# Pliki kratowe

# Definicja pliku kratowego

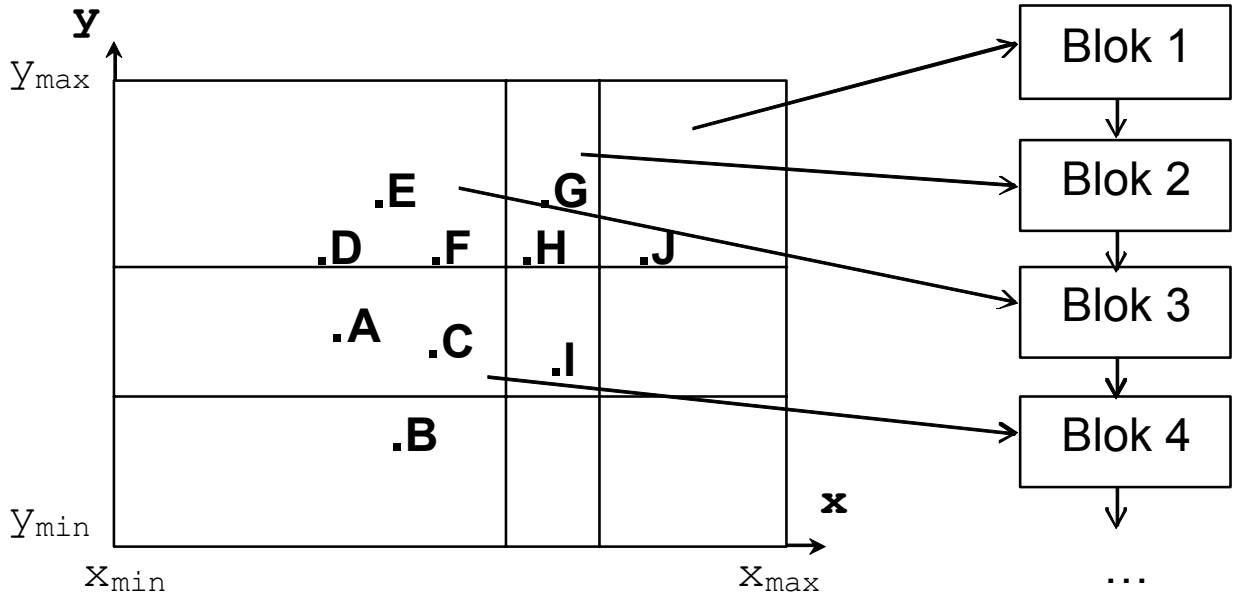
***Plik kratowy*** (ang. *grid file*) jest strukturą wspierająca realizację zapytań wielowymiarowych. Uporządkowanie rekordów, zawierających dane wielowymiarowe w pliku kratowym, zależy od ich lokalizacji w przestrzeni wymiarów.

Plik kratowy implementuje odwzorowanie:

***lokalizacja przestrzenna -> alokacja na dysku***

Blok dyskowy, w którym zostanie umieszczony dany rekord, jest wyznaczany za pomocą wielowymiarowej tablicy, która przechowuje adresy wszystkich bloków dyskowych pliku. Poszczególne elementy tablicy odpowiadają *regionom* - przestrzennym prostopadłościanom, na które podzielona jest wielowymiarowa przestrzeń dziedzin atrybutów. Kolejność bloków w pliku nie odzwierciedla sąsiedniości regionów.

Regiony są parami rozłączne, a ich suma tworzy całą przestrzeń (dziedzinę).



# Struktura pliku kratowego

Struktura pliku kratowego przechowującego dane z przestrzeni k-wymiarowej składa się z trzech elementów:

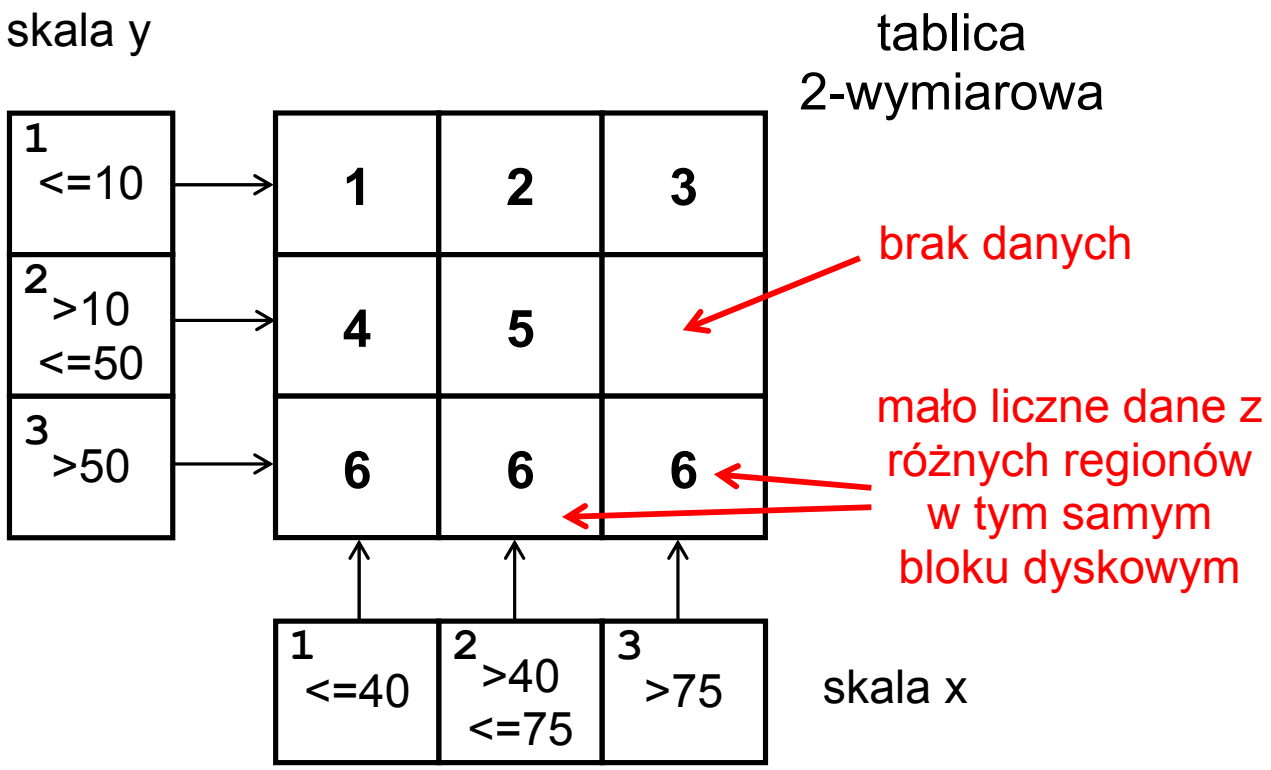
**Zbiór *k* skal** – skale są jednowymiarowymi tablicami opisującymi sposób podziału przestrzeni na regiony; służą do adresowania tablicy wielowymiarowej;

**Tablica k-wymiarowa** – jej elementy reprezentują poszczególne regiony i zawierają wskaźniki na bloki danych. W przypadku słabego wypełnienia bloków kilka elementów tablicy może adresować ten sam blok. Jeżeli dany region nie zawiera danych, to odpowiadający mu element tablicy jest pusty.

**Bloki danych** – zawierają rekordy danych, zlokalizowane w tej samej podprzestrzeni. Wypełnienie bloków może być mniejsze (dla wielu wymiarów znacznie mniejsze) niż 50%.

W modelu zapytań przyjmuje się, że tablica wielowymiarowa i skale są przechowywane w pamięci operacyjnej.

# Struktura pliku kratowego




| | |
|---|---|
| 1 | rekord_1 <29, 8, …><br>rekord_2 <35, 2, …><br>rekord_3 <0, 9, …> |
| 2 | rekord_4 <42, 7, …><br>rekord_5 <70, 1, …><br>rekord_6 <51, 5, …> |
| … | … |
| 6 | rekord_16 <25, 55, …><br>rekord_17 <38, 80, …><br>rekord_18 <49, 69, …><br>rekord_19 <100, 53, …> |

# Wyszukiwanie danych za pomocą plików kratowych

Algorytm wyszukiwania danych dla modelu zapytań punktowych:

**Znajdź**(**x,y**)

1. Wyszukaj na skali **X** przedział zawierający **x**. Zwróć indeks **i** przedziału.
2. Wyszukaj na skali **Y** przedział zawierający **y**. Zwróć indeks **j** przedziału.
3. Pobierz z wielowymiarowej tablicy adres bloku **p** pamiętany w elemencie o adresie (**i,j**).
4. Odczytaj blok danych o adresie **p**. Blok zawiera szukany rekord lub danego rekordu nie ma w bazie danych.

Algorytm wyszukiwania danych dla modelu zapytań o częściowym dopasowaniu:

**Znajdź**(**x**)

1. Wyszukaj na skali **X** przedział zawierający **x**. Zwróć indeks **i** przedziału.
2. Pobierz z wielowymiarowej tablicy adresy bloku **p** pamiętanych w elementach o adresach (**i,?**).
3. Odczytaj bloki danych o adresie **p**. Bloki zawierają szukane rekordy lub danych rekordów nie ma w bazie danych.

# Wyszukiwanie danych za pomocą plików kratowych

Algorytm wyszukiwania najbliższego sąsiada:

**ZnajdźNajbliższegoSąsiada**(x,y)

1. Znajdź blok **p** zawierający punkt o współrzędnych (x,y).
2. Szukaj najbliższego sąsiada w bloku **p**. Jeżeli odległość od znalezionego sąsiada jest mniejsza niż odległości danego punktu od granic podprzestrzeni oznacza to koniec wyszukiwania.
3. Jeżeli odległość od znalezionego sąsiada jest większa niż odległość danego punktu od granic przestrzeni, to odczytaj sąsiednie bloki przy tych granicach i szukaj w nich punktów bliżej położonych.

Dla wszystkich modeli zapytań stosowanie pliku kratowego w większości przypadków zmniejsza, a w najgorszych przypadkach, nie pogarsza kosztów wyszukiwania.

# Utrzymywanie plików kratowych

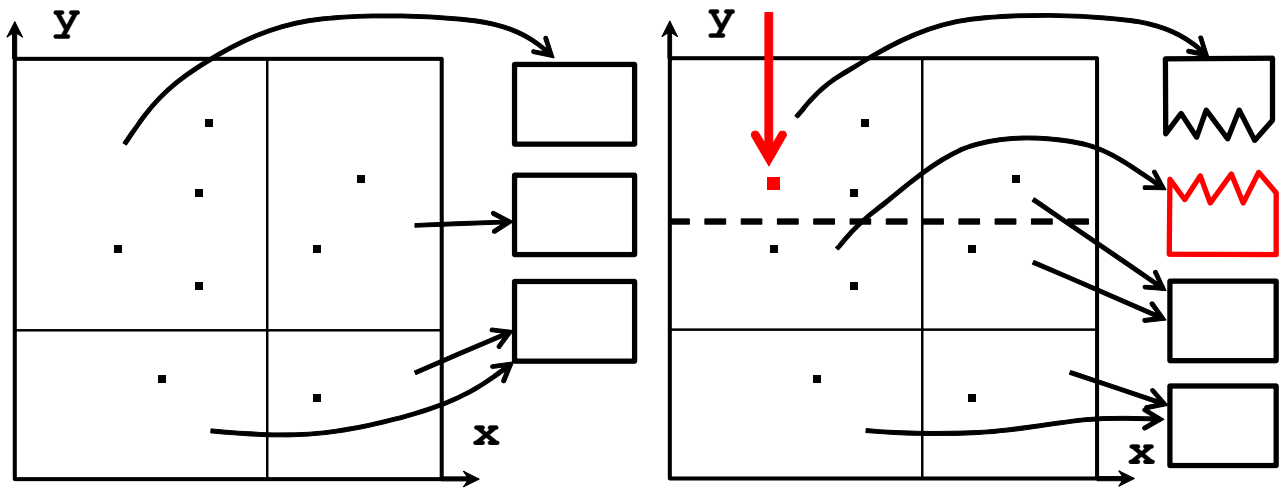
## Wstawianie nowych rekordów

Jeżeli blok do którego powinien trafić nowy rekord jest pełny należy dokonać podziału odpowiadającej mu podprzestrzeni. W pliku kratowym podział danej podprzestrzeni będzie propagowany do wszystkich podprzestrzeni leżących na linii cięcia.

Wybór osi podziału może być sekwencyjny - podział według osi: x, y, z, x, y, z, itd., lub priorytetowy - podział według osi: x, x, x, y, y, z, x, x, x, itd.

Powyższe algorytmy podziału bloku nie gwarantują równomiernego wypełnienia. Równomierny podział przepełnionego bloku optymalny w momencie przepełnienia, nie gwarantuje równomiernego podziału innych dzielonych przy okazji regionów.

Dzielenie nieprzepełnionych obszarów nie wiąże się z tworzeniem nowych bloków.



# Problemy z utrzymaniem plików kratowych

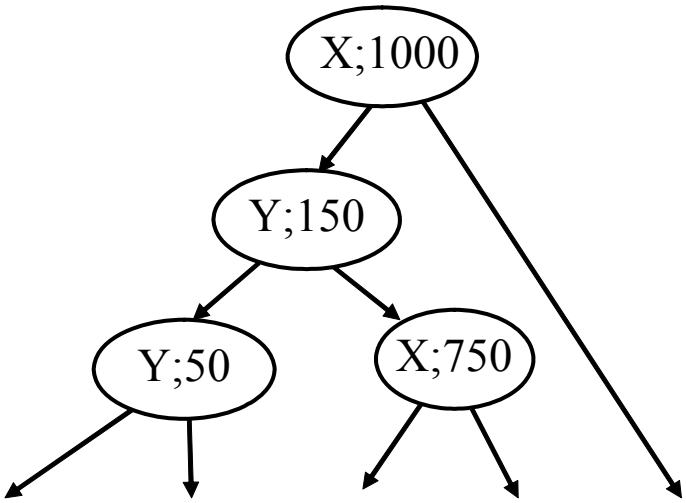
- Wraz z podziałem bloków rośnie, ekspotencjalnie od liczby wymiarów, liczba elementów tablicy wielowymiarowej. Dla tabeli wielowymiarowej o rozmiarze: `m` x `n` x `p`, podział wzdłuż osi p oznacza utworzenie `m` x `n` nowych regionów.
- Dla wielu wymiarów większość regionów tablicy wielowymiarowej jest pusta.
- Brak skalowalności dla bardzo dużych zbiorów danych. Rozmiar tablicy wielowymiarowej i skal może przekroczyć rozmiar dostępnej pamięci operacyjnej.

# Wielowymiarowe drzewa binarne

# kd-drzewa

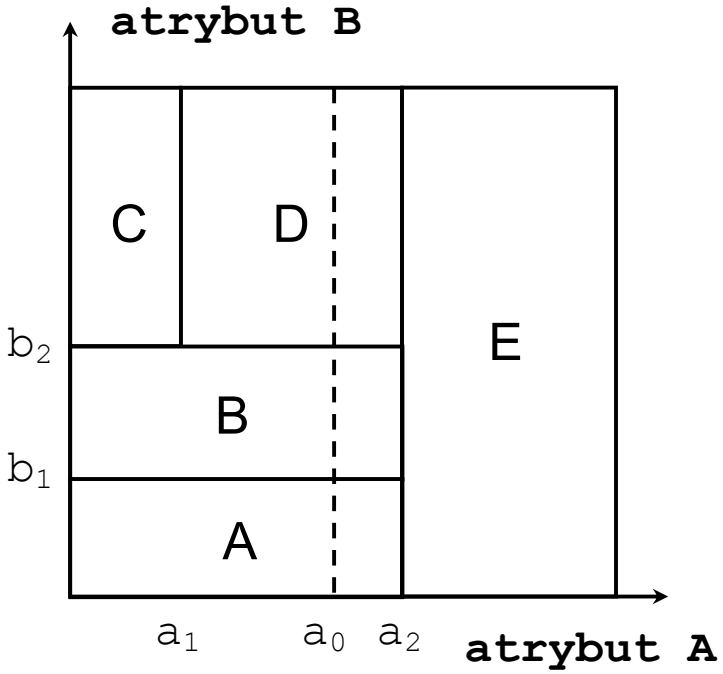
## k-d-drzewo – wielowymiarowe drzewo binarne

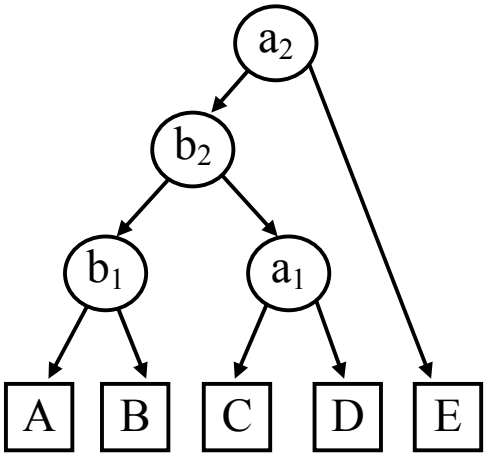
- Każdy węzeł drzewa reprezentuje wartość jednego z atrybutów reprezentujących wymiary przeszukiwanej przestrzeni.
- Kolejność występowania atrybutów w kolejnych poziomach drzewa może być przypadkowa. Wynika stąd potrzeba pamiętania w węzłach oprócz wartości również typów atrybutów.
- W procesie szukania jeżeli wartość odpowiedniego atrybutu szukanego klucza jest mniejsza od wartości danego węzła proces będzie kontynuowany dla potomka wskazywanego przez lewą krawędź węzła, w przeciwnym przypadku dla potomka wskazywanego przez prawą krawędź.




Dane mogą być pamiętane w węzłach drzewa lub w wyróżnionych liściach drzewa.

# Zastosowanie kd-drzew do podziału przestrzeni na regiony



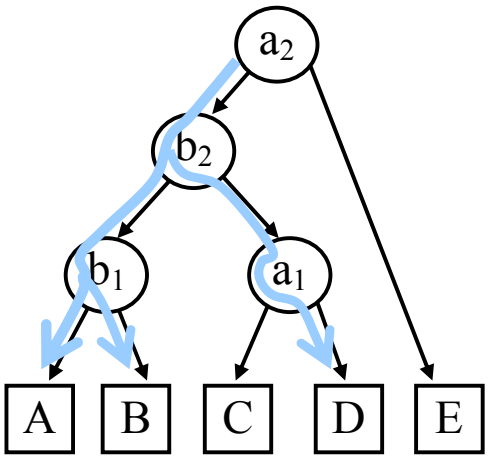



# Szukanie danych za pomocą kd-drzew

Operacja wyszukiwania punktowego dla w pełni zrównoważonego drzewa wymaga maksymalnie przejrzenia $\lceil \log_2 r \rceil$ węzłów, gdzie $r$ jest liczbą wszystkich danych.

## Zapytania o częściowym dopasowaniu

Znajdź wszystkie dane, dla których wartość atrybutu $A$ jest równa $a_0$, gdzie $a_1 < a_0 < a_2$.




Wyszukiwanie wymaga w tym wypadku równoległego nawigowania kilku ścieżek.

# Własności kd-drzew

**Zalety:**

- rząd drzewa jest niezależny od liczby wymiarów *k*
- bardzo efektywne algorytmy utrzymania drzewa
- nie nakładanie się regionów

**Wady:**

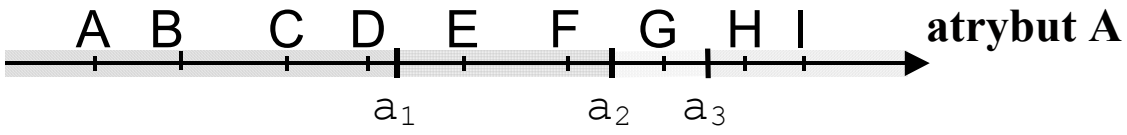
- pokrycie pustych regionów
- struktura zależna od kolejności wykonywania operacji
- metoda dedykowana jedynie dla danych w pamięci operacyjnych

# Indeksy wielowymiarowe

# Indeksy wielowymiarowe

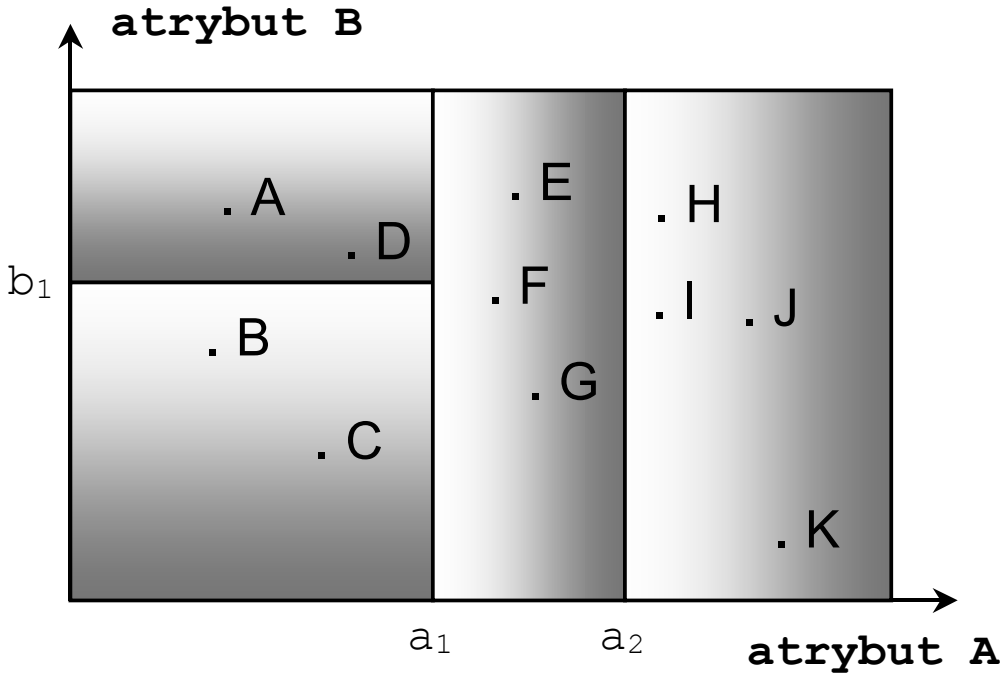
## Węzeł indeksu jednowymiarowego

- podział jednowymiarowej dziedziny na przedziały




## Węzeł indeksu wielowymiarowego

- podział wielowymiarowej dziedziny na regiony



# kdB drzewo

## Idea

Rozszerzenie stosowalności na przeszukiwanie plików dyskowych: połączenie struktur *k-d-drzew* i *B-drzew*.

## Założenia

1. Format rekordów indeksu:

   $klucz_0, klucz_1, \ldots klucz_{K-1}, lokalizacja$

   gdzie: $klucz_i \in dziedzina_i$,
   $K$ - jest stałą określającą liczbę wymiarów,
   *lokalizacja* - jest adresem rekordu danych o podanych wartościach kluczy.

2. Wsparcie dla zapytań typu:

   $$min_i \leq klucz_i \leq max_i, \quad 0 \leq i \leq K - 1$$

3. Dynamiczne utrzymywanie struktury: wsparcie dla operacje wstawiania i usuwania rekordów przeplatanych z operacjami zapytań.

4. Bardzo duża liczba rekordów wykluczająca utrzymanie struktury w pamięci operacyjnej.

- Dla $K = 1$ rozwiązaniem spełniającym powyższe wymagania są $B^+$*-drzewa*.
- Trzy pierwsze wymagania dla dowolnego $K$ spełniają *k-d-drzewa*.

# Struktura kdB drzewa

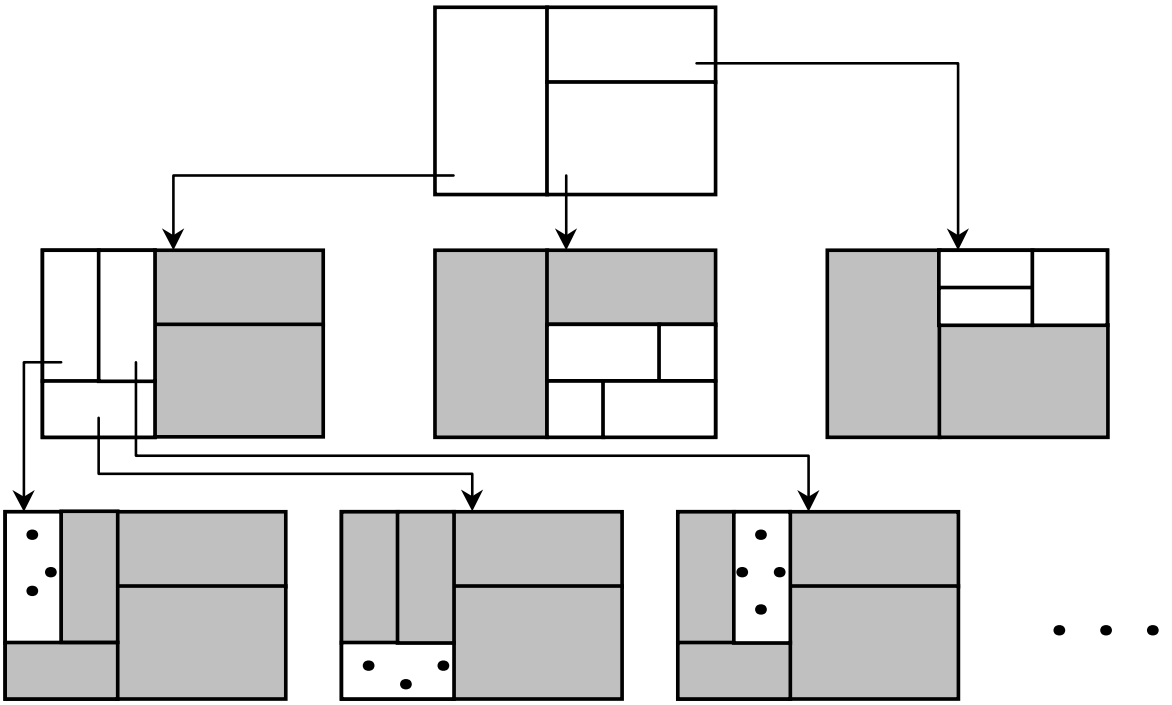
*Punkt* ($x_0, x_1, ..., x_{K-1}$) jest elementem iloczynu kartezjańskiego: $dziedzina_0 \times dziedzina_1 \times ... \times dziedzina_{K-1}$.

*Region* ($l_0, l_1, ..., l_{K-1}$), *gdzie* $l_i = [min_i, max_i)$, jest zbiorem wszystkich punktów ($x_0, x_1, x_2,..., x_{K-1}$) takich, że:

$min_i \leq x_i < max_i$, $0 \leq i \leq K-1$, oraz $min_i , max_i \in dziedzina_i$.

***k-d-B-drzewo*** jest kolekcją dwóch rodzajów węzłów składowanych na odrębnych stronach dyskowych:

1. ***węzłów wewnętrznych*** (ang. *region pages*), które są zbiorami par *(region, wskaźnik)*, gdzie wskaźniki są adresami stron dyskowych zawierających węzły niższego poziomu;
2. ***liści*** (ang. *point pages*), które są kolekcjami par *(punkt, wskaźnik)*, gdzie wskaźniki są adresami rekordów danych o określonych wartościach atrybutów.



**Przykład 2dB drzewa**

# Struktura kdB drzewa

1. k-d-B-drzewa są zrównoważone; droga od korzenia do wszystkich liści jest taka sama.
2. Regiony znajdujące się wewnątrz każdego węzła wewnętrznego są rozłączne, a ich suma tworzy region.
3. Jeżeli korzeń drzewa nie jest liściem, to suma wszystkich jego regionów tworzy region: $dziedzina_0$ *x* $dziedzina_1$ *x ... x* $dziedzina_{K-1}$.
4. Jeżeli dla danego elementu węzła wewnętrznego ($region_i$, $wskaźnik_i$), wskaźnik adresuje potomny węzeł wewnętrzny, wtedy suma regionów węzła potomnego tworzy $region_i$.
5. Jeżeli dla danego elementu węzła wewnętrznego ($region_i$, $wskaźnik_i$), wskaźnik adresuje liść drzewa, wtedy wszystkie punkty należące do tego liścia zawierają się w $region_i$.
6. Nie jest gwarantowany stopień wypełnienia węzłów indeksu ponad 50%.

# Realizacja zapytań

Algorytm realizacji zapytania wyszukuje rekordy spełniające warunki zapytania określone przez *region zapytania*:

$$I_0 \text{ x } I_1 \text{ x } \ldots \text{ x } I_{K-1}.$$

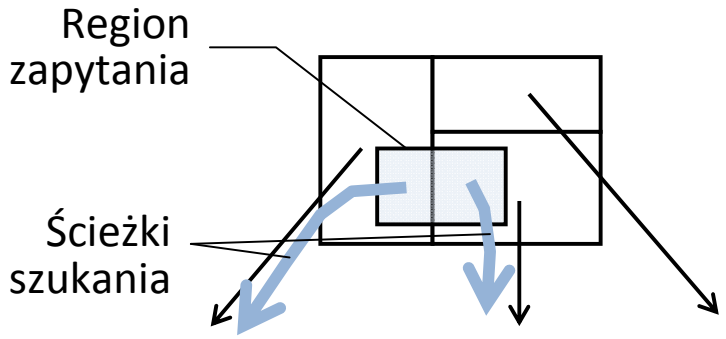
Jeżeli dla wszystkich przedziałów $I_i$: $min_i = max_i$ , gdzie $0 \leq i \leq K - 1$ zapytanie jest nazywane *zapytaniem punktowym*.

Algorytm realizujący zapytanie jest następujący:

(1) Odczytaj stronę dyskową zawierającą korzeń drzewa i podstaw ją pod zmienną *strona*.

(2) Jeżeli *strona* zawiera liść, to dla każdej pary (*punkt*$_i$, *wskaźnik*$_i$) na tej stronie, dla której *punkt*$_i$ należy do *regionu zapytania*, odczytaj i zwróć jako wynik rekord adresowany przez *wskaźnik*$_i$.

(3) Jeżeli *strona* zawiera węzeł wewnętrzny, to dla każdej pary (*region*$_i$, *wskaźnik*$_i$) tego węzła takiej, że iloczyn *region*$_i$ i *region zapytania* jest niepusty, podstaw pod zmienną *strona* stronę dyskową wskazywaną przez wskaźnik$_i$ i rekurencyjnie wywołaj punkt (2).



# Modyfikacje kdB drzew

## Definicja 1

Wynikiem podziału regionu $l_0$ x $l_1$ x ... x $l_{K-1}$ względem elementu $x_i \in$ *dziedziny* są dwa nowe regiony:

(1) *lewy:* $l_0$ x ... x $[min_i, x_i)$ x ... x $l_{K-1}$,

(2) *prawy:* $l_0$ x ... x $[x_i, max_i)$ x ... x $l_{K-1}$.

## Definicja 2

Dla danego wymiaru zachodzą następujące relacje między punktami i między regionami a punktami:

- Region leży na lewo od $x_i$ , jeżeli $x_i < min_i$ regionu.
- Region leży na prawo od $x_i$ , jeżeli $x_i \geq max_i$ regionu.
- Punkt $(y_0, y_1, y_2, ..., y_{K-1})$ leży na lewo od $x_i$ , jeżeli $y_i < x_i$ .
- Punkt $(y_0, y_1, y_2, ..., y_{K-1})$ leży na prawo od $x_i$ , jeżeli $y_i \geq x_i$ .

## Definicja 3

Wynikiem podziału *liścia* względem elementu $x_i$ są dwa nowe *liście*: *lewy liść* i *prawy liść*. Wszystkie pary (*punkt*, *wskaźnik*) *liścia*, dla których *punkt* leży na lewo od $x_i$ zostaną przeniesione do *lewego liścia*. Natomiast w *prawym liściu* zostaną umieszczone wszystkie pary, dla których *punkt* leży na prawo od $x_i$. Na końcu operacji podziału stary *liść* jest usuwany.

# Modyfikacje kdB drzew

## Definicja 4

Wynikiem podziału *węzła wewnętrznego* względem elementu $x_i$ są dwa nowe *węzły*: *lewy* i *prawy*. Nowe węzły są następnie wypełniane *regionami* wywiedzionymi z *regionów* dzielonego *węzła wewnętrznego*, który na końcu operacji jest usuwany. Procedura wypełniania nowych *węzłów wewnętrznych regionami* jest następująca.

Dla każdej pary (*region*, *wskaźnik*) w dzielonym węźle wewnętrznym:

S1. Jeżeli *region* leży na lewo od $x_i$, wstaw parę (*region*, *wskaźnik*) do lewego *węzła wewnętrznego*.

S2. Jeżeli *region* leży na prawo od $x_i$, wstaw parę (*region*, *wskaźnik*) do prawego *węzła wewnętrznego*.

S3. W pozostałych przypadkach:

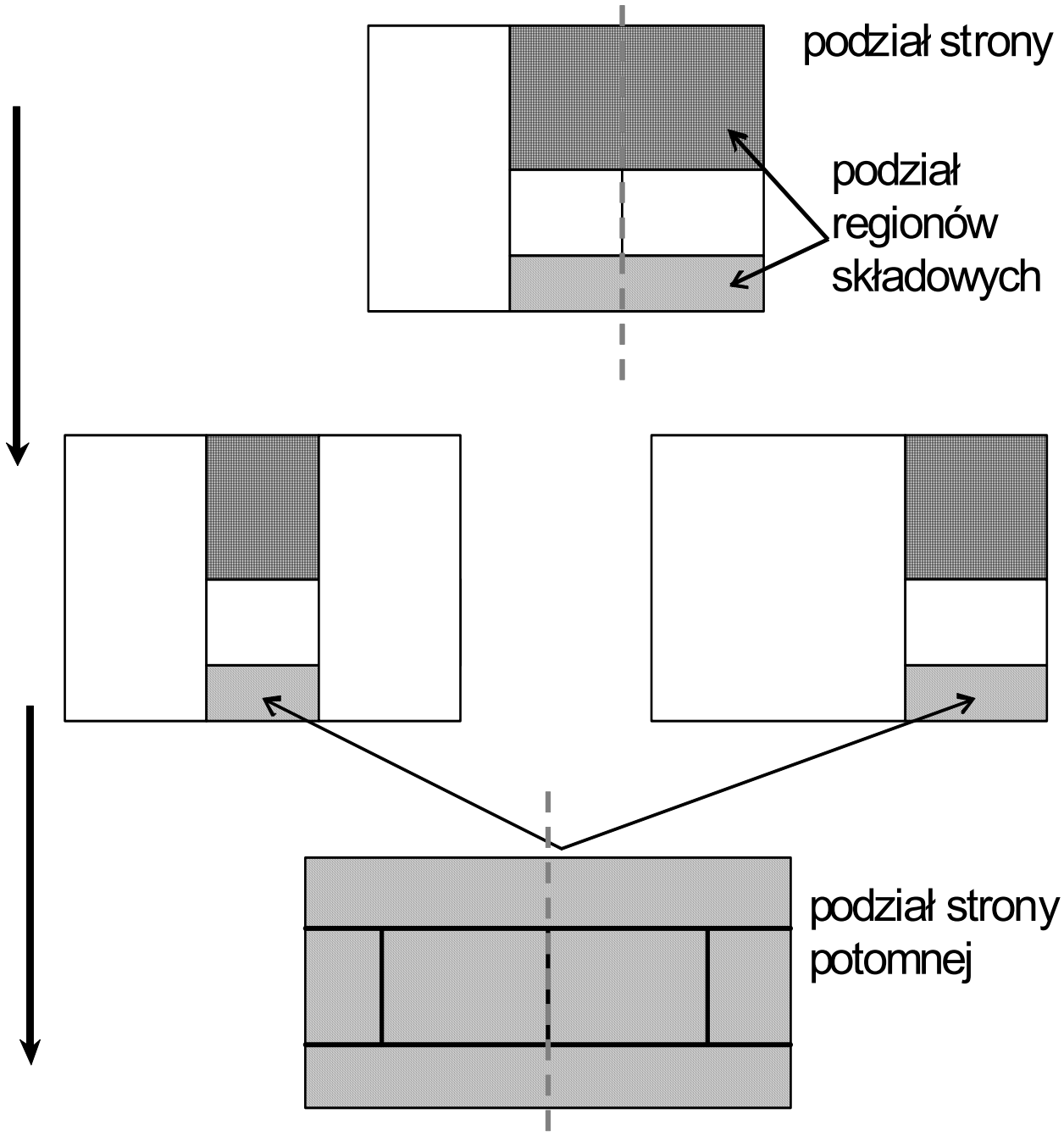
S3.1. Podziel rekurencyjnie stronę adresowaną przez wskaźnik według $x_i$, w wyniku czego powstaną dwie nowe strony adresowane przez *lewy wskaźnik* i *prawy wskaźnik*.

S3.2. Podziel *region* według $x_i$, w wyniku czego powstaną *lewy region* i *prawy region*.

S3.3. Dodaj parę (*lewy region*, *lewy wskaźnik*) do *lewej strony*, a parę (*prawy region*, *prawy wskaźnik*) do *prawej strony*.

# Modyfikacje kdB drzew

**Ilustracja operacji podziału strony:**



# Modyfikacje kdB drzew

Procedura wstawiania rekordu indeksu (*punkt*, *wskaźnik*) jest następująca:

I1. Wykonaj operację *zapytania punktowego* dla punktu: *punkt*, w wyniku której zostanie odnaleziony *liść*. Podstaw adres znalezionego *liścia* do zmiennej *strona*.

I2. Wstaw parę (*punkt*, *wskaźnik*) do znalezionego liścia. Jeżeli w liściu nie wystąpi przepełnienie, zakończ procedurę wstawiania.

I3. Jeżeli wystąpiło przepełnienie, wybierz *dziedzinę*$_i$ i element tej dziedziny $x_i$, takie żeby żadna z dwóch nowych stron, które powstaną w wyniku podziału *strony* względem $x_i$ nie była przepełniona. Podziel stronę względem $x_i$, w wyniku czego powstaną dwie nowe strony: *lewa strona* i *prawa strona*, adresowane przez *lewy wskaźnik* i *prawy wskaźnik*.

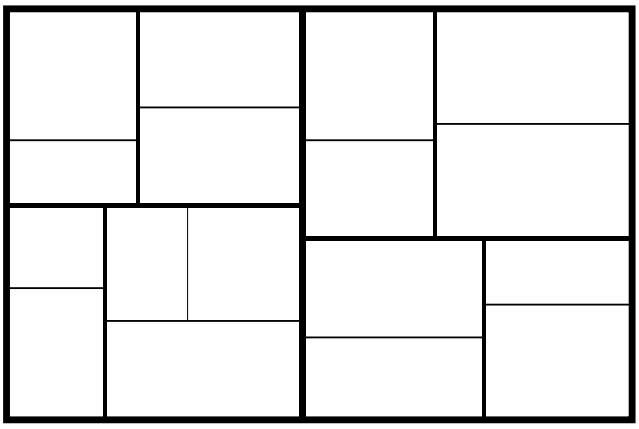
I4. Jeżeli *strona* była korzeniem przejdź do kroku (I5). Jeżeli nie, pod zmienną *strona* podstaw adres przodka *strony* (została ona odnaleziona podczas wykonywania kroku (I1). Zamiast pary (*region*, *strona*) wstaw na stronie dwie pary (*lewy region*, *lewy wskaźnik*) oraz (*prawy region*, *prawy wskaźnik*), gdzie *lewy region* i *prawy region* powstały w wyniku podziału regionu *region* względem $x_i$. Jeżeli w wyniku tego wstawienia wystąpi przepełnienie wróć do kroku (I3). W przeciwnym przypadku zakończ działanie algorytmu.

I5. Utwórz nowy korzeń zawierający pary: ((*dziedzina*$_0$ x ... x $[-\infty, x_i)$ x ... x $I_{K-1}$), *lewy wskaźnik*) i ((*dziedzina*$_0$ x ... x $[x_i, \infty)$ x ... x $I_{K-1}$), *prawy wskaźnik*)

# Strategie wyboru dziedziny podziału regionów

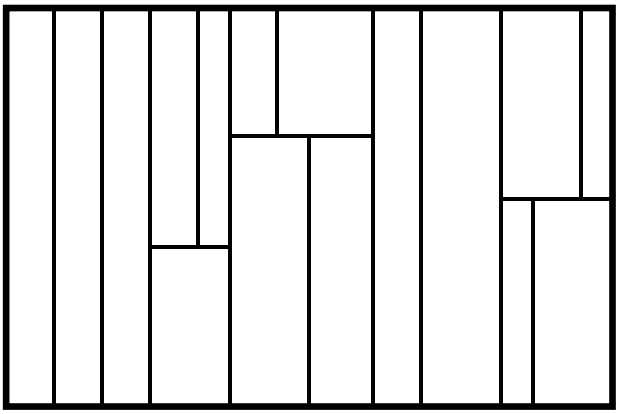
## Metoda cykliczna

W każdym węźle jest składowana dodatkowa zmienna: *dziedzina podziału*. W korzeniu drzewa wartość tej zmiennej jest inicjowana na 0. Podczas podziału węzłów zmienna ta jest wykorzystywana do inicjacji zmiennych nowych stron na wartość: (*dziedzina podziału + 1) mod K*.



## Metoda cykliczna z priorytetami

Niektóre dziedziny ze względu na przykład na rozkład zapytań mogą być faworyzowane przez propagowanie na kilku kolejnych poziomach.

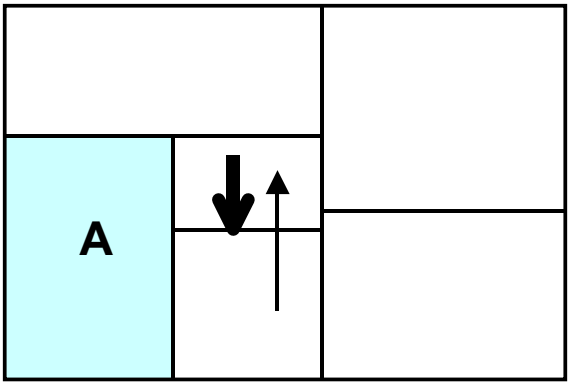


# Usuwanie rekordów

Dwie techniki przeciwdziałające zbyt niskiemu wypełnieniu stron dyskowych zawierających węzły wewnętrznych lub liście drzewa:

- łączenie stron w wyniku łączenia regionów,
- redystrybucja elementów stron (zmiana granic między regionami).

Nie wszystkie strony mogą być poddane temu procesowi.




Dla regionu A nie ma *dziedziny*$_i$, dla której istniałby na tej samej stronie inny region *R* taki, że:

$$minA_i = minR_i \text{ i } maxA_i = maxR_i.$$

# Algorytm reorganizacji słabo wypełnionej strony

R1. *P* jest słabo wypełnioną stroną poddaną procesowi reorganizacji, a *strona* jest stroną rodzicielską *P* zawierającą element (*region*, *wskaźnik na P*).

R2. Znajdź elementy (*region*$_1$, *wskaźnik*$_1$), (*region*$_2$, *wskaźnik*$_2$), ..., na *stronie* takie, że regiony *region*, *region*$_1$, *region*$_2$, ... tworzą razem region. W najgorszym przypadku taki region tworzą wszystkie regiony strony.

R3. Połącz strony adresowane przez *wskaźnik na P*, *wskaźnik*$_1$, *wskaźnik*$_2$, ..., a następnie dziel je tak długo, aż utworzone w kolejnym kroku strony nie są przepełnione.

R4. Zamień elementy (*region*, *wskaźnik*), (*region*$_1$, *wskaźnik*$_1$), (*region*$_2$, *wskaźnik*$_2$), ..., na elementy opisujące strony powstałe w kroku (R3).

# Własności kdB-drzew

**Zalety:**

- wsparcie dla przeszukiwania danych w pamięciach masowych,
- nie nakładanie się regionów.

**Wady:**

- pokrycie pustych regionów,
- stopień wypełnienia stron indeksu,
- operacje podziału mogą być propagowane w dół drzewa.