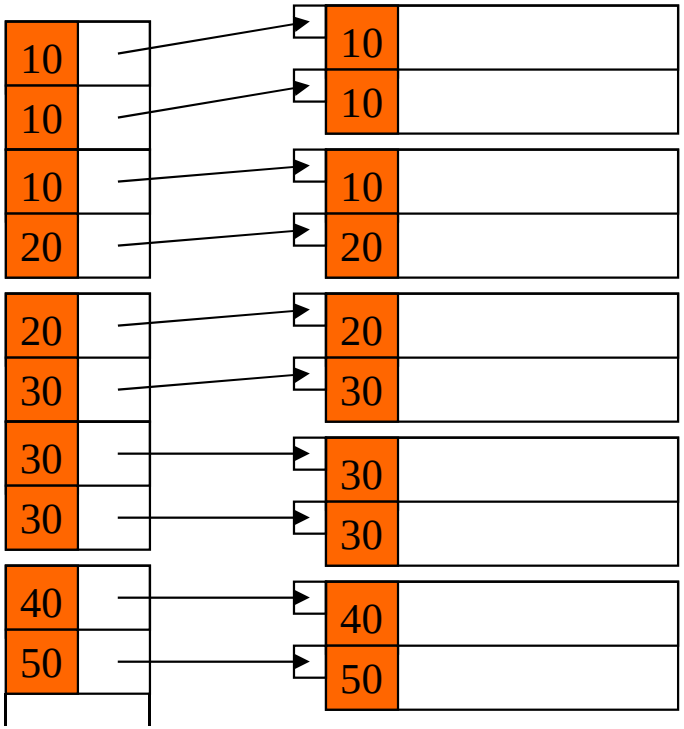
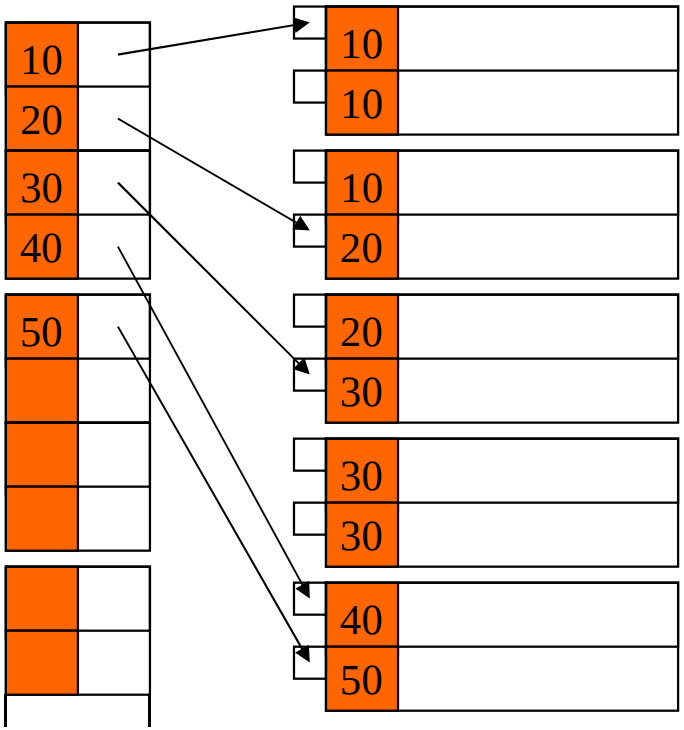
Indeks gęsty dla klucza nieunikatowego

- Wariant 1: Indeks gęsty dopuszcza powtórzenia



Indeks gęsty dla klucza nieunikatowego

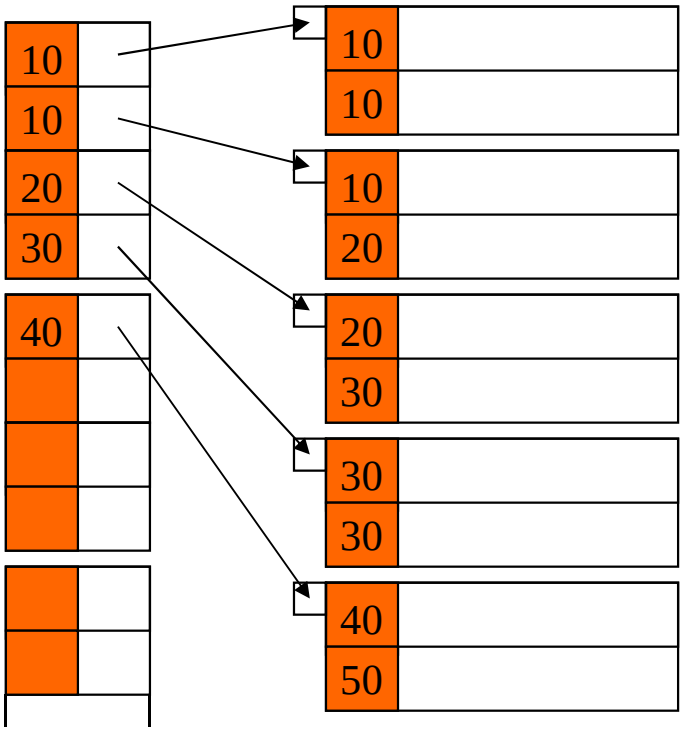
- Wariant 2: (efektywniejszy) wartości klucza występują w indeksie tylko raz



- To nadal jest wskazanie na adres rekordu z szukanym kluczem (pozostałe rekordy są w tym samym lub kolejnych blokach).
- Bloki wiąże się w listy (w każdym bloku jest wskaźnik do następnego)

Indeks rzadki dla klucza nieunikatowego

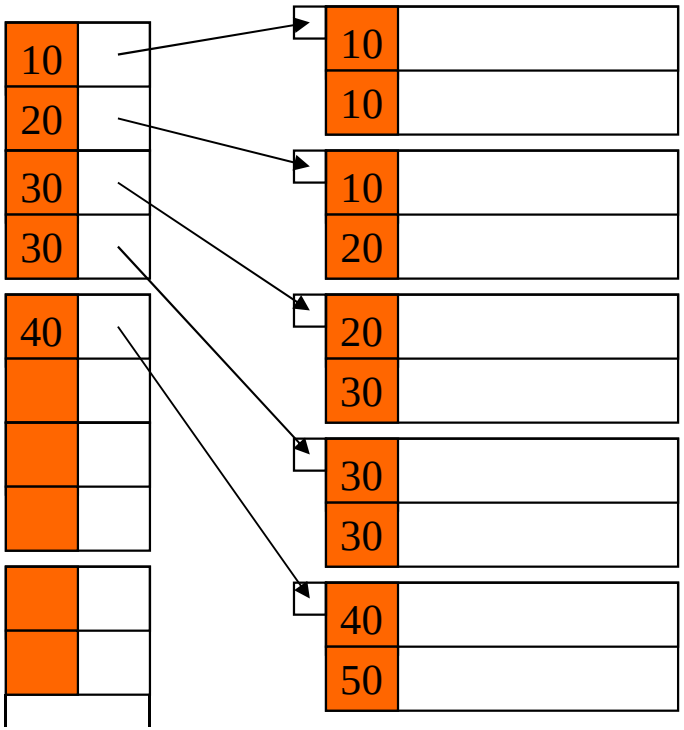
- Wariant 1: Podobnie jak w przypadku klucza unikatowego plik indeksu zawiera pary klucz-wskaźnik do bloku.



- Aby odszukać rekord trzeba znaleźć ostatnią pozycję indeksu z wartością =< K,
- Przesunąć się w pliku indeksu wstecz, szukając wartości mniejszej od K.
- Wszystkie bloki danych, w których może wystąpić szukany klucz K znajdują się między tymi dwoma wskaźnikami.

Indeks rzadki dla klucza nieunikatowego

- Wariant 2: Indeks wskazuje najmniejszą wartość nowego klucza w bloku.



- para klucz-wskaźnik odnosi się tu do bloku z nowym kluczem (tj. takim, który nie wystąpił w poprzednich blokach), lub do bloku, w którym są tylko identyczne wartości klucza.

Zmiany w plikach sekwencyjnych i indeksu

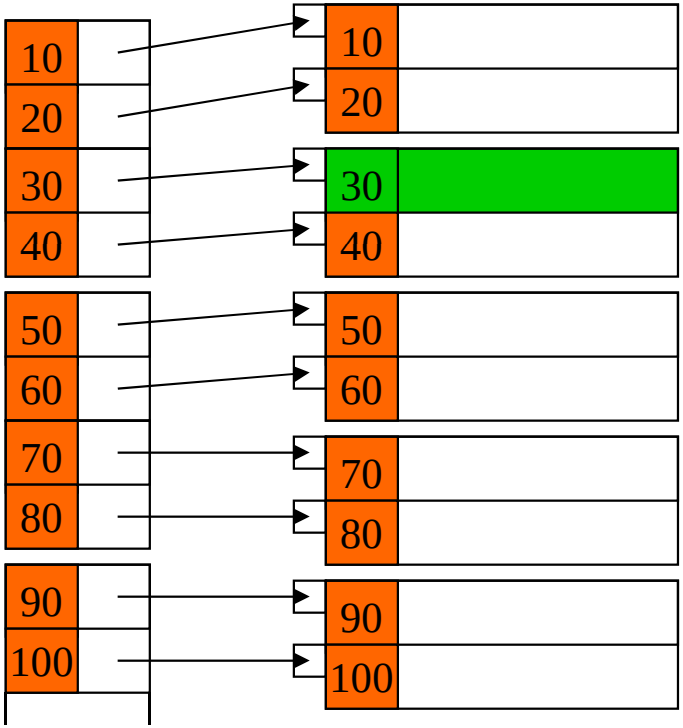
- W razie potrzeby tworzy się (lub usuwa) bloki nadmiarowe (wyłącznie jako poszerzenie bloków oryginalnych, czyli bez wskazań na nie w pliku indeksu rzadkiego)
- Zamiast nadmiarowych możliwe jest tworzenie nowych bloków w pliku sekwencyjnym (z zachowaniem kolejności rekordów)
- w razie potrzeby rekordy przesuwane są między sąsiednimi blokami.

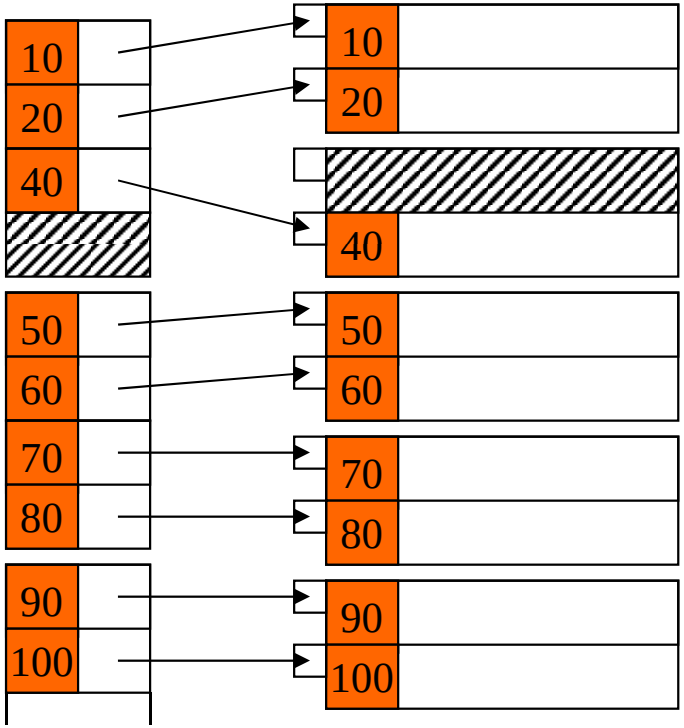
Skutki działań na pliku z danymi w pliku indeksowym:

	Gęsty	**Rzadki**
Tworzenie pustego bloku nadmiarowego	brak	brak
Usuwanie pustego bloku nadmiarowego	brak	brak
Tworzenie pustego bloku sekwencyjnego	brak	wstawianie
Usuwanie pustego bloku sekwencyjnego	brak	usuwanie
Wstawianie rekordu	wstawianie	(?)aktualizacja
Usuwanie rekordu	usuwanie	(?)aktualizacja
Przesuwanie rekordu	aktualizacja	(?)aktualizacja

Zmiany w plikach sekwencyjnych i indeksu

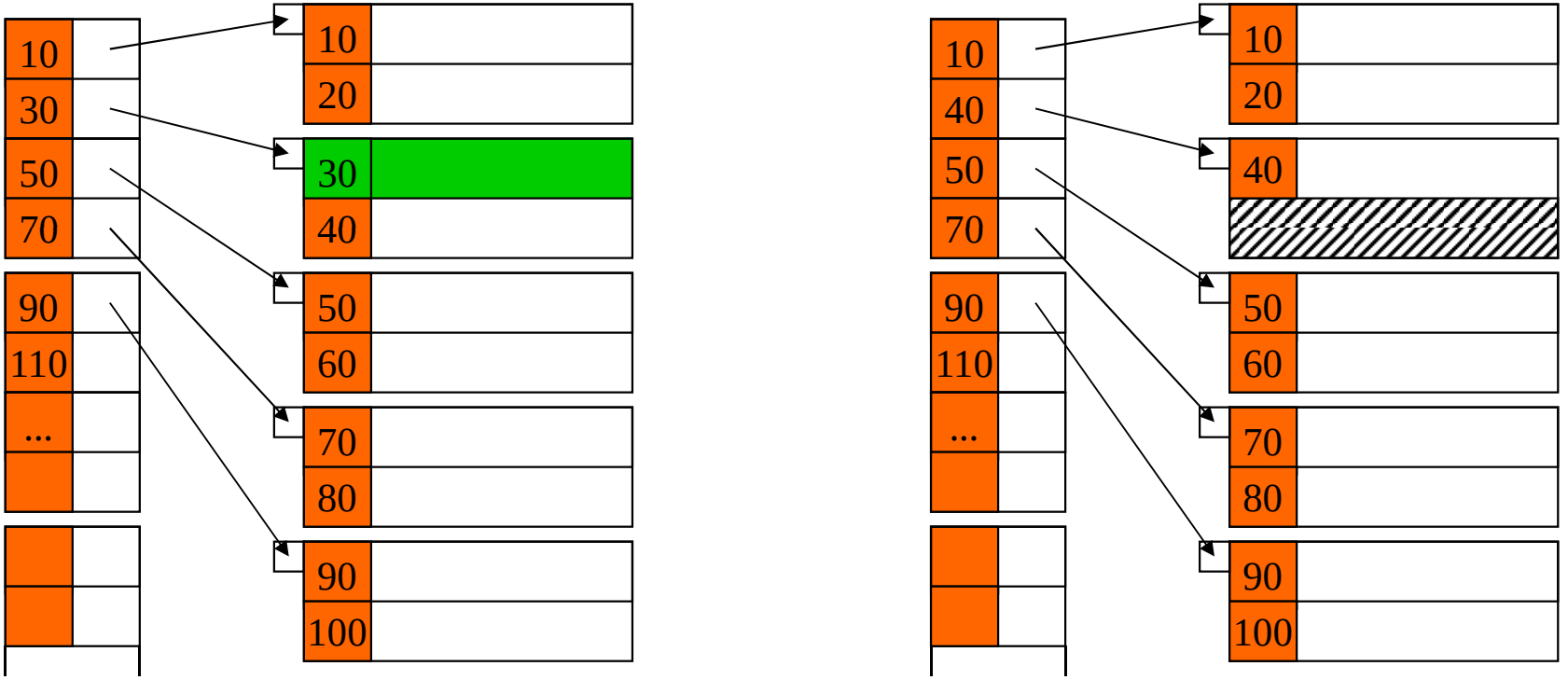
- Przykład usuwania z pliku sekwencyjnego z indeksem gęstym (rekord o kluczu 30)





Zmiany w plikach sekwencyjnych i indeksu

- Przykład usuwania z pliku sekwencyjnego z indeksem rzadkim (rekord o kluczu 30)



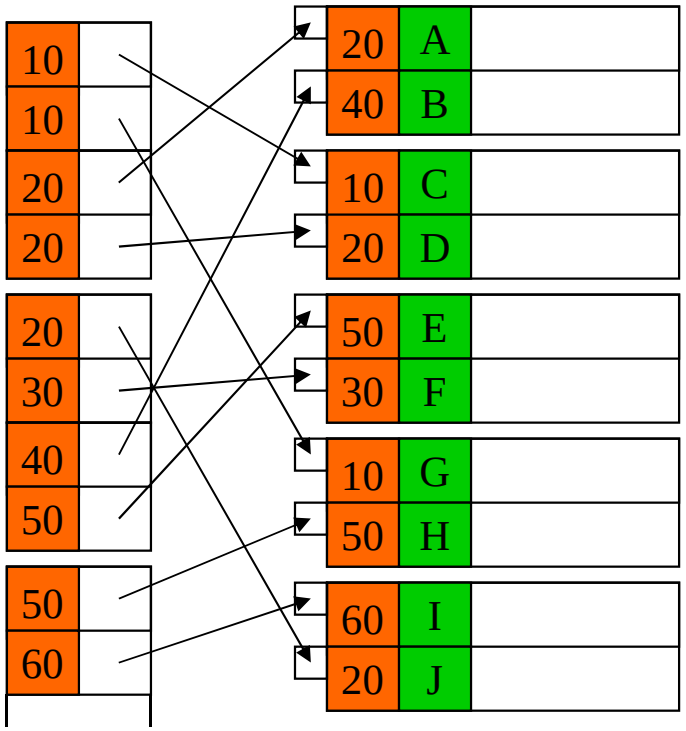
Indeksy pomocnicze

- Dotychczasowe indeksy (**indeksy główne**) określały położenie rekordów.

```
SELECT nazwisko, adres FROM OSOBY
  WHERE data_urodzenia = to_date('15.11.1974','DD.MM.YYYY');
```

- Indeks główny na polu nazwisko nie wspiera zapytania z warunkiem na datę urodzenia;
- **Indeks pomocniczy** utworzony na polu data_urodzenia może wspierać takie zapytanie
- Indeks pomocniczy nie wyznacza pozycji rekordów w pliku z danymi, wskazuje jedynie ich bieżącą lokalizację.
- Nie ma więc sensu mówić o rzadkim indeksie pomocniczym (nie dałoby się ustalić wg takiego indeksu rekordów jawnie w nim nie występujących).
- Można stworzyć drugi poziom indeksu, który może już być rzadki.

Indeksy pomocnicze



Indeksy bitmapowe

- Niech zbiór danych ma stałe numery 1,...,n
- Indeks bitmapowy pola F jest zbiorem wektorów bitowych o długości n, z których każdy odpowiada jednej wartości, jaka może wystąpić w polu F
- Na i-tej pozycji wektora odpowiadającego wartości x występuje:
  - bit równy 1, jeśli wartość pola F w i-tym rekordzie wynosi x
  - bit równy 0, jeśli jest to inna wartość

Przykład:

A	B
30	FU
30	BAR
40	BAZ
50	FU
40	BAR
30	BAZ

Indeks bitmapowy dla pola A:

30:	110001
40:	001010
50:	000100

Indeks bitmapowy dla pola B:

FU:	100100
BAR:	010010
BAZ:	001001

Indeksy bitmapowe

- Zaleta - indeks bitmapowy wspiera zapytania z częściowym dopasowaniem.

Przykład: Tabela zawiera bazę klientów kupujących biżuterię. Wśród informacji w rekordzie znajdują się pola *wiek* i *zarobki*

wiek	zarobki
25	60
45	60
50	75
50	100
50	120
70	110
85	140
30	260
25	400
45	350
50	275
60	260

Indeks bitmapowy dla pola *wiek*

25:	100000001000
30:	000000010000
45:	010000000100
50:	001110000010
60:	000000000001
70:	000001000000
85:	000000100000

Indeks bitmapowy dla pola *zarobki*

60:	110000000000
75:	001000000000
100:	000100000000
110:	000001000000
120:	000010000000
140:	000000100000
260:	000000010001
275:	000000000010
350:	000000000100
400:	000000001000

Indeksy bitmapowe

Szukamy nabywców biżuterii w wieku **45-55** i zarobkach **100-200**.

Indeks bitmapowy dla pola *wiek*

25:	100000001000
30:	000000010000
45:	**010000000100**
50:	**001110000010**
60:	000000000001
70:	000001000000
85:	000000100000
	011110000110

or

Indeks bitmapowy dla pola *zarobki*

60:	110000000000
75:	001000000000
100:	**000100000000**
110:	**000001000000**
120:	**000010000000**
140:	**000000100000**
260:	000000010001
275:	000000000010
350:	000000000100
400:	000000001000
	000111100000

or

011110000110

and

000111100000

000110000000

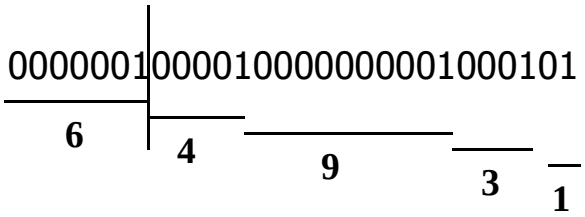
Indeksy bitmapowe

Wynik ***000110000000*** oznacza, że rekordy czwarty i piąty spełniają zadane kryteria.

	wiek	zarobki
1	25	60
2	45	60
3	50	75
4	50	100
5	50	120
6	70	110
7	85	140
8	30	260
9	25	400
10	45	350
11	50	275
12	60	260

Indeksy bitmapowe skompresowane

- Obserwacja: indeksy bitmapowe mogą wymagać dużej ilości miejsca do ich zapisania.
- Łączna ilość bitów jest iloczynem liczby rekordów (*n*) i liczby różnych wartości w polu (*m*)
- przy dużym *m* (w stosunku do *n*) liczba jedynek w wektorze bitowym jest niewielka (wartość rzadko się powtarza.
- Warto wówczas zakodować wektory bitowe tak, aby zajmowały mniej miejsca (mniej niż *n* bitów).
- Prosta metoda - **kodowanie długości serii:**
  - Zapis przebiegu (czyli ciągu *k* zer i jednej jedynki) za pomocą liczby *k* (binarnie)



Takich binarnych zapisów przebiegów nie można wprost skonkatenować, bo nie jest to jednoznaczne

Indeksy bitmapowe skompresowane

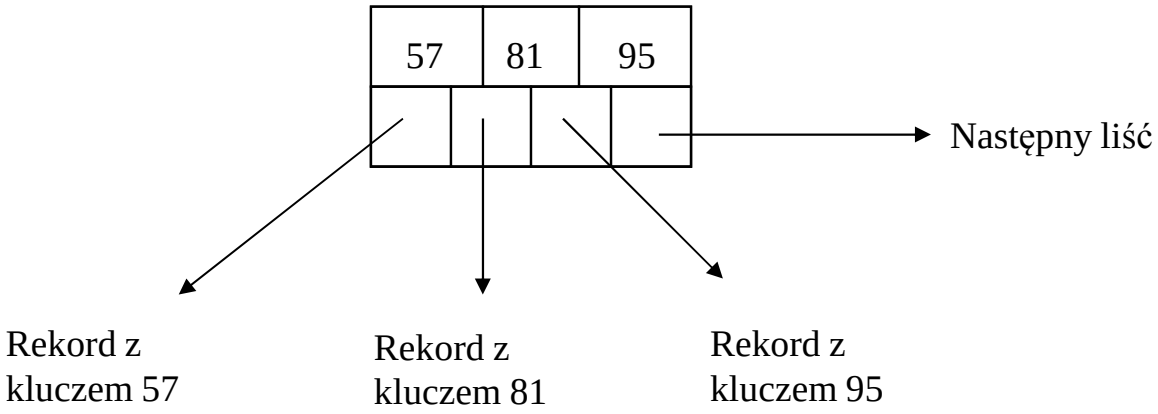
- metoda z określeniem długości zapisu binarnego liczby *k* poprzedzającym samą liczbę.
- Niech $j = \lg k$ zapisane „jedynkowo” (*j*-1 jedynek i zero)
- Dla przebiegu 00000000000001 (czyli *k=13*) mamy *j=4*.

  Zapis 11101101
- Szczególne przypadki:
  - *k* = 1 (zapisujemy 01)
  - *k* = 0 (zapisujemy 00)

- Rozkodowanie przebiega analogicznie, np:
  11101101001011

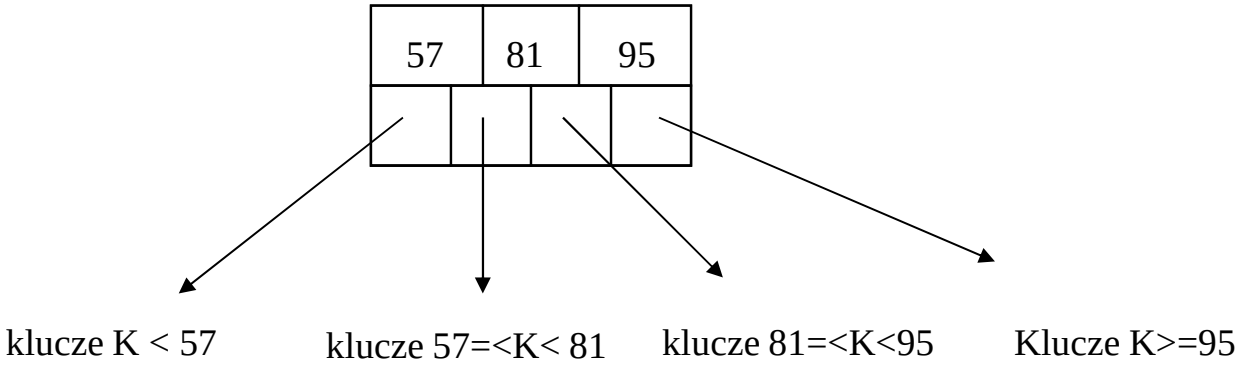
B-drzewa

- Zamiast indeksów jedno lub wielopoziomowych często wykorzystywane są B-drzewa.
- Przestrzeń dyskowa jest tak zorganizowana, że każdy blok jest zapełniany więcej niż w połowie, ale nie do samego końca.
- Powstające drzewo jest zrównoważone.
- **Liść** B-drzewa

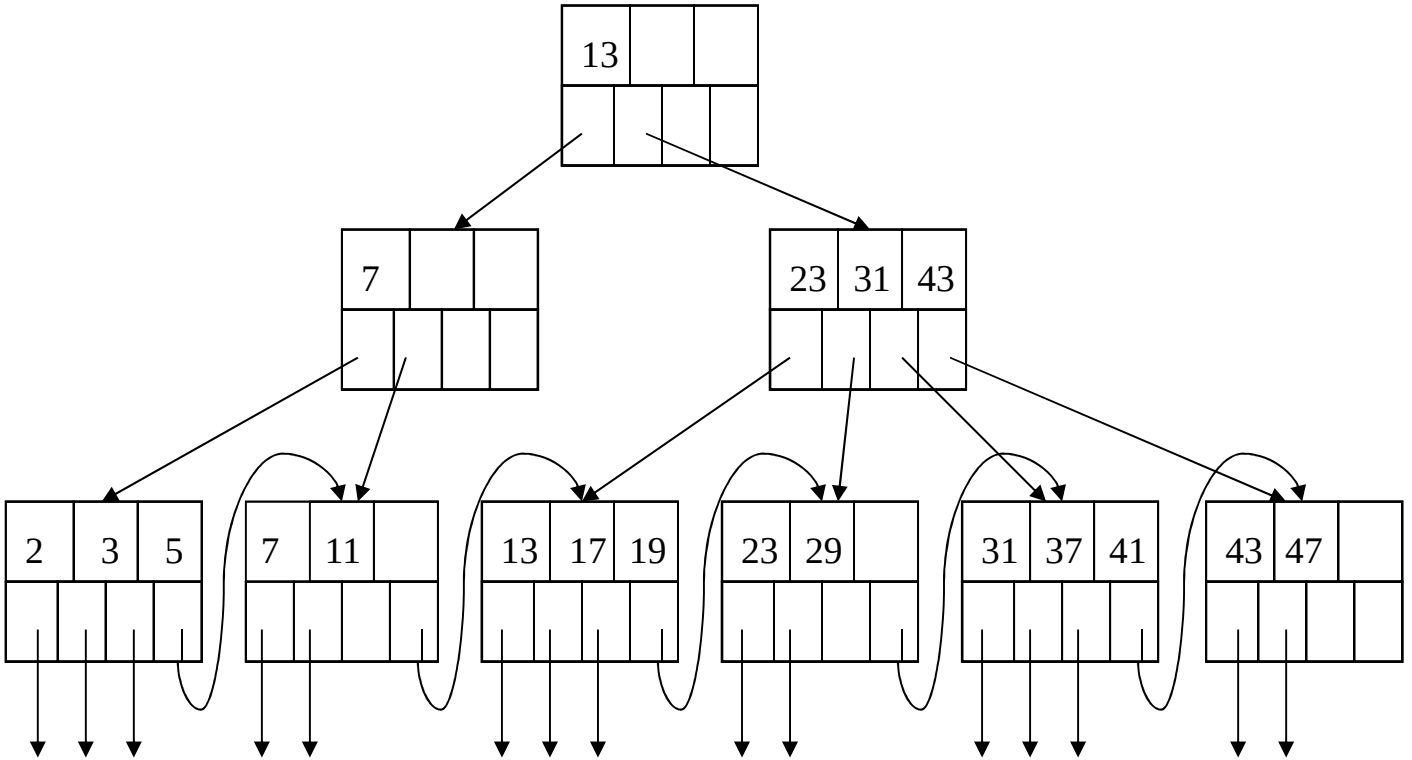


B-drzewa

- **Węzeł wewnętrzny** B-drzewa



B-drzewa



B-drzewa

- z indeksem typu B-drzewo związany jest parametr n wyznaczający jego układ.
- W każdym bloku znajduje się n wartości klucza wyszukiwania i n+1 wskaźników.
- W poprzednim przykładzie n=3.
- **Reguły:**
  - Z korzeniem stowarzyszone są co najmniej dwa wskaźniki, wskazujące na bloki z niższych poziomów.
  - Ostatni wskaźnik w liściu wskazuje na blok sąsiedniego liścia (czyli ten, w którym znajduje się kolejny klucz).
  - Pośród pozostałych n pozostałych wskaźników ca najmniej $\left\lfloor \frac{n+1}{2} \right\rfloor$ wskazuje na rekordy z danymi.
  - Wszystkie n+1 wskaźników w węźle wewnętrznym może wskazywać na bloki kolejnego niższego poziomu, wykorzystane są co najmniej $\left\lceil \frac{n+1}{2} \right\rceil$ wskaźniki
  - dla kluczy K1, ...,Kj wskaźniki wskazują odpowiednio: pierwszy - bloki B-drzewa z wskaźnikami do rekordów o kluczach mniejszych od K1, drugi - bloki z wskaźnikami do rekordów o kluczach większych lub równych od K1, ale mniejszych od K2, itd....

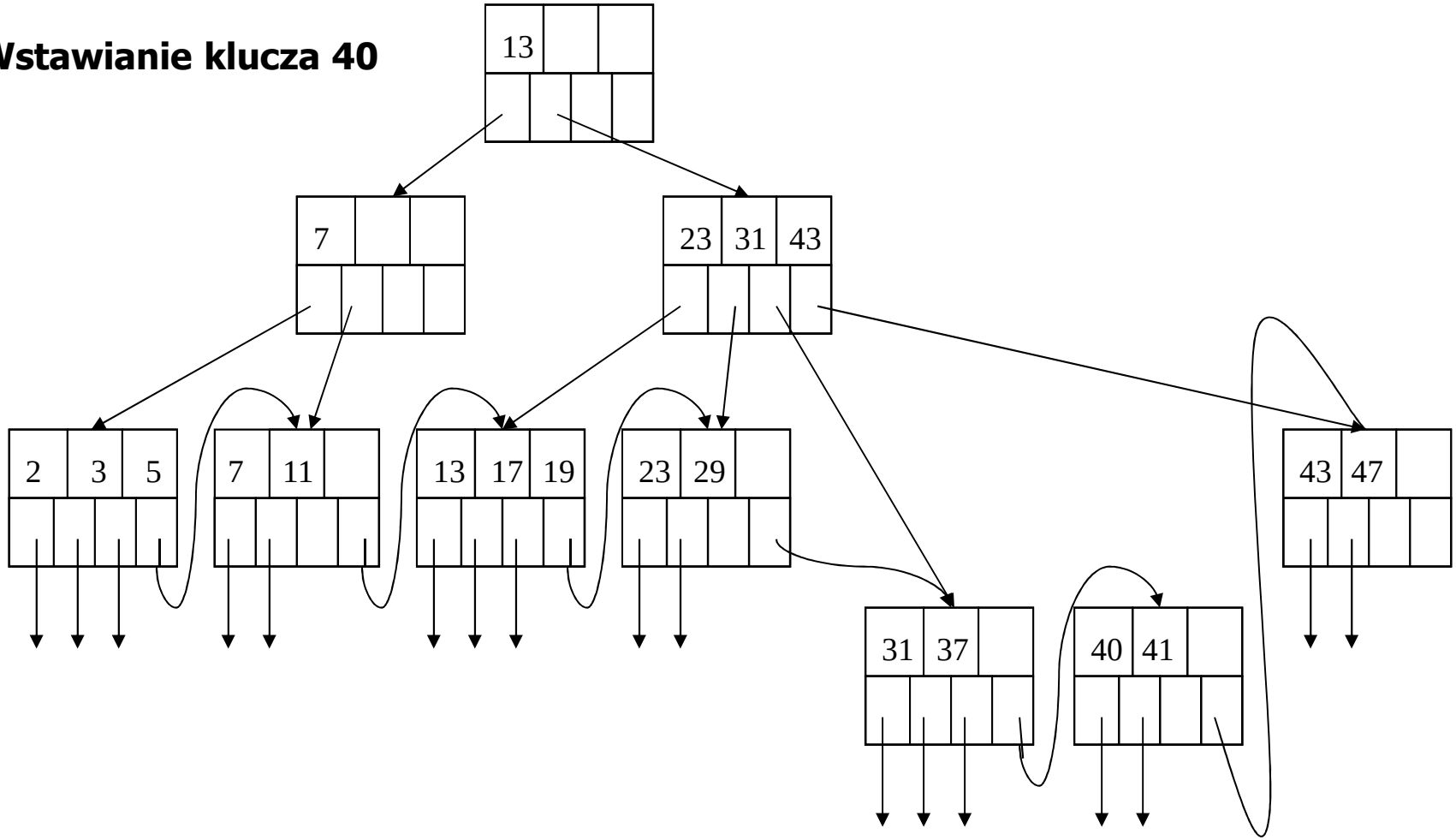
- Wyszukiwanie w B-drzewach odbywa się rekurencyjnie (zakładamy, że klucze w liściach nie powtarzają się):
  - KROK PODSTAWOWY: Jeśli znajdujemy się w liściu, to przeglądamy klucze w poszukiwaniu klucza K. Gdy i-ty klucz ma wartość K, to i-ty wskaźnik prowadzi do szukanego rekordu.
  - KROK INDUKCYJNY: Jeśli znajdujemy się w węźle wewnętrznym z kluczami K1,K2,...,Kn, to wg powyższej reguły ustalamy blok z jedynym „potomkiem" prowadzącym do liścia z kluczem K.

B-drzewa - wstawianie

- Wstawianie również odbywa się rekurencyjnie:
  - W odpowiednim liściu znajdujemy miejsce na nowy klucz
  - jeśli miejsce jest, to klucz zostaje wstawiony.
  - Jeśli miejsca nie ma, to rozdzielamy liść na dwa węzły tak, aby każdy był zapełniony w połowie (lub nieco ponad połowę)
  - odnotowujemy to rozbicie na wyższym poziomie (nowa para klucz-wskaźnik), rekurencyjnie stosując tę samą zasadę.
  - Jeśli zachodzi potrzeba rozbicia korzenia, to tworzony jest nowy korzeń z potomkami powstałymi z podziału poprzedniego korzenia.

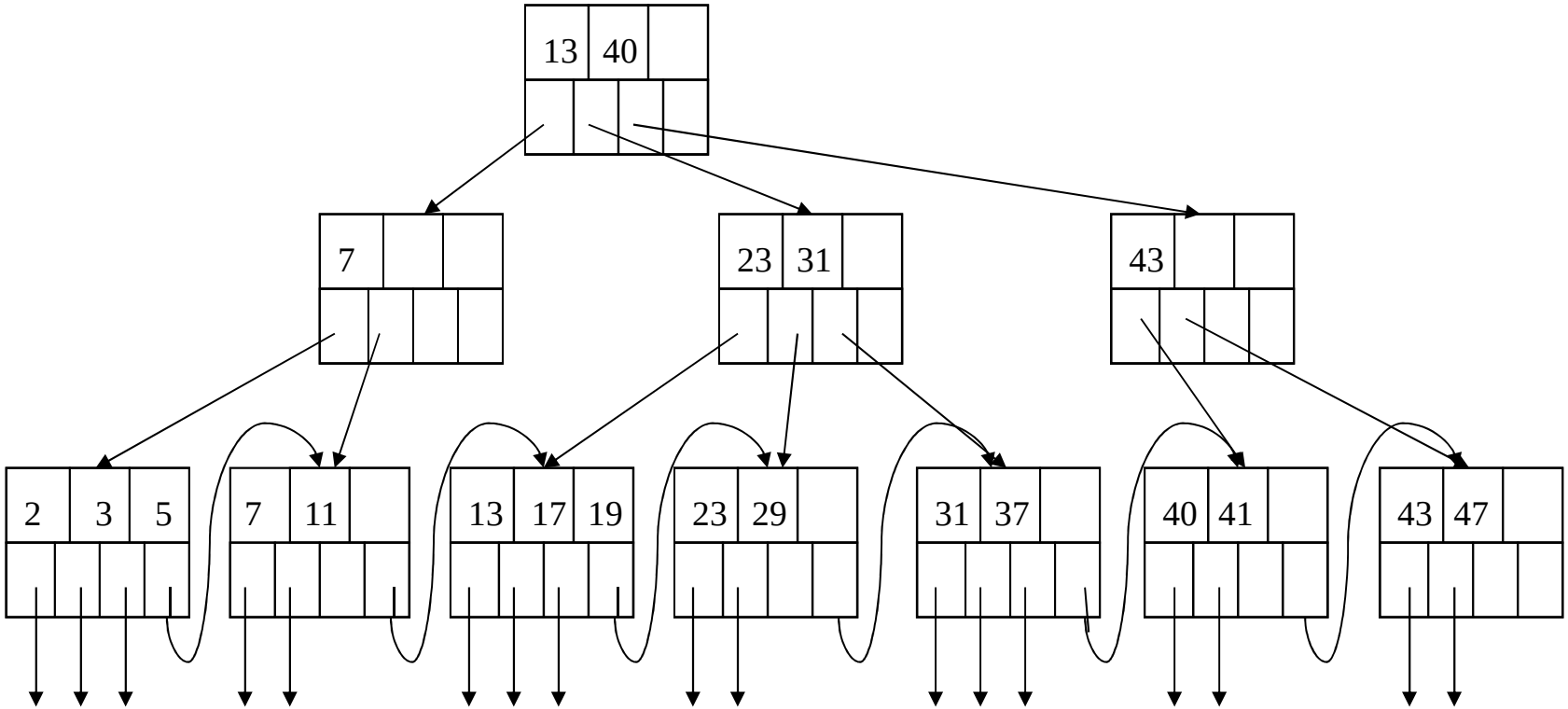
B-drzewa - wstawianie

- **Wstawianie klucza 40**



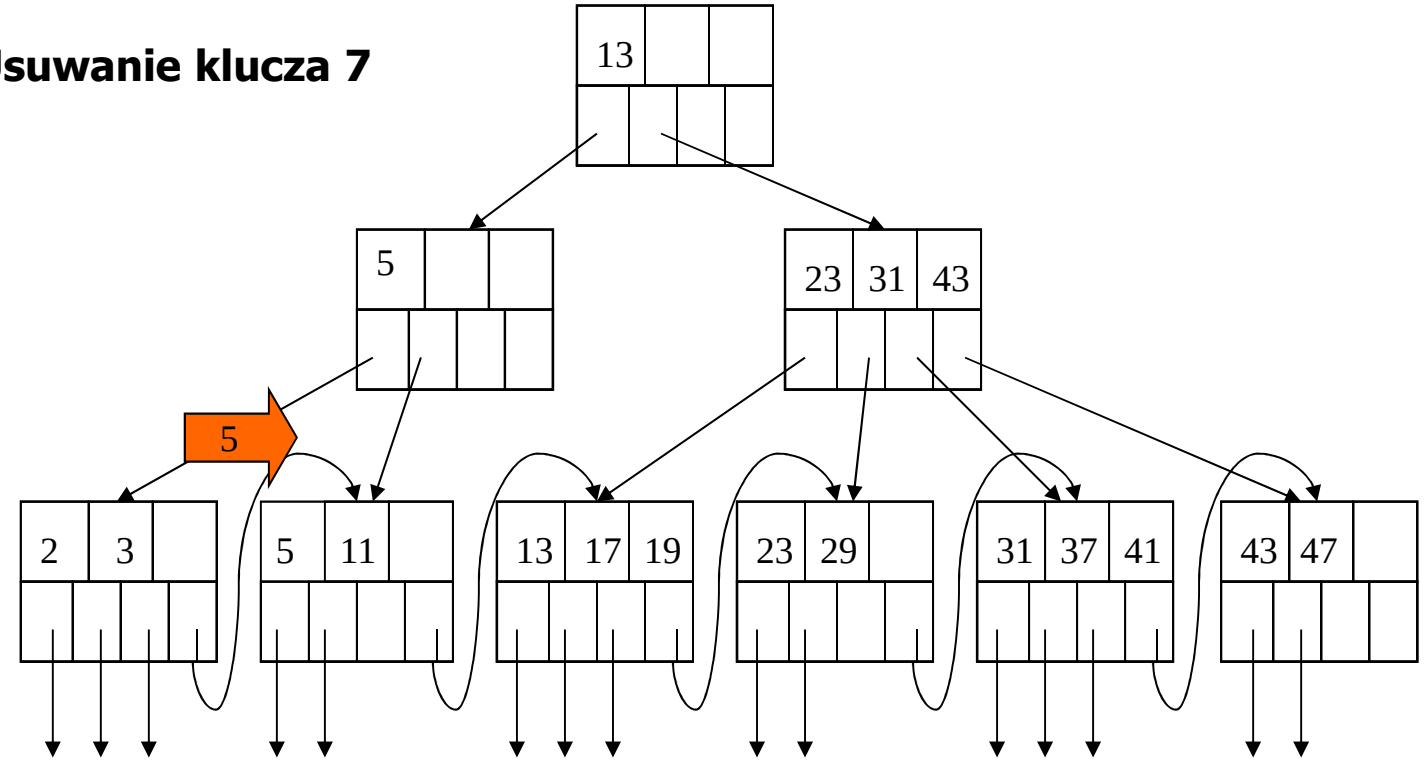
B-drzewa - wstawianie

- **aktualizacja węzłów wyższych poziomów**



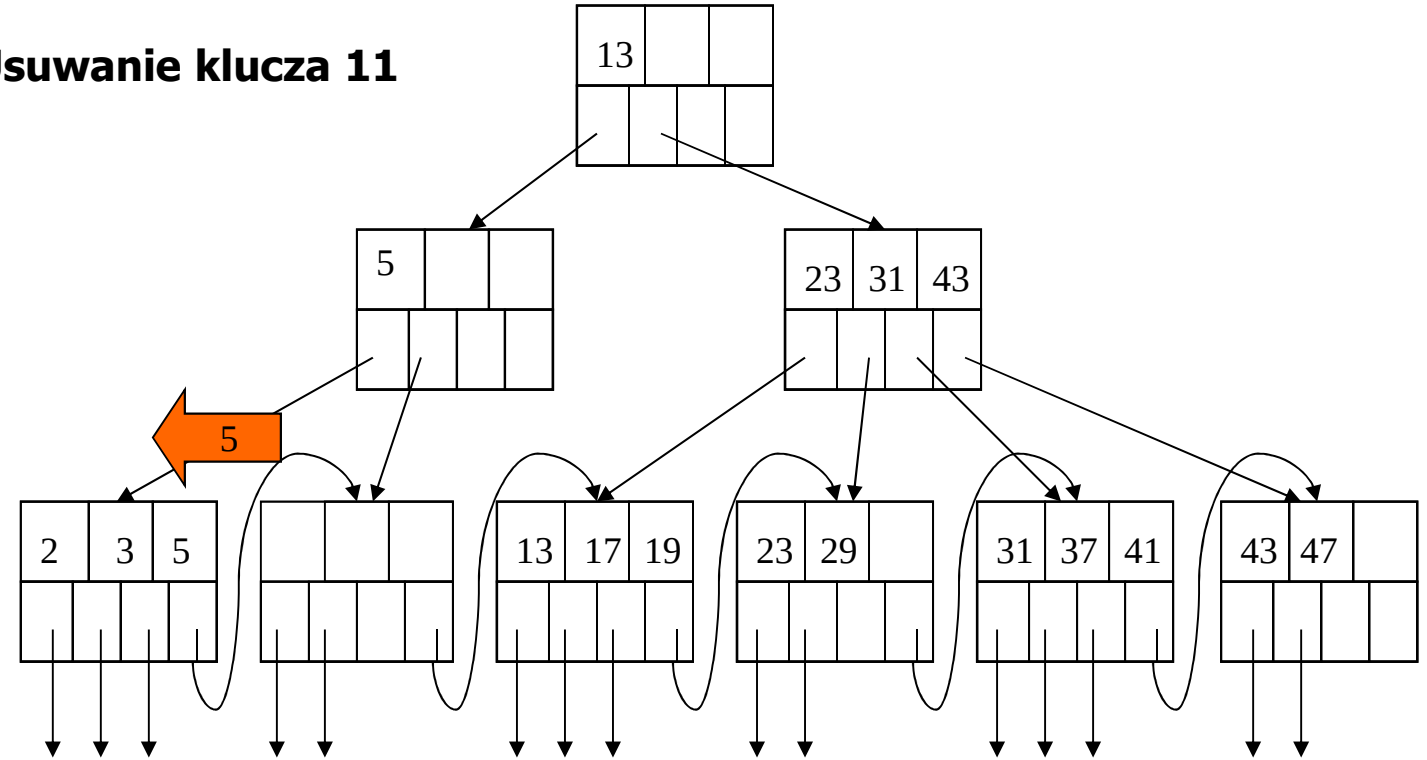
B-drzewa - usuwanie

- **Usuwanie klucza 7**



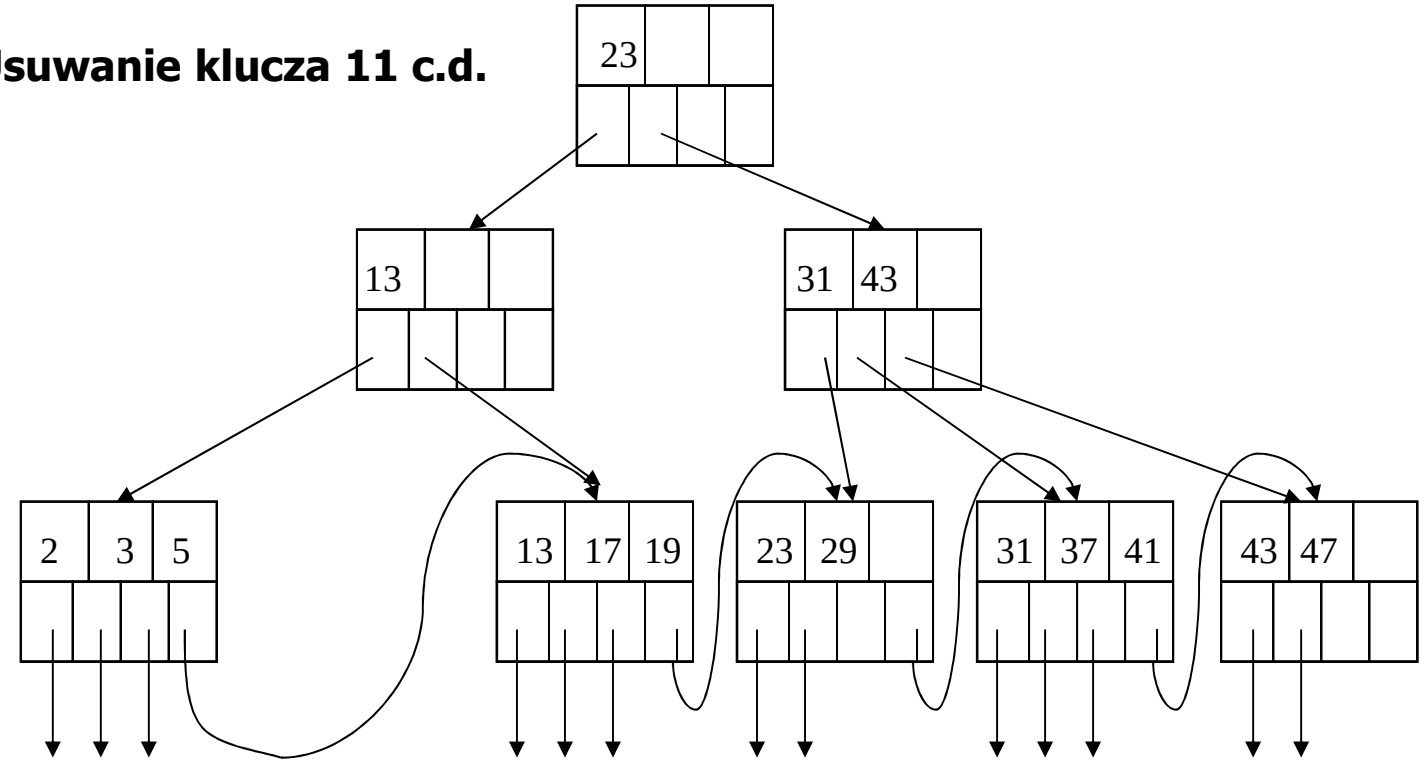
B-drzewa - usuwanie

- **Usuwanie klucza 11**



B-drzewa - usuwanie

- **Usuwanie klucza 11 c.d.**



B-drzewa - efektywność

- Przy dużej ilości n (liczba kluczy w bloku, 10 lub więcej) scalanie lub podział nie zdarza się zbyt często, a gdy się zdarzy, to zwykle dotyczy tylko liści (zmiana więc obejmuje dwa sąsiednie liście i ich rodzica). Koszt będący czasem potrzebnym na reorganizację można więc w zasadzie zaniedbać.
- Wyszukiwanie wymaga przejścia przez wszystkie poziomy B-drzewa
- Ile poziomów ma B-drzewo?

- **Przykład:** w bloku 4KB dla kluczy 4-bajtowych i 8-bajtowych wskaźników mieści się 340 par klucz-wskaźnik.
- Uwzględniając ograniczenia dotyczące ilości wskaźników w bloku, niech średnio znajduje się w nim 255 wskaźników.
- Dla 3-poziomowego drzewa mamy korzeń, 255 węzłów pośrednich i 65025 liści.
- W każdym liściu znajduje się 255 wskaźników, czyli mamy około 16.6 miliona wskazań na rekordy.
- Jedynie bardzo duże relacje wymagać będą drzew większych niż 3 poziomowe.