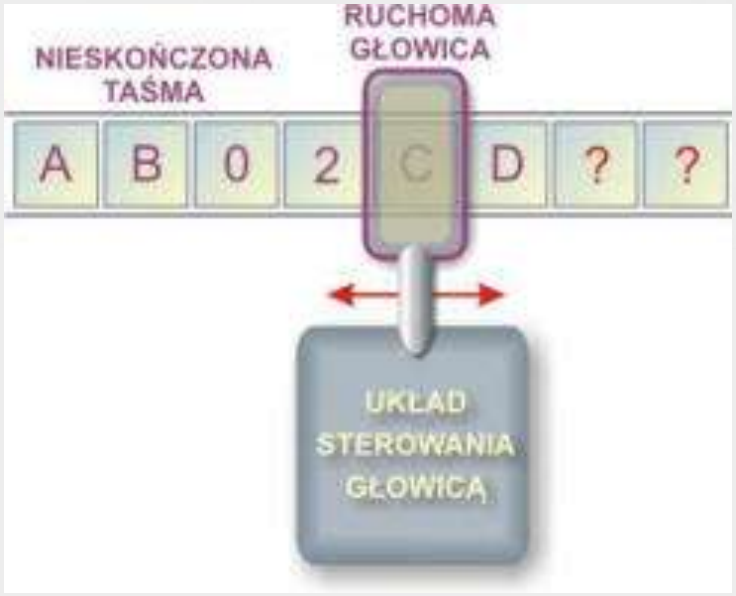
Algorytm Euklidesa:

```
1. Dopóki b jest różne od 0
   1. c:= a mod b
   2. a:= b
   3. b:= c
2. Pisz a
```

Maszyna Turinga

Maszyna Turinga zbudowana jest z trzech głównych elementów:

- Nieskończonej taśmy zawierającej komórki z przetwarzanymi symbolami
- Ruchomej głowicy zapisująco-odczytującej.
- Układu sterowania głowicą.



Maszyna Turinga - taśma

Nieskończona taśma jest odpowiednikiem współczesnej pamięci komputera.

Taśma dzieli się na komórki,
w których umieszczone zostały symbole, czyli po prostu znaki przetwarzane przez maszynę Turinga.

Maszyna Turinga - głowica

Głowica odczytuje i zapisuje dane na taśmę i zawsze znajduje się nad jedną z komórek.

Przed rozpoczęciem pracy maszyny Turinga głowica jest zawsze ustawiana nad komórką taśmy zawierającą pierwszy symbol do przetworzenia.

Maszyna Turinga – układ sterowania

Układ odczytuje za pomocą głowicy symbole z komórek taśmy oraz przesyła do głowicy symbole do zapisu w komórkach.

Dodatkowo nakazuje on głowicy przemieścić się do sąsiedniej komórki w lewo lub w prawo.

Maszyna Turinga - programowanie

Instrukcją dla maszyny Turinga jest następująca piątka symboli:

Instrukcja maszyny Turinga	Znaczenia symboli	
$(S_o, q_i, S_z, q_j, \boldsymbol{L/P})$	S_o	symbol odczytany przez głowicę z bieżącej komórki na taśmie
	q_i	bieżący stan układu sterowania
	S_z	symbol, jaki zostanie zapisany w bieżącej komórce na taśmie
	q_j	nowy stan, w który przejdzie układ sterowania po wykonaniu tej operacji
	$\boldsymbol{L/R}$	ruch głowicy o jedną komórkę w lewo (L) lub w prawo (R)

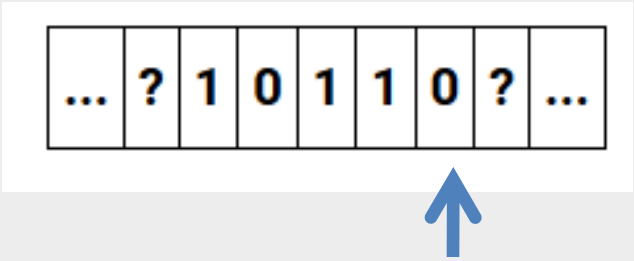
Maszyna Turinga – przykład

Przykładowy program:

Program

```
0,q0,1,q0,L bit 0 zamień na 1
1,q0,0,q0,L bit 1 zamień na 0
```

Przykładowy zapis na taśmie:



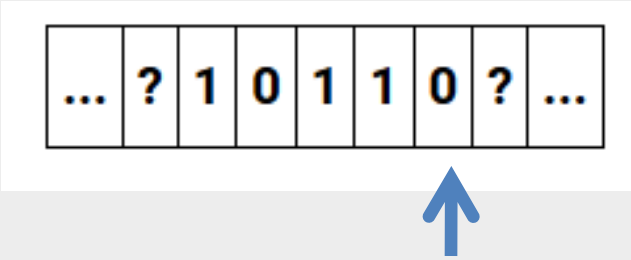
Co robi ten program?

Maszyna Turinga – przykład 2

A co robi ten:

```
0,q0,0,q0,L stan początkowy, zapamiętujemy 0
1,q0,0,q1,L stan początkowy, zapamiętujemy 1
0,q1,1,q0,L zapisujemy zapamiętane 1,zapamiętujemy 0
1,q1,1,q1,L zapisujemy zapamiętane 1,zapamiętujemy 1
?,q0,0,q2,L koniec, zapisujemy zapamiętane 0
?,q1,1,q2,L koniec, zapisujemy zapamiętane 1
```

Dla zapisu na taśmie:



Teza Churcha-Turinga

Każdy problem, dla którego przy nieograniczonej pamięci oraz zasobach istnieje efektywny algorytm jego rozwiązywania, da się rozwiązać na maszynie Turinga.

Złożoność algorytmów

Złożoność algorytmów możemy podzielić na:

- Złożoność obliczeniową
- Złożoność pamięciową

Złożoność algorytmów – wyszukiwanie liniowe

Wyszukiwanie na nieuporządkowanej liście elementu **X**(np. numeru telefonu)

W typowym algorytmie wykonujemy dwa sprawdzenia:

- *Czy znaleźliśmy X*
- *Czy dotarliśmy do końca listy*

Możemy założyć, że mają one decydujący wpływ na czas działania algorytmu.

Czy możemy to poprawić?

Tak. Wstawiając tzw. wartownika listy

Złożoność algorytmów – notacja *O*

- Gdy w naszym algorytmie mamy do przetworzenia **N** elementów i liczba ta jest dość duża, wówczas nie dbamy o to, czy algorytm wykona 3*N, czy 100 *N, czy N/5 operacji elementarnych.
- Liczy się wówczas to, że wraz ze wzrostem ilości danych czas wydłuża się liniowo.
- Wówczas mówimy, że w najgorszym przypadku czas działania algorytmu jest linowy i zapisujemy, go używając **notacji duże O:** ***O(N)***

Złożoność algorytmów – notacja *O*

Wyszukiwanie na uporządkowanej liście elementów można znacznie przyspieszyć:

1. *Szukany element porównujemy z elementem środkowym.*
2. *Jeśli jest on szukanym elementem to kończymy działanie z odpowiedzią pozytywną*
3. *Jeśli, nie to bierzemy pierwszą lub drugą połówkę listy zależnie od wyniku porównania*
4. *Jeśli lista nie jest pusta wracamy do punktu 1, w przeciwnym przypadku kończymy działanie z odpowiedzią negatywną*

Złożoność tego algorytmu wynosi $\mathbf{O(\log_2 N)}$

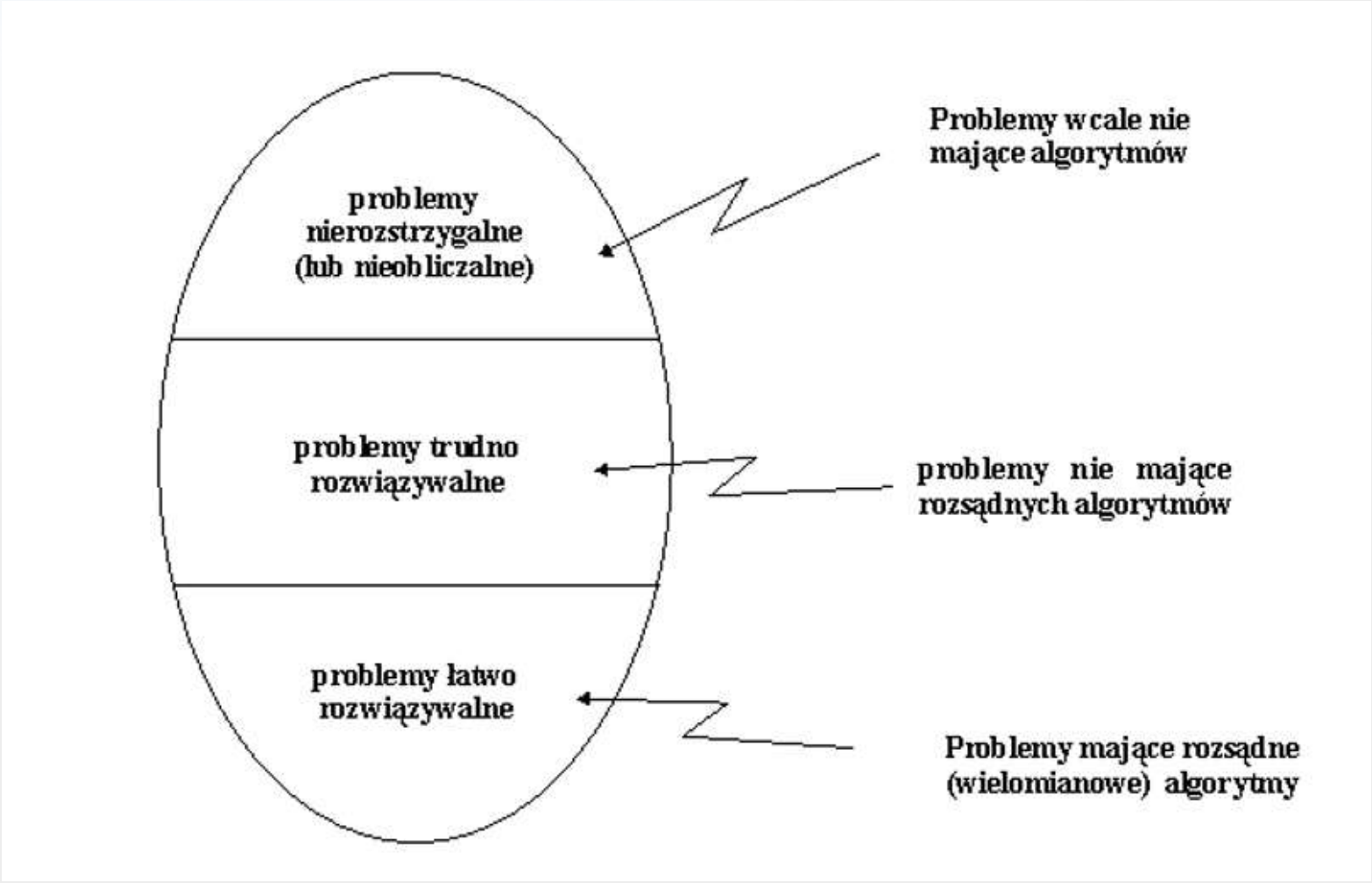
Złożoność algorytmów – notacja ***O***

Sortowanie bąbelkowe

I. Wykonaj N-1 razy co następuje:
   1. Wskaż pierwszy element
   2. Wykonaj N-1 razy co następuje:
      a. Porównaj element wskazany z elementem następnym
      b. Jeśli porównane elementy są w złej kolejności to je zamień
      c. Wskaż następny element

Można łatwo wykazać, że złożoność tego algorytmu wynosi $O(n^2)$

Klasy problemów algorytmicznych ze względu na złożoność



Klasy problemów algorytmicznych ze względu na złożoność

- Problemy łatwo rozwiązywalne, to np. sortowanie elementów listy, wyszukiwanie liniowe
- Problemy trudno rozwiązywalne, to np.: problem komiwojażera, lub problem wież Hanoi.
- Problemy nieobliczalne lub nierozstrzygalne, to np.: problem kafelkowania (problem układania domina)

Algorytmy zachłanne

- Problem z wydawaniem reszty
- Problem kinomaniaka
- Minimalne drzewo rozpinające

Programowanie dynamiczne

- Algorytm plecakowy
- Podział majątku
- Problem minimalnej ścieżki
- Najdłuższe wspólne podsłowo (LCS)
- Optymalne mnożenie macierzy

Struktury danych

- STOS (LIFO)
- KOLEJKA (FIFO)
- LISTA
- KOLEJKA PRIORYTETOWA
- DRZEWA BINARNE

Dziękuję za uwagę