Programowanie podstawowe

Łukasz Bojarski

Algorytm

Alan Turing

- angielski matematyk i kryptolog
- jeden z twórców informatyki
- twórca pojęcia maszyny Turinga
- twórca urządzenia służącego do łamania kodu Enigmy

Algorytm

Maszyna Turinga

- skończonego **alfabetu** symboli;
- skończonego zbioru **stanów**;
- nieskończonej **taśmy** z zaznaczonymi kwadratami, z których każdy może zawierać pojedynczy symbol;
- ruchomej **głowicy** odczytująco - zapisującej, która może wędrować wzdłuż taśmy przesuwając się na raz o jeden kwadrat
- diagramu przejść między **stanami**, zawierającego instrukcje, które powodują, że zmiany następują przy każdym zatrzymaniu się

Algorytm

jest skończonym, uporządkowanym ciągiem jasno zdefiniowanych czynności, koniecznych do wykonania postawionego zadania

Algorytm

Cechy algorytmów:

- poprawność
- jednoznaczność
- skończoność
- sprawność

Algorytm

Sposoby zapisywania:

- opis słowny
- lista kroków
- schemat blokowy
- pseudokod
- język programowania np. Java

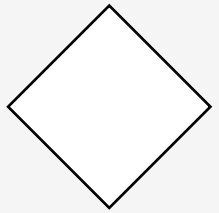
Algorytm

Schemat blokowy

- początek działań schematu

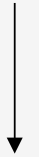
- blok wejścia-wyjścia

- blok obliczeniowy

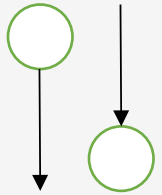


- blok decyzyjny

- zakończenie działania schematu



- łącznik poszczególnych bloków



- łącznik wewnątrzstronicowy

Algorytm

Zadanie

Narysuj schemat blokowy sprawdzający czy suma dwóch liczb jest podzielna przez 2.

Algorytm

Pytanie

Kto wie jak zrobić pętlę *for* w schemacie blokowym?

Algorytm

Pseudokod

:= albo ← przypisanie wartości do zmiennej – Java =

= równe – Java ==

<> nierówne/różne – Java !=

< mniejsze od

> większe od

<= mniejsze lub równe

>= większe lub równe

Algorytm

Pseudokod

set a – pobranie zmiennej a

print a – wypisanie zmiennej a

if a > 0 **then** - instrukcja warunkowa

    <instrukcja>

else

    <instrukcja>

Algorytm

Pseudokod

```
for i := 1 to 10 do                - pętla for
   <instrukcja>

while i < 10 do                    - pętla while
   <instrukcja>

begin                              - blok kodu
  <instrukcja>
end
```

Algorytm

Zadanie

Napisz pseudokod dla poprzedniego zadania.

Struktury danych

sposób przechowywania informacji w komputerze

Struktury danych

Przykłady:

- rekord
- tablica
- stos
- kolejka
- lista
- drzewo
- graf

Struktury danych

Tablica – kontener uporządkowanych danych, w którym poszczególne elementy dostępne są za pomocą kluczy (indeksu).

int[] tab = new int[10] – tablica jednowymiarowa typów int

int[][] tab = new int[10][10] – tablica dwuwymiarowa typów int

Struktury danych

Zadanie

Dana jest tablica liczb całkowitych. Wypisz:

- wszystkie liczby od końca będące na parzystych indeksach,
- sumę tylko tych liczb podzielnych przez 3,
- wynik sumy 5 początkowych liczb i odejmując ostatni element tablicy, zakładając, że tablica jest rozmiaru co najmniej 6.

Struktury danych

Stos - liniowa struktura danych, w której dane dokładane są na wierzch stosu i z wierzchołka stosu są pobierane

(LIFO – Last In, First Out)

Operacje:

- push – dodanie na szczyt stosu nowego elementu
- pop – usunięcie ze szczytu stosu elementu
- isEmpty– zwraca *true*, jeżeli stos nie zawiera żadnego elementu, wpp. *false*
- peek– zwraca element ze szczytu stosu

Struktury danych

Zadanie

Stwórz stos za pomocą tablicy.

Struktury danych

Zadanie

Stwórz stos za pomocą listy.

Struktury danych

Kolejka - liniowa struktura danych, w której nowe dane dopisywane są na końcu kolejki, a z początku kolejki pobierane są dane do dalszego przetwarzania.

(FIFO – First In, First Out)

Operacje:

- add/engueue – dodanie nowego elementu na koniec kolejki
- poll/degueue – usunięcie pierwszego elementu z kolejki
- isEmpty – zwraca *true*, jeżeli kolejka nie zawiera żadnego elementu, wpp. *false*
- peek – zwraca pierwszy element w kolejce

Zadanie

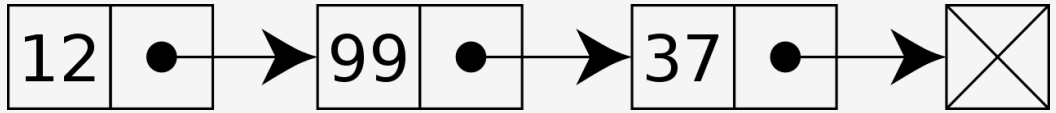
Korzystając z poprzedniego zadania, stwórz kolejkę za pomocą listy.

Struktury danych

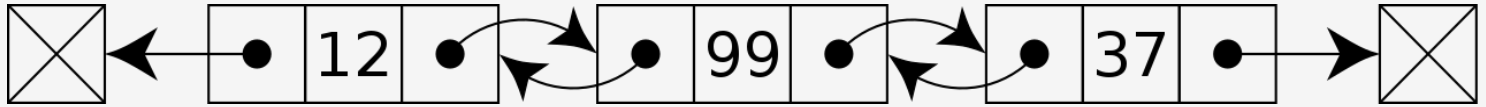
Lista - struktura danych służąca do reprezentacji zbiorów dynamicznych, w której elementy ułożone są w liniowym porządku.

Przykłady:

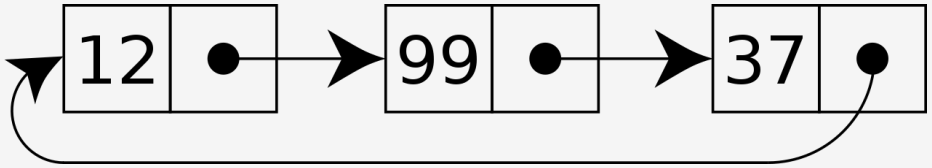
- jednokierunkowa



- dwukierunkowa



- cykliczna



Struktury danych

Zadanie domowe

Na podstawie zdobytej wiedzy, stwórz listę dwukierunkową.

Elementy pojedynczego obiektu:

- value
- prev
- next

Elementy listy:

- first albo head
- last albo tail

Metody listy:

- addFirst/addLast
- peekFirst/peekLast
- pollFirst/pollLast
- isEmpty
- show/showReverse

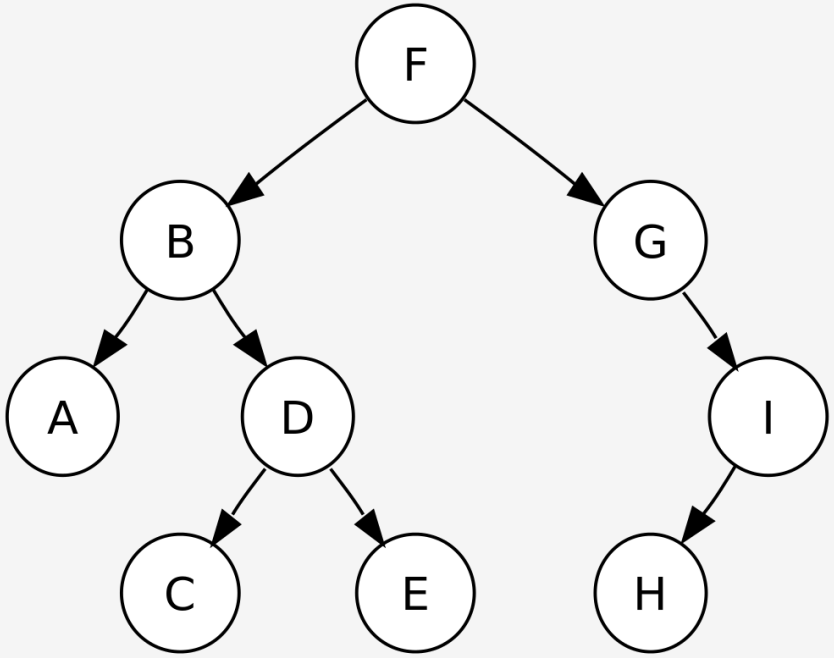
Struktury danych

Drzewo - jest strukturą danych zbudowaną z elementów, które nazywamy **węzłami**. Dane przechowuje się w węzłach drzewa. Węzły są ze sobą powiązane w sposób hierarchiczny za pomocą **krawędzi**, które zwykle przedstawia się za pomocą strzałki określającej hierarchię. Pierwszy węzeł drzewa nazywa się **korzeniem**. Od niego "wyrastają" pozostałe węzły, które nazywamy **dziećmi**. Węzeł nadrzędny w stosunku do dziecka nazywamy **rodzicem.** Jeśli węzeł nie posiada dzieci, to nazywa się **liściem.**

Drzewo, w którym węzły mogą posiadać co najwyżej dwoje dzieci, nazywa się **drzewem binarnym**

Zadanie

Dla przedstawionego obok drzewa wyznacz korzeń, rodziców, dzieci oraz liście. Czy jest to drzewo binarne?



Struktury danych

Drzewo

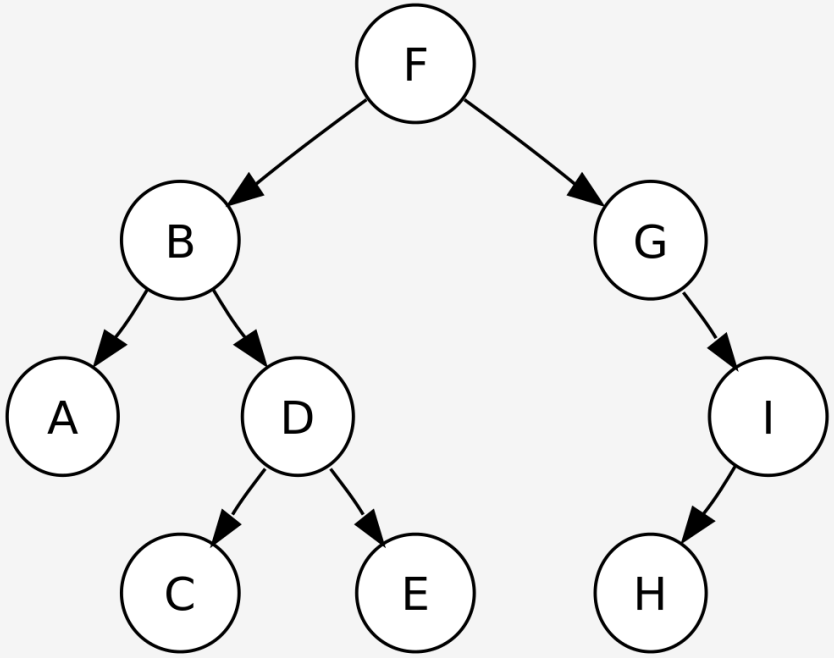
Ciąg krawędzi łączących węzły nazywa się **ścieżką**.

Liczba krawędzi w ścieżce od korzenia do węzła jest nazywana **długością** – liczba ta określa **poziom** węzła.

Wysokością drzewa jest największy poziom istniejący w drzewie.

Zadanie

Dla przedstawionego obok drzewa wyznacz przykładowe ścieżki oraz ich długości. Jaka jest wysokość drzewa?



Struktury danych

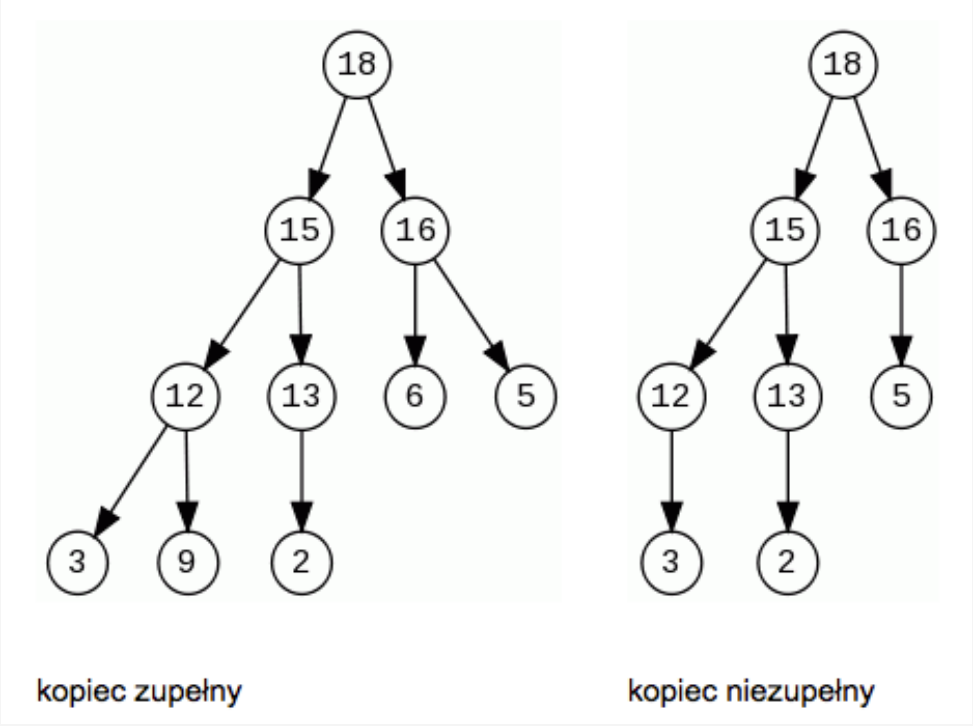
Kopiec (inaczej stóg) – jest drzewem binarnym, spełniającym warunek, że każdy następnik jest niewiększy od swojego poprzednika.

czyli:

- w korzeniu kopca znajduje się największy lub jeden z grupy największych o identycznej wartości
- na ścieżkach (połączeniach między węzłami), od korzenia do liścia, elementy są posortowane nierosnąco

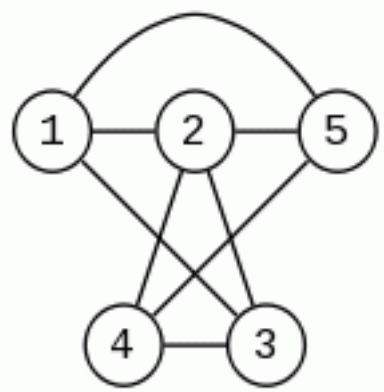
Przykład wykorzystania: sortowanie przez kopcowanie (heapsort)

Kopiec zupełny – to kopiec będący **zupełnym** drzewem binarnym.

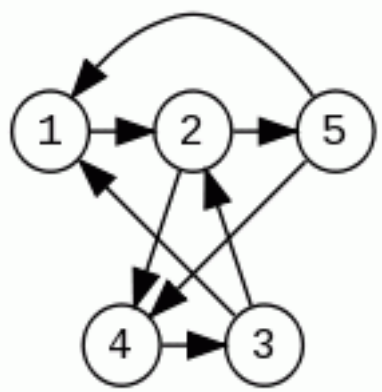


Struktury danych

Graf - struktura danych, która składa się z wierzchołków i krawędzie, przy czym poszczególne wierzchołki (węzły) mogą być połączone krawędziami (skierowanymi lub nieskierowanymi) w taki sposób, iż każda krawędź zaczyna się i kończy w którymś z wierzchołków.



graf nieskierowany



graf skierowany

Algorytmy

klasyfikacje i sposoby implementacji

Algorytmy

Klasyfikacja algorytmów:

- **dziel i zwyciężaj** – dzielimy problem na kilka mniejszych, a te znowu dzielimy, aż ich rozwiązania staną się oczywiste
- **metoda zachłanna** – nie analizujemy podproblemów dokładnie, tylko wybieramy najbardziej obiecującą w danym momencie drogę rozwiązania
- **programowanie dynamiczne** – problem dzielony jest na kilka, ważność każdego z nich jest oceniana i po pewnym wnioskowaniu wyniki analizy niektórych prostszych zagadnień wykorzystuje się do rozwiązania głównego problemu
- **programowanie liniowe** – oceniamy rozwiązanie problemu przez pewną funkcję jakości i szukamy jej minimum
- **wyszukiwanie wyczerpujące (*brute force)*** – przeszukujemy zbiór danych, aż do odnalezienia rozwiązania
- **heurystyka** – człowiek na podstawie swojego doświadczenia tworzy algorytm, który działa w najbardziej prawdopodobnych warunkach

Algorytmy

Programowanie liniowe – przykład:

Mieszanka	Kawa			zysk
	brazylijska [kg]	kolumbijska [kg]	peruwiańska [kg]	[zł/kg]
Northwest Passage	2	4	4	80
Sunrise Blend	4	5	1	60
Harbormaster	3	3	4	40
French Expedition	7	2	1	50
zapas [kg]	800	640	600	

$$\text{zmaksymalizować} \quad f(x_1, x_2, x_3, x_4) = 80x_1 + 60x_2 + 30x_3 + 50x_4$$

Algorytmy

Sposoby implementacji:

- **proceduralność** – podział algorytmu na szereg procedur
- **sekwencyjność** – wykonywanie poszczególnych procedur algorytmu, według kolejności ich wywołań
- **równoległość** – część zadań wykonuje się jednocześnie, wymieniając się danymi
- **rekurencyjność** – funkcja wywołuje sama siebie, aż do uzyskania wyniku
- **obiektowość** – procedury i dane łączy się w pewne klasy reprezentujące najważniejsze elementy algorytmu oraz stan wewnętrzny wykonującego je systemu
- **probabilistyczność** (**randomizowość)** – działa szybko, ale wynik nie jest pewny

Przykładowe algorytmy

Zadanie

Napisz program, który będzie sprawdzał czy w tablicy znajduje się dana liczba.

Przykładowe algorytmy

Binary search (wyszukiwanie binarne) – szukanie danego elementu w tablicy uporządkowanej. Metoda „dziel i zwyciężaj”.

Przykładowe algorytmy

Binary search (wyszukiwanie binarne) – pseudokod:

```
A := [...]       { n-elementowa tablica uporządkowana }
lewo := 1        { indeks początku przedziału }
prawo := n     { indeks końca przedziału - początkowo cała tablica A }

y := poszukiwana wartość
indeks := pusty

while lewo < prawo do
   begin
      środek := (lewo + prawo) div 2; { dzielenie całkowitoliczbowe }

      if A[środek] < y then
         lewo := środek + 1
      else
         prawo := środek;
   end;

if A[lewo] = y then
   indeks := lewo
else
   indeks := brak;
```

Przykładowe algorytmy

Zadanie

Narysuj schemat blokowy, a następnie zaimplementuj algorytmu wyszukiwania binarnego.

Przykładowe algorytmy

Algorytm Euklidesa (GCD) - wyznacza największy wspólny dzielnik (NWD) dwóch liczb.

Przykładowe algorytmy

Algorytm Euklidesa (GCD) - wyznacza największy wspólny dzielnik (NWD) dwóch liczb.

Lista kroków:

1. Weźmy dwie liczby całkowite dodatnie a i b.
2. Jeśli b = 0 to przejdź do kroku 3., wpp. wykonaj:
   2.1 Jeśli a > b to a := a – b
   2.2 Wpp. b := b – a
   2.3 Przejdź do kroku 2.
3. a jest szukanym największym dzielnikiem.
4. Koniec

Przykładowe algorytmy

Zadanie

Narysuj schemat blokowy, napisz pseudokod a następnie zaimplementuj algorytm Euklidesa wykorzystujący odejmowanie.

Przykładowe algorytmy

Algorytm Euklidesa (GCD) – wykorzystujący resztę z dzielenia.

Lista kroków:

1. Weźmy dwie liczby całkowite dodatnie a i b.
2. Jeśli b = 0 to przejdź do kroku 3., wpp. wykonaj:
   2.1 r := a mod b
   2.2 a := b
   2.3 b := r
   2.4 Przejdź do kroku 2.
3. a jest szukanym największym dzielnikiem.
4. Koniec

Przykładowe algorytmy

Zadanie

Zaimplementuj algorytm Euklidesa wykorzystujący resztę z dzielenia.