A1
B1
B2
C1

Drzewo – struktura języków programowania

M@я3k Pų∂€£kØ

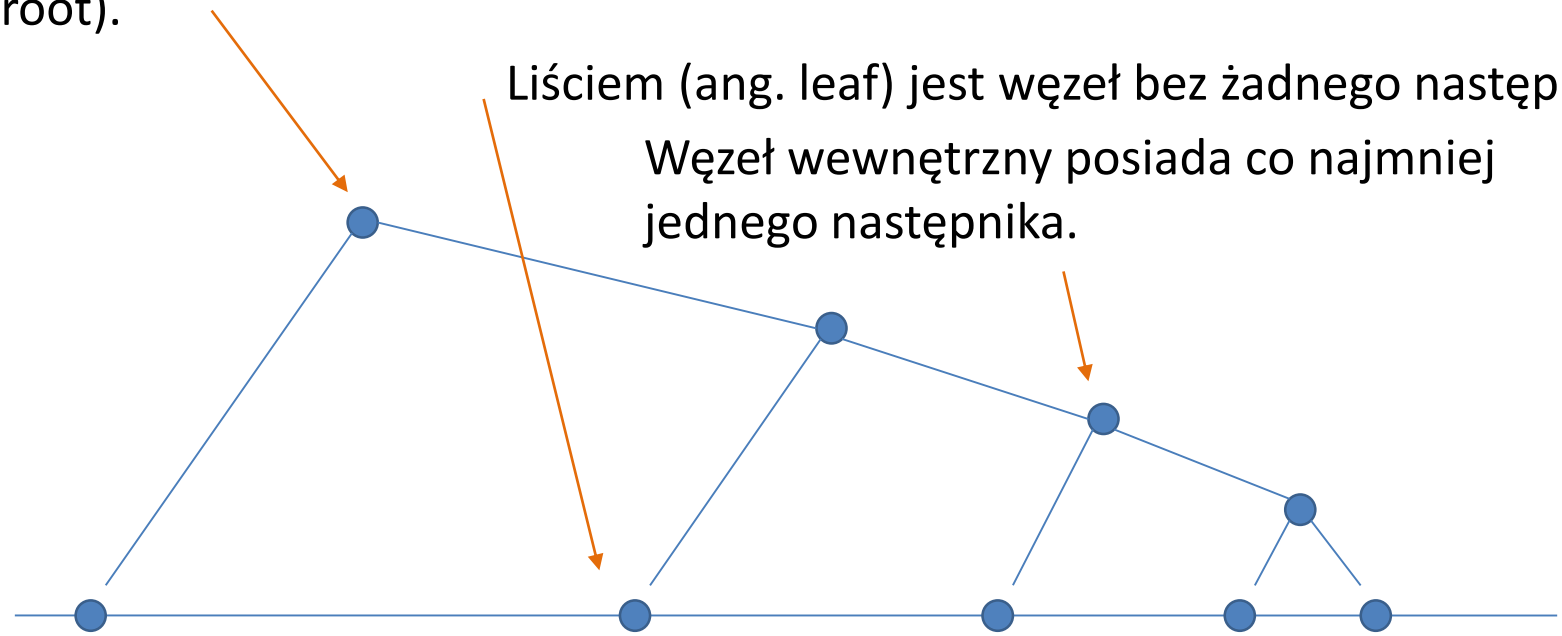
Programowanie w Pythonie

Spis treści

Drzewo jako struktura danych

- zbiór węzłów powiązanych wskaźnikami, spójny i bez cykli.
- Drzewo jest strukturą hierarchiczną.

- Drzewo posiada wyróżniony węzeł początkowy nazywany korzeniem (ang. root).
- Liściem (ang. leaf) jest węzeł bez żadnego następnika.
- Węzeł wewnętrzny posiada co najmniej jednego następnika.

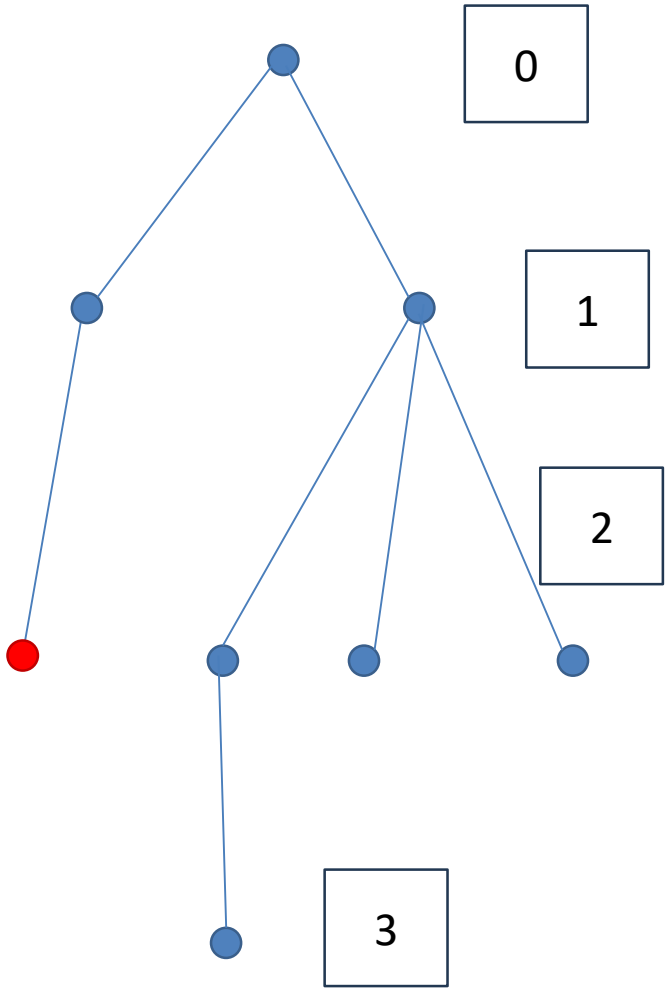


Zastosowanie drzewa

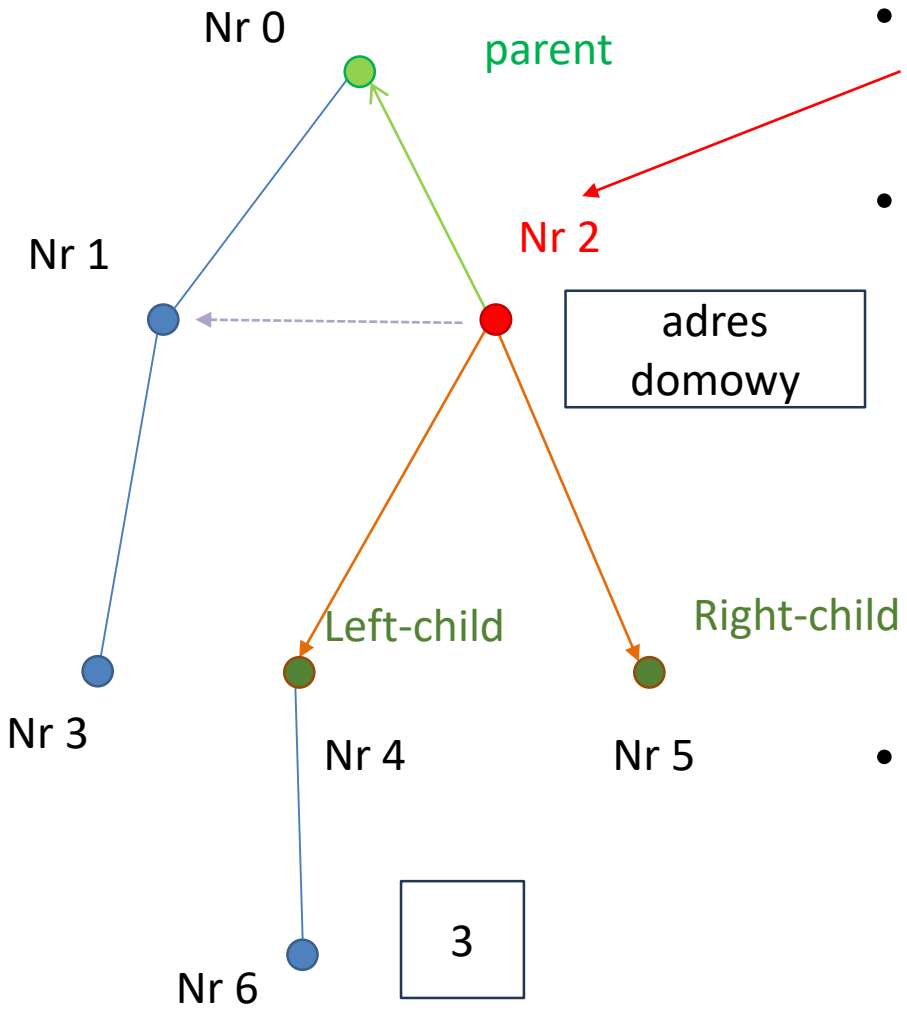
- Drzewa to nieliniowe struktury danych służące do reprezentowania hierarchii, umożliwiające szybkie wyszukiwanie, wstawianie i usuwanie elementów w czasie logarytmicznym.
- **Główne zastosowania struktur drzewiastych:**
- **Systemy plików i katalogów:** Struktura folderów w systemach operacyjnych (Windows, Linux) bazuje na drzewie, gdzie katalogi są węzłami nadrzędnymi, a pliki podrzędnymi.
- **Bazy danych (indeksowanie):** Drzewa B i B+ są używane do szybkiego wyszukiwania, wstawiania i usuwania rekordów w systemach baz danych.
- **Wyszukiwanie i sortowanie:** Drzewa poszukiwań binarnych (BST) oraz zrównoważone drzewa (np. AVL, Czerwono-Czarne) zapewniają szybki dostęp do uporządkowanych danych.
- **Frontend i dokumenty:** Drzewo DOM (Document Object Model) w przeglądarkach internetowych reprezentuje strukturę strony HTML.
- **Algorytmy tekstowe:** Drzewa trie (drzewa przedrostkowe) są stosowane do autouzupełniania tekstu (autocomplete) oraz w słownikach.
- **Wyrażenia matematyczne:** Drzewa wyrażeń służą do reprezentacji i obliczania wartości wyrażeń arytmetycznych.
- **Reprezentacja hierarchii:** Drzewa genealogiczne, struktury organizacyjne w firmach, drzewa decyzji w uczeniu maszynowym.

Parametry drzewa

- Wysokość drzewa – długość najdłuższej ścieżki od korzenia do liścia (liczba węzłów)
  - liczba poziomów na których zapisane jest drzewo.
- Głębokość węzła – długość ścieżki od korzenia do tego węzła (liczba węzłów)
  - numer poziomu, na którym znajduje się węzeł.
- Poziomy w drzewie numeruje się od 0
  - korzeń znajduje się na poziomie 0.



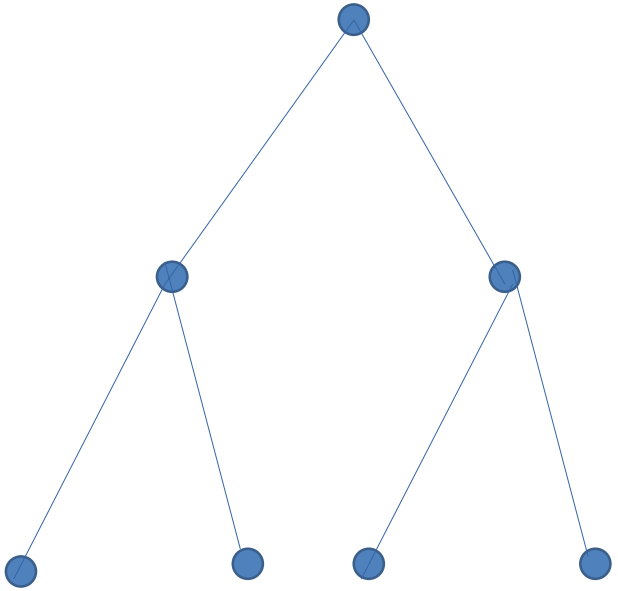
Drzewo uporządkowane



- Drzewo uporządkowane ma ponumerowanych (oznaczonych) następników.
- Drzewo uporządkowane składa się z węzłów, które zawierają następujące pola:
  - info – wartość pamiętana w węźle,
  - parent – wskaźnik na poprzednika,
  - left-child – wskaźnik na lewego syna (lista następników),
  - sibling – wskaźnik na prawego brata.
- Każdy węzeł w drzewie przechowuje jedną wartość.

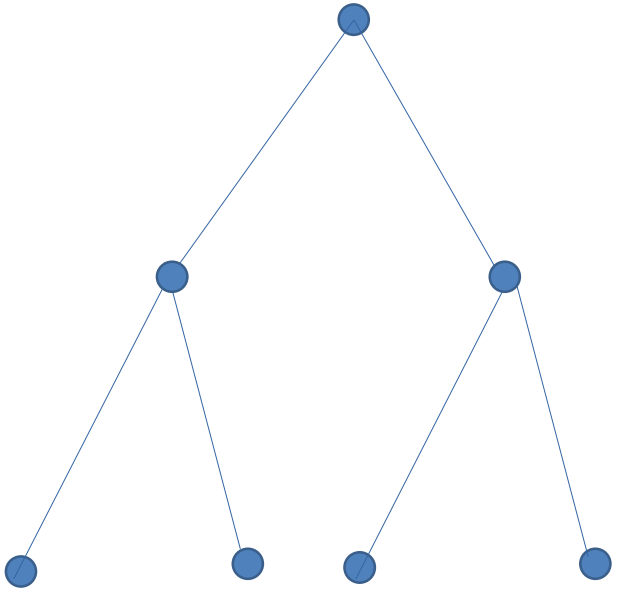
Drzewo binarne

- Drzewo binarne (ang. binary tree) to drzewo ukorzenione, w którym każdy wierzchołek ma dwóch synów – lewego i prawego.
- Węzeł drzewa binarnego ma pole z kluczem oraz wskaźnik na lewe poddrzewo i wskaźnik na prawe poddrzewo.
- Wysokość binarnego drzewa n-elementowego jest nie mniejsza niż log(n) i nie większa niż n.



Drzewo binarne dwukierunkowe

- W drzewie binarnym poruszamy się tylko od korzenia w kierunku jakiegoś liścia.
- Jeśli w węźle drzewa binarnego znajduje się wskaźnik na ojca, to w drzewie takim można się przemieszczać w obu kierunkach – od korzenia w kierunku jakiegoś liścia oraz w drugą stronę w kierunku korzenia.
- Takie drzewo nazywa się dwukierunkowym.



Szczególne przypadki drzew binarnych

Drzewo regularne	drzewo binarne, w którym każdy węzeł wewnętrzny na dwóch synów
Drzewo pełne	drzewo regularne, w którym wszystkie liście są na tym samym poziomie
	Drzewo pełne o wysokości h ma 2^h-1 węzłów.
Drzewo zupełne	drzewo pełne, z którego usunięto część liści z prawej strony
	Drzewo zupełne o wysokości h ma od 2^{h-1} do 2^h-1 węzłów.

Przykładowy kod

```
drzewo =    [['a',1,2],
             ['b',3,4],
             ['c',5,6],
             ['d',-1,-1],
             ['e',-1,-1],
             ['f',-1,-1],
             ['g',-1,-1],
             ]

print (drzewo)

for i in range (len (drzewo)):
    print (drzewo[i][0])
    lewy  = drzewo[i][1]
    prawy = drzewo[i][2]
    if lewy != -1:
        print (, lewy', lewy)
    if prawy != -1:
        print ('prawy', prawy)

odleglosc = 16
n = 1
for i in range (3):
    for k in range (odleglosc):
        print (' ', end="")
    print (drzewo[i][0])
    odleglosc = int (odleglosc /2)
```

```
drzewo =    [['a', 1, 2],
             ['b', 3, 4],
             ['c', 5, 6],
             ['d',-1,-1],
             ['e',-1,-1],
             ['f',-1,-1],
             ['g',-1,-1],
            ]

print (drzewo)

for i in range (len (drzewo)):
    print (drzewo[i][0])
    lewy  = drzewo[i][1]
    prawy = drzewo[i][2]
    if lewy != -1:
        print (" \\")
        print ('  `-lewy nr', lewy,drzewo[lewy][0])
    if prawy != -1:
        print ("  |")
        print ('   prawy nr', prawy, drzewo[prawy][0])
    if prawy == -1 and lewy == -1:
        print ("   liść")
print ("-----------------------")
odleglosc = 16
n = 1

for i in range (3):
    for j in range (n-1, 2*n-1):
        for k in range (odleglosc):
            print (' ', end="")
        print (drzewo[j][0], end = "")

    odleglosc = int (odleglosc /2) + 1
    n *= 2
    print ()
```

Przykładowy kod c.d.

```
for i in range (3):
    for j in range (n-1, 2*n-1):
        for k in range (odleglosc):
            print (' ', end="")
        if j % 2 == 0:
            print ("\\", end="")
        else:
            print ("/", end="")
    print ()
    for j in range (n-1, 2*n-1):
        for k in range (odleglosc):
            print (' ', end="")
        if j % 2 == 0:
            print (" ", drzewo[j][0], end = "")
        else:
            print (drzewo[j][0], end = "")
```