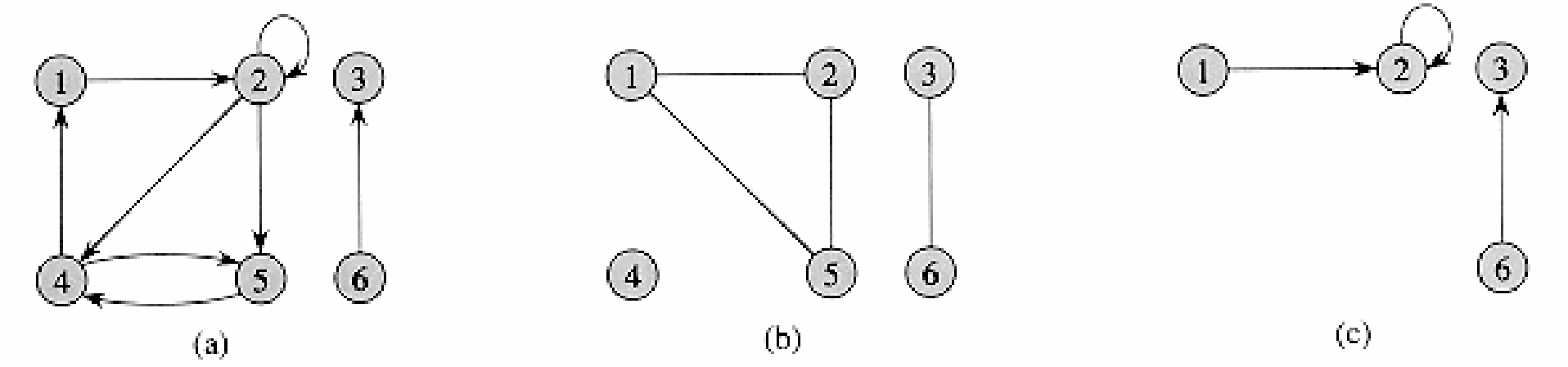
Grafy

Graf skierowany (**digraf**) G jest opisany parą (V,E), gdzie V jest zbiorem skończonym a E relacją binarną w V.
Zbiór V nazywamy zbiorem wierzchołków G a jego elementy nazywamy wierzchołkami.
Zbiór E jest zbiorem krawędzi G, a jego elementy nazywamy krawędziami.
Krawędź jest zatem zbiorem (u,v), gdzie $u,v \in V$.
W grafie skierowanym mogą występować pętle, łączące ten sam wierzchołek.

W grafie **nieskierowanym** $G=(V,E)$, zbiór krawędzi E jest zbiorem nieuporządkowanych par wierzchołków. Krawędź jest zatem zbiorem $(u,v) = (v,u)$, gdzie $u,v \in V$ i $u \neq v$. W grafie nieskierowanym nie mogą występować pętle, łączące ten sam wierzchołek.



a) Graf skierowany `G=(V,E)`, gdzie `V = {1, 2, 3, 4, 5, 6}` i `E = {(1, 2), (2, 2), (2, 4), (2, 5), (4, 1), (4, 5), (5, 4), (6, 3)}`. Krawędź `(2, 2)` jest pętlą.

b) Graf nieskierowany `G = (V, E)`, gdzie `V = {1, 2, 3, 4, 5, 6}` i `E = {(1, 2), (1, 5), (2, 5), (3, 6)}`. Wierzchołek `4` jest izolowany.

(c) Podgraf grafu z punktu a)

Jeżeli (u,v) jest krawędzią grafu skierowanego, to mówimy, że krawędź ta jest wychodząca z wierzchołka u i wchodząca do wierzchołka v.

Jeżeli (u,v) jest krawędzią grafu nieskierowanego, to mówimy, że krawędź ta jest incydentna z wierzchołkami u i v.

Jeżeli (u,v) jest krawędzią grafu, to mówimy, że wierzchołek v jest sąsiedni do u. Jeżeli graf jest symetryczny, relacja ta jest symetryczna.

Stopień wierzchołka – liczba incydentych z nim krawędzi (w grafie nieskierowanym).

Stopień wyjściowy wierzchołka – liczba krawędzi wychodzących z niego; stopień wejściowy – liczba krawędzi wchodzących; stopień wierzchołka – suma obu powyższych stopni (w grafie skierowanym).

Ścieżka (droga) długości k z wierzchołka u do u' jest ciągiem wierzchołków $\langle v_0, v_1, v_2, \ldots, v_k \rangle$ takich, że $u=v_0$, $u'=v_k$ oraz $(v_{i-1}, v_i) \in E$ dla $i=1,2,\ldots,k$. Ścieżka zawiera zarówno wierzchołki $v_0, v_1, v_2, \ldots, v_k$ jak i krawędzie (v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k). Ścieżka o wszystkich różnych wierzchołkach jest prosta.
Długość ścieżki jest liczbą krawędzi ścieżki.

W grafie skierowanym ścieżka $\langle v_0, v_1, v_2, \ldots, v_k \rangle$ tworzy cykl, jeżeli $v_0=v_k$ oraz zawiera co najmniej jedną krawędź. Cykl $\langle i, i \rangle$ złożony z krawędzi (i, i) jest pętlą.
W grafie nieskierowanym ścieżka $\langle v_0, v_1, v_2, \ldots, v_k \rangle$ tworzy cykl, jeżeli $v_0=v_k$ oraz $v_1, v_2, \ldots, v_k$ są różne i $k \geq 2$.

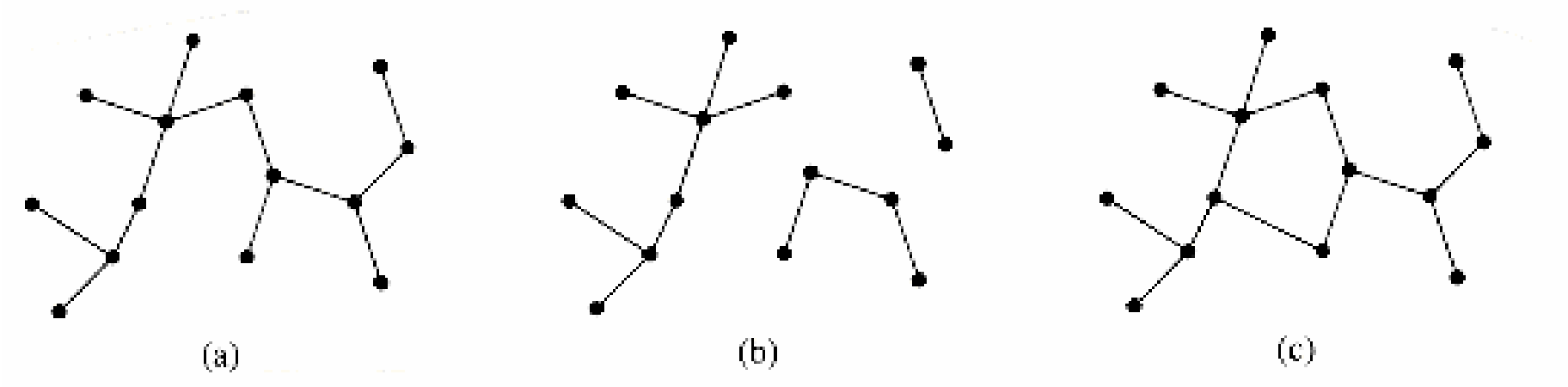
Graf nie zawierający cykli nazywamy acyklicznym.

Graf nieskierowany jest spójny, jeżeli każda para wierzchołków jest połączona ścieżką.

Drzewa

Drzewo wolne

– spójny, acykliczny graf nieskierowany
– jeżeli acykliczny, nieskierowany graf nie jest spójny, nazywamy go lasem.



a) drzewo wolne
b) las
c) ani las, ani drzewo – zawiera cykl

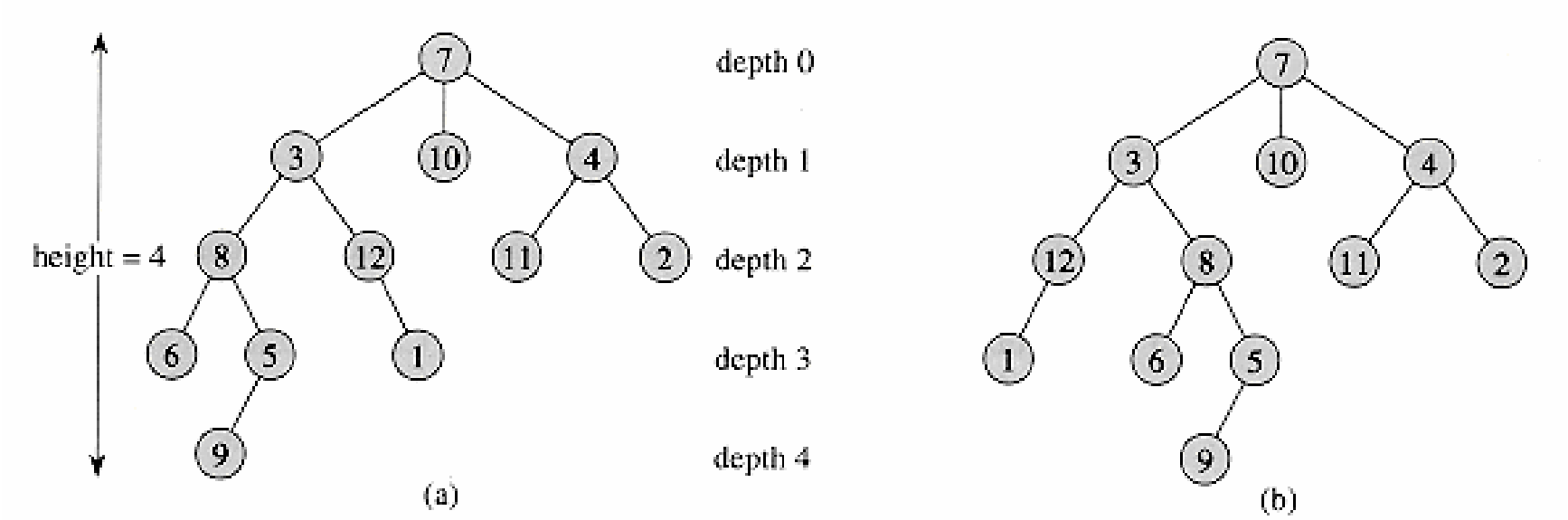
Twierdzenie

Niech G=(V,E) będzie grafem nieskierowanym. Poniższe fakty są równoważne.

1) G jest drzewem wolnym
2) Każde dwa wierzchołki G są połączone ze sobą dokładnie jedną ścieżką prostą
3) G jest spójny, ale jeśli usuniemy którąś z krawędzi E, to powstały graf jest niespójny
4) G jest spójny i |E|=|V-1|
5) G jest acykliczny i |E|=|V-1|
6) G jest acykliczny, lecz jeśli dodamy do E dowolną krawędź, to powstały graf zawiera cykl.

Drzewa ukorzenione i uporządkowane

– drzewo wolne, z wyróżnionym jednym wierzchołkiem – korzeniem drzewa.
– wierzchołki drzew ukorzenionych nazywamy węzłami
– w drzewie ukorzenionym nie ma znaczenia porządek węzłów potomnych na danym poziomie, w drzewie uporządkowanym ma on znaczenie.



a), b) drzewa o wysokości 4; jako drzewa ukorzenione są identyczne, ale jako drzewa uporządkowane różnią się porządkiem węzłów

Rozważmy węzeł x drzewa ukorzenionego T o korzeniu r. Każdy węzeł y na ścieżce z r do x nazywamy przodkiem węzła x. Jeżeli y jest przodkiem x, to x jest potomkiem y.

Wniosek: każdy węzeł jest jednocześnie swoim przodkiem i potomkiem.

Jeżeli y jest przodkiem x i $x \neq y$, to y jest właściwym przodkiem x a x jest właściwym potomkiem y.

Poddrzewo o korzeniu x jest drzewem utworzonym z potomków x.

Jeżeli ostatnią krawędzią na ścieżce od korzenia do węzła x jest (y,x), to y jest poprzednikiem (ojcem) x, a x jest następnikiem (synem) y.

Jeżeli dwa węzły mają ten sam poprzednik, to nazywamy je braćmi. Węzeł nie posiadający następników jest węzłem zewnętrznym lub liściem. Węzeł, który nie jest liściem, jest węzłem wewnętrznym.

Liczba następników węzła x drzewa ukorzenionego T nazywamy stopniem x.
Długość ścieżki od korzenia do węzła x nazywamy głębokością (poziomem) węzła x w drzewie.

Drzewo uporządkowane to zatem drzewo ukorzenione, w którym następniki każdego węzła są uporządkowane.

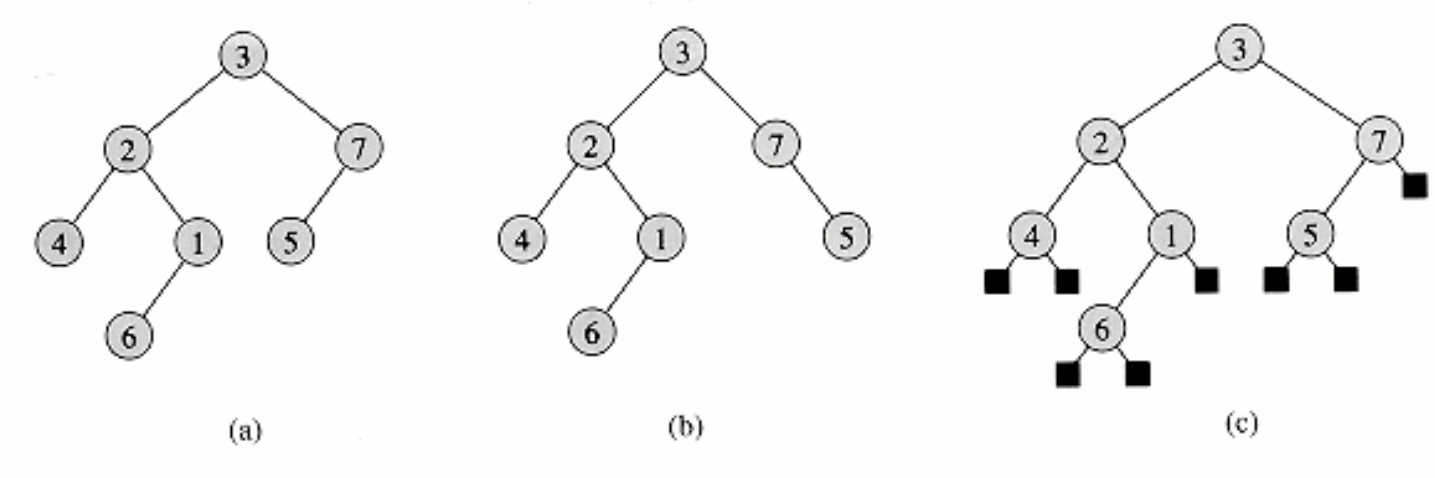
Drzewo binarne

Definicja rekurencyjna: drzewo binarne jest strukturą zdefiniowaną na skończonym zbiorze węzłów, które:
– nie zawiera żadnych węzłów, lub
– składa się z trzech rozłącznych zbiorów węzłów: korzenia, drzewa binarnego zwanego lewym poddrzewem oraz drzewa binarnego zwanego prawym poddrzewem.

Drzewo binarne, które nie ma żadnych węzłów, nazywamy pustym lub zerowym.

Jeżeli lewe poddrzewo jest niepuste, to jego korzeń jest lewym następnikiem korzenia drzewa głównego. Korzeń niepustego prawego poddrzewa jest prawym następnikiem.

Drzewo binarne nie jest po prostu drzewem uporządkowanym o stopniu węzłów co najwyżej 2. Dlaczego? Bo gdy węzeł ma tylko jeden następnik, to w drzewie binarnym ważne jest jego położenie: czy jest lewym czy też prawym następnikiem.



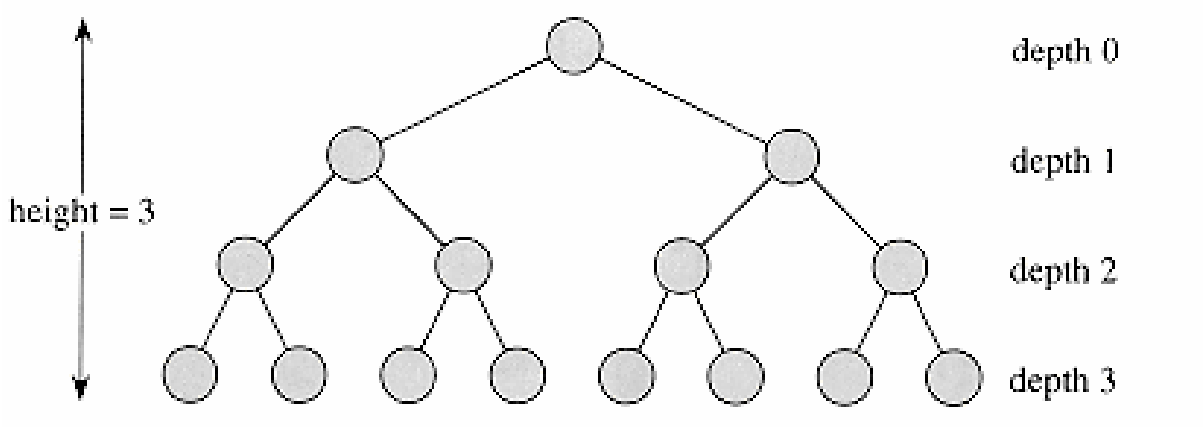
a), b) drzewa binarne
c) drzewo binarne regularne (■ – liście)

Drzewo binarne jest regularne, jeżeli każdy jego węzeł ma stopień 2 lub jest liściem.

Drzewo pozycyjne może posiadać więcej niż 2 następniki dla tego samego węzła – są one oznaczane liczbami całkowitymi.

Drzewo rzędu k (k-arne) to drzewo pozycyjne, w którym każdy węzeł ma następniki oznaczane liczbami nie przekraczającymi k.

Pełne drzewo rzędu k to drzewo rzędu k, w którym wszystkie liście mają identyczną głębokość.



Pełne drzewo binarne o wysokości 3.

Ile liści ma pełne drzewo rzędu k o wysokości h? Korzeń ma k następników, z których każdy ma k następników o głębokości 2 itd. Zatem otrzymujemy k^h liści.
Odwracając problem, wysokość pełnego drzewa rzędu k o n liściach wynosi $\log_k n$.
Liczba węzłów wewnętrznych:

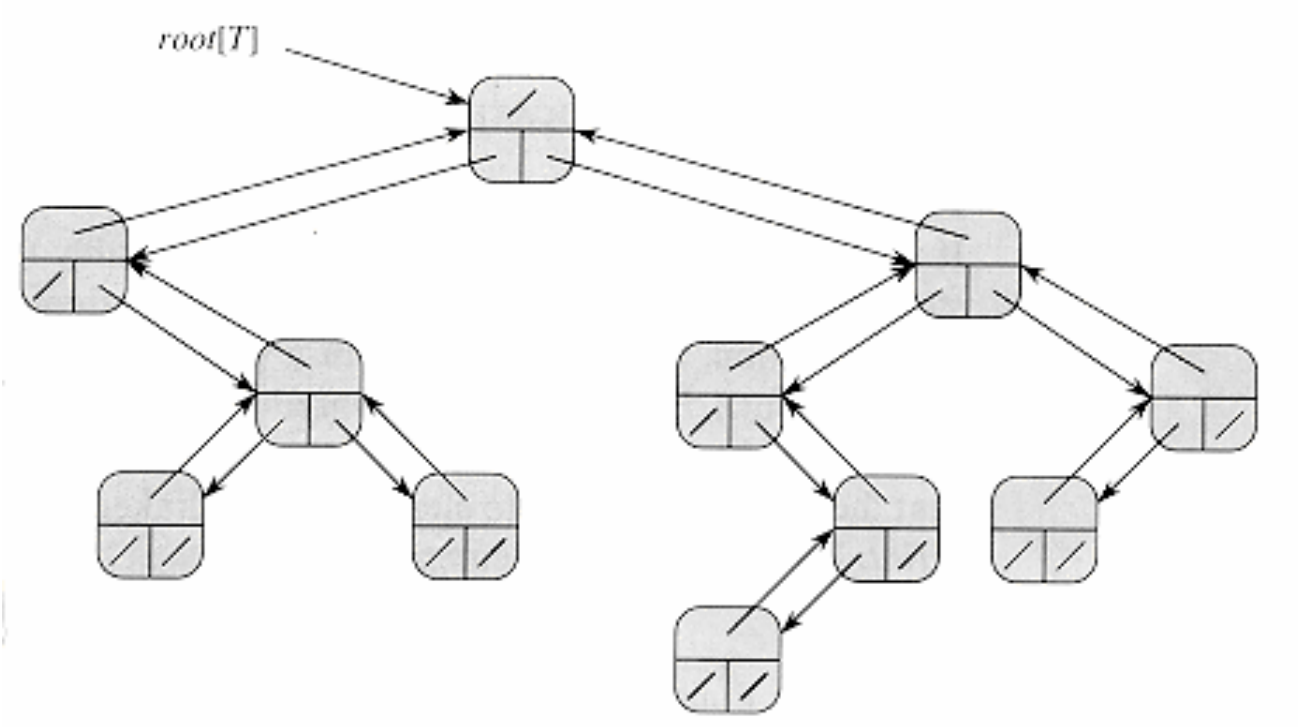
$$1 + k + k^2 + \cdots + k^{h-1} = \sum_{i=0}^{h-1} k^i = \frac{k^h - 1}{k - 1}$$

Dla drzewa binarnego mamy więc $2^h - 1$ węzłów wewnętrznych.

Reprezentowanie drzew ukorzenionych

Węzły reprezentujemy za pomocą rekordów; podobnie jak w listach, zakładamy że każdy węzeł ma pole `key`. Pozostałe pola będą służyć do przechowywania wskaźników do innych węzłów.

Drzewa binarne

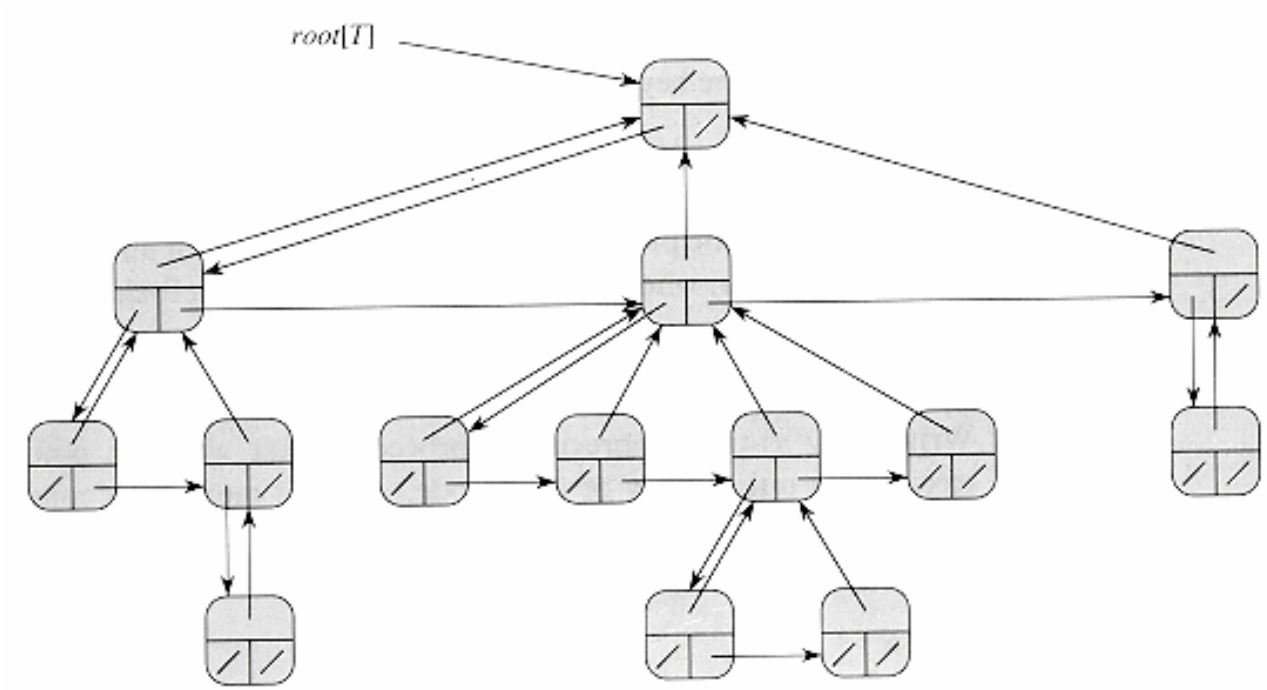


W polach `p`, `left` i `right` przechowujemy wskaźniki do ojca, lewego syna oraz prawego syna.
`root[T]` – wskaźnik do korzenia.

Drzewa ukorzenione o dowolnym stopniu rozgałęzień

Gdy liczba synów węzła jest ograniczona, pola `left` i `right` można zastąpić przez pola $\texttt{son}_1$, $\texttt{son}_2$, $\texttt{son}_3$, ..., $\texttt{son}_k$. Dodatkowo, sposób ten marnuje pamięć.

Alternatywna reprezentacja – za pomocą drzew binarnych: „na lewo syn, na prawo brat”



Każdy węzeł `x` ma pola: `p[x]` (górne), `left-child[x]` (lewe dolne) `right-sibiling[x]` (prawe dolne)

Alternatywne reprezentacje drzew

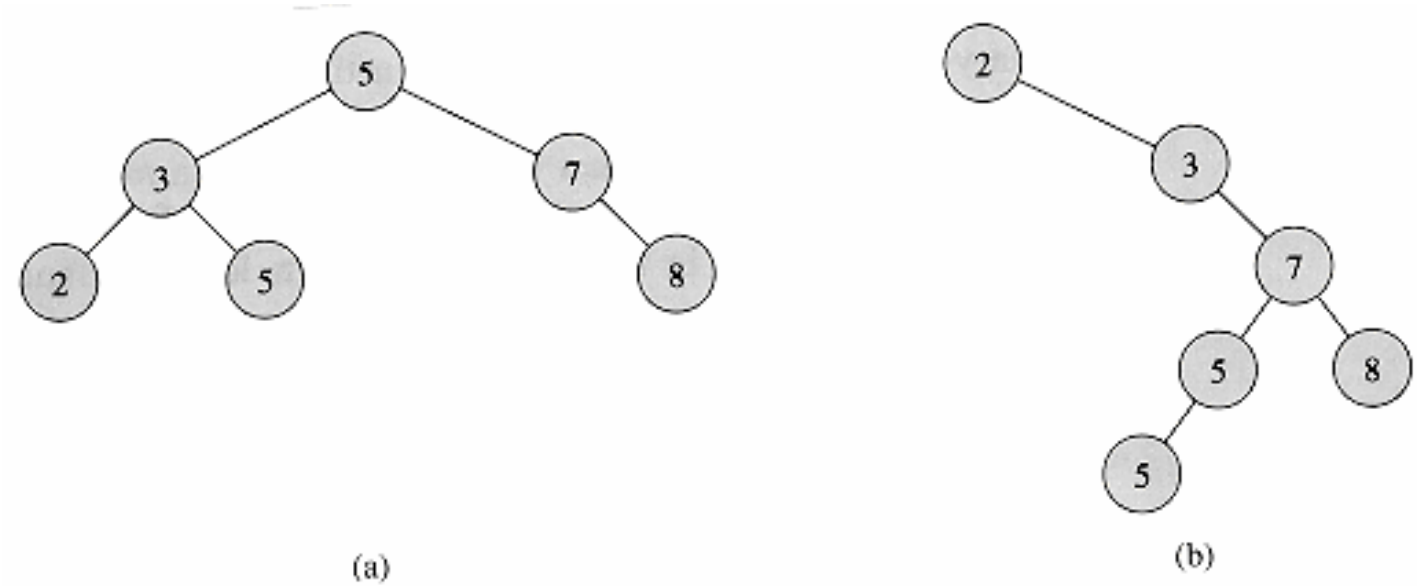
Przy sortowaniu przez kopcowanie wprowadzona została struktura kopca, która miała kształt pełnego drzewa binarnego i była reprezentowana za pomocą tablicy.

Drzewa poszukiwań binarnych (BST)

– struktury drzewa binarnego, dla których zdefiniowane są operacje właściwie dla zbiorów dynamicznych: `Search, Insert, Delete, Minimum, Maximum, Successor, Predecessor.`

Własność drzewa BST

Niech `x` będzie węzłem drzewa BST. Jeśli y jest węzłem znajdującym się w lewym poddrzewie węzła `x`, to `key[y]`$\leq$`key[x]`. Jeśli `y` jest węzłem znajdującym się w prawym poddrzewie węzła `x`, to `key[x]`$\leq$`key[y]`.



a) drzewo BST o wysokości 2
b) drzewo BST o wysokości 4 składające się z tych samych węzłów

Własność drzewa BST umożliwia wypisanie wszystkich znajdujących się w nim kluczy w uporządkowany sposób przy użyciu algorytmu rekurencyjnego, nazywanego przechodzeniem drzewa metodą inorder – klucz korzenia poddrzewa zostaje wypisany między wartościami z jego lewego poddrzewa a wartościami z prawego poddrzewa.

Metoda preorder wypisuje klucz korzenia przed wypisaniem wartości z obu poddrzew, a metoda postorder wypisuje klucz korzenia po wypisaniu wartości znajdujących się w poddrzewach.

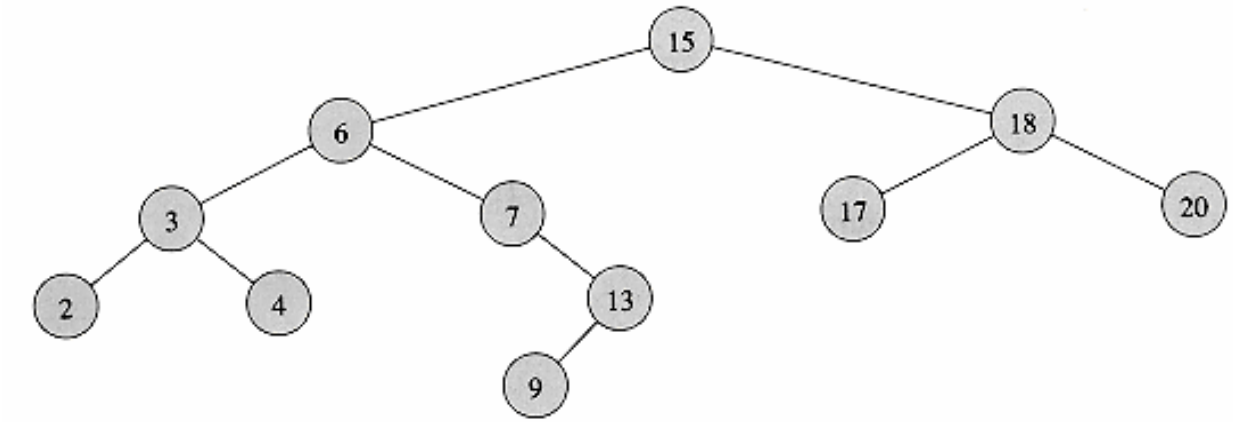
```
INORDER-TREE-WALK(x)
   if x ≠ NIL
      INORDER-TREE-WALK (left[x])
      wypisz key[x]
      INORDER-TREE-WALK (right[x])
```

Dla drzew z rysunku, algorytm wypisze wartości: `2, 3, 5, 5, 7, 8`.

Czas działania algorytmu dla `n` węzłów: `O(n)`

Wyszukiwanie w BST

```
TREE-SEARCH (x, k)
 if x = NIL or k = key[x]
      return x
  if k < key[x]
      return TREE-SEARCH (left[x], k)
  else
      return TREE-SEARCH (right[x], k)
```



Wyszukanie klucza 13 wymaga przejścia po ścieżce 15→6→7→13

Wersja iteracyjna kodu:

```
ITERATIVE-TREE-SEARCH (x,k)
  while x ≠ NIL and k ≠ key[x]
      do if k < key[x]
            x := left[x]
         else
            x := right[x]
  return x
```

Czas działania procedur dla drzewa o wysokości `h`: `O(h)`

Szukanie minimum i maksimum

– podążamy wg. wskaźników `left` (`right`) od korzenia aż napotkamy `NIL`)

```
TREE-MINIMUM (x)
  while left[x] ≠ NIL
      do x := left[x]
  return x
```

analogicznie:

```
TREE-MAXIMUM (x)
  while right[x] ≠ NIL
      do x := right[x]
  return x
```

Czas działania procedur dla drzewa o wysokości h: O(h)

Następniki i poprzedniki

Następnik węzła – następny węzeł odwiedzany podczas przechodzenia drzewa w porządku inorder.
Jeżeli wszystkie klucze są różne, to następnikiem węzła `x` jest węzeł o najmniejszym kluczu większym niż klucz węzła `x`.

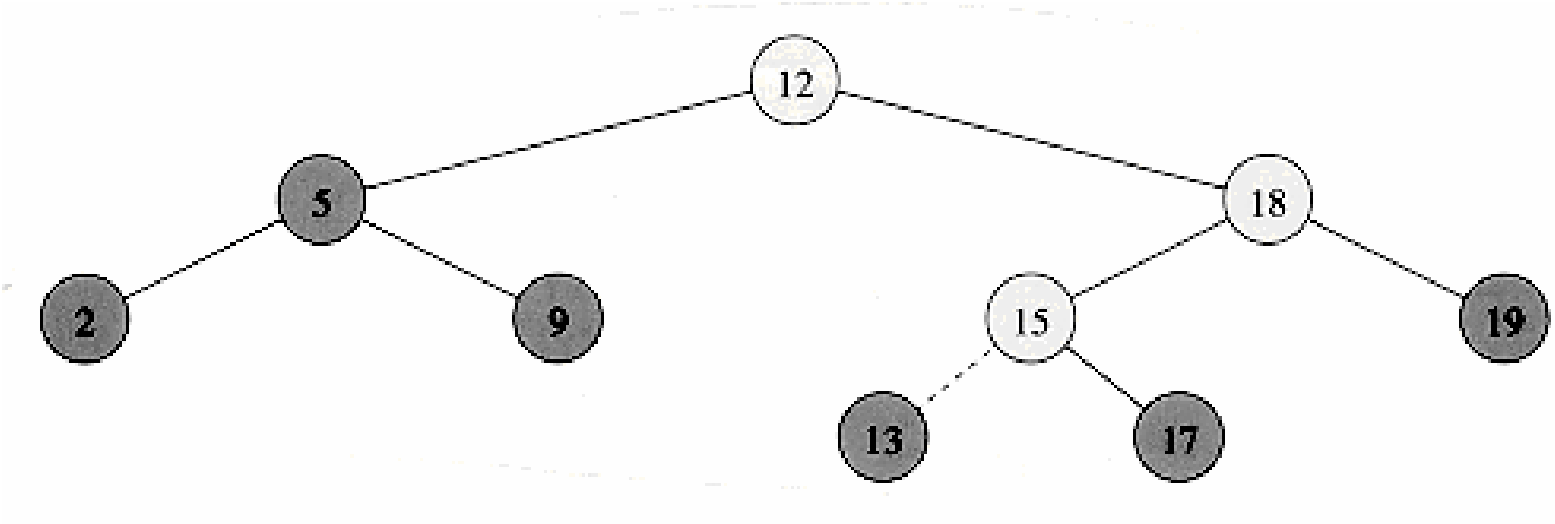
Jeżeli prawe poddrzewo węzła `x` jest niepuste, to następnikiem tego węzła jest najbardziej na lewo położony węzeł w prawym drzewie. Jeżeli węzeł nie ma prawego poddrzewa, ale ma następnik `y`, to `y` jest najniższym przodkiem węzła `x`, którego lewy syn także jest przodkiem `x`. Czas: `O(h)`

```
TREE-SUCCESSOR(x)
  if right[x] ≠ NIL
      return TREE-MINIMUM(right[x])
  y := p[x]
  while y ≠ NIL and x = right[y]
       do x := y
          y := p[y]
  return y
```

Procedura `TREE-PREDECESSOR(x)` może być skonstruowana analogicznie.

Wniosek: operacje `Search, Minimum, Maximum, Successor, Predecessor` są na drzewie binarnym o wysokości `h` wykonywane w czasie `O(h)`.

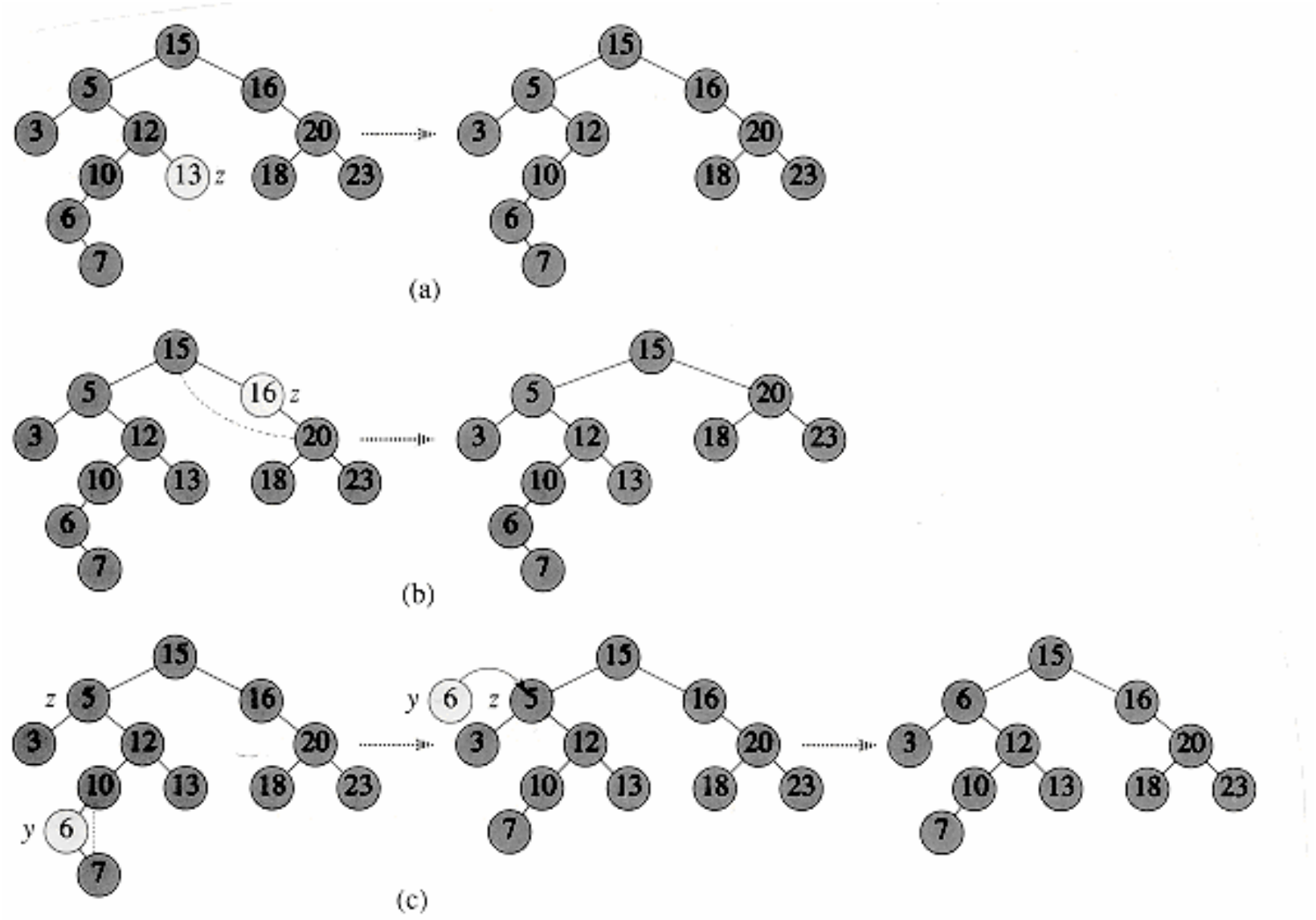
Wstawianie węzłów do drzewa



Wstawienie elementu 13 na odpowiednie miejsce. Czas: `O(h)`

```
TREE-INSERT(T,z) // z – wstawiany węzeł
  y := NIL
  x := root[T]
  while x ≠ NIL
       do y := x
          if key[z] < key[x]
             x := left[x]
          else
             x := right[x]
  p[z] := y
  if y = NIL
      root[T] := z
  else
      if key[z] < key[y]
           left[y] := z
      else
           right[y] := z
```

Usuwanie węzła



Usuwanie z BST węzłów z – trzy przypadki. Czas: $O(h)$

```
TREE-DELETE(T, z)
  if left[z] = NIL or right[z] = NIL
      y := z
  else
      y := TREE-SUCCESSOR(z)
  if left[y] ≠ NIL
      x := left[y]
  else
      x := right[y]
  if x ≠ NIL
      p[x] := p[y]
  if p[y] = NIL
      root[T]  x
  else
      if y = left[p[y]]
          left[p[y]]  x
      else
          right[p[y]]  x
  if y ≠ z
      key[z]  key[y]
      skopiuj pozostałe pola y
  return y
```