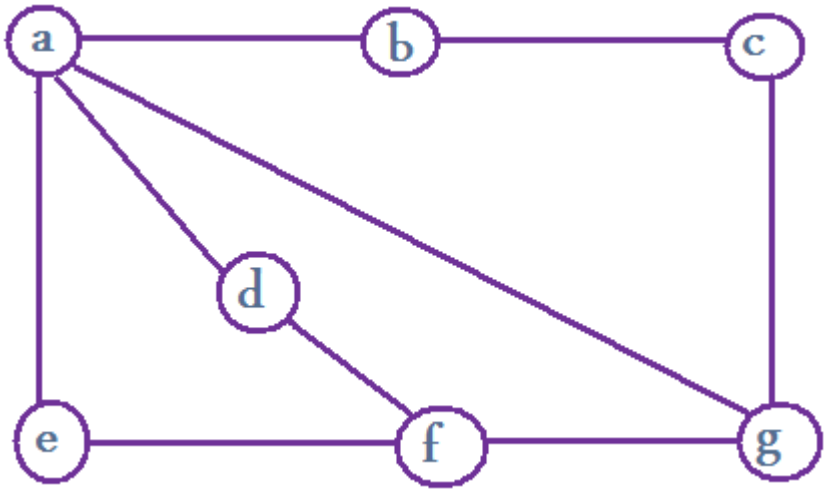
DRZEWA BINARNE

Graf

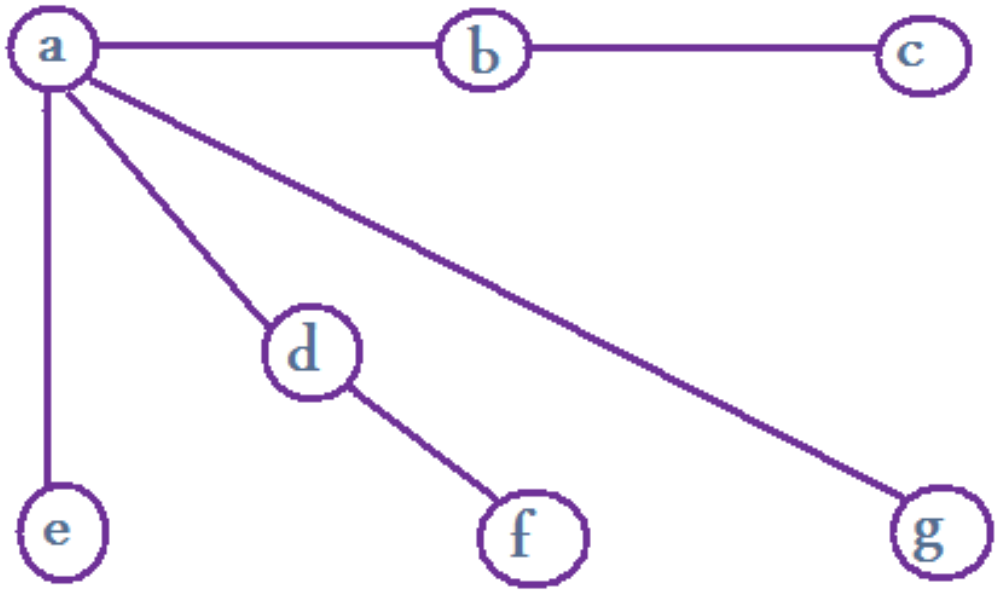
- Graf nieskierowany G=(V, E) jest to para składająca się ze skończonego zbioru wierzchołków V oraz ze zbioru krawędzi E, gdzie krawędzie to pary wierzchołków
- Przykład grafu:



Drzewo

- ***Drzewo*** to spójny i acykliczny graf
- Graf jest ***spójny***, gdy dla każdych dwóch wierzchołków u, v grafu istnieje ścieżka je łącząca
- ***Droga*** lub ***ścieżka*** w grafie to ciąg jego wierzchołków, w którym sąsiednie wierzchołki są połączone krawędzią
- Drogę nazywamy ***cyklem*** jeżeli wierzchołek początkowy drogi jest również wierzchołkiem końcowym, a wszystkie pozostałe wierzchołki tej drogi są różne

Przykład drzewa

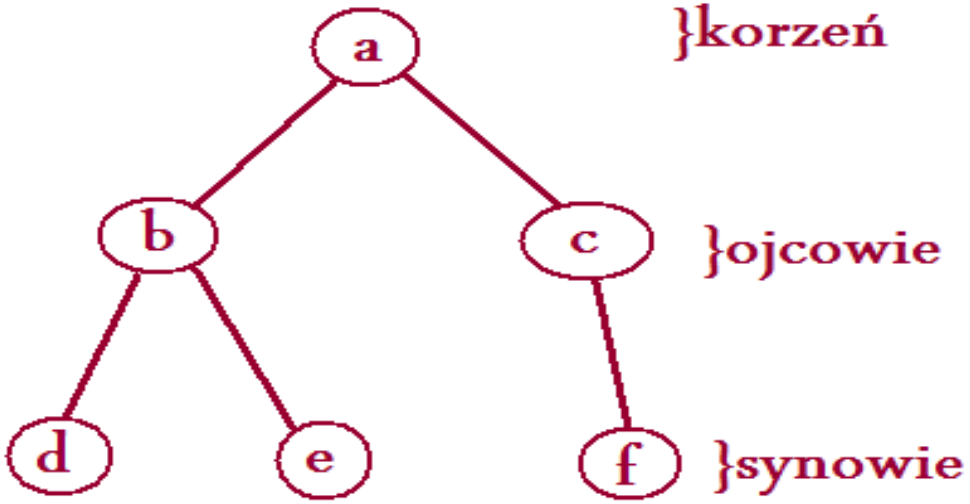


Drzewo ukorzenione

- Drzewo ukorzenione to drzewo z wyróżnionym jednym wierzchołkiem. Wyróżniony wierzchołek nazywamy ***korzeniem drzewa.***

Do opisu zależności między wierzchołkami drzewa używa się terminologii drzewa genealogicznego.

- Przykład

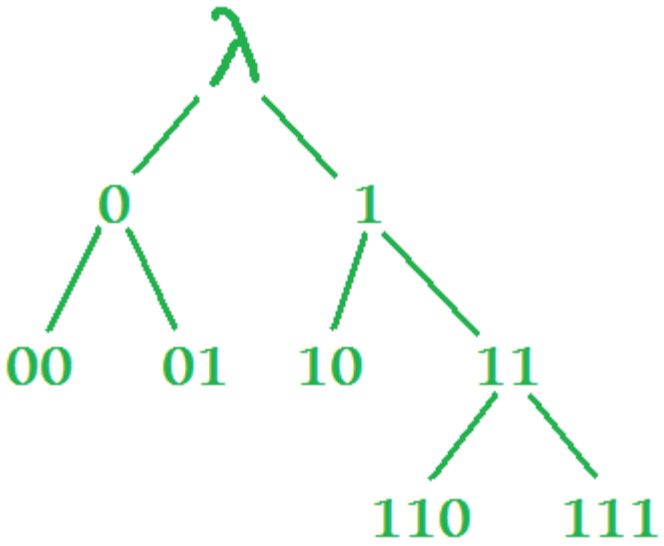


Drzewo ukorzenione ciąg dalszy

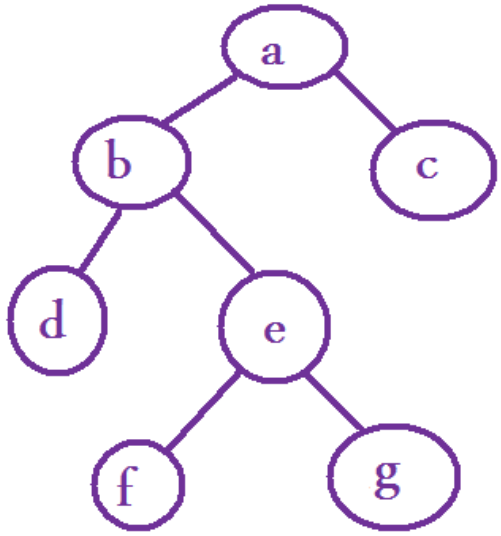
- Każdy wierzchołek *v* różny od korzenia jest z nim połączony dokładnie jedną **drogą prostą**(wszystkie występujące w niej wierzchołki są różne).
- Sąsiad leżący na drodze do korzenia nazywamy **ojcem** wierzchołka v.
- Pozostali sąsiedzi wierzchołka v są **synami** wierzchołka v.
- Synów tego samego wierzchołka nazywamy **braćmi.**
- Korzeń drzewa nie ma ojca, czyli elementu poprzedniego, wszyscy jego sąsiedzi są jego synami.
- Wierzchołek, który nie posiada syna, nazywamy **liściem**.

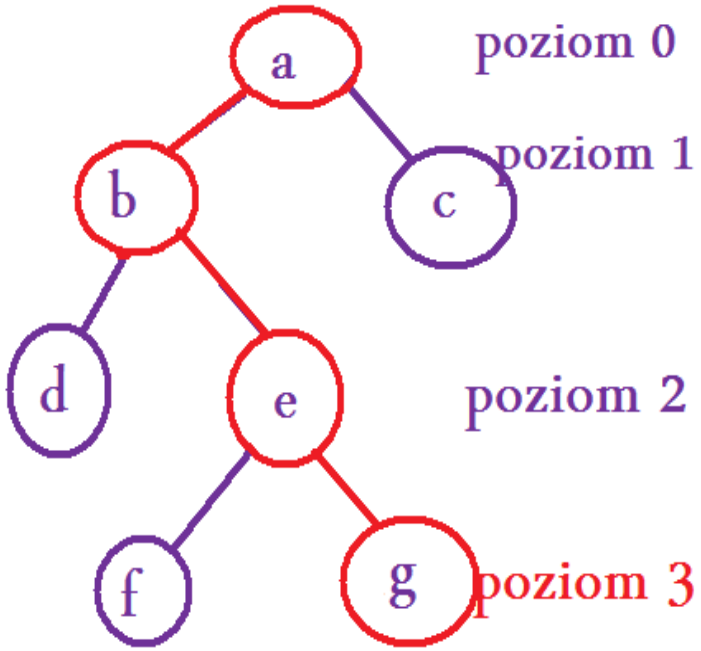
DRZEWA BINARNE

- **Drzewo binarne** w teorii grafów to drzewo, w którym stopień każdego wierzchołka jest nie większy od 3.
- Innymi słowy, to takie drzewo ukorzenione, w którym każdy wierzchołek ma co najwyżej dwóch synów.
- Do oznaczania wierzchołków w drzewie binarnym zwykle używa się ciągów zer i jedynek. Wierzchołki są oznaczane następująco:
  - Korzeń drzewa oznaczamy przez λ-pusty ciąg
  - Jeżeli jakiś wierzchołek jest oznaczony przez x, to jego synowie oznaczeni są przez x0 i x1.



- Drzewo binarne jest **regularne** jeśli stopień wyjściowy każdego wierzchołka jest równy 2 lub zero.
- **Numerem poziomu** wierzchołka v nazywamy długość drogi prostej od korzenia do v. Sam korzeń ma numer poziomu 0.
- **Wysokość** drzewa z wyróżnionym korzeniem jest równa największemu numerowi poziomu wierzchołka.



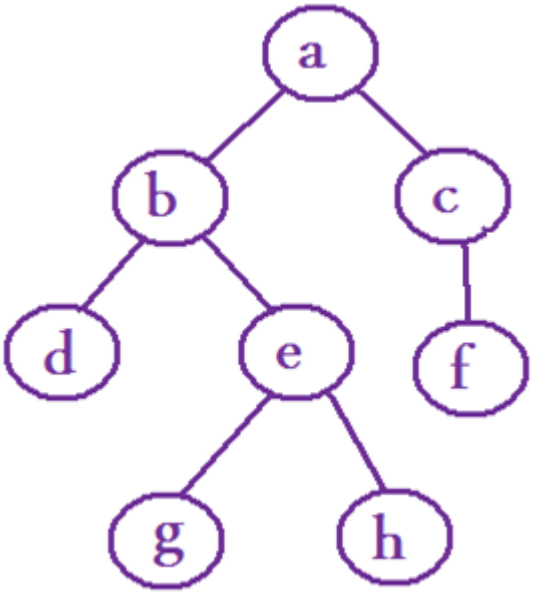


Przechodzenie drzewa

- **Pre-order,** przejście wzdłuż (λ, T_L, T_P)
  - Odwiedzamy najpierw korzeń, później lewe poddrzewo, a następnie prawe poddrzewo
- **In-order,** przejście poprzeczne (T_L, λ, T_P)

(przejście grafu w głąb)

  - Odwiedzamy poddrzewo lewe, następnie przechodzimy do korzenia, a później do poddrzewa prawego
- **Post-order,** przejście wsteczne (T_L, T_P, λ)
  - Odwiedzamy lewe poddrzewo, później poddrzewo prawe, a następnie odwiedzamy korzeń

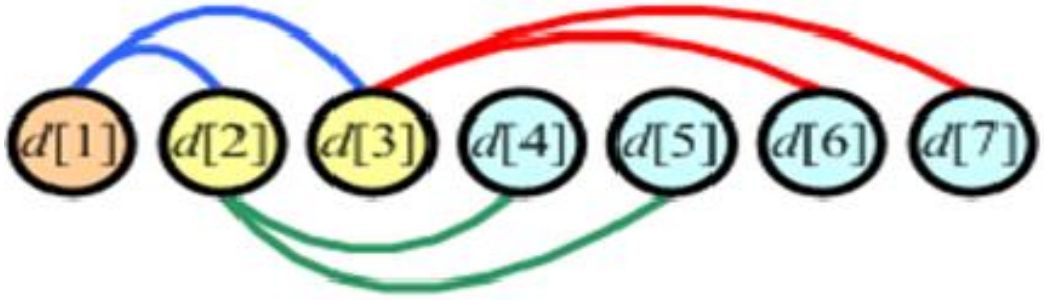


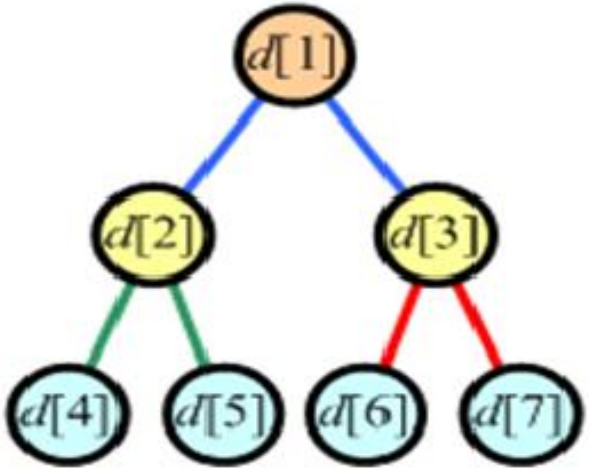
Orientacja	Kolejność wierzchołków
Pre-order	a, b, d, e, g, h, c, f
In-order	d, b, g, e, h, a, f, c
Post-order	d, g, h, e, b, f, c, a

Odwzorowanie drzewa binarnego

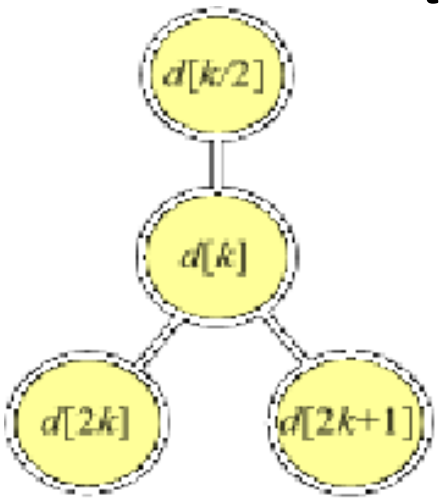
- Najprostszym sposobem przetwarzania za pomocą komputera struktur drzew binarnych jest zastosowanie zwykłej tablicy n elementowej. Każdy element tej tablicy będzie reprezentował jeden węzeł(wierzchołek) drzewa binarnego.
- Zastosujmy odwzorowanie:
  - Element d[1] będzie zawsze korzeniem drzewa.
  - i-ty poziom drzewa binarnego wymaga 2^{i-1} węzłów. Będziemy je kolejno pobierać z tablicy.

  Otrzymamy wtedy następujące odwzorowanie elementów tablicy w drzewo binarne:





- Ogólny wzór dla węzła k:



węzły potomne mają indeksy równe:

2k - lewy potomek

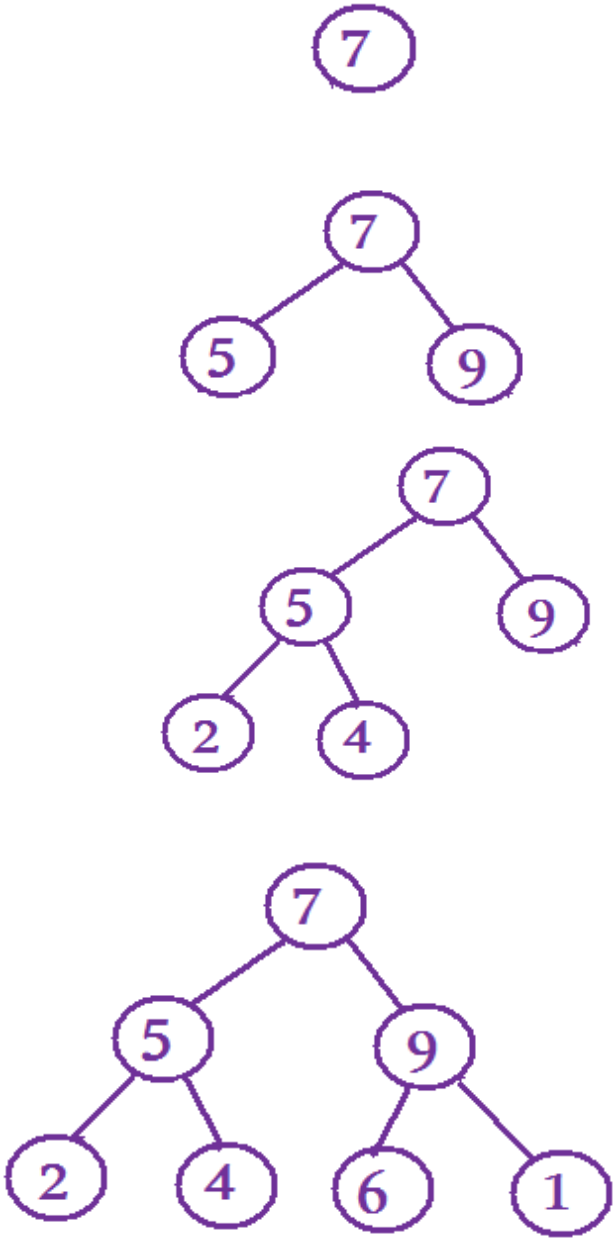
2k+1 - prawy potomek

eł nadrzędny ma indeks

Przykład

Skonstruować drzewo binarne z elementów zbioru {7 5 9 2 4 6 1}

1. Konstrukcję drzewa binarnego rozpoczynamy od korzenia, który jest pierwszym elementem zbioru, czyli liczbą 7.
2. Do korzenia dołączamy dwa węzły potomne, które leżą obok w zbiorze. Są to dwa kolejne elementy, 5 i 9.
3. Do lewego węzła potomnego (5) dołączamy jego węzły potomne. Są to kolejne liczby w zbiorze, czyli 2 i 4.
4. Pozostaje nam dołączyć do prawego węzła ostatnie dwa elementy zbioru, czyli liczby 6 i 1. Drzewo jest kompletne.



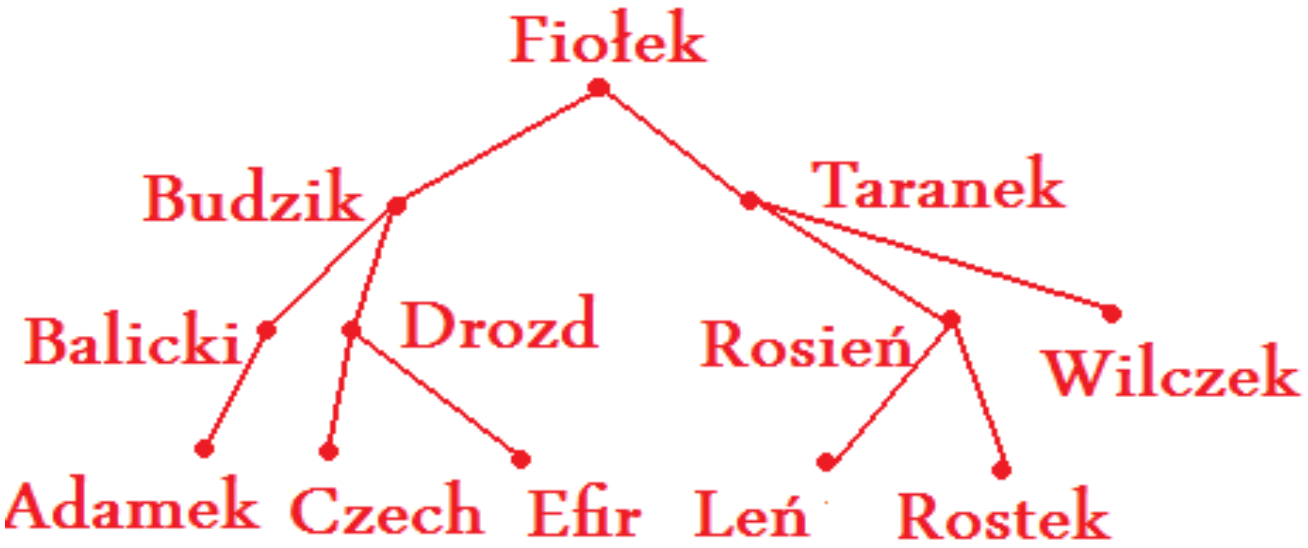
Drzewo poszukiwań binarnych

Ciąg liczb lub plik może być zorganizowany w specjalny rodzaj drzewa z wyróżnionym korzeniem, jest to **drzewo poszukiwań binarnych.** Jest to drzewo binarne, w którym każdy węzeł ma przyporządkowany jedną wartość, przy czym: wartość przechowywana w węźle jest **większa** od wartości przechowywanej w lewym synu i **mniejsza** od wartości przechowywanej w prawym synu.

Przechodząc drzewo metodą ***in-order*** uzyskuje się ciąg wartości posortowanych rosnąco.

- Aby znaleźć węzeł drzewa, w którym jest wartość x, porównujemy x z wartością k w korzeniu i w zależności od wyniku porównania:
- $x = k$ – kończymy poszukiwania
- $x < k$ – szukamy w lewym poddrzewie
- $x > k$ – szukamy w prawym poddrzewie

- Rysunek poniżej prezentuje drzewo poszukiwań binarnych zawierające zbiór rekordów klientów uporządkowany alfabetycznie.



- W celu znalezienia rekordu klienta **Rostek** postępujemy następująco:

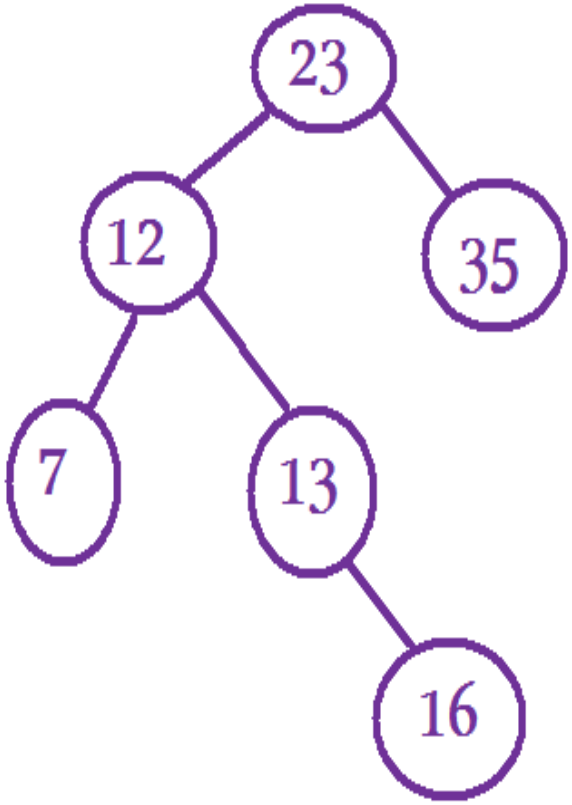
- Porównujemy nazwisko Rostek z nazwiskiem z korzenia, Fiołek. Rostek w porządku alfabetycznym znajduje się później, więc wybieramy prawą gałąź.
- Szukane nazwisko porównujemy z nazwiskiem Taranek. Rostek jest wcześniej w porządku alfabetycznym, wybieramy gałąź lewą.
- Porównujemy nazwisko Rostek z Rosieniem. Szukane nazwisko występuje później w porządku alfabetycznym wybieramy prawą gałąź.
- Dotarliśmy do szukanego wierzchołka, kończymy poszukiwania.

Opisana procedura jest skuteczna, ponieważ z każdego wierzchołka wychodzą co najwyżej dwie krawędzie w dół, więc wystarczy tylko kilka porównań by znaleźć właściwy adres, nawet jeśli liczba rekordów jest duża.

Przykład

- Drzewo utworzone dla ciągu wartości {23, 12, 35, 13, 16, 7}

1. Konstrukcję drzewa binarnego rozpoczynamy od korzenia, który jest pierwszym elementem zbioru, czyli liczbą 23.
2. Do korzenia dołączamy dwa węzły potomne, które leżą obok w zbiorze. Są to dwa kolejne elementy, 12 i 35.Ich rozmieszczenie nie jest przypadkowe, mniejszą wartość-12 umieszczamy po lewej stronie, natomiast 35 po stronie prawej.
3. Kolejną liczbą z listy jest 13. Wartość ta jest mniejsza od 23 i większa od 12 dlatego zostaje prawym potomkiem 12.
4. Liczba 16 jest mniejsza od 23, większa od 12 i od 13 dlatego umieszczamy ją na pozycji potomka prawego 13.
5. Pozostała liczba 7 zostaje lewym potomkiem liczby 12, jako że jest mniejsza od 23 i 12.



Algorytmy przeszukiwania drzew binarnych

- ***Lista***-uporządkowany ciąg elementów
- ***Kolejka***-lista z trzema operacjami
  - Dodawanie nowego elementu na koniec kolejki
  - Zdejmowanie pierwszego elementu z początku kolejki
  - Sprawdzanie, czy kolejka jest pusta

  Taki sposób dodawania i odejmowanie elementów jest określany FIFO(ang. First in first out-pierwszy, który wszedł pierwszy wyjdzie) Przykładem jest kolejka w sklepie.

- ***Stos***-lista z trzema operacjami:
  - Dodawanie elementu na wierzch stosu
  - Zdejmowanie elementu z wierzchu stosu
  - Sprawdzanie, czy stos jest pusty

  Na stosie dodajemy i odejmujemy elementy z tego samego końca. Sposób ten określany jest LIFO(ang. Last in first out-ostatni, który wszedł, pierwszy wyjdzie)

  Przykład układanie korespondencji na stosie, lub zmywanie stosu talerzy.

Przykład: Algorytm szukania książki wśród znajomych

- Tworzymy stos pusty
- Wkładamy na stos numery telefonów znajomych, którzy mogą mieć książkę
- Dopóki na stosie są numery, powtarzamy:
  - Zdejmujemy z wierzchu stosu jeden numer telefonu
  - Dzwonimy pod numer
  - Jeżeli osoba, do której dzwonimy, posiada szukaną książkę, to kończymy poszukiwania
  - W przeciwnym razie pytamy ją o znajomych, którzy mogą mieć książkę i numery dopisujemy do stosu

Algorytm przeszukiwani drzew w głąb (z użyciem stosu)

Dane wejściowe: drzewo **T**

- Odwiedzamy korzeń λ i wkładamy go na stos
- Dopóki stos nie jest pusty powtarzamy (niech *v* będzie wierzchołkiem na wierzchu stosu)
  - Sprawdzamy, czy istniej syn *u* wierzchołka *v*, który nie był odwiedzony:
    - Jeśli takie *u* znajdziemy, to

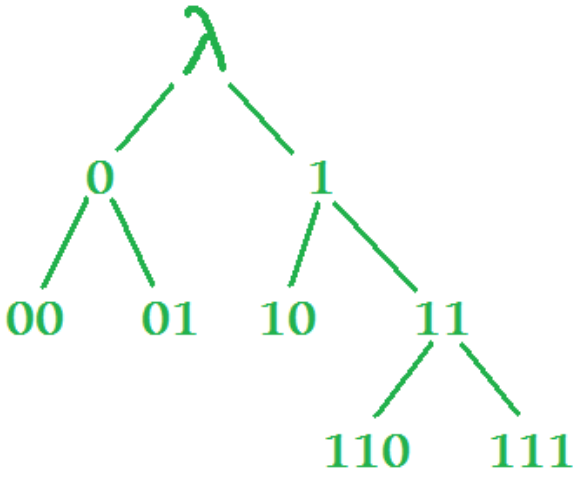
      odwiedzamy *u* i wkładamy go na wierzch stosu
    - Jeżeli takiego *u* nie znajdziemy, to

      zdejmujemy *v* ze stosu

Dodatkowo algorytm ten powinien zapamiętywać, jaki wierzchołek był zdjęty ze stosu. Ułatwia to stwierdzenie, który z synów wierzchołka znajdującego się na stosie pod spodem należy rozpatrzyć.

Przykład

Tabela pokazuje, jak wierzchołek jest odwiedzany i jaka jest zawartość stosu po każdej iteracji, gdy przeszukiwane jest następujące drzewo



Wierzchołek	Stos
λ	λ
0	λ,0
00	λ,0,00
0	λ,0
01	λ,0,01
0	λ,0
λ	λ
1	λ,1
10	λ,1,10
1	λ,1
11	λ,1,11
110	λ,1,11,110
11	λ,1,11
111	λ,1,11,111
11	λ,1,11
1	λ,1

W metodzie tej po każdym kroku algorytmu wierzchołki znajdujące się na stosie tworzą ścieżkę od wierzchołka wejściowego do wierzchołka aktualnie odwiedzanego.

Algorytm przeszukiwania drzewa wszerz (z użyciem kolejki)

- Dane wejściowe drzewo T
- Odwiedzamy korzeń λ i wkładamy go do kolejki.
- Dopóki kolejka nie jest pusta, powtarzamy:
  - bierzemy jeden wierzchołek *v* z początku kolejki,
  - odwiedzamy wszystkich synów wierzchołka *v,*
  - każdego wkładamy na koniec kolejki.

W metodzie przeszukiwania wszerz wierzchołki są przeszukiwane w kolejności od wierzchołków będących najbliżej wierzchołka początkowego do wierzchołków będących dalej

Przykład

Tabela ukazuje odpowiednie wierzchołki i zawartości kolejki po każdej iteracji algorytmu dla drzewa z poprzedniego przykładu.

Wierzchołki	Kolejka
λ	λ
0,1	0,1
00,01	1,00,01
10,11	00,01,10,11
-	01,10,11
-	10,11
-	11
110,111	110,111
-	111
-	-

Drzewo wyrażeń arytmetycznych

- Jest to przykład zastosowania drzew binarnych.
- W drzewie tym każdy wierzchołek ma etykietę.
- Liście etykietowane są stałymi bądź zmiennymi.
- Wierzchołki niebędące liśćmi etykietowane są operacjami arytmetycznymi.
- Każdemu wierzchołkowi w drzewie można

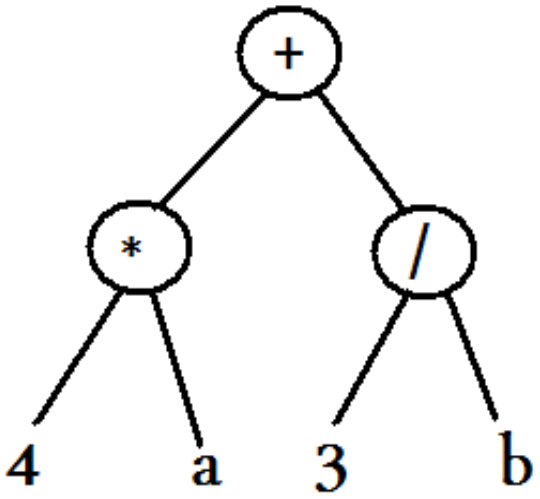
przypisać wyrażenie arytmetyczne według

zasady:

- Dla liści wyrażeniami są etykiety tych liści(stałe lub zmienne)
- Jeżeli wierzchołek v ma

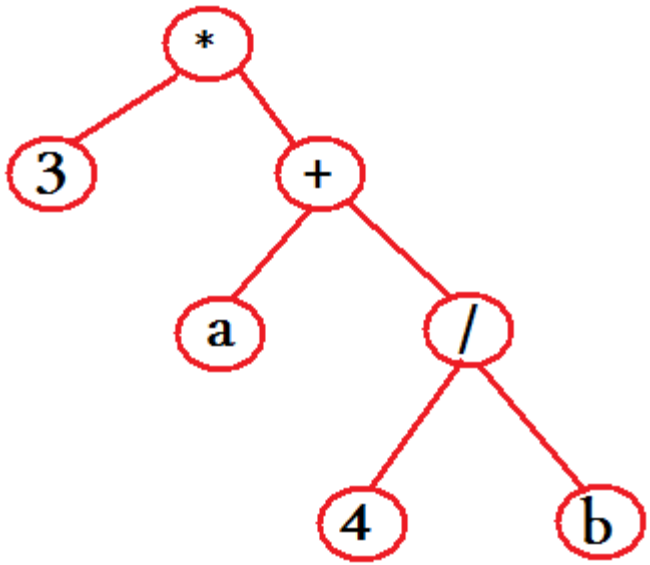
etykietę op, a jego synom przypisano wyrażenie W(v0) i W(v1), to wierzchołkowi v przypisujemy wyrażenie W(v)=(W(v0)opW(v1))

Drzewo wyrażenia



- Opuszczanie nawiasów w wyrażeniach może prowadzić do niejednoznaczności, np. wyrażenie 3(a+4/b) po opuszczeniu nawiasów jest identyczne z wyrażeniem 3a+4/b i zmienia sens.
- Drzewo odpowiadające

wyrażeniu 3(a+4/b)



Sposobem przedstawiania

wyrażeń bez użycia nawiasów

jest notacja polska(Łukasiewicza),

Inaczej postfiksowa (znak operacji stoi na końcu wyrażenia).

Wyrażenie w postaci postfiksowej	Wyrażenie w postaci infiksowej
3,a+	3+a
3,a*4,b/+	3a+4/b

Algorytm obliczania wyrażenia w postaci postfiksowej

- Dla kolejnych elementów zapisu wyrażenia:
  - jeżeli element jest stałą lub zmienną, to
    - wkładamy jego wartość na stos
  - jeżeli element jest znakiem operacji, to
    - zdejmujemy dwie wartości z wierzchu stosu,
    - wykonujemy operację na tych wartościach,
    - obliczoną wartość wkładamy na wierzch stosu
- Po przejściu całego wyrażenia jego wartość znajduje się w stosie

Przykład

Wyrażenie abc+*de/+

Zakładamy, że zmienne mają następujące wartości: a=3,b=2,c=1,d=4,e=2

Czytany element	Stos
a	3,
b	3,2,
c	3,2,1,
+	3,3,
*	9,
d	9,4,
e	9,4,2,
/	9,2,
+	11.

Podsumowanie

- Drzewa znakomicie oddają istotę organizacji hierarchicznej
- Są jedną z najważniejszych nieliniowych struktur danych używanych w algorytmach kombinatorycznych
- Ich struktura znalazła wiele zastosowań, m.in. w informatyce

Literatura:

- „Matematyka dyskretna” A. Szepietowski
- „Matematyka dyskretna” K. A. Ross, C. R. B. Wright
- http://edu.i-lo.tarnow.pl/inf/alg/003_sort/index.php