TEORETYCZNE PODSTAWY INFORMATYKI

8/10/2018

WFAiS UJ, Informatyka Stosowana
I rok studiów, I stopień

Zakres tematyki wykładu

- Co to jest informacja?
- Algorytmy i struktury danych, poprawność algorytmu
- Złożoność obliczeniowa
- Rekursja, indukcja, iteracja, kombinatoryka i elementy teorii prawdopodobieństwa
- Modele danych: drzewa, listy, zbiory, relacje, grafy
- Wzorce, automaty, wyrażenia regularne i gramatyki

Literatura

- H. Abelson, et al., Struktura i intepretacja programów komputerowych
- A. V. Acho, J. D. Ullman, Wykłady z informatyki z przykładami w języku C
- T. H. Cormen, Ch. F. Leiserson, R. L. Rivest, Wprowadzenie do algorytmów
- L. Banachowski, K. Diks, W. Rytter, Algorytmy i struktury danych
- A. Drozdek, D. L. Simon, Struktury danych w języku C
- D. Harel, Rzecz o istocie informatyki
- J.E. Hopcroft, J. Ullman, Wprowadzenie do teorii automatów, języków i obliczeń
- S. Kowalski, A. W. Mostowski, Teoria automatów i lingwistyka matematyczna
- Ch. H. Papadimitriou, Złożoność obliczeniowa
- W. Sikorski, Wykłady z podstaw informatyki
- W. M. Turski, Propedeutyka Informatyki
- N. Wirth, Algorytmy i struktury danych = programy

Warunki zaliczenia przedmiotu

- **Zaliczenie ćwiczeń z zadań tablicowych (1/3 oceny końcowej)**
- **Egzamin pisemny: (2/3 oceny końcowej)**
  - **Na końcu każdego wykładu podana jest lista pytań do danego wykładu. Z tej listy pytań zostaną wybrane pytania na egzaminie pisemnym.**
    - **15 pytań, każde punktowane w skali 0-2.**
  - **Aby zaliczyć egzamin należy uzyskać co najmniej 15pkt**

Informatyka: mechanizacja abstrakcji

- **Informatyka jest miedzy innymi nauką o abstrakcji, czyli nauką o tworzeniu właściwego modelu reprezentującego problem i znajdowaniu odpowiedniej techniki mechanicznego (automatycznego) jego rozwiązywania.**
- **Informatycy tworzą abstrakcje rzeczywistych problemów w formach które mogą być rozumiane i przetwarzane w pamięci komputera.**

Informatyka: mechanizacja abstrakcji

- **Abstrakcja**
  oznacza pewne uproszczenie, zastąpienie skomplikowanych i szczegółowych okoliczności występujących w świecie rzeczywistym zrozumiałym modelem umożliwiającym rozwiązanie naszego problemu.
  Oznacza to że pomijamy szczegóły które nie mają wpływu lub mają minimalny wpływ na rozwiązanie problemu. Ten etap rozwiązywania zadania nazywamy konstruowaniem modelu.
- **Opracowanie odpowiedniego modelu ułatwia zajęcie się istotą problemu który mamy rozwiązać.**

Informatyka: mechanizacja abstrakcji

W ramach tego wykładu omówimy

- **Modele danych: abstrakcje wykorzystywane do opisywania problemów**
- **Struktury danych: konstrukcje języka programowania wykorzystywane do reprezentowania modelów danych. Przykładowo język C udostępnia wbudowane takie jak struktury czy wskaźniki, które umożliwiają reprezentowanie bardziej skomplikowanych struktur takich jak np. drzewa i grafy**
- **Algorytmy: techniki wykorzystywane do otrzymywania rozwiązań na podstawie operacji wykonywanych na danych reprezentowanych przez struktury danych lub na inne sposoby**

Wykład 1a:

Informacja i zasady jej zapisu

- **Czym jest informacja?**
  - **Bity i Bajty**
  - **Kodowanie informacji**
- **Systemy zapisu liczb**
  - **System binarny (dwójkowy)**
  - **Sposoby kodowania: liczb naturalnych, całkowitych, rzeczywistych**
  - **Dlaczego pojawiają się błędy i zaokrąglenia**
- **Znaki i teksty**
- **Obrazy i dźwięki**
- **Kompresja i szyfrowanie**

Czym jest informacja?

Istnieje kilka różnych definicji pojęcia informacja (encyklopedia PWN)

- **Konstatacja stanu rzeczy, świadomość.**
- **Obiekt abstrakcyjny, który w sposób zakodowany może być przesyłany, przetwarzany i używany do sterowania.**
- **Powiadamianie społeczeństwa lub określonych zbiorowości w sposób zobiektyzowany, systematyczny i konkretny za pomocą środków masowego przekazu.**

Interesuje nas ta druga definicja, ponadto:

- **Informacją zajmuje się nauka zwana Teorią Informacji. Dotyczy ona przekazywania wiadomości ze źródła wiadomości do ich przeznaczenia – odbiorcy.**
- **Informację możemy mierzyć ilościowo i jakościowo.**

Teoria informacji

- **Teoria informacji – dyscyplina zajmująca się problematyką informacji oraz metodami przetwarzania informacji, np. w celu transmisji lub kompresji.**
- **Naukowo teoria informacji jest blisko powiązana z matematyką dyskretną, a z jej osiągnięć czerpią takie dyscypliny jak informatyka i telekomunikacja.**

Teoria informacji: historia

- Za twórcę teorii informacji uważa się Claude'a E. Shannona, który prawdopodobnie po raz pierwszy użył tego terminu w 1945 roku w swojej pracy zatytułowanej "A Mathematical Theory of Cryptography".
- Natomiast w 1948 roku w kolejnej pracy pt. "A Mathematical Theory of Communication" przedstawił najważniejsze zagadnienia związane z tą dziedziną nauki.
- Shannon stworzył podstawy ilościowej teorii informacji, późniejsi autorzy próbowali stworzyć teorie wyjaśniające wartość (cenność) informacji.
- W Polsce Marian Mazur stworzył oryginalną teorię opisującą zarówno ilość jak i jakość informacji. Opisał ją m.in. w wydanej w 1970 roku książce *Jakościowa teoria informacji*. Wprowadził w niej rozróżnienie między *informacjami opisującymi* a *informacjami identyfikującymi*

Jak przekazujemy informację?

Informację przekazuje możliwość porównania dwóch stanów.

- **Dzwonek dzwonka informuje nas, że ktoś nacisnął przycisk.**
  Kiedy przycisk się zatnie i dzwonek dzwoni dalej, już nie informuje nas o niczym. Gdy przestanie dzwonić a my porównamy dwie sytuacje, uzyskamy informację, że usterka została usunięta.
- **Brak zmian to brak przekazu informacji: niezmienny sygnał nosi nazwę szumu. Nie można go jednak ignorować, gdyż często zakłóca przekaz właściwej informacji.**

Jednostka informacji: bit

Podstawową jednostką informacji jest bit, oznaczany też poprzez „b”(w ang. kawałek, skrót od binary digit, czyli cyfra dwójkowa).

- **Bit jest to podstawowa elementarna jednostka informacji: wystarczająca do zakomunikowania jednego z co najwyżej dwóch jednakowo prawdopodobnych zdarzeń.**
- **Bit stanowi podstawę zapisu informacji w różnych typach pamięci komputera. Wszystkie inne jednostki składają się z jego wielokrotności.**
- **Bit przyjmuje jedną z dwóch wartości, które zwykle oznacza się jako „0” lub „1”. Jest to oznaczenie stosowane w matematyce (wartość logiczna: „0” – fałsz, „1” - prawda) oraz przy opisie informacji przechowywanej w pamięci komputera i opisie sposobów kodowania informacji.**

Bajt

- Jest to najmniejsza adresowalna jednostka informacji pamięci komputerowej, składająca się z bitów, w praktyce przyjmuje się że jeden bajt to 8 bitów (zostało to uznane za standard w 1964 r.).
- Jeden bajt może reprezentować zatem $2^8 = 256$ różnych wartości, które mogą oznaczać zapisywane informacje.
- Bajt oznaczany jest poprzez „B".
- Stosowanie przedrostków kilo, mega, giga itp. jako do określania odpowiednich potęg liczby dwa (np. 2^{10}) jest niezgodne z wytycznymi układu SI (np. kilo oznacza 1000, a nie 1024).
- W celu odróżnienia przedrostków o mnożniku 1000 od przedrostków o mnożniku 1024 (2^{10}), w styczniu 1997 r. pojawiła się propozycja ujednoznacznienia opracowana przez IEC (ang. International Electrotechnical Commission) polegająca na dodawaniu litery "i" po symbolu przedrostka dwójkowego, oraz "bi" po jego nazwie.

Wielokrotności bajtów

Przedrostki dziesiętne (SI)			Przedrostki binarne (IEC)		
Nazwa	Symbol	Mnożnik	Nazwa	Symbol	Mnożnik
Kilobajt	kB / KB	$10^3 = 1000^1$	Kibibajt	KiB	$2^{10} = 1024^1$
Megabajt	MB	$10^6 = 1000^2$	Mebibajt	MiB	$2^{20} = 1024^2$
Gigabajt	GB	$10^9 = 1000^3$	Gibibajt	GiB	$2^{30} = 1024^3$
Terabajt	TB	$10^{12} = 1000^4$	Tebibajt	TiB	$2^{40} = 1024^4$

Kodowanie informacji

- **Jak to się dzieje że w pamięci komputera można przechowywać teksty, obrazy, dźwięki i liczby znacznie różniące się od zestawu 0 – 255?**
  - **Dzięki kodowaniu informacji**
- **Bez kodowania nie ma zapisu różnorodnych informacji w pamięci komputera. Kodowanie występuje w każdym programie i na każdym poziomie.**

Systemy zapisu liczb

- **System liczbowy – to inaczej zbiór reguł zapisu i nazewnictwa liczb.**

- **Do zapisu liczb zawsze używa się pewnego skończonego zbioru znaków, zwanych cyframi (np. arabskimi lub rzymskimi), które jednak można zestawiać ze sobą na różne sposoby otrzymując nieskończoną liczbę kombinacji.**

System jedynkowy

- **Najbardziej prymitywnym systemem liczbowym jest jedynkowy system liczbowy, w którym występuje tylko jeden znak (np. 1).**
- **W systemie tym kolejne liczby są tworzone przez proste powtarzanie tego znaku.**

- **Przykład:**
  - **3 w tym systemie zapisujemy jako 111,**
  - **5 w tym systemie zapisujemy jako 11111.**

Systemy addytywne

- W tych systemach liczby tworzy się przez dodawanie kolejnych symboli.
- Przykładem addytywnego systemu jest dobrze znany i wciąż stosowany rzymski system liczbowy z podstawowymi wielokrotnościami 10 i 5.
- Jego cyfry to: „I” = 1, „V” = 5, „X” = 10, „L” = 50, „C” = 100, „D” = 500, „M” = 1000
- W tym systemie w niektórych przypadkach występuje odejmowanie, a nie tylko dodawanie.
- Przykład:
  - jeśli "X"=10,"V"=5,"I"=1 to „XVI” = 10+5+1 = 16
  - jeśli "X"=10,"V"=5,"I"=1 to „XIV” = 10+5 -1 = 14

Systemy addytywne

- **Sześćdziesiątkowy system liczbowy, stosowany w Mezopotamii, w którym podstawowymi wielkościami były 10 i 60, był częściowo addytywny, częściowo pozycyjny. Jest on najstarszym znanym systemem każdego z tych dwóch rodzajów.**
- **W życiu codziennym spotykamy ślady babilońskiego systemu w podziale godziny na 60 minut, a minuty na 60 sekund, oraz w podziale kąta na minuty i sekundy kątowe.**
- **Zaletą systemów addytywnych jest możliwość zapisu nawet dużych liczb całkowitych (pod warunkiem że są okrągłe) za pomocą jednego znaku, a wadą złożoność, kłopoty interpretacyjne i zbyt wielka liczba cyfr przy mało okrągłych liczbach, oraz bardzo skomplikowany sposób dokonywania za ich pomocą prostych operacji arytmetycznych, wymagający zapamiętywania długich tabel.**

Systemy pozycyjne

- Są to systemy które posiadają symbole (cyfry) tylko dla kilku najmniejszych liczb naturalnych: 0, 1, 2, ..., g – 1, gdzie g to tzw. podstawa systemu, która może być dowolną liczbą naturalną większą niż 1.
  - Np. 5004,3 (podstawa systemu g = 10)
- Cyfry te są kolejno umieszczane w ściśle określonych pozycjach i są mnożone przez odpowiednią potęgę g. W sytuacji, gdy dana potęga nie jest potrzebna do zapisu danej liczby, zostawia się w zapisie puste miejsce, lub częściej specjalny symbol. Współcześnie jest to cyfra 0.

Systemy pozycyjne

- **Na przykład liczbę 5004,3 w dziesiętnym systemie liczbowym (czyli systemie, którego podstawą jest 10) odczytuje się jako:**
  - $5 \cdot 10^3 + 0 \cdot 10^2 + 0 \cdot 10^1 + 4 \cdot 10^0 + 3 \cdot 10^{-1}$

    $= 5 \cdot 1000 + 4 \cdot 1 + 3 \cdot 0{,}1$

    $= 5004{,}3$

Systemy pozycyjne

- Zaletą systemów pozycyjnych jest ich
  - Klarowność
  - Łatwość dokonywania nawet złożonych operacji arytmetycznych
  - Możliwość zapisu dowolnie dużej liczby, jednak do zapisu bardzo dużych liczb (nawet okrągłych) jest potrzebna duża liczba cyfr.
- Współcześnie powszechnie używany jest system dziesiątkowy.
- W informatyce często stosowany jest system dwójkowy (binarny), ósemkowy i szesnastkowy (heksadecymalny).

Systemy liczbowe w informatyce

- **Z racji reprezentacji liczb w pamięci komputerów za pomocą bitów, najbardziej naturalnym systemem w informatyce jest system dwójkowy.**

- **Ze względu na specyfikę architektury komputerów, gdzie często najszybszy dostęp jest do adresów parzystych, albo podzielnych przez 4, 8 czy 16, często używany jest szesnastkowy system liczbowy. Sprawdza się on szczególnie przy zapisie dużych liczb takich jak adresy pamięci, zakresy parametrów, itp.**

Systemy liczbowe w informatyce

- **Na przykład:**
  - $2^{16} = 65536_{10} = 10000_{16}$
  - $2^{32} = 4294967296_{10} = 100000000_{16}$
- **10000_{16} i 100000000_{16} są znacznie łatwiejsze do zapamiętania.**
- **System szesnastkowy często spotykany jest też na stronach WWW (HTML), gdzie stosowany jest do zapisu kolorów.**

Liczby naturalne w systemie binarnym

- Podobnie jak w dziesiętnym systemie pozycyjnym, w systemie dwójkowym liczby naturalne przedstawiamy jako sumę potęg bazy (2) z odpowiednimi wagami: 0 i 1.
- Każda liczba naturalna ma zatem reprezentację postaci: $\sum_{k=0}^{m} a_k 2^k$, gdzie $a_k \in \{0,1\}$
- Reprezentacja ta jest jednoznaczna, jeśli przyjmiemy, że nie stosujemy wiodących zer.

Liczby naturalne w systemie binarnym

Pierwsze 16 wartości			
$(k)_{10}$	$(k)_2$	$(k)_{10}$	$(k)_2$
0	0	8	1000
1	1	9	1001
2	10	10	1010
3	11	11	1011
4	100	12	1100
5	101	13	1101
6	110	14	1110
7	111	15	1111

Liczby naturalne w syst. binarnym

- **Jeśli ustalimy z góry pewną liczbę n bitów, za pomocą których będziemy reprezentowali liczby naturalne, to uzyskamy n - bitowe reprezentacje, uzupełniając je do pełnych n bitów zerami z lewej strony.**
- **Dla n = 8 (1 bajt), mamy kolejno : 00000000, 00000001, 00000010, … , 11111111**
  - **możemy w ten sposób zapisać liczby od 0 do 255.**
- **Różnych wartości n – bitowych jest 2n: od 0 do 2n - 1**

Zapis dziesiętny → binarnego

Metoda pierwsza:

- Oznaczmy poprzez m liczbę w zapisie dziesiętnym i załóżmy $m > 0$
- Znajdujemy największą liczbę $d = 2^k$ nie większą niż m.
- Piszemy jedynkę, odejmujemy od m wartość d, a następnie kolejno dla wszystkich mniejszych potęg dwójki sprawdzamy, czy mieszczą się one w tym co zostało z m.
- Jeśli dana potęga dwójki się nie mieści to dopisujemy zero, wpp. dopisujemy jedynkę i odejmujemy tę wartość od tego, co zostało z m.
- Przykładowo dla $m = 13$ patrz tabelka

m	Potęga dwójki	Wypisano
13	8	1
5	4	11
1	2	110
1	1	1101
0		

Liczby całkowite

- Umówmy się, że przeznaczymy określoną liczbę n bitów, aby reprezentować liczby całkowite. Powstaje pytanie: jak zapisywać liczby ujemne?
- Istnieją co najmniej 3 sposoby.
  - Kodowanie w systemie znak-moduł
  - Kodowanie w systemie znak-moduł odwrotny
  - Kodowanie w systemie uzupełnieniowym

System znak - moduł

Kodowanie w systemie znak-moduł:

- Umawiamy się że jeden bit, np. pierwszy z lewej, rezerwujemy na określenie znaku liczby.
- Pozostałe n-1 bitów reprezentuje moduł liczby w tradycyjny sposób.
- Jeśli pierwszy bit znaku jest równy 0, to liczba jest nieujemna, a jeśli 1, to jest niedodatnia.

Bity	Wartość	Bity	Wartość
0000	0	1000	-0
0001	1	1001	-1
0010	2	1010	-2
0011	3	1011	-3
0100	4	1100	-4
0101	5	1101	-5
0110	6	1110	-6
0111	7	1111	-7

Ułamki

- Podobnie jak w systemie dziesiętnym korzystamy z ujemnych potęg bazy (dziesiątki) po przecinku, tak tu będziemy rozważali binarne rozwinięcia ułamków za pomocą ujemnych potęg dwójki.
- Po przecinku, oddzielającym część całkowitą od ułamkowej, kolejne bity będą odpowiadały wartościom kolejno: ½, ¼, …

System	Ułamki		
Dziesiętny	¼	½	¾
Binarny	0.01	0.1	0.11

- Zapisanie ułamka dziesiętnego binarnie jest proste, jeśli tylko mianownik ułamka jest potęgą dwójki. Wystarczy zapisać licznik binarnie, a następnie przesunąć przecinek o tyle pozycji w lewo, ile wynosi wykładnik potęgi dwójki.
  - Np.: 5/16 zapisane binarnie to 0.0101, jest to po prostu 5 czyli 101, przesunięte o 4 pozycje w prawo.

Ułamki

Co zrobić gdy mianownik nie jest potęgą dwójki?

- Przez u oznaczmy ułamek, który chcemy zapisać binarnie. Niech $0 < u < 1$.
- Dopóki $0 < u$, bądź otrzymujemy nie napotkane wcześniej wartości u, będziemy wykonywać:
  - Pomnóż u przez 2
  - Jeżeli u jest mniejsze od 1, wpisz cyfrę 0
  - Wpp. wpisz cyfrę 1 i odejmij od u cyfrę 1.
- Gdy powtórzy się wartość u, otrzymamy szukany wynik, przy czym wypisany ciąg jest okresowy, a jego okresem jest ciąg bitów między powtórzeniami u.

Ułamki - przykład

- **Zauważmy iż nawet tak prosta liczba jak 1/10 ma nieskończone binarne rozwinięcie okresowe. Gdy chcemy reprezentować ją w komputerze, jesteśmy zmuszeni do zaokrąglenia tej wartości i w rzeczywistości otrzymujemy tylko coś koło 1/10.**

Bity	Wartość **u**
0.	2/7
0	4/7
1	1/7
0	2/7
0	4/7

Binarne rozwinięcie: $2/7 = 0.(010)$

Bity	Wartość **u**
0.	1/10
0	2/10
0	4/10
0	8/10
1	6/10
1	2/10

Binarne rozwinięcie: $1/10 = 0.0(0011)$

Ułamki - zaokrąglenia

- Skoro nie da się dokładnie reprezentować wartości wymiernych w komputerze, trzeba je zaokrąglać. Reguły są bardzo proste.
- Jeśli chcemy zaokrąglić na k - tej pozycji, to patrzymy na cyfrę na następnej pozycji (k+1)
  - Jeśli jest ona równa 0, to zaokrąglamy w dół, 'obcinamy ogon' rozwinięcia binarnego
  - Wpp, zaokrąglamy w górę, również 'odrzucamy ogon', jednakże na k - tym miejscu przybliżenia dodajemy 1.

Ułamki – zaokrąglenia, przykład

- **Przybliżenie ułamka:**
  $(1/10)_{10} = (0.0(0011))_2$

- **Najlepsze przybliżenia uzyskujemy dla ułamków o mianownikach będących kolejnymi potęgami dwójki.**

Numer bitu zaokrąglenia	Zaokrąglenie	Wartość zaokrąglenia
1	0.0	0/2
2	0.00	0/4
3	0.001	1/8
4	0.0010	2/16
5	0.00011	3/32
6	0.000110	6/64
7	0.0001101	13/128
8	0.00011010	26/256
9	0.000110011	51/512
…		

System stałopozycyjny

- **Liczby rzeczywiste mają część całkowitą i ułamkową, układ stałopozycyjny charakteryzuje się tym że przeznaczamy w nim stałą, z góry określoną liczbę (k) bitów na część całkowitą, tak jak i na ułamkową (u).**
- **Tym sposobem możemy przedstawić liczby z zakresu od -2^{k-1} do $2^{k-1} - 2^{-u}$, i wartości w nim reprezentowane są rozłożone równomiernie co 2^{-u}.**
- **Jest to sposób nieekonomiczny. Gdy operujemy na dużych liczbach (np. w astronomii) nie potrzebujemy przeznaczać pamięci na część ułamkową. Gdy za to operujemy bardzo małymi liczbami (np. fizyka cząstek elementarnych), nie potrzebujemy przeznaczać dużo pamięci na część całkowitą…**

System zmiennopozycyjny

- **System zmiennopozycyjny jest intuicyjny i powszechnie stosowany. Gdy chcemy zapisać np. stałą Plancka nikt nie będzie pisał:**
  - **h = 0,00000000000000000000000000000000663 J· s gdyż byłoby to nieczytelne.**

  Zostanie użyty natomiast zapis:
  - **h = 6,63· 10^{-34} J· s**

- **W takim zapisie podaje się kilka cyfr znaczących (mantysę) oraz określamy rząd wielkości (cechę) poprzez podanie potęgi podstawy systemu.**

System zmiennopozycyjny

- **Pojawia się problem jednoznaczności, np. liczbę 3/8 można przedstawić jako:**
  - $3/8 \cdot 2^0$
  - $3/4 \cdot 2^{-1}$
  - $3/2 \cdot 2^{-2}$
  - $3 \cdot 2^{-3}$ ….
  - $3/16 \cdot 2^1$
  - $3/32 \cdot 2^2$ …
- **Rozwiązaniem dla problemu jednoznaczności jest przyjęcie pewnego standardu.**
  - **Np. dla systemu o podstawie 2):**
    - **Mantysa musi mieścić się w przedziale (½, 1] dla wartości dodatnich oraz [-1, -½) dla wartości ujemnych.**
    - **Wyjątkiem jest reprezentacja zera.**

System zmiennopozycyjny

- **Każdą niezerową liczbę rzeczywistą reprezentujemy za pomocą przybliżenia wymiernego w postaci pary (m, c), takich że:**

  $m \in [-1, -½) \cup (½, 1]$

  gdzie m jest mantysą, zaś c jest cechą.

- **Interpretacja takiej reprezentacji wyraża się wzorem:**

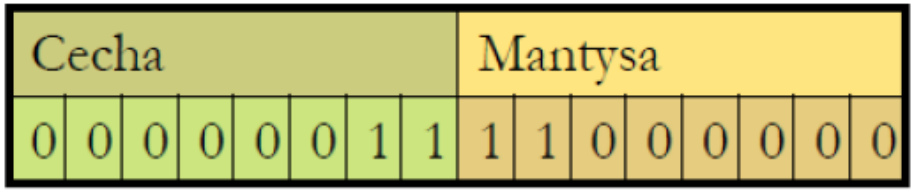
  $x = m P^c$

  gdzie dla tego przykładu $P = 2$ (podstawa systemu)

System ten umożliwia zapis liczb rzeczywistych z ustalonym błędem względnym.

System zmiennopozycyjny

- Liczba binarna zapisana w postaci cecha – mantysa na dwóch bajtach:



-

Tutaj: c = 3, m = - 1.

- W praktyce zwykle na cechę przeznaczamy jeden bajt, na mantysę minimum trzy bajty.
  - **Ilość bajtów przeznaczonych na cechę decyduje o zakresie.**
  - **Ilość bajtów przeznaczona na mantysę decyduje o błędzie.**
- Liczby w mantysie są kodowane w systemie znak – moduł.
- Zaś dla cechy w systemie uzupełnieniowym.

Standard zapisu liczb zmiennopozycyjnych

- **Standard IEEE 754 (ang. Institute of Electrical and Electronics Engineers) dla liczby rzeczywistej: (4 bajty)**

Kolejne bity (od lewej)	Znaczenie
1 (jeden)	Znak mantysy
2-9 (osiem)	Cecha
10-33 (dwadzieścia trzy)	Mantysa

- **Standard IEEE dla liczby podwójnej precyzji: (8 bajtów)**

Kolejne bity (od lewej)	Znaczenie
1 (jeden)	Znak mantysy
2-12 (jedenaście)	Cecha
13-64 (pięćdziesiąt dwa)	Mantysa

Standard IEEE – wartości specjalne

Wartość	Zapis
+0	Wszystkie bity są zerami.
-0	Bit znaku jest ustawiony, reszta jest zerami.
liczby małe (*ang. denormalized numbers*)	Wykładnik równy zero, mantysa różna od 0, nie zakłada się wiodącego niezerowego bitu; są to liczby zbyt małe aby mogły być reprezentowane z taką samą precyzją jak "zwykłe" liczby
$\pm\infty$	Ustawione są wszystkie bity cechy, mantysa jest równa 0, może się pojawić np. jako wynik dzielenia przez 0.
NaN (*ang. Not a Number*)	Ustawione są wszystkie bity cechy, mantysa różna od 0, może się pojawić np. jako wynik pierwiastkowania liczby ujemnej.

System zmiennopozycyjny

- W tym przykładzie przeznaczymy 3 bity na cechę, i 5 na mantysę.
- Jedyne legalne dodatnie mantysy to: 00101, 00110, 00111, 01000, gdyż $\mathbf{m} \in [-1, -\frac{1}{2}) \cup (\frac{1}{2}, 1]$
- Maksymalną możliwą do zapisania liczbą jest 8.
- Tworząc analogiczną tabelkę dla cech ujemnych, zauważymy iż minimalną dodatnią liczbą, możliwą do zapisania jest 5/128.
- Reprezentowane wartości nie są rozłożone równomiernie, im dalej od zera tym rzadziej.
- W obrębie jednej cechy wartości są rozłożone równomiernie

Cecha	Mantysa	Wartość
000	00101	5/8
000	00110	6/8
000	00111	7/8
000	01000	8/8
001	00101	10/8
001	00110	12/8
001	00111	14/8
001	01000	16/8
010	00101	20/8
010	00110	24/8
010	00111	28/8
010	01000	32/8

System zmiennopozycyjny a zero

- **Nie da się zera przedstawić w podanej postaci, gdyż żadna z liczb $m2^c$ nie może być zerem dla mantys co do modułu większych od ½ .**

- **Najczęściej stosowane rozwiązanie polega na tym, że wyłącza się jedną z cech (najmniejszą) i ustala, że jeśli liczba ma tę cechę, to jest równa zero niezależnie od mantysy.**

Błąd obliczeniowy

- **Błąd bezwzględny:** różnica między wartością zmierzoną/obliczoną x a wartością dokładną x_0

$$\Delta x = x - x_0$$

  - W systemie stałopozycyjnym obliczenia są wykonywane ze stałym max błędem bezwzględnym

- **Błąd względny:** błąd bezwzględny podzielony przez wartość dokładną

$$\delta x = \frac{\Delta x}{x_0}$$

  - W systemie zmiennopozycyjnym obliczenia są wykonywane ze stałym max błędem względnym

Znaki i teksty

- **Teksty składają się ze znaków.**
- **Podstawą zapisu jest jeden bajt.**
- **1 bajt przyjmuje 256 różnych wartości.**
- **Aby zapisać tekst posługujemy się kodowaniem**
- **Ważną cechą kodowania jest jednoznaczność:**
  - **przyjęcie pewnego sposobu kodowania powinno być powszechne:**
  - **ASCII: 0 – 127 standardowe, 128 – 255 zależne od kraju**

Np.
litera W: 01010111, czyli 87.

Znaki i teksty

Np.
litera W: 01010111, czyli 87.

Znaki specjalne	0-31, 127
Spacja	32
Cyfry	48-57
Duże litery	65-90
Małe litery	97-122
Pozostałe znaki: Kropka, nawiasy, itd....	33-47, 58-64, 91-96, 123-127

ASCII

- **ASCII (ang. American Standard Code for Information Interchange) - 7-bitowy kod przyporządkowujący liczby z zakresu 0-127 literom (alfabetu angielskiego), cyfrom, znakom przystankowym i innym symbolom oraz poleceniom sterującym.**
- **Litery, cyfry oraz inne znaki drukowane tworzą zbiór znaków ASCII. Jest to 95 znaków o kodach 32-126. Pozostałe 33 kody (0-31 i 127) to tzw. kody sterujące służące do sterowania urządzeniem odbierającym komunikat, np. drukarką czy terminalem.**

ASCII

- Ponieważ kod ASCII jest 7-bitowy, a większość komputerów operuje na 8-bitowych bajtach, dodatkowy bit można wykorzystać na powiększenie zbioru kodowanych znaków.
- Powstało wiele różnych rozszerzeń ASCII wykorzystujących ósmy bit (np. norma ISO 8859), nazywanych stronami kodowymi.
- Również kodowanie UTF-8 można uważać za rozszerzenie ASCII, tutaj jednak dodatkowe znaki są kodowane na 2 i więcej bajtach.

Znaki i teksty

- **W rozszerzonym kodzie ASCII znajdują się niektóre znaki matematyczne oraz znaki symulujące elementy grafiki na komputerach.**
- **Przetwarzanie informacji nie oznacza samego zapisywania tekstów. Dodatkowe informacje (wytłuszczenie, różne czcionki, akapity…) też trzeba zakodować.**
  - **Przykład: W kodzie ASCII znaki 0-31 i 127 nie są wykorzystane. Jeżeli umówimy się że po jednym z tych znaków następny zmienia znaczenie, to mamy 255 dodatkowych kodów.**
    Np. kod 65 występujący po tym wybranym znaku nie będzie oznaczać litery A tylko jedną z funkcji sterujących pracą edytora.

Obrazy, dźwięki, …

- Ciągi bajtów muszą przechowywać teksty, liczby, muzykę, animacje: wszystkie informacje zapisywane w wyniku wykonywanych działań.
- Potrzebne jest zakodowanie informacji, inne niż w przypadku liczb czy też tekstów.
- Kodowanie koloru: model RGB, model YUV.
- Kodowanie obrazu: formaty: BMP (bitmapa), GIF, JPEG
- Kodowanie muzyki: formaty: MP1, MP2, MP3, MP4, WAV, OGG, …
- Warto zauważyć pewną zależność: im większa precyzja, tym większy rozmiar pliku.

Kodowanie koloru – RGB

- Jeden z modeli przestrzeni barw, opisywanej współrzędnymi RGB.
  - Jego nazwa powstała ze złożenia pierwszych liter angielskich nazw barw: R – red (czerwonej), G – green (zielonej) i B – blue (niebieskiej), z których model ten się składa.
- Jest to model wynikający z właściwości odbiorczych ludzkiego oka, w którym wrażenie widzenia dowolnej barwy można wywołać przez zmieszanie w ustalonych proporcjach trzech wiązek światła o barwie czerwonej, zielonej i niebieskie.
- Model RGB jest jednak modelem teoretycznym a jego odwzorowanie zależy od urządzenia, co oznacza, że w każdym urządzeniu każda ze składowych RGB może posiadać nieco inną charakterystykę widmową, a co za tym idzie, każde z urządzeń może posiadać własny zakres barw możliwych do uzyskania.

Kodowanie koloru – RGB

- **Model RGB miał pierwotnie zastosowanie do techniki analogowej, obecnie ma również do cyfrowej. Jest szeroko wykorzystywany w urządzeniach analizujących obraz (np. aparaty cyfrowe, skanery) oraz w urządzeniach wyświetlających obraz (np. telewizory, monitory komputerowe).**
- **Zapis koloru jako RGB często stosuje się w informatyce (np. palety barw w plikach graficznych, w plikach html). Najczęściej stosowany jest 24-bitowy zapis kolorów (po 8 bitów na każdą z barw składowych), w którym każda z barw jest zapisana przy pomocy składowych, które przyjmują wartość z zakresu 0-255. W modelu RGB 0 (dla każdej ze składowych) oznacza kolor czarny, natomiast 255 (analogicznie) kolor biały.**
- **Kolor RGB można obliczyć tak:**
  - **numer koloru = R * 256^2 + G * 256 + B**

Kodowanie koloru – YUV

- Model barw, w którym Y odpowiada za jasność obrazu (luminancję), a pod UV zaszyta jest barwa - dwie chrominancje.
- Model YUV był wykorzystywany w czasie przechodzenia od telewizorów czarno-białych na kolorowe. Czarno-białe odbiorniki wyświetlały jedynie jasność obrazu, a kolorowe dodawały kolor, co pozwoliło posiadaczom czarno-białych nie pozbywać się odbiorników od razu. Y - luminacja (dla obrazu czarno-białego) U - przeskalowana składowa BV - przeskalowana składowa R
- Y = 0.299 *R + 0.587*G +0.114 *B
- U = B – Y
- V = R - Y

Kompresja

- **Jest to działanie mające na celu zmniejszanie objętości pliku. Przy kompresji wykorzystuje się podobieństwa i regularności występujące w plikach. Program przeprowadza analizę i wybiera fragmenty, które można zapisać w sposób zajmujący mniejszą liczbę bajtów. Wyróżniamy dwa typy:**
  - **Kompresja bezstratna: odtworzona informacja jest identyczna z oryginałem, dekompresja jest w pełni odwracalna (np. GIF).**
  - **Kompresja stratna: polega ona na eliminowaniu pewnych elementów oryginału, w celu lepszej efektywności kompresji (np. JPEG).**
- **Możemy powiązać jakość ze stopniem kompresji.**

Szyfrowanie

- **Szyfr – rodzaj kodu, system umownych znaków stosowany w celu utajnienia wiadomości, żeby była ona niemożliwa (lub bardzo trudna) do odczytania przez kogoś kto nie posiada odpowiedniego klucza.**
- **Szyfrowanie to procedura przekształcania wiadomości nie zaszyfrowanej w zaszyfrowaną.**
- **Wiadomość przed zaszyfrowaniem nazywa się tekstem jawnym (plaintext), a wiadomość zaszyfrowaną – szyfrogramem (ciphertext).**
- **Kryptologia – nauka o przekazywaniu informacji w sposób zabezpieczony przed niepowołanym dostępem.**
- **Kryptologię dzieli się na:**
  - **Kryptografię, czyli naukę o układaniu systemów kryptograficznych,**
  - **Kryptoanalizę, czyli naukę o ich łamaniu.**

Szyfrowanie

- Dwa najpopularniejsze algorytmy kryptografii asymetrycznej (czyli takiej, w której się używa zestawów dwu lub więcej powiązanych ze sobą kluczy, umożliwiających wykonywanie różnych czynności kryptograficznych) to RSA i ElGamal.
- RSA: Został stworzony w 1978 przez zespół: Ronald Rivest, Adi Shamir, Leonard Adleman (nazwa RSA jest akronimem utworzonym z pierwszych liter nazwisk jego twórców). RSA opiera się na trudności faktoryzacji dużych liczb. Znalezienie szybkiej metody faktoryzacji doprowadziłoby do złamania RSA, aczkolwiek nie ma dowodu, że nie da się go złamać w inny sposób.
- ElGamal: natomiast jest oparty na trudności problemu logarytmu dyskretnego w ciele liczb całkowitych modulo duża liczba pierwsza. Algorytm w połowie lat 80-tych przedstawił Egipcjanin Taher Elgamal. Algorytm ElGamala umożliwia szyfrowanie oraz obsługę podpisów cyfrowych.
- Setki modyfikacji algorytmu ElGamala (podobnie jak modyfikacje algorytmu RSA) mają różne inne zastosowania.

Wykład 1b

Struktury danych i algorytmy

- Typy danych i struktury danych
- Analiza algorytmów
- Sposoby zapisu algorytmów
- Rodzaje algorytmów
- Schematy blokowe i algografy
- Wybór algorytmu

* Niektóre przykłady z wykł prof. T. Roughgarden, Standford, USA

Struktury danych i algorytmy

- **Struktury danych to narzędzia do reprezentowania informacji która ma być przetworzona przez program komputerowy,**
- **Algorytmy to przepisy wykonania czynności niezbędnych do jej przetworzenia.**
- **Wybór algorytmu do rozwiązania konkretnego problemu programistycznego pomaga w ustaleniu, jaką strukturę danych należałoby użyć, ale i odwrotnie – wybrana struktura danych ma ogromny wpływ na szczegóły realizacji i efektywności algorytmu.**

Typy danych i struktury danych

- Dane są to „obiekty” którymi manipuluje algorytm.
- Te obiekty to nie tylko dane wejściowe lub wyjściowe (wyniki działania algorytmu), to również obiekty pośrednie tworzone i używane w trakcie działanie algorytmu.
- Dane mogą by różnych typów, do najpospolitszych należą liczby (całkowite, dziesiętne, ułamkowe) i słowa zapisane w rozmaitych alfabetach.

Typy danych i struktury danych

- **Interesują nas sposoby w jaki algorytmy mogą organizować, zapamiętywać i zmieniać zbiory danych oraz „sięgać" do nich.**
  - **Zmienne czyli „pudelka" w których chwilowo przechowujemy jakąś wartość,**
  - **Tablice czyli tabele (macierze), w których to możemy odwoływać się do indeksów,**
  - **Listy i wektory**
  - **Kolejki i stosy,**
  - **Drzewa, czyli hierarchiczna struktura danych,**
  - **Zbiory…. Grafy…. Relacje….**

Typy danych i struktury danych

- W wielu zastosowaniach same struktury danych nie wystarczają.
- Czasami potrzeba bardzo obszernych zasobów danych, stanowiących dla wielu algorytmów potencjalne dane wejściowe, a więc mające ustaloną strukturę i nadające się do odszukiwania i manipulowania nimi. Nazywa się je bazami danych (relacyjne i hierarchiczne).
- Kolejny krok to bazy wiedzy, których elementami są bazy danych, a które zawierają również informacje o związkach pomiędzy danymi.

Algorytmy

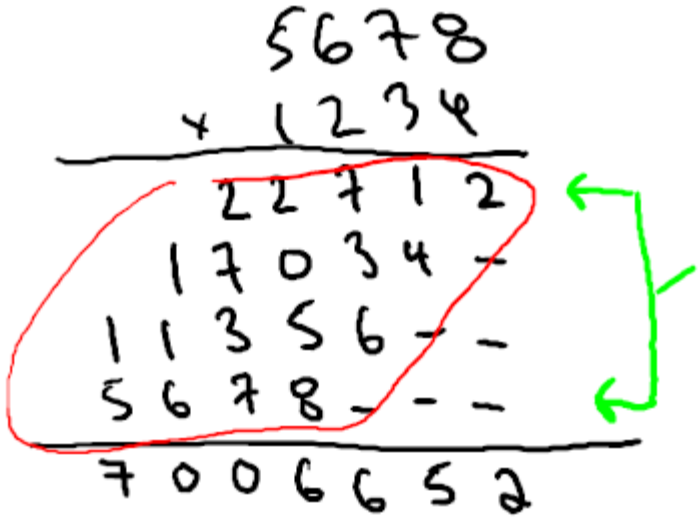
- **Algorytm to „przepis postępowania" prowadzący do rozwiązania konkretnego zadania; zbiór poleceń dotyczących pewnych obiektów (danych) ze wskazaniem kolejności w jakiej mają być wykonane". Jest jednoznaczną i precyzyjną definicją (specyfikacją) kroków które mogą być wykonywane „mechanicznie".**
- **Algorytm odpowiada na pytanie „jak to zrobić" postawione przy formułowaniu zadania. Istota algorytmu polega na rozpisaniu całej procedury na kolejne, możliwie elementarne kroki.**
- **Algorytmiczne myślenie można kształtować niezależnie od programowania komputerów, chociaż każdy program komputerowy jest zapisem jakiegoś algorytmu.**

Analiza algorytmów

- **Analiza algorytmów i powiązanych z nimi struktur danych.**
  - **Znalezienie najlepszych sposobów wykonywania najczęściej spotykanych poleceń,**
    - **musimy nauczyć się podstawowych technik projektowania dobrych algorytmów.**
  - **Zrozumienie w jaki sposób wykorzystywać struktury danych i algorytmy tak, by tworzyć efektywne (szybkie) programy.**

Mnożenie dwóch liczb całkowitych (szkolny algorytm)

- **Wejście: dwie n-cyfrowe liczby x, y**
- **Wyjście: z = x * y**
- **Operacje: mnożenie i dodawanie dwóch 1-dno cyfrowych liczb**



Ilość operacji:
~ n dla każdego wiersza

Ilość wszystkich operacji:
~ n * n = n^2
z dokładnością do stałej

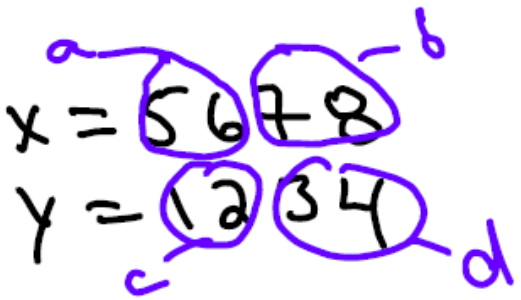
Mnożenie dwóch liczb całkowitych (algorytm Karatsuba)

- **Krok 1: policz** $a \cdot c = 672$
- **Krok 2: policz** $b \cdot d = 2652$
- **Krok 3: policz**

$(a+b)(c+d) = 134 \cdot 46 = 6164$

- **Krok 4: policz** $③ - ② - ① = 2840$
- **Krok 5:**

$$\begin{array}{r} 6720000 \\ 2652 \\ 284000 \\ \hline 7006652 \end{array} = (1234)(5678)$$



Mnożenie dwóch liczb całkowitych (algorytm Karatsuby)

- **Rozpisujemy liczby x, y:**

$$x = 10^{n/2}a + b \qquad y = 10^{n/2}c + d$$

a,b,c,d są liczbami n/2 cyfrowymi

- **Reprezentujemy iloczyn jako**

$$x.y = (10^{n/2}a + b)(10^{n/2}c + d)$$
$$= 10^n ac + 10^{n/2}(ad + bc) + bd$$

- Należy obliczyć:
  - (1) ac (2) bd (3) (a+b) (c+d) = ac+ad+bc+bd
  - Gauss trick: (3) – (1) – (2) = ad + bc

Kilka mnożeń i dodawań liczb n/2 cyfrowych, czy to jest lepszy algorytm?

Sposoby zapisu algorytmu

- Najprostszy sposób zapisu to zapis słowny
  - Pozwala określić kierunek działań i odpowiedzieć na pytanie, czy zagadnienie jest możliwe do rozwiązania.
- Bardziej konkretny zapis to lista kroków
  - Staramy się zapisać kolejne operacje w postaci kolejnych kroków które należy wykonać.
- Bardzo wygodny zapis to zapis graficzny
  - schematy blokowe i grafy.
- Bardziej zaawansowana forma to zapis przy pomocy uproszczonego kodu języka programowania tzw. pseudo-kod

Algorytm - przykład

Przykład: dodanie dwóch liczb

- **Sformułowanie zadania:**
  oblicz sumę dwóch liczb naturalnych: a,b. Wynik oznacz przez S.
- **Dane wejściowe: dwie liczby a i b**
- **Cel obliczeń: obliczenie sumy S = a + b**
- **Dodatkowe ograniczenia: sprawdzenie warunku dla danych wejściowych np. czy a, b są naturalne.**

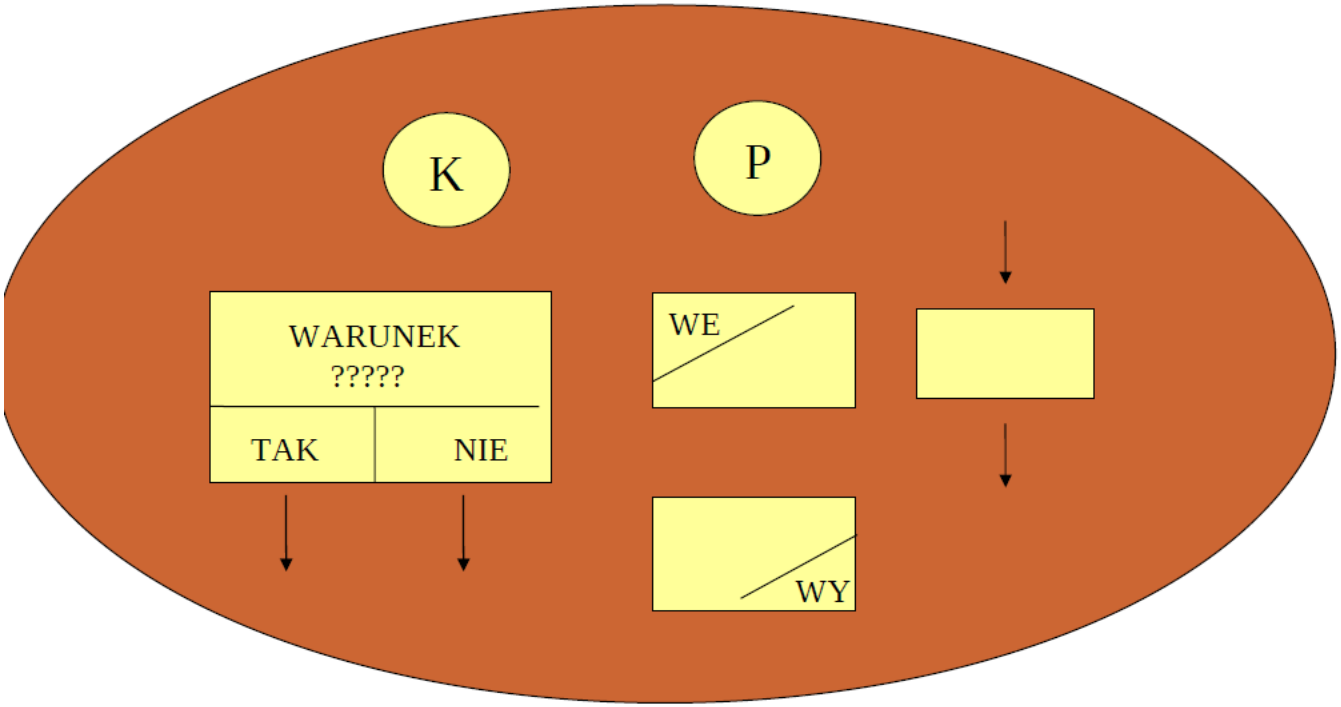
Zapis algorytmu: lista kroków

- **Zapis algorytmu przy pomocy listy kroków:**
  - **sformułowanie zagadnienia (zadanie algorytmu),**
  - **określenie zbioru danych potrzebnych do rozwiązania zagadnienia (określenie czy zbiór danych jest właściwy),**
  - **określenie przewidywanego wyniku (wyników): co chcemy otrzymać i jakie mogą być warianty rozwiązania,**
  - **zapis kolejnych ponumerowanych kroków, które należy wykonać, aby przejść od punktu początkowego do końcowego.**

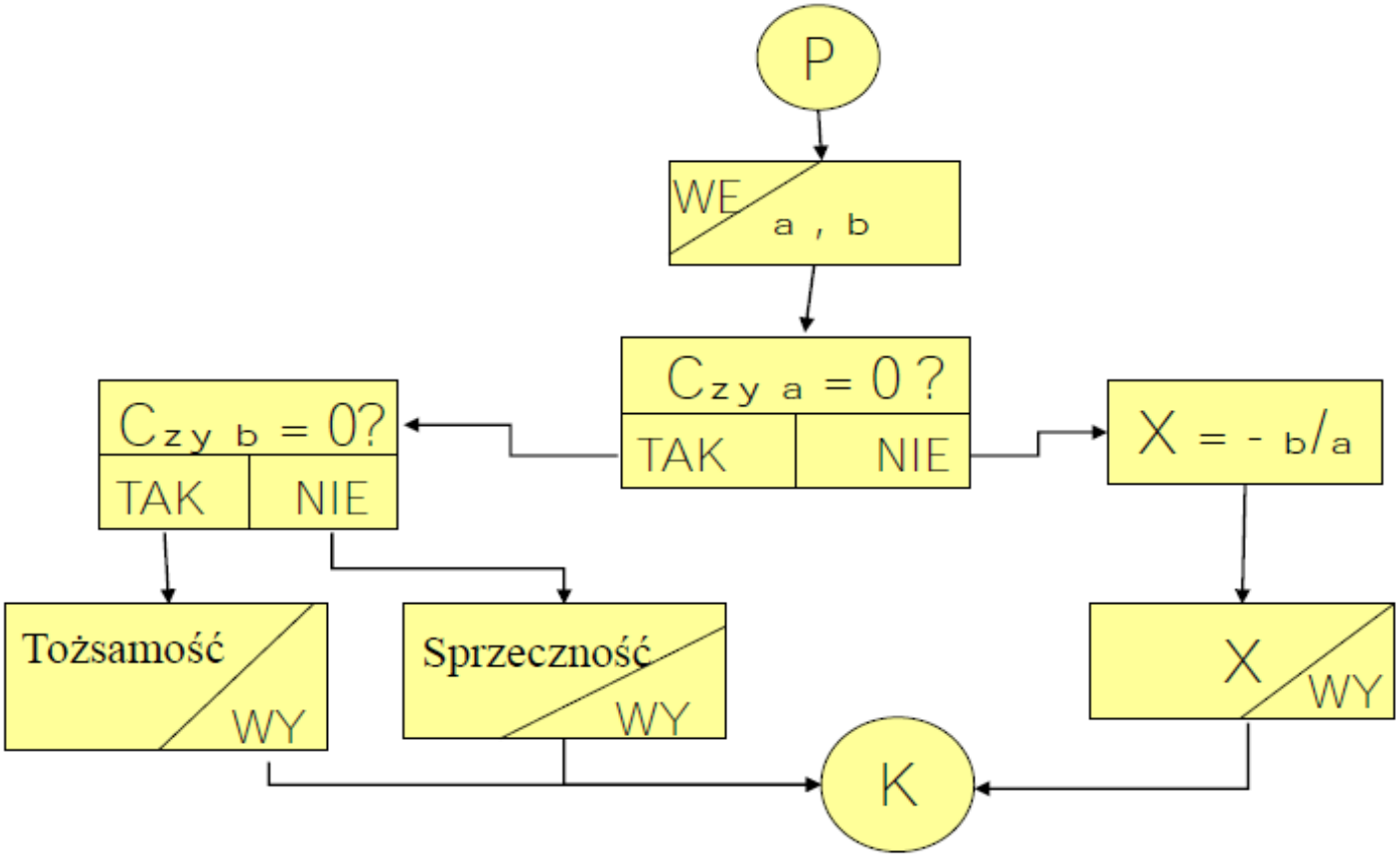
Algorytm - przykład

- **Sformułowanie zadania**
  Znajdź rozwiązanie równania liniowego postaci $a \cdot x + b = 0$.
  Wynikiem jest wartość liczbowa lub stwierdzenie dlaczego nie ma jednoznacznego rozwiązania.
- **Dane wejściowe**
  Dwie liczby rzeczywiste a i b
- **Cel obliczeń (co ma być wynikiem)**
  Obliczenie wartości x lub stwierdzenie, że równanie nie ma jednoznacznego rozwiązania.
  - **gdy $a = 0$ to sprawdź czy $b = 0$, jeśli tak to równanie sprzeczne lub tożsamościowe**
  - **gdy $a \neq 0$ to oblicz $x = -b/a$**

Schematy blokowe i algografy



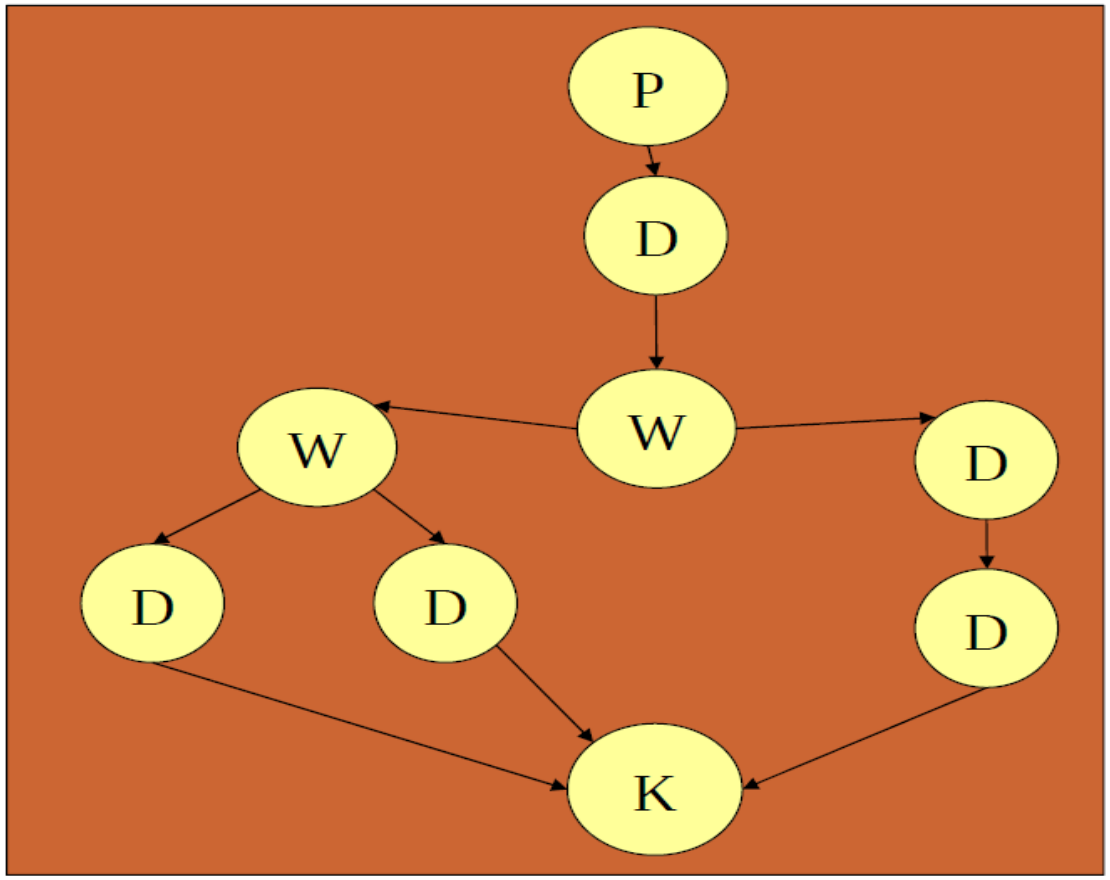
Schemat blokowy rozwiązania równania liniowego



Grafy

- **Graf symbolizuje przepływ informacji.**
- **Graf składa się z węzłów i ścieżek.**
- **W przypadku algorytmów graf można wykorzystać aby w uproszczonej formie zilustrować ilość różnych dróg prowadzących do określonego w zadaniu celu.**
- **Graf pozwala wykryć drogi, które nie prowadzą do punktu końcowego, których to poprawny algorytm nie powinien posiadać.**

Graf algorytmu rozwiązania równania liniowego



- P – początek
- K – koniec
- D – działanie
- W – warunek

Grafy

- **Jeżeli w grafie znajduje się ścieżka, która nie doprowadza do węzła końcowego, to mamy do czynienia z niepoprawnym grafem.**
- **W programie przygotowanym na podstawie takiego grafu, mamy do czynienia z przerwaniem próby działania i komunikatem o zaistnieniu jakiegoś błędu w działaniu.**
- **Węzeł grafu może mieć dwa wejścia jeżeli ilustruje pętle. Wtedy liczba ścieżek początek-koniec może być nieskończona, gdyż nieznana jest liczba obiegów pętli.**

Schemat blokowy czy graf ?

- Graf to tylko schemat kontrolny służący do sprawdzenia algorytmu.
  - Brak informacji o wykonywanych operacjach
- Schemat blokowy służy jako podstawa do tworzenia programów.

Rodzaje algorytmów

Algorytm liniowy:

- **Ma postać ciągu kroków których jest liniowa ilość (np. stała albo proporcjonalna do liczby danych) które muszą zostać bezwarunkowo wykonane jeden po drugim.**
- **Algorytm taki nie zawiera żadnych warunków ani rozgałęzień: zaczyna się od podania zestawu danych, następnie wykonywane są kolejne kroki wykonawcze, aż dochodzimy do wyniku.**

Rodzaje algorytmów

Algorytm z rozgałęzieniem:

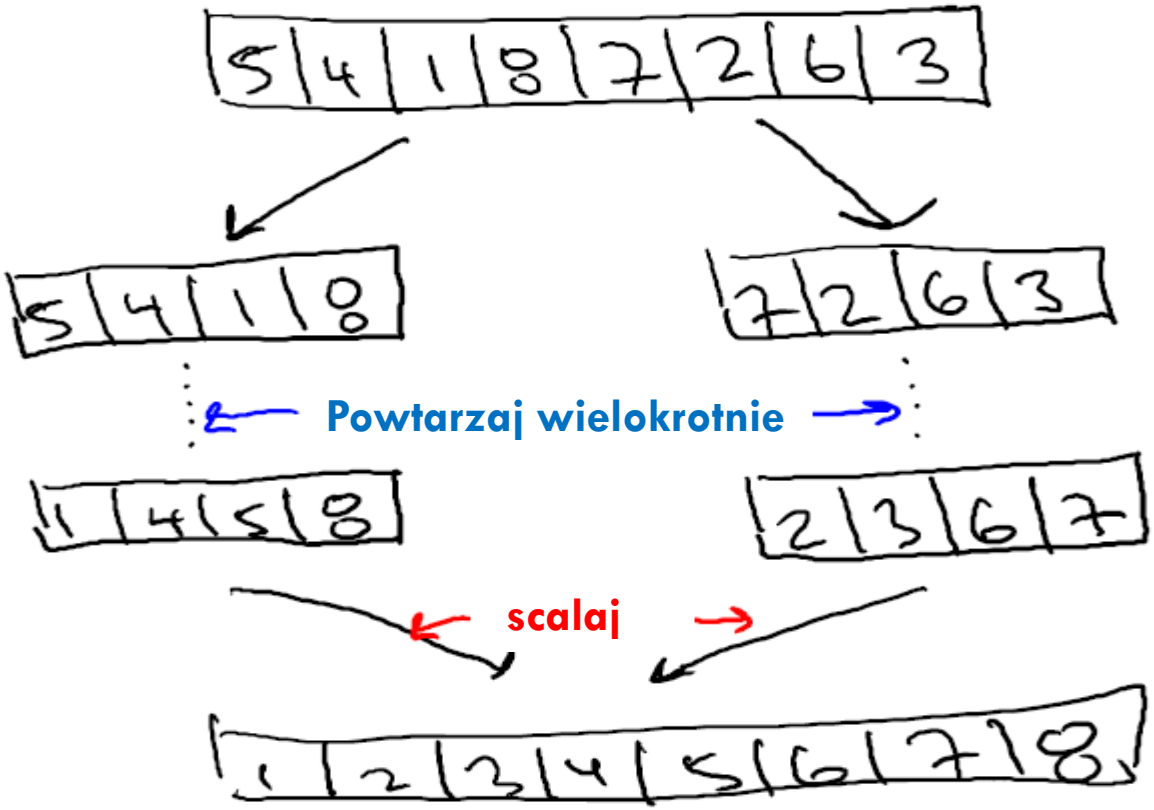
- Większość algorytmów zawiera rozgałęzienia będące efektem sprawdzania warunków. Wyrażenia warunkowe umożliwiają wykonanie zadania dla wielu wariantów danych i rozważanie różnych przypadków.
- Powtarzanie różnych działań ma dwojaką postać:
  - liczba powtórzeń jest z góry określona (przed rozpoczęciem cyklu), alg. najczęściej związany z działaniami na tablicach,
  - liczba powtórzeń jest nieznana (zależy od spełnienia pewnego warunku), alg. najczęściej związany z obliczeniami typu iteracyjnego.

Algorytmy: „dziel i zwyciężaj”

- **Metoda: „dziel i zwyciężaj” :**
  - **Dzielimy problem na mniejsze części tej samej postaci co pierwotny.**
  - **Teraz te pod-problemy dzielimy dalej na coraz mniejsze, używając tej samej metody, aż rozmiar problemu stanie się tak mały, że rozwiązanie będzie oczywiste lub będzie można użyć jakiejś innej efektywnej metody rozwiązania.**
  - **Rozwiązania wszystkich pod-problemów muszą być połączone w celu utworzenia rozwiązania całego problemu.**
- **Ten typ algorytmów zazwyczaj jest implementowany z zastosowaniem technik rekurencyjnych.**

Algorytmy: „dziel i zwyciężaj”

- **Mamy posortować tablicę liczb, zakładamy że są różn**



Algorytmy: „dziel i zwyciężaj”

- **Jak znaleźć minimum ciągu liczb?**
  - **Dzielimy ciąg na dwie części, znajdujemy minimum w każdej z nich, bierzemy minimum z obu liczb jako minimum ciągu.**

- **Jak sortować ciąg liczb?**
  - **Dzielimy na dwie części, każdą osobno sortujemy a następnie łączymy dwa uporządkowane ciągi (scalamy).**

Algorytmy oparte na programowaniu dynamicznym

- **Można stosować wówczas, kiedy problem daje się podzielić na wiele pod-problemów, których rozwiązania są możliwe do zapamiętania w jedno-, dwu- lub wielowymiarowej tablicy w taki sposób że w pewnej określonej kolejności można je wszystkie (a więc i cały problem) efektywnie rozwiązać.**

Jak obliczać ciąg Fibonacciego?

$$F(i) = \begin{cases} 1 & \text{jeśli } i = 1 \\ 1 & \text{jeśli } i = 2 \\ F(i-2)+F(i-1) & \text{jeśli } i > 2 \end{cases}$$

- **Aby obliczyć F(n), wartość F(k), gdzie k<n musimy wyliczyć F(n-k) razy.**
- **Liczba obliczeń rośnie wykładniczo.**
- **Korzystnie jest więc zachować (zapamiętać w tablicy) wyniki wcześniejszych obliczeń (F(k)).**

Jak obliczać liczbę kombinacji?

- **Liczba kombinacji (podzbiorów) r-elementowych ze zbioru n-elementowego oznaczana ($\frac{n}{r}$), dana jest wzorem:**

$$\binom{n}{r} = n!/(r!\,(n-r)!)$$

Możemy użyć wzorów:

$$\binom{n}{r} = 1, \text{ jeśli } r = 0 \text{ lub } n = r$$

$$\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r} \text{ dla } 0 < r < n$$

Obliczamy rząd po rzędzie w **trójkącie Pascala**

n \ r	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

Algorytmy z powrotami

- Często możemy zdefiniować jakiś problem jako poszukiwanie rozwiązania wśród wielu możliwych przypadków.
- Dane:
  - Pewna przestrzeń stanów, przy czym stan jest to sytuacją stanowiąca rozwiązanie problemu albo mogąca prowadzić do rozwiązania
  - Sposób przechodzenia z jednego stanu do drugiego.
  - Mogą istnieć stany które nie prowadzą do rozwiązania.

Przykładami tego typu algorytmów są gry.

Algorytmy z powrotami

Metoda powrotów

- Wymaga zapamiętania wszystkich wykonanych ruchów czy też wszystkich odwiedzonych stanów aby możliwe było cofanie się (powroty).
- Stanów mogą być tysiące lub miliony więc bezpośrednie zastosowanie metody powrotów, mogące doprowadzić do odwiedzenia wszystkich stanów, może być zbyt kosztowne.
- Inteligentny wybór następnego posunięcia, tzw. funkcja oceniająca, może znacznie poprawić efektywność algorytmu.
  - Np. aby uniknąć przeglądania nieistotnych fragmentów przestrzeni stanów.

Wybór algorytmu

- **Regułą jest że należy implementować algorytmy najprostsze, które wykonują określone zadanie.**
- **Prosty algorytm to**
  - **łatwiejsza implementacja, czytelniejszy kod**
  - **łatwość testowania**
  - **łatwość pisania dokumentacji,….**
- **Jeśli program ma działać wielokrotnie, jego wydajność i wykorzystywany algorytm stają się bardzo ważne.**
- **Błędy zaokrągleń, powstające przy reprezentacji liczb, a także przy wykonywaniu działań na nich rozwinęły się w samodzielna dziedzinę tzw. analiza numeryczna.**

Wybór algorytmu

- **Istnieją również inne zasoby, które należy niekiedy oszczędnie wykorzystywać w pisanych programach:**
  - **ilość przestrzeni pamięciowej wykorzystywanej przez zmienne**
  - **generowane przez program obciążenie sieci komputerowej**
  - **ilość danych odczytywanych i zapisywanych na dysku**
  - **mniej obliczeń to lepsza dokładność numeryczna (zaokrąglenia)**

Algorytm optymalnego dodawania liczb naturalnych

- **Posortuj dane od liczby najmniejszej do największej względem modułów.**
- **Dopóki są co najmniej dwa elementy w zbiorze powtarzaj następujące działanie:**
  - **Pobierz z posortowanego zbioru dwie najmniejsze wartości (dlaczego najmniejsze?)**
  - **Dodaj do siebie**
  - **Włóż do zbioru z powrotem wynik dodawania**

Problemem nietrywialnym jest wykonywanie tego szybko → użyj stosu.

Wybór algorytmu

- **Zrozumiałość i efektywność: to są często sprzeczne cele. Typowa jest sytuacja w której programy efektywne dla dużej ilości danych są trudniejsze do napisania/zrozumienia.**
  - **Np. sortowanie przez wybieranie (łatwy, nieefektywny dla dużej ilości danych) i sortowanie przez „dzielenie i scalanie" (trudniejszy, dużo efektywniejszy).**
- **Zrozumiałość to pojęcie względne, natomiast efektywność można obiektywnie zmierzyć: testy wzorcowe, analiza złożoności obliczeń.**

Efektywność algorytmu

- **Czas działania:**
  - **Oznaczamy przez funkcje T(n) liczbę jednostek czasu, które zajmuje wykonanie programu lub algorytmu w przypadku problemu o rozmiarze n.**
  - **Funkcje te nazywamy czasem działania. Dość często czas działania zależy od konkretnych danych wejściowych, nie tylko ich rozmiaru. W takim przypadku, funkcje T(n) definiuje się jako najmniej korzystny przypadek z punktu widzenia kosztów czasowych. Inną wyznaczaną wielkością jest też czas średni, czyli średni dla różnych danych wejściowych.**

Testy wzorcowe

- Podczas porównywania dwóch lub więcej programów zaprojektowanych do wykonywania tego samego zadania, opracowujemy niewielki zbiór typowych danych wejściowych które mogą posłużyć jako dane wzorcowe (ang. benchmark).
- Powinny być one reprezentatywne i zakłada się że program dobrze działający dla danych wzorcowych będzie też dobrze działał dla wszystkich innych danych.
- Np. test wzorcowy umożliwiający porównanie algorytmów sortujących może opierać się na jednym małym zbiorze danych, np. zbiór pierwszych 20 cyfr liczby ∏; jednym średnim, np. zbiór kodów pocztowych województwa krakowskiego; oraz na dużym zbiorze takim jak zbiór numerów telefonów z obszaru Krakowa i okolic.
- Przydatne jest też sprawdzenie jak algorytm działa dla ciągu już posortowanego (często działają kiepsko).

Uwagi końcowe

- **Na wybór najlepszego algorytmu dla tworzonego programu wpływa wiele czynników, najważniejsze to:**
  - **prostota,**
  - **łatwość implementacji**
  - **efektywność**

Pytania do wykładu

1) **Co to jest algorytm i jakie znasz sposoby jego zapisu?**
2) **Scharakteryzuj, na czym polegają następujące typy algorytmów:**
   - **liniowy**
   - **z rozgałęzieniem**
   - **z powrotami**
   - **„dziel i zwyciężaj”**
   - **zachłanny**
   - **oparty o programowanie dynamiczne**
3) **Według jakich kryteriów efektywność algorytmu?**
4) **W jaki sposób badamy czas działania algorytmu?**