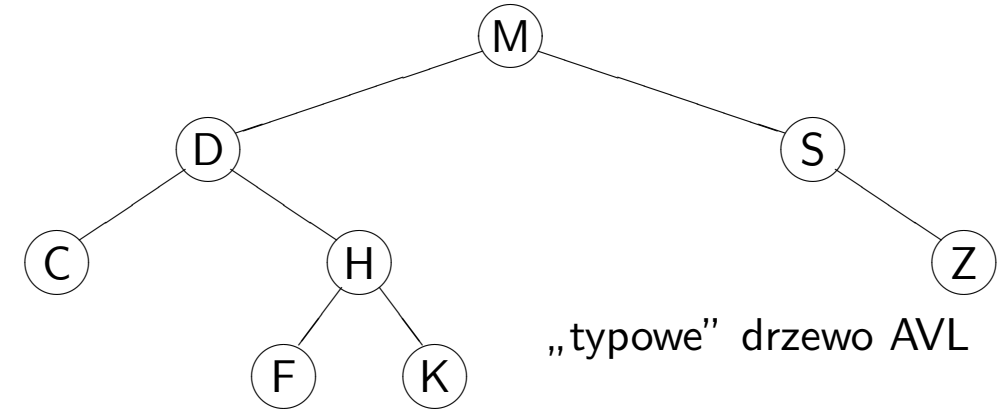
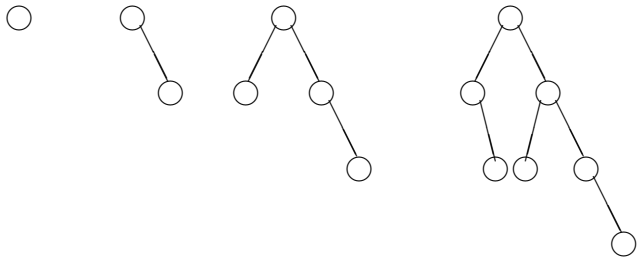
Drzewa AVL — definicje

Uporządkowane drzewo binarne jest **drzewem AVL**[1], jeśli dla każdego węzła różnica wysokości dwóch jego poddrzew wynosi co najwyżej 1.



„typowe" drzewo AVL



„minimalne" drzewa AVL

Własności drzew AVL gwarantują, że nawet w najgorszym przypadku wysokość drzewa wyniesie $1.44 \times \log(N + 2)$.

[1] Drzewa te wprowadzili w 1962 roku dwaj rosyjscy matematycy: G.M.Adelson-Welskij i E.M.Landis, i od ich nazwisk (w angielskojęzycznej transkrypcji) pochodzi nazwa AVL.

Drzewa AVL — tworzenie

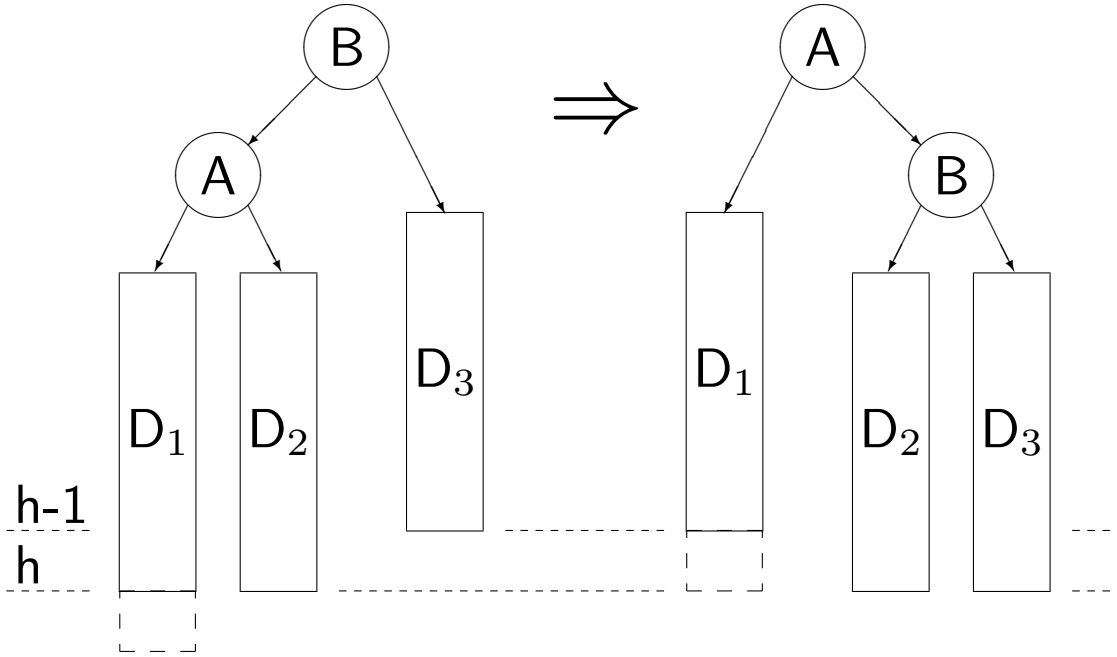
Procedura wstawiania elementu do drzewa AVL jest następująca:

1. użyj standardowej procedury wstawiania elementu do uporządkowanego drzewa binarnego, oraz,
2. jeżeli drzewo utraciło własność AVL, to przywróć ją poprzez wykonanie jednej z czterech rotacji omówionych poniżej.

Implementacja tego schematu wykorzystuje rekurencyjną procedurę dodawania elementu i dodatkowe trójwartościowe pole `balans` w każdym węźle drzewa.

```
TYPE DrzewoAVL = ^WezelAVL;
     WezelAVL = RECORD
                  info: T_Element;
                  lewe, prawe: DrzewoAVL;
                  balans : -1..1; { 1 oznacza prawe poddrzewo wyzsze,
                                   -1 ozn.wyzsze jest lewe poddrzewo,
                END;                0 ozn.poddrzewa rownej wysokosci}
```

Drzewa AVL — pojedyncza prawa rotacja



zauważmy ciekawe własności rotacji:

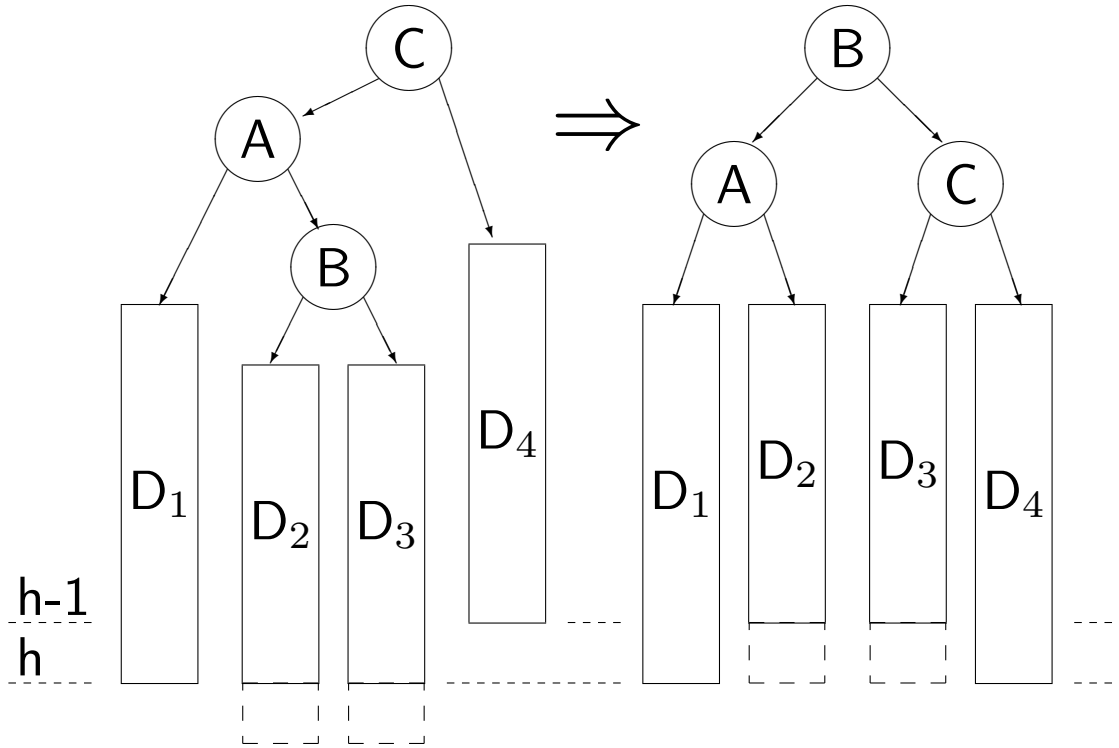
- jeśli całe drzewo było uporządkowane przed dodaniem elementu, to jest również uporządkowane po wykonaniu rotacji

- jeśli przed dodaniem elementu poddrzewa D_1,D_2,D_3 miały własność AVL to całe drzewo ma tę własność po dodaniu elementu i wykonaniu rotacji

Drzewa AVL — podwójna prawa rotacja

ciąg dalszy własności rotacji:

- rotacja pozostawia drzewo o tej samej wysokości co przed dodaniem elementu, zatem cała operacja nie narusza własności AVL z punktu widzenia żadnego „nadrzędnego" elementu drzewa (inaczej mówiąc, rotacja „kasuje" naruszenie własności AVL, jeśli takie nastąpiło)



Drzewa AVL — implementacja dodawania elementu

dodawanie elementu do drzewa można zrealizować zwykłym rekurencyjnym algorytmem dodawania elementu do drzew uporządkowanych, jeśli uwzględni się, za pomocą pola balans, możliwy wzrost wysokości drzewa i konieczność wykonania rotacji (patrz funkcja `DodajAVL`):

poprzedni balans	przyrost lewego poddrzewa	przyrost prawego poddrzewa	nowy balans	przyrost drzewa	komentarz
-1	0	0	b.z.	0	drzewo bez zmian
-1	1(sk)/-1(we)	0	$-2 \longrightarrow 0$	0	prawa rotacja
-1	0	1(sk)/-1(we)	0	0	poprawiło się
0	0	0	b.z.	0	drzewo bez zmian
0	1(sk)/-1(we)	0	-1	1	dopuszcz.wzrost
0	0	1(sk)/-1(we)	1	1	dopuszcz.wzrost
1	0	0	b.z.	0	drzewo bez zmian
1	1(sk)/-1(we)	0	0	0	poprawiło się
1	0	1(sk)/-1(we)	$2 \longrightarrow 0$	0	lewa rotacja

```
FUNCTION DodajAVL(VAR korzen : DrzewoAVL;
                  element    : DrzewoAVL;
                  FUNCTION PorownajEl(e1, e2 : T_Element) : CHAR) : INTEGER;
{zwraca 0 gdy wysokosc drzewa nie wzrosla; -1/1 gdy wzrosla przez lewe/prawe poddrzewo}
VAR l_wzr, p_wzr : -1..1;
BEGIN
   IF korzen=NIL THEN
   BEGIN
      korzen := Element;
      korzen^.balans := 0;
      DodajAVL := 1; {w tym przypadku wartosc 1 lub -1 nie ma znaczenia}
   END
   ELSE
   BEGIN
      l_wzr := 0;    {wartosc wzrostu lewego lub prawego poddrzewa potrzebna dla ...}
      p_wzr := 0;    {wybrania rotacji: pojedynczej lub podwojnej, i prawej lub lewej}
      CASE PorownajEl(element^.info, korzen^.info) OF
       '<' : l_wzr := DodajAVL(korzen^.lewe, element, PorownajEl);
       '=' : ; (* blad - element jest juz w/na drzewie *)
       '>' : p_wzr := DodajAVL(korzen^.prawe, element, PorownajEl);
      END;
      (**** teraz rozwaz mozliwe przypadki (patrz tabela):
          - wykonaj odpowiednia rotacje jesli to konieczne,
          - ustaw pole balans, i wylicz wlasciwa wartosc funkcji. *)
   END;
END; {DodajAVL}
```

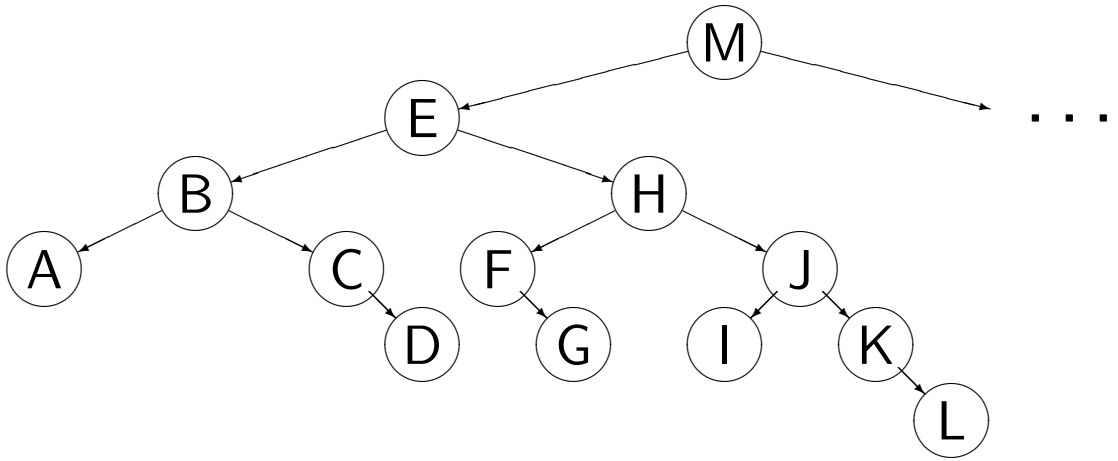
Drzewa AVL — usuwanie elementu

Usuwanie elementu z drzewa AVL jest zrealizowane analogicznie: element jest usuwany procedurą rekurencyjną, i po powrocie z wywołań rekurencyjnych na kolejnych poziomach aktualizowane jest pole balans, a w razie potrzeby wykonywana jest odpowiednia rotacja.

Jednak w przypadku usuwania węzłów rotacja nie pozostawia drzewa o tej samej wysokości, co przed usunięciem węzła, a o 1 niższe. Dlatego jednorazowe wykonanie rotacji nie kasuje do końca zaburzenia, które propaguje się dalej i na wyższych poziomach mogą być konieczne kolejne rotacje.

Skrajnym przykładem takiej sytuacji jest „ekstremalne" drzewo AVL, które na każdym poziomie posiada maksymalne niezrównoważenie dopuszczalne przez własność AVL. Usunięcie w tym drzewie węzła z niższej gałęzi drzewa powoduje kaskadę rotacji na wszystkich poziomach.

Skrajny przykład usuwania węzła z drzewa AVL:

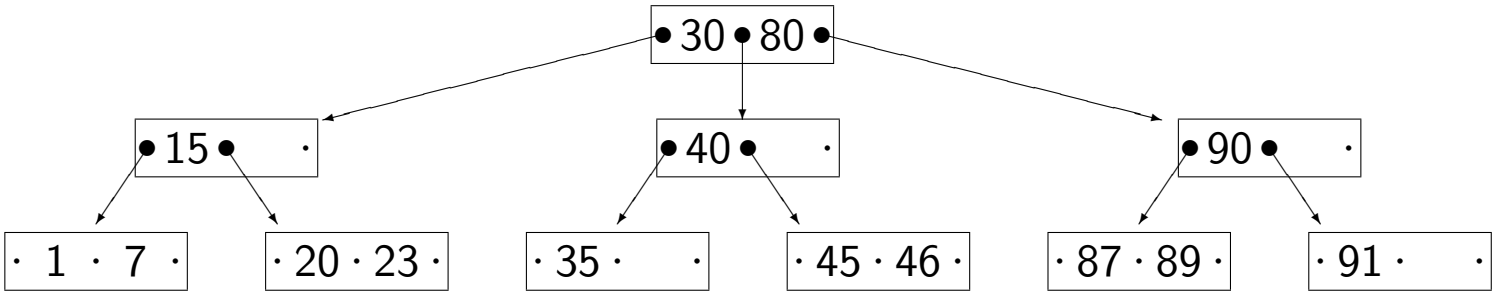


Usunięcie elementu „A” powoduje naruszenie równowagi w węźle „B”. Aby to skorygować należy wykonać lewą rotację wynoszącą element „C” w górę, a element „B” w dół, i zmniejszającą w efekcie wysokość drzewa „B-C-D”. To jednak powoduje naruszenie równowagi w węźle „E” i konieczność ponownej lewej rotacji wynoszącej do góry węzeł „H”. Wysokość tego poddrzewa maleje, i może z kolei spowodować naruszenie równowagi w węźle „M”, powodując konieczność wykonania w tym punkcie rotacji, itd.

B-drzewa — definicja i własności

B-drzewem[2] nazywamy sortowane drzewo rzędu $2D + 1$ (niebinarne), w którym każdy węzeł, z wyjątkiem korzenia, zawiera od D do $2D$ elementów (kluczy), a wszystkie ścieżki od korzenia do wszystkich liści mają tę samą długość.

Przykładowe B-drzewo rzędu 3 (liczba $D = 1$):



Jak wynika z definicji, w B-drzewie liście występują tylko na najniższym poziomie. Z konstrukcji drzewa rzędu > 2 wynika, że każdy węzeł drzewa, który zawiera k kluczy, posiada dokładnie $k + 1$ poddrzew, przy czym w B-drzewie wszystkie te poddrzewa muszą być niepuste, z wyjątkiem liści.

[2] B-drzewa zostały wprowadzone w roku 1972 przez dwóch matematyków: Niemca Rudolfa Bayera i Amerykanina Edwarda M. McCreighta. Nie wiadomo skąd wzięli nazwę **B-drzewa**.

Zauważmy również, że korzeń B-drzewa jest jedynym węzłem, który może zawierać nieograniczoną liczbę od 1 do $2D$ elementów.

Częściowe zrównoważenie B-drzew powoduje, że ich wysokość w najgorszym przypadku jest również logarytmem (rzędu D) liczby węzłów. Nadają się one zatem do przechowywania dużych ilości danych, w szczególności w plikach dyskowych, zwłaszcza jeśli wartość D jest tak duża, że wielkość węzła zajmuje całą jednostkę alokacji pamięci dyskowej, lub jej wielokrotność. Dlatego typowy rząd B-drzewa może wynosić nawet kilkaset, i wtedy nawet bardzo duże drzewo może mieć tylko 3-4 poziomy.

B-drzewa — definicja typu danych

```
CONST BDrzewoMax = 500;
TYPE TypElementu = INTEGER;
     BDrzewo = ^WezelB;
     WezelB = RECORD
                naddrzewo:BDrzewo;
                elementy :ARRAY[1..BDrzewoMax] OF TypElementu;
                poddrzewa:ARRAY[0..BDrzewoMax] OF BDrzewo;
                liczba   :INTEGER;
              END;
```

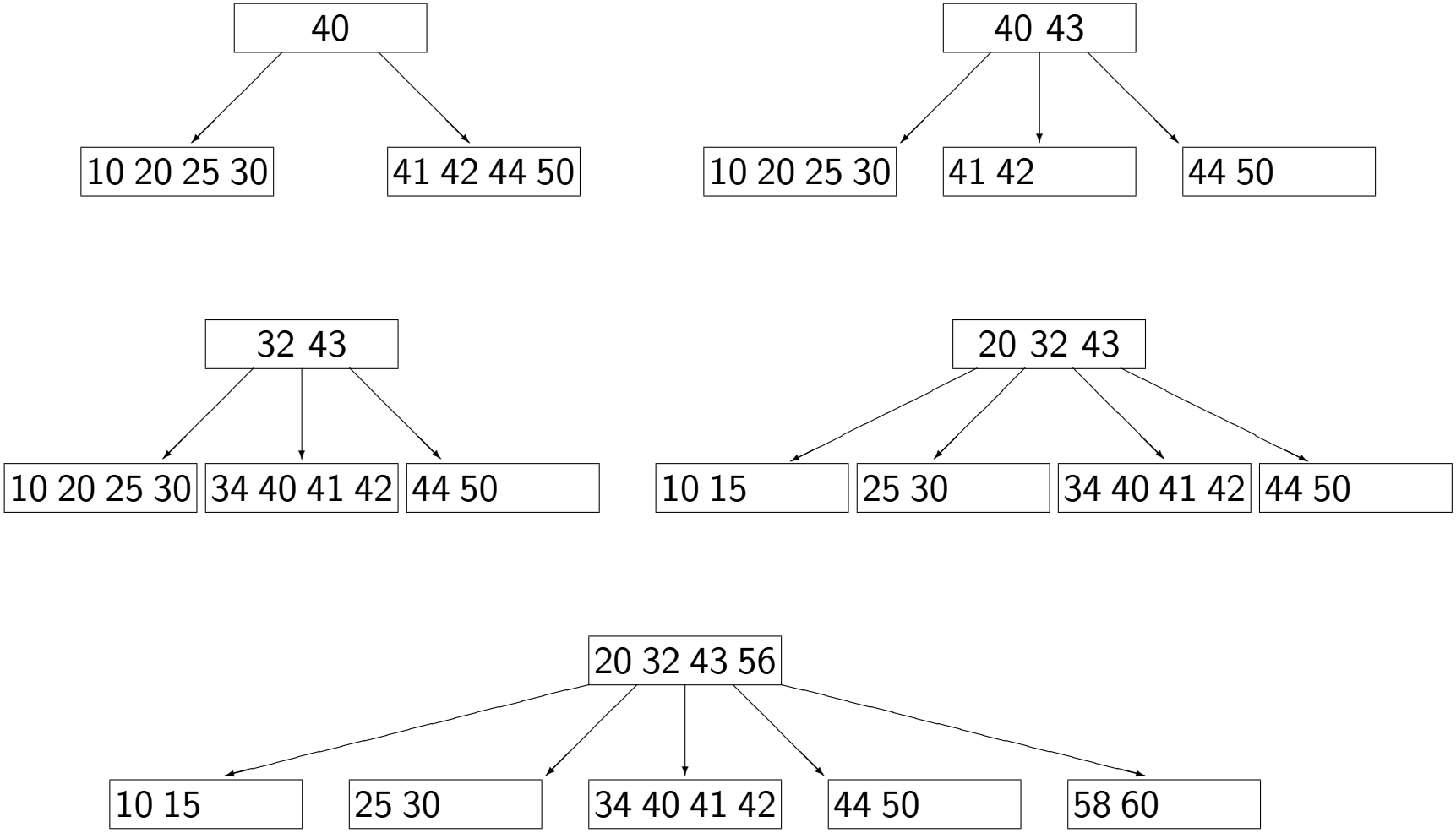
B-drzewa — dodawanie kluczy

- proste dodawanie kluczy - zawsze do liścia

- „przelewanie" kluczy między sąsiednimi liściami

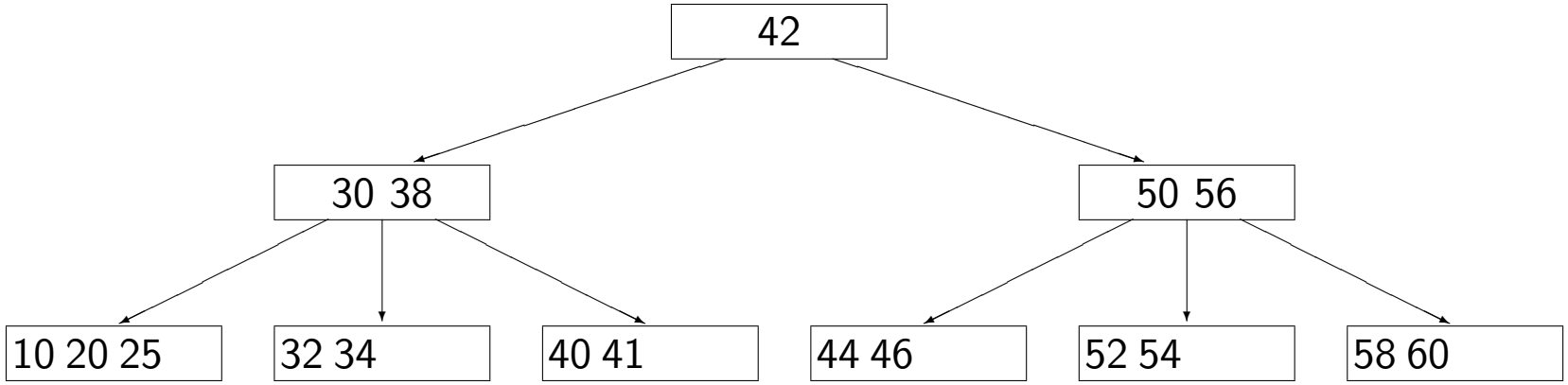
B-drzewa — dodawanie kluczy (2)

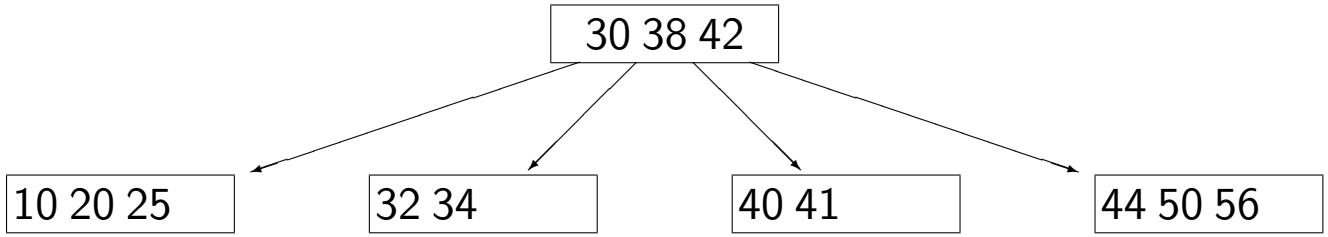
- „rozpoławianie” węzła z propagacją klucza w górę



B-drzewa — usuwanie kluczy

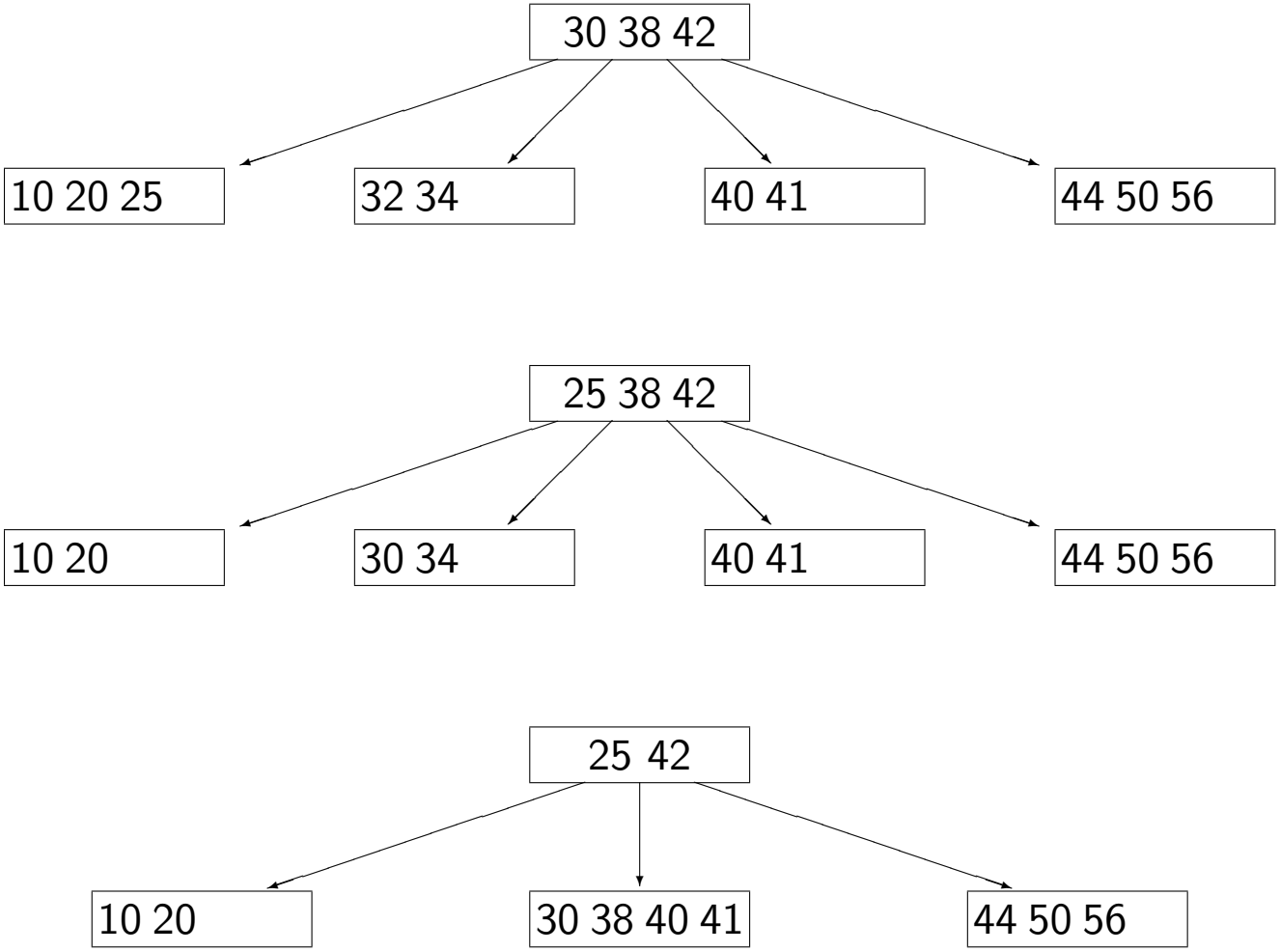
- usuwanie kluczy z liści i „zapadanie się” drzewa





B-drzewa — usuwanie kluczy (2)

- usuwanie kluczy z liści i łączenie sąsiednich liści



Operacje na poznanych typach drzew

Drzewa:	Wysokość drzewa w przypadku:		
	najgorszym	najlepszym	średnim
zrównoważone	$\log_2(N+1)$	$\log_2(N+1)$	$\log_2(N+1)$
niezrównoważone	N	$\log_2(N+1)$	$1.39 \times \log_2(N)$
drzewa AVL	$1.44 \times \log_2(N)$	$\log_2(N+1)$	$\log_2(N) + 0.25$
B-drzewa	$\log_{D+1}(\frac{N+1}{2}) + 1$	$\log_{2D+1}(N+1)$	$\log_{1.38D+1}(\frac{N}{D+1}) + 1$

Drzewa:	Złożoność obliczeniowa operacji:			
	wyszukiwania elementu	dodawania elementu	usuwania elementu	przeglądania uporządkowanego
nieuporządkowane	O(N)	O(1)	O(1)	O(N log N)
uporządkowane	O(N)	O(N)	O(N)	O(N)
zrównoważone	O(log N)	O(log N)	O(log N)	O(N)
drzewa AVL	O(log N)	O(log N)	O(log N)	O(N)
B-drzewa	O(log N)	O(log N)	O(log N)	O(N)