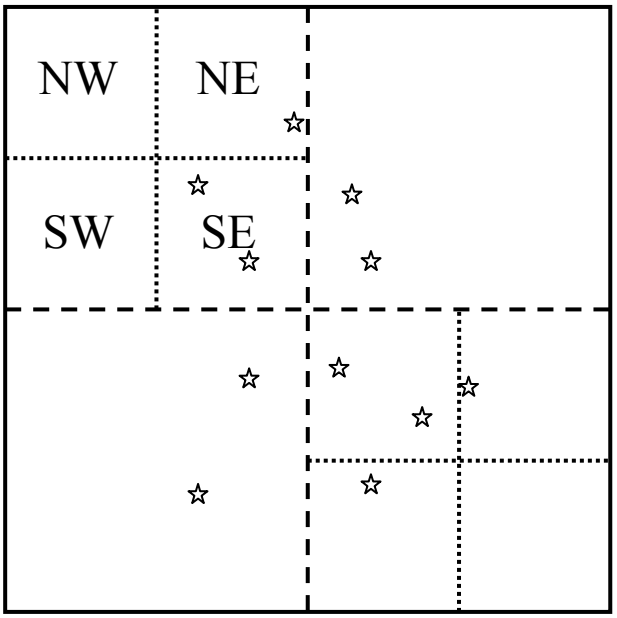
Drzewa ćwiartek (ang. Quadtree) i R-drzewa (ang. R-tree)

Drzewa ćwiartek
Quadtree

Drzewa ćwiartek – rozległa klasa hierarchicznych struktur danych, których wspólną cechą jest rekurencyjna dekompozycja przestrzeni wielowymiarowej na 2^d podregionów.

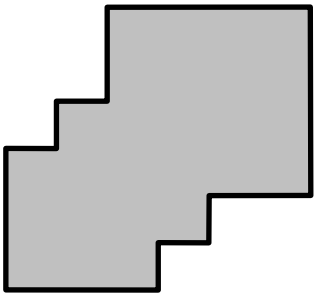


Klasyfikacje drzew ćwiartek ze względu na:

- zastosowanie
- metody dekompozycji
- rozdzielczość

Quadtree

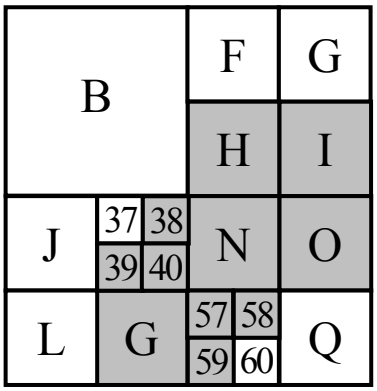
Zastosowanie quadtree do reprezentacji danych przestrzennych. Drzewa jest konstruowane poprzez algorytm `split and merge`



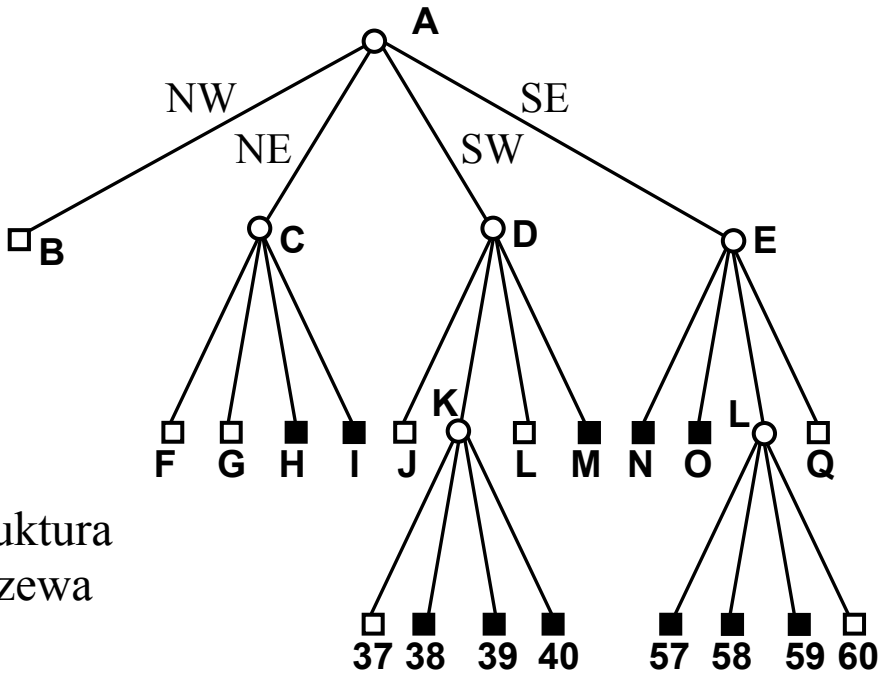
region

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	0	0
0	0	1	1	1	0	0	0

tablica binarna



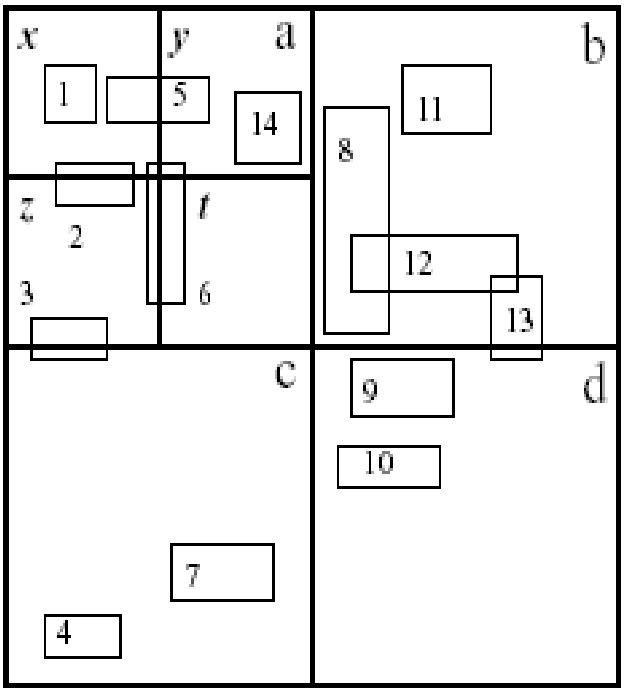
dekompozycja przestrzeni

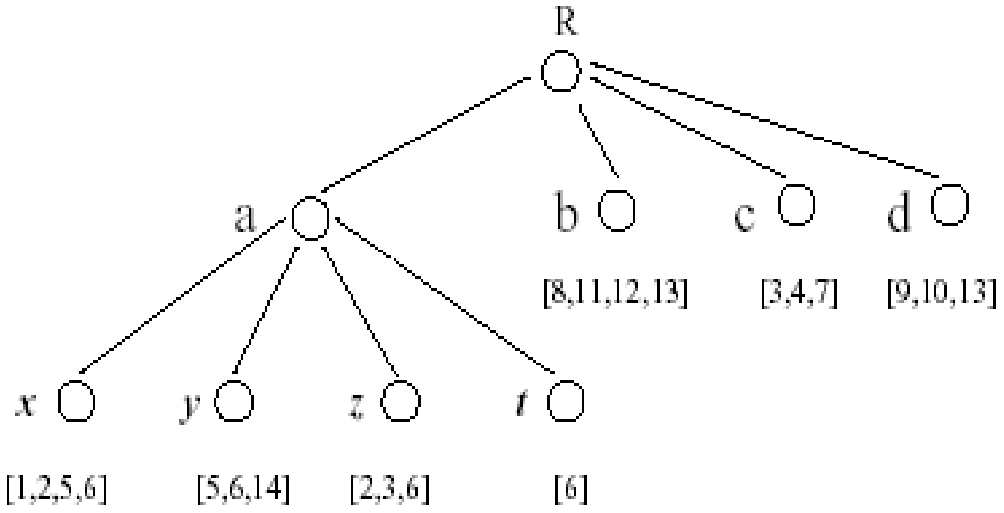


Struktura drzewa

Quadtree

Zastosowanie quadtree do wyszukiwania danych przestrzennych. Zadany poziom dopasowywania.





R-drzewa

R-drzewa są dynamicznymi strukturami danych służącymi do wyszukiwania obiektów wielowymiarowych (**niepunktowych!!!**) w przestrzeni wielowymiarowej. W R-drzewach obiekty wielowymiarowe są aproksymowane za pomocą minimalnych regionów pokrywających (**MBR** – minimum bounding rectangle).

Realizowane funkcje:

- **Zapytania punktowe**: Znajdź identyfikatory obiektów przestrzennych, których MBR zawiera punkt P
- **Zapytania regionowe**: Znajdź identyfikatory obiektów przestrzennych, których MBR ma część wspólną z regionem R
- **Najbliższy sąsiad**: Znajdź identyfikatory obiektów przestrzennych, które są najbliżej punktu P

Struktura węzłów R-drzewa

- struktura węzłów pośrednich:

  $((MBR_1, p_1), (MBR_2, p_2), \ldots , (MBR_n, p_n)),$

  MBR jest specyfikacją regionu o k wymiarach postaci: $[x_{1,min} : x_{1,max}] X \ldots X [x_{k,min} : x_{k,max}]$

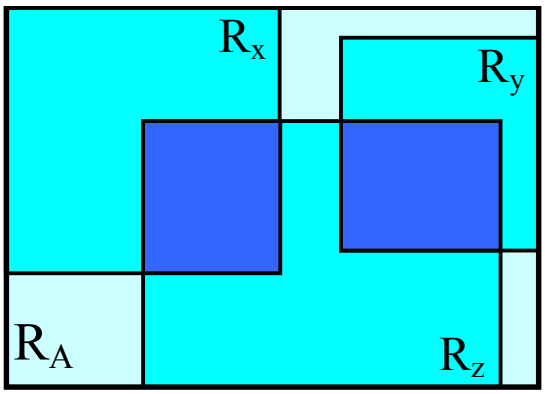
  $n \in (m, M)$, M jest maksymalnym, a m jest minimalnym wypełnieniem węzłów i $m = c * \lfloor M/2 \rfloor$, gzie c jest pewną stałą; z wyjątkiem korzenia, dla którego $m = 2$;
- struktura liści:

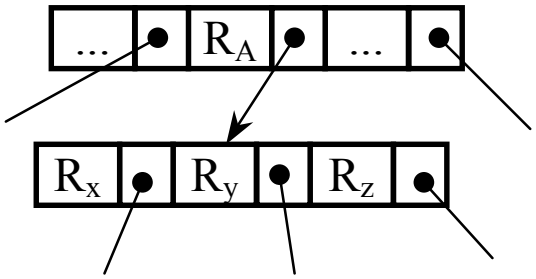
  $((MBR_1, oid_1), (MBR_2, oid_2), \ldots , (MBR_n, p_n)),$

  gdzie *oid* jest wskaźnikiem na indeksowany obiekt.

Charakterystyka R-drzewa

1. Wszystkie liście znajdują się na tym samym poziomie drzewa – jest to drzewo zrównoważone.
2. Regiony znajdujące się w liściach są najmniejszymi z regionów obejmujących przestrzennie indeksowane obiekty.
3. Regiony znajdujące się w węzłach wewnętrznych są najmniejszymi z regionów obejmujących przestrzennie wszystkie regiony węzłów potomnych.
4. Regiony znajdujące się w tym samym węźle R-drzewa mogą siępokrywać.



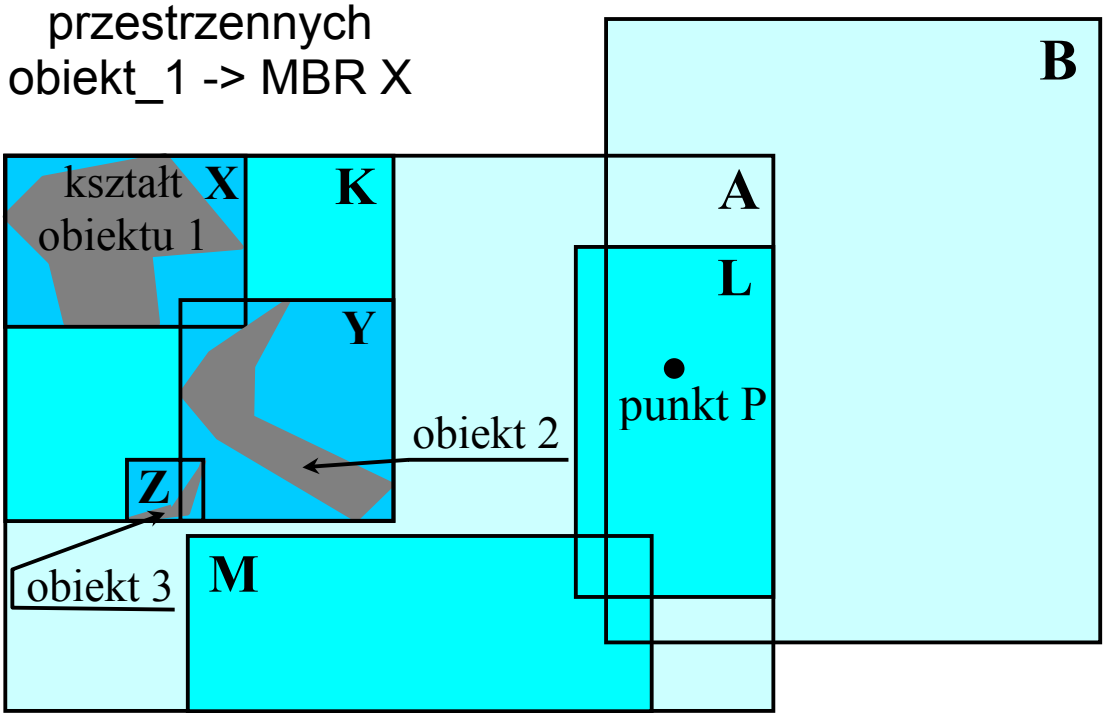


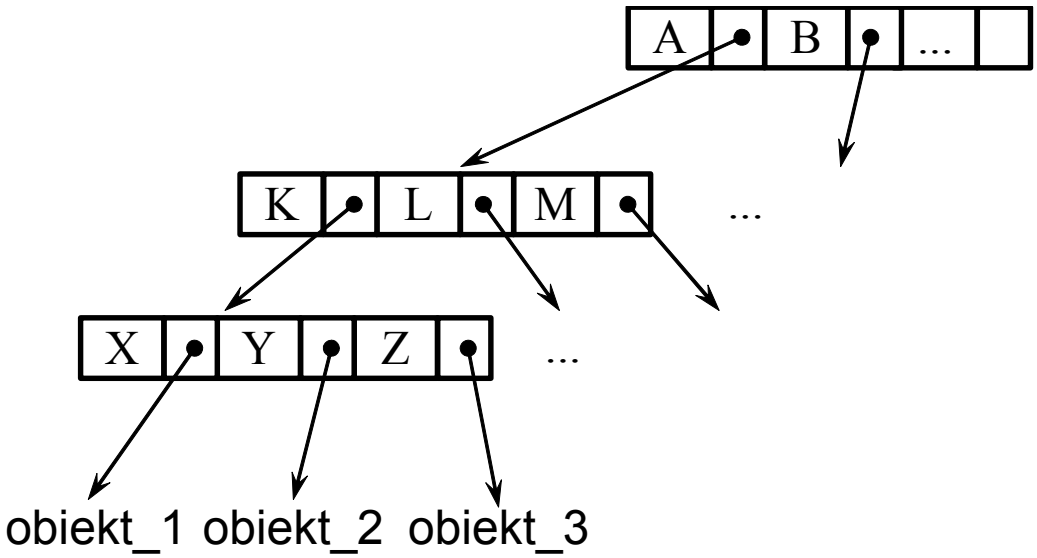
5. Suma wszystkich regionów znajdujących się w danym węźle nie musi tworzyć regionu i w konsekwencji nie musi być równa zawierającemu je regionowi w węźle rodzicielskim.
6. Minimalna wysokość drzewa wynosi $\lceil \log_m N \rceil - 1$ gdzie N jest liczbą indeksowanych obiektów przestrzennych

Dwie ostatnie własności są konsekwencją faktu, że podział przepełnionych węzłów nie polega na podziale regionu, lecz na pogrupowaniu regionów składowych i wyznaczeniu dla nich MBR.

Przykład

Aproksymacja obiektów przestrzennych
obiekt_1 -> MBR X





Wyszukiwanie danych za pomocą R-drzew

Procedura *Search* (T, R_s)

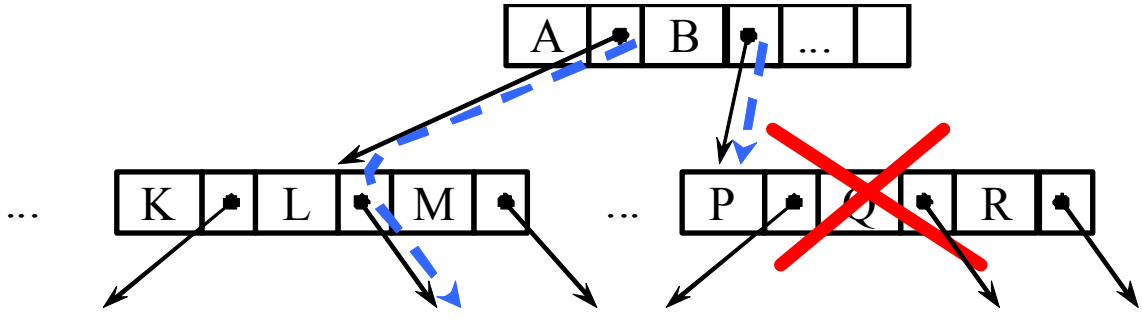
(gdzie **T** jest korzeniem drzewa, a $\boldsymbol{R_s}$ jest regionem zapytania)

(S1) Jeżeli T nie jest liściem, sprawdź dla każdej pary (R_i, P_i), czy region R_i pokrywa się częściowo z regionem R_s. Jeżeli tak, to dla takiego regionu wywołaj rekurencyjnie funkcję *Search* (P_i, R_s). Jeżeli region R_s nie ma części wspólnej z żadnym z regionów zakończ poszukiwanie wzdłuż danej ścieżki.

(S2) Jeżeli T jest liściem, sprawdź dla każdej pary (R_i, oid_i), czy region R_i pokrywa się częściowo z regionem R_s. Jeżeli tak, to umieść oid_i w zbiorze wynikowy procedury.

Efektywność operacji punktowych zapytań wykonywanych za pomocą R-drzew będzie obniżona w przypadku, gdy poszukiwany region znajdują się w obszarze należącym do kilku regionów tego samego poziomu szukanie może przebiegać wzdłuż wielu równoległych ścieżek. Z drugiej strony ścieżki wyszukiwania mogą zanikać przed osiągnięciem poziomu liści drzewa.

Przykład: znajdź obiekt zawierający punkt P.



Utrzymywanie R-drzew

Procedura *Insert (r*)

Brak jednoznaczności wyboru ścieżki wstawiania

Wstaw do R-drzewa rekord indeksu *r*(*R*, *oid*).

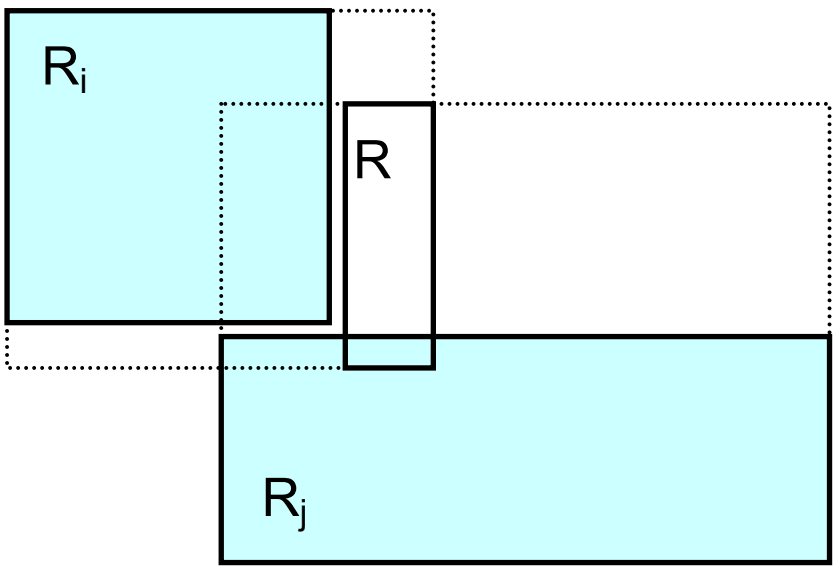
(I1) Znajdź właściwa pozycję dla nowego rekordu indeksu:

(I1.1) Pod *L* podstaw identyfikator strony dyskowej zawierającej korzeń drzewa.

(I1.2) Jeżeli *L* jest liściem, to przejdź do kroku (I2).

(I1.3) Jeżeli *L* nie jest liściem, przeszukaj go w celu znalezienia pozycji (R_i, P_i), której region R_i wymaga najmniejszego powiększenia dla całkowitego pokrycia regionu *R*. Jeżeli wynik poszukiwania nie jest jednoznaczny, wybierz najmniejszy region R_i.

(I1.4) Pod *L* podstaw P_i i wróć do kroku (I1.2).



Procedura *Insert (r)* - cd

(I2) Dodaj rekord (*R*, *oid*) do *L*. Jeżeli w *L* jest wystarczająca ilość wolnego miejsca, przejdź do kroku (I3). Jeżeli nie, **podziel *L* za pomocą procedury *SplitNode*, która utworzy nowy liść *L'* i rozmieści rekordy liścia *L* i wstawiany rekord *r* w liściach *L* i *L'*.**

(I3) Zmodyfikuj regiony R-drzewa:

(I3.1) Jeżeli *L* jest korzeniem i *L'* jest puste zakończ procedurę.

(I3.2) Jeżeli nie, pod *P* podstaw identyfikator węzła rodzicielskiego dla węzła *L*. Jeżeli *L* jest korzeniem utwórz nowy węzeł, który będzie pełnił rolę korzenia i podstaw jego identyfikator pod *P*. Zmodyfikuj region R_L, tak, aby był to MBR dla regionów składowych węzła *L*.

(I3.3) Jeżeli węzeł *L* był dzielony, **dodaj do *P* nową parę ($R_{L'}$, *L'*), gdzie $R_{L'}$ jest MBR regionów składowych węzła *L'*.** Jeżeli w *P* jest wystarczająca ilość wolnego miejsca, przejdź do kroku (I3.4). Jeżeli nie, podziel węzeł *P* za pomocą procedury *SplitNode*, która utworzy nowy liść *P'* i rozmieści równomiernie pozycje węzła *P* i parę ($R_{L'}$, *L'*) w węzłach *P* i *P'*.

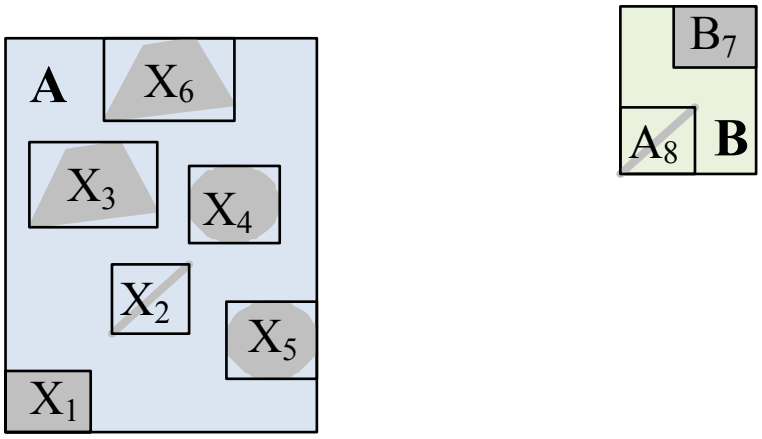
(I3.4) Pod *L* podstaw *P* i jeżeli *P* wymagało podziału pod, pod *L'* podstaw *P'*. Wróć do kroku (I3.1).

Procedura podziału węzła

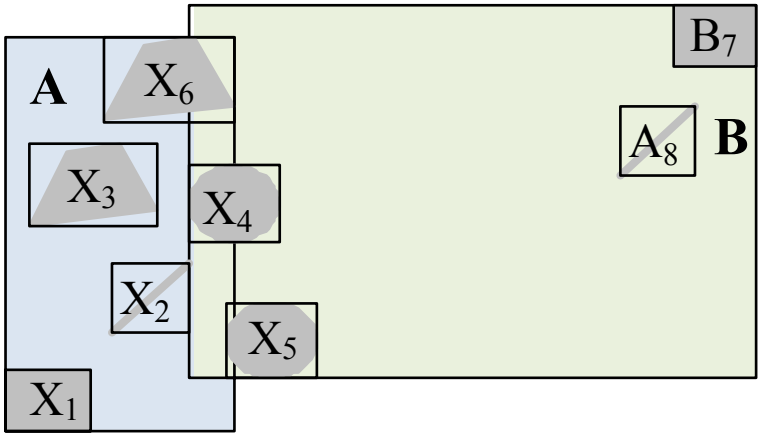
Kryterium jakości podziału

- Równomierne wypełnienie węzłów
- Minimalizacja wspólnej części MBR
- Minimalizacja sumy objętości MBR
- Regularne kształty MBR

przypadek (1)



przypadek (2)



W R-drzewach jako kryterium podziału węzłów przyjęto minimalizację sumarycznego pokrycia w połączeniu z optymalizacją równomiernego wypełnienia węzłów

Algorytm `SplitNode`

Złożoność algorytmu, gwarantującego równomierny podział zbioru regionów składowych należących do dzielonego węzła na dwa podzbiory przy równoczesnej minimalizacji sumarycznego wolumenu pokrywających te zbiory MBR, jest ekspotencjalna.

Guttman zaproponował wydajny algorytm o rzędzie złożoności $O(n^2)$.

Algorytm `QuadraticSplitNode`

// podziel M+1 regionów na dwie grupy

QS1 Wywołaj **`PickSeeds`** dla wyboru pierwszych elementów grup

QS2 `Repeat`

`DistributeEntry`

`until`

wszystkie regiony są rozproszone lub jedna z grup zawiera M-m+1 elementów

QS3 Pozostałe elementy dodaj do drugiej grupy

Eksperymentalnie ustalono, że optymalna wartość `m` wynosi 0,4*M.

Algorytm `SplitNode`

Algorytm `PickSeeds`

// Znajdź dwa najbardziej oddalone regiony

PS1 Dla każdej pary regionów składowych (R_i, R_j) wyznacz region R, który jest ich MBR;

Oblicz wolumen v = vol(R) - vol(R_i) - vol(R_j);

PS2 Wybierz parę z największym v;

Algorytm `DistibuteEntry`

DE1 Wywołaj **`PickNext`** dla wyboru następnego regionu składowego;

DE2 Dodaj go do grupy, której MBR będzie wymagał mniejszego powiększenia; jeżeli powiększenie będzie równe to do grupy o mniejszym MBR; dla równych MBR do mniej licznej grupy;

Algorytm `PickNext`

PN1 Dla każdego nieprzydzielonego jeszcze regionu składowego R_i, oblicz wolumeny v_1 i v_2, o które trzeba by powiększyć MBR pierwszej i drugiej grupy w wyniku dodania tego regionu;

PN2 Wybierz region o największej różnicy $|v_1 - v_2|$;

Własności R-drzew

- Indeksowanie dowolnych typów danych przestrzennych w przestrzeni wielowymiarowej
- Aproksymacja indeksowanych danych
- W wypadku przepełnienia węzłów indeksu następuje podział zbioru regionów, nie przestrzeni
- Unikanie pokrywania pustej przestrzeni
- Wysoki stopień wypełnienia węzłów indeksu (> 50%)
- Proste algorytmy utrzymania
- Zależność efektywności wyszukiwania od rozkładu indeksowanych danych
- Zależność rzędu drzewa od liczby wymiarów
- Silna zależność efektywności wyszukiwania od liczby wymiarów

Modyfikacje R-drzew

Dla małej liczby wymiarów:

- R^+-drzewa
- R^*-drzewa
- R-drzewa Hilberta

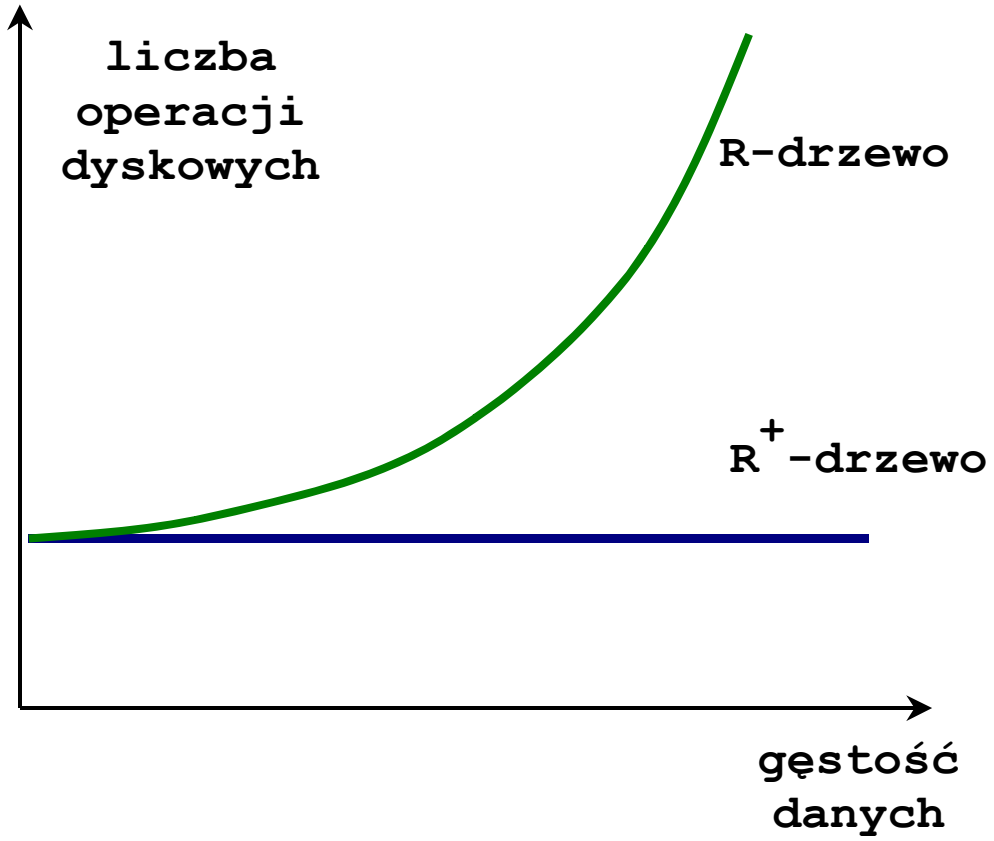
Dla dużej liczby wymiarów:

- TV-drzewa (Telescope-Vectors)
- X-drzewa (eXtended nodes)
- SS-drzewa (Similarity Search)
- SR-drzewa (Similarity search R-tree)

R^+-drzewa

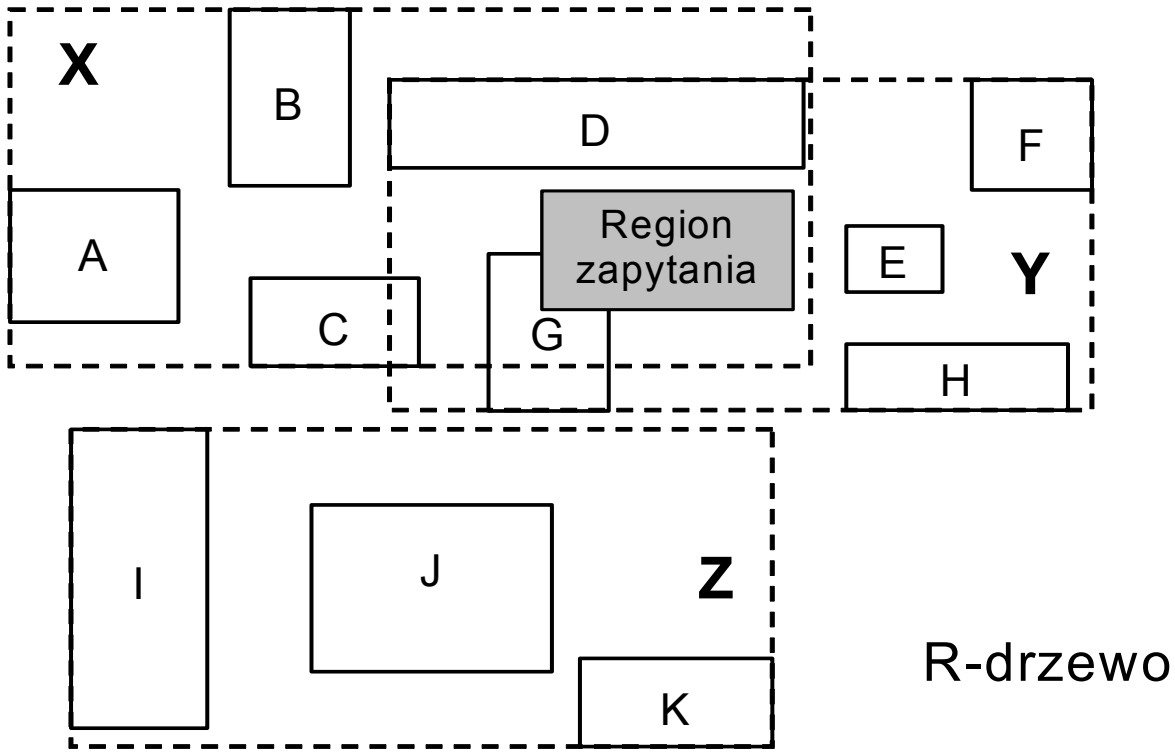
Cel:

- Zwiększenie efektywności operacji wyszukiwania w wyniku eliminacji wzajemnego pokrywania się regionów tego samego poziomu indeksu



Struktura R^+- drzew

Trudne regiony zapytań

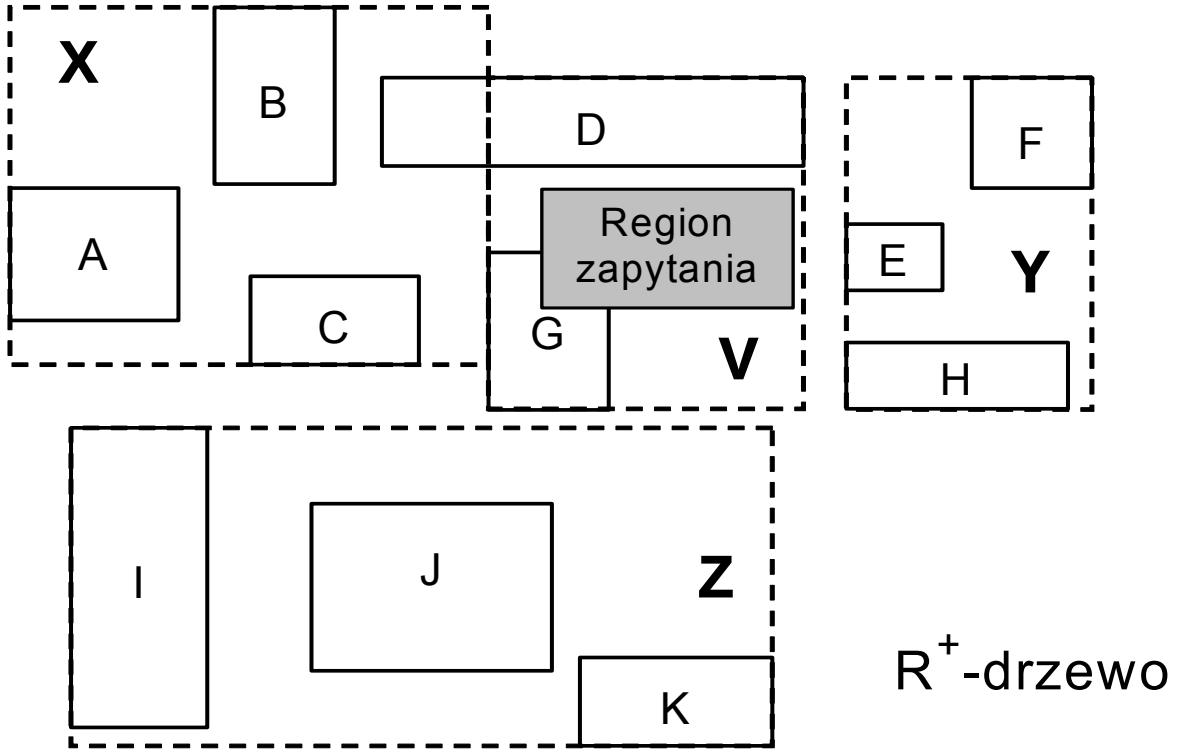


R-drzewo

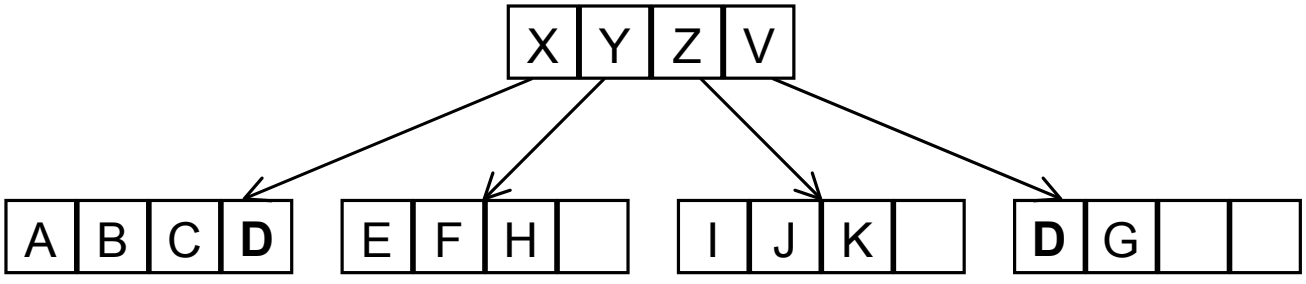
Struktura R^+- drzew

Nowe cechy struktury:

- regiony każdego poziomu R^+-drzewa są rozłączne
- regiony pokrywające obiekty mogą należeć do kilku regionów wyższego poziomu indeksu

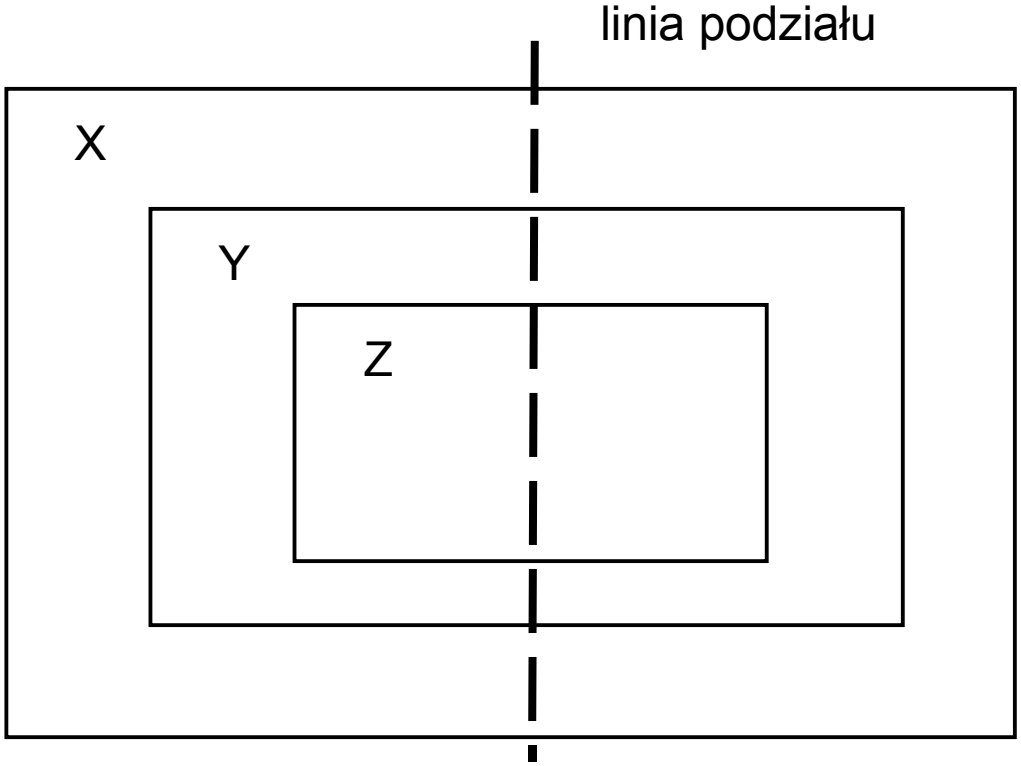


R^+-drzewo



Operacje na R^+- drzewach

- Algorytm zapytania analogiczny jak dla R-drzew; charakteryzuje się jednak większą wydajnością - punktowe zapytania są wykonywane pojedynczą ścieżką nawigacji
- Algorytm wstawiania może być wielościeżkowy - wstawiane obiekty mogą być dodane do więcej niż jednego liścia
- Podział węzłów w wypadku ich przepełnienia może pociągnąć za sobą podział węzłów niższych poziomów – analogicznie jak w kdB-drzewach
- Stopień wypełnienia węzłów jest niższy niż w R-drzewach
- Okresowa reorganizacja struktury



R^*- drzewa

Modyfikacja algorytmów utrzymania R-drzewa przy niezmienionej strukturze dla optymalizacji podstawowych kryteriów efektywności drzewa:

- Minimalizacja objętości regionów
- Minimalizacja pokrywania się regionów
- Minimalizacja powierzchni regionów
- Maksymalizacja wypełnienia węzłów drzewa

Występują złożone zależności między powyższymi kryteriami.

Eksperymentalnie wyznaczone modyfikacje algorytmów utrzymania drzewa.

1. Modyfikacja algorytmu wyboru ścieżki wstawiania; minimalizacja wzajemnego pokrywania się regionów na poziomie liści i minimalizacja sumarycznego pokrycia dla węzłów pośrednich.
2. Modyfikacja algorytmu podziału – optymalizowane są równolegle: sumaryczne pokrycie, wzajemne pokrycie i obszar powierzchni dla różnych proporcji wypełnienia węzłów
3. Wymuszone powtórne wstawianie elementów do węzła – dane są porządkowane według odległości środka regionu składowego od regionu pokrywającego

Wybór ścieżki wstawiania

Niech R_1, …,R_n będą regionami składowanymi w węźle. Wtedy sumaryczne pokrycie danego regionu R_k z pozostałymi regionami przechowywanymi w węźle jest równe:

$$overlap(R_k) = \sum_{i=1,\, i\neq k}^{n} vol(R_k \cap R_i)$$

Algorytm `ChooseSubtree(R)`

CS1 Zmiennej `N` przypisz korzeń drzewa

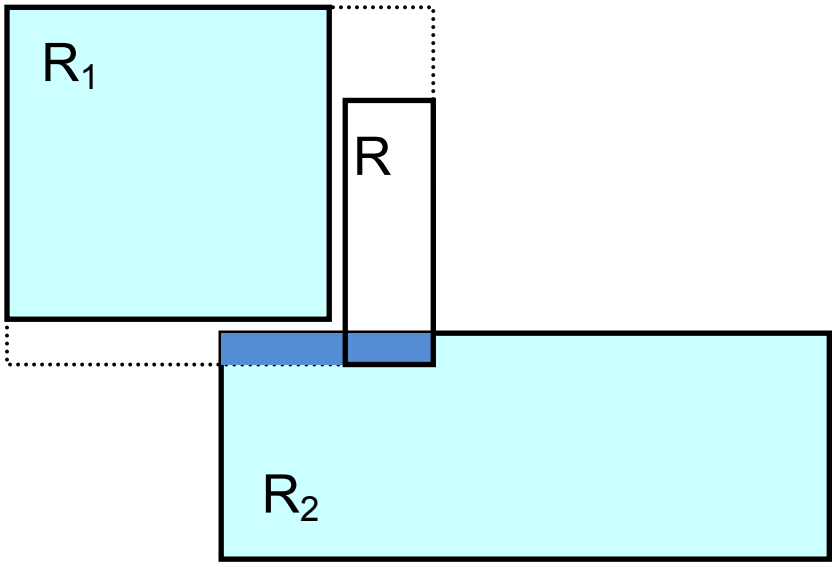
CS2 `If N` jest liściem
`return N`
`elseif` wskaźniki w `N` wskazują na liście
`then` // minimalizacja `overlap`
Dla każdego regionu składowego R_i wyznacz $\Delta overlap(R_i)=overlap(R'_i)-overlap(R_i)$ gdzie R'_i jest regionem R_i zmodyfikowanym tak by obejmował również region R; wybierz region, dla którego $\Delta overlap$ jest najmniejsze; w wypadku niejednoznacznego wyboru wybierz rekord, którego region zostanie najmniej powiększony; w dalszej kolejności region najmniejszy; // $O(n^2)$
`else` // minimalizacja obszaru regionów
wybierz region, który wymaga najmniejszego powiększenia dla całkowitego pokrycia regionu *R*; w wypadku niejednoznacznego wyboru, wybierz region najmniejszy; // O(n)
`endif`

CS3 Zmiennej `N` przypisz węzeł wskazywany przez wskaźnik powiązany z wybranym regionem

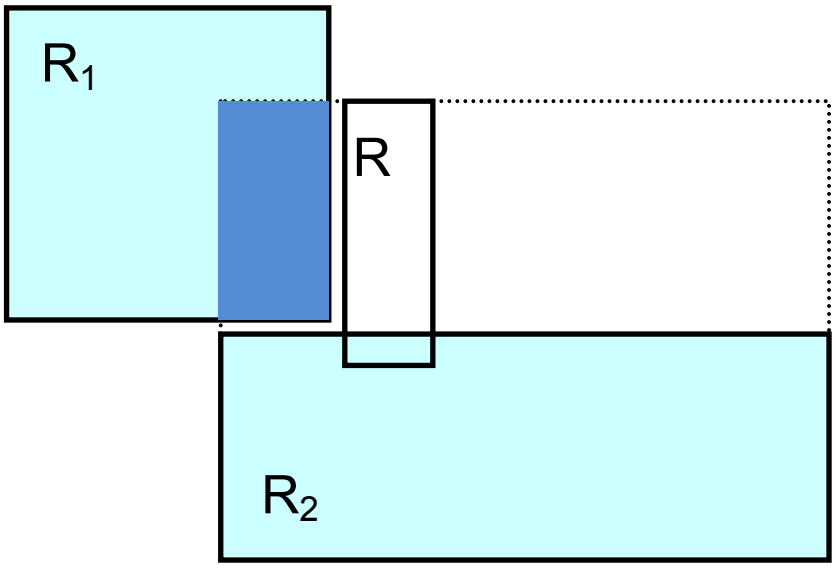
Wybór ścieżki wstawiania

Overlap(R_1) = vol($R_1 \cap R_2$) = 0

Overlap(R'_1) = vol($R'_1 \cap R_2$)



Overlap(R'_2) = vol($R_1 \cap R'_2$)



Algorytm podziału węzła

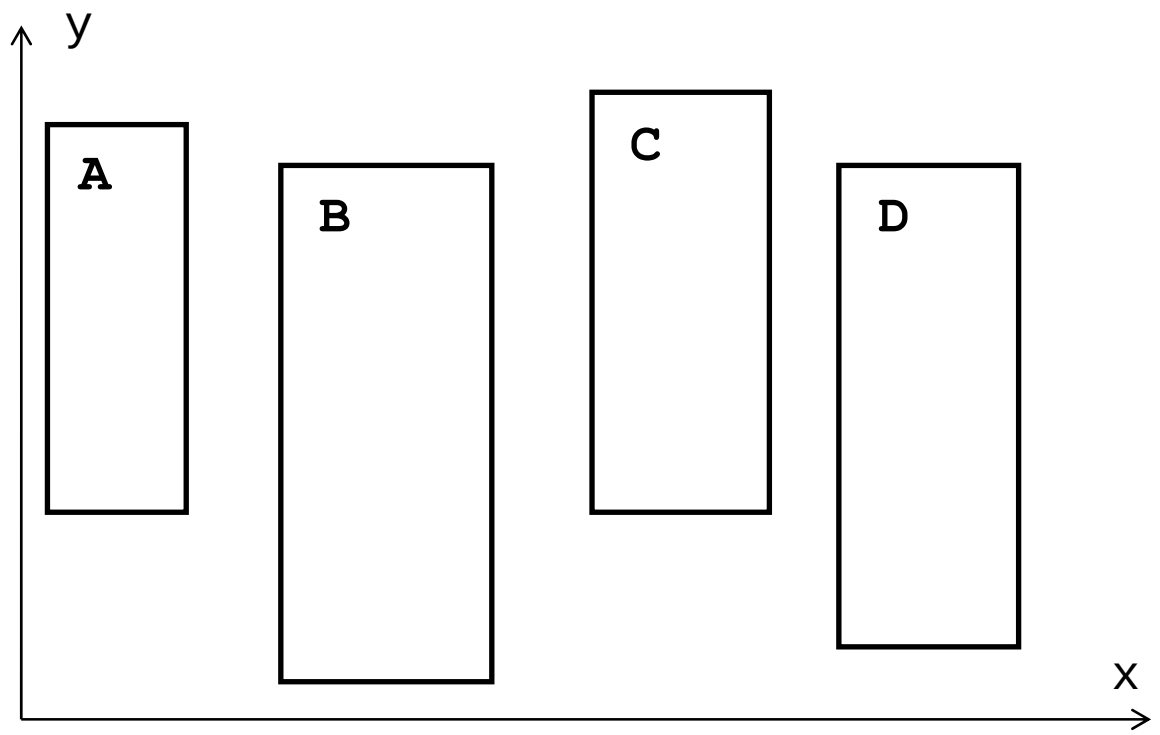
Algorytm `SplitNode`

SN1 Wywołaj **`ChooseSplitAxis`** dla wyznaczenia osi względem, której zostanie wykonany podział;

SN2 Wywołaj **`ChooseSplitIndex`** dla określenia najlepszego podziału regionów na dwie grupy względem wybranej osi;

SN3 Rozprosz regiony między dwie wybrane grupy;

`ChooseSplitAxis` - heurystyczne ograniczenie dziedziny rozwiązania



Poszczególne wymiary w różny sposób różnicują zbiór MBR. Algorytm **`SplitNode`** szuka najbardziej różnicującego wymiaru.

Ograniczenie dziedziny wyszukiwania

Dla danej osi porządkujemy regiony składowe węzła względem kolejno najmniejszych i największych wartości granic regionów.

Dla każdego z tych dwóch porządków wykonujemy M-2m+2[1] (gdzie $2 \le m \le \lfloor M/2 \rfloor$) podziałów M+1 regionów na dwie grupy, gdzie m jest minimalną liczbą rekordów w węźle indeksu. K-ty podział ($k \in <1; M-2m+2>$) jest zdefiniowany następująco:

> Pierwsza grupa obejmuje pierwszych (m-1)+k regionów, a druga - pozostałe regiony.

Badania eksperymentalne wykazały, że optymalna wartość m wynosi 40%.

Dziedzina rozwiązania zostanie ograniczona do wyznaczonych w tym kroku podziałów.

W kolejnych krokach algorytmu zostaną zastosowane trzy miary jakości grupowania regionów:

- obszar:

  vol(MBR(grupy I)) + vol(MBR(grupy II))

- powierzchnia:

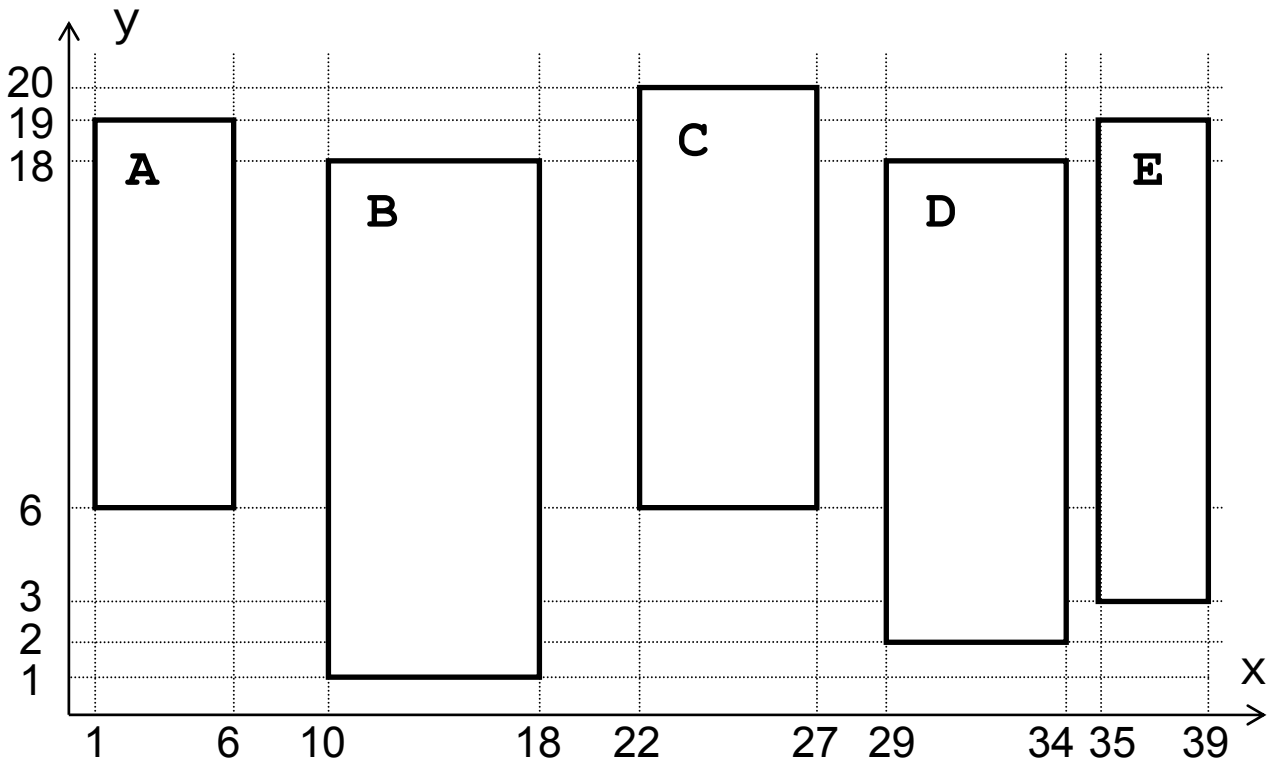
  margin(MBR(grupy I)) + margin(MBR(grupy II))

- wzajemne pokrycie:

  $\mathrm{vol}(\mathrm{MBR}(\text{grupy I})) \cap \mathrm{vol}(\mathrm{MBR}(\text{grupy II}))$

[1] Dla M parzystych. Dla M nieparzystych: (M-2m+1) podziałów.

Przykład



Niech: M=4, m=2

Oś X:

$X_{min}(A)=1$, $X_{min}(B)=10$, $X_{min}(C)=22$, $X_{min}(D)=29$, $X_{min}(E)=35$
$X_{max}(A)=6$, $X_{max}(B)=18$, $X_{max}(C)=27$, $X_{max}(D)=34$, $X_{max}(E)=39$

Podział 1 (k=1, **min**, (m-1)+k=2): {A, B}; {C, D, E}
Podział 2 (k=2, **min**, (m-1)+k=3): {A, B, C}; {D, E}
Podział 3 (k=1, **max**, (m-1)+k=2): {A, B}; {C, D, E}
Podział 4 (k=2, **max**, (m-1)+k=3): {A, B, C}; {D, E}

Oś Y:

$Y_{min}(B)=1$, $Y_{min}(D)=2$, $Y_{min}(E)=3$, $Y_{min}(A)=5$, $Y_{min}(C)=5$
$Y_{max}(B)=18$, $Y_{max}(D)=18$, $Y_{max}(E)=19$, $Y_{min}(A)=19$, $Y_{max}(C)=20$

Podział 1 (k=1, **min**, (m-1)+k=2): {B, D}; {E, A, C}
Podział 2 (k=2, **min**, (m-1)+k=3): { B, D, E}; {A, C}
Podział 3 (k=1, **max**, (m-1)+k=2): {B, D}; {E, A, C}
Podział 4 (k=2, **max**, (m-1)+k=3): {B, D, E}; {A, C}

Algorytm `ChooseSplitAxis`

CSA1 `For each` wymiaru x

Posortuj regiony względem dolnej i górnej granicy regionu (x_{min}, x_{max}) w danym wymiarze i określ wszystkie rozkłady regionów na dwie grupy gr_1^i i gr_2^i. Wyznacz sumę powierzchni MBR dla wszystkich rozkładów:

S=$\sum_i$ `margin`(MBR(gr_1^i)) + `margin`(MBR(gr_2^i))

`end`

CSA2 Jako oś podziału wybierz wymiar z najmniejszą wartością S;

Oceniamy sumaryczną jakość podziałów dla poszczególnych osi:

Oś X:

gr 1, 3: `margin`(MBR{A, B}) = 70;
`margin`(MBR {C, D, E}) = 70;

gr 2, 4: `margin`(MBR{A, B, C}) = 90;
`margin`(MBR {D, E}) = 54; **=284**

Oś Y:

gr 1, 3: `margin`(MBR{B, D}) = 82;
`margin`(MBR {E, A, C}) = 112;

gr 2, 4: `margin`(MBR{B, D, E}) = 96;
`margin`(MBR {A, C}) = 80; **=370**

Algorytm `ChooseSplitIndex`

CSI1 Dla wybranego wymiaru wybierz podział z minimalną wartością pokrywania się regionów; w wypadku niejednoznacznego wyboru, wybierz podział z minimalnym obszarem;

1)

MBR{A, B} $\cap$ MBR {C, D, E} = $\varnothing$

MBR{A, B, C} $\cap$ MBR {D, E} = $\varnothing$

2)

`vol`(MBR{A, B}) + `vol`(MBR{C, D, E}) = 306 + 306 = 612

`vol`(MBR{A, B, C}) + `vol`(MBR{D, E}) = 494 + 170 = 664

Wynik:

{A, B} i **{C, D, E}**

Utrzymanie R*-drzewa dla operacji Delete

Węzły, których wypełnienie spada poniżej wartości `m` są usuwane, a składowane w nich rekordy są powtórnie wstawiane do R*-drzewa.

Linearyzacja przestrzeni

Podstawowa różnica między drzewami jedno- i wielowymiarowymi polega na braku globalnego porządku elementów indeksów wielowymiarowych odpowiadającego lokalizacji danych w przestrzeni wielowymiarowej.

Heurystyczne rozwiązanie problemu:

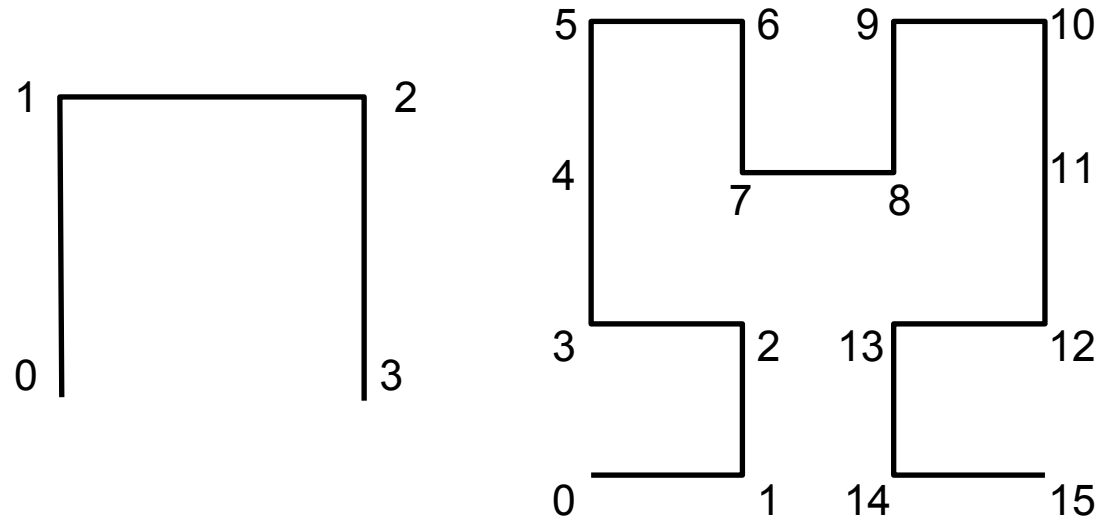
> **Przypisanie elementom wielowymiarowym - jednowymiarowych wartości numerycznych, spełniających następujące wymaganie: jeżeli dwa obiekty znajdują się blisko siebie w przestrzeni wielowymiarowej, to istnieje duże prawdopodobieństwo, że mają bliskie sobie wartości jednowymiarowe. Dane oddalone od siebie w przestrzeni wielowymiarowej muszą mieć przypisane odległe wartości jednowymiarowe.**

Sposobem na przypisanie poszczególnym lokalizacjom wielowymiarowym wartości jednowymiarowych odzwierciedlających charakterystykę wielowymiarową, jest tzw. **linearyzacja przestrzeni**.

Linearyzacja przestrzeni

Przestrzeń jest dzielona na elementarne komórki, którym przypisane są unikalne etykiety zgodnie z lokalizacją w globalnym porządku określonym przez *krzywą wypełniającą przestrzeń* (ang. space-filling curve). Krzywa taka musi przechodzić przez wszystkie komórki nigdy się nie przecinając.

Przykładowe krzywe: Hilberta, Peano, Z-ordering, itp.

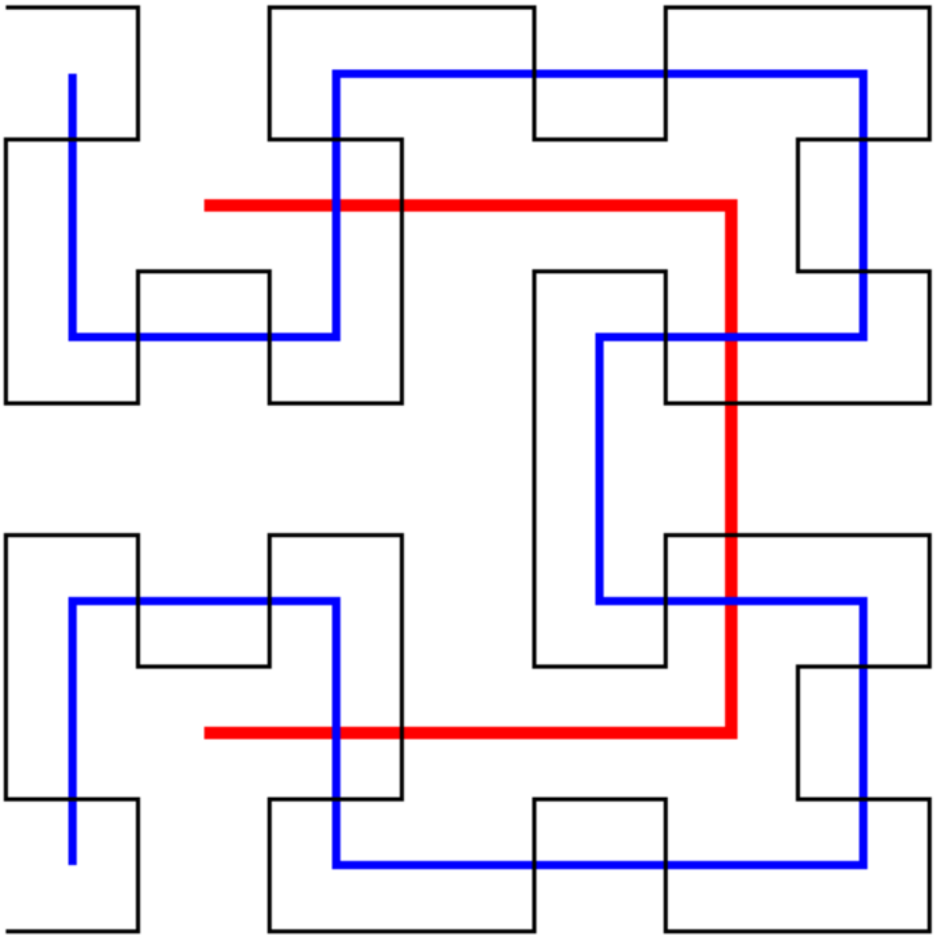


Krzvwe Hilberta rzedu 1 i 2

Generowanie krzywej Hilberta

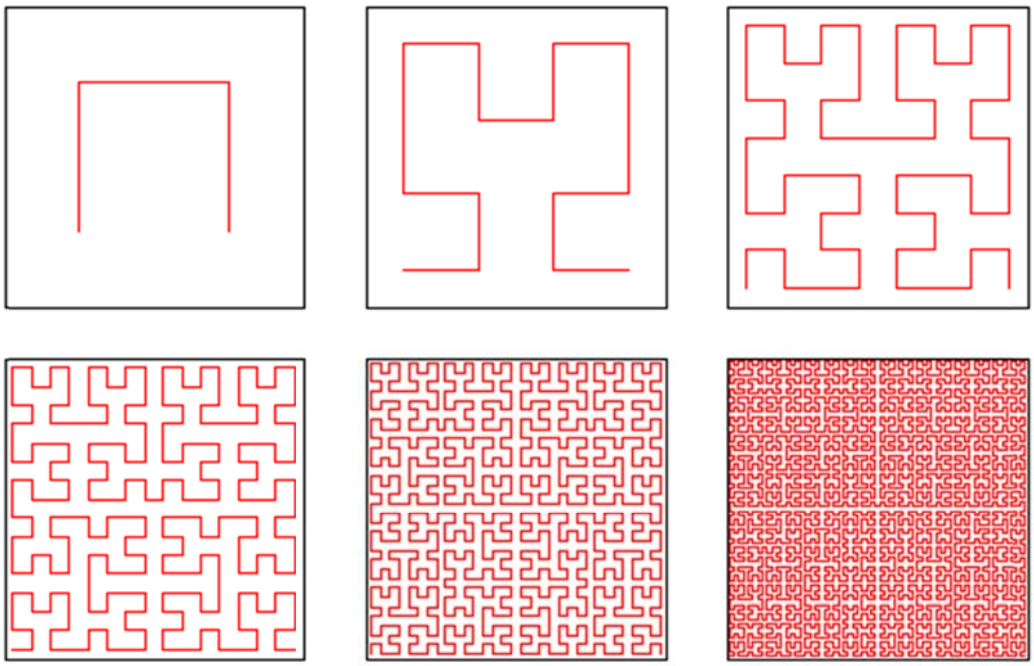
Rząd 2 na podstawie rzędu 3:

Rząd 3 na podstawie rzędu 2:

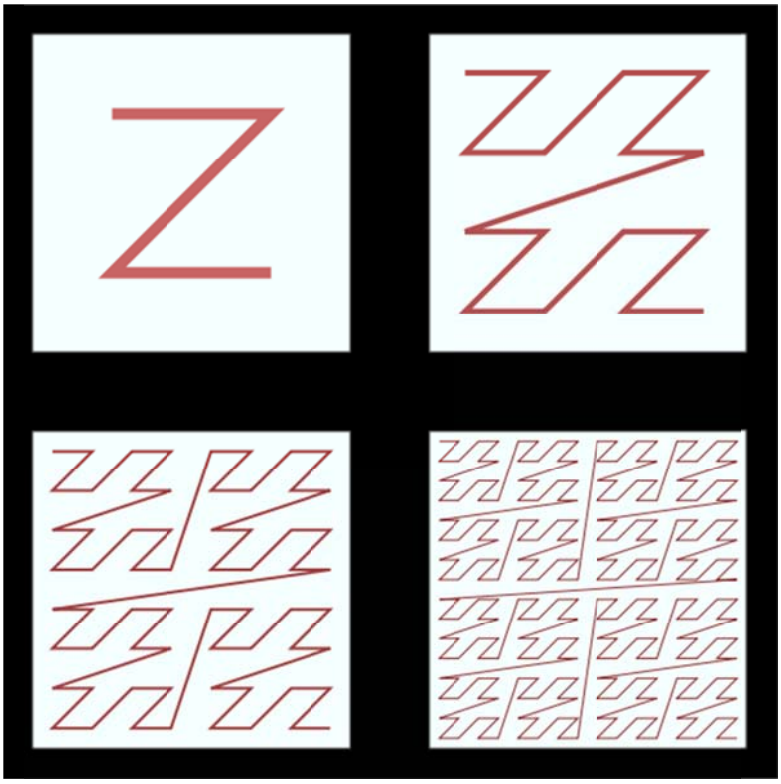


Linearyzacja przestrzeni

Krzywe Hilberta rzędu 1, 2, 3, 4, 5 i 6:



Krzywa Z-ordering rzędu 1, 2, 3 i 4:



Krzywe wypełniające przestrzeń są samopodobne.

R-drzewo Hilberta

R-drzewo Hilberta jest odmianą klasycznego R-drzewa. Dzięki wprowadzeniu miary charakteryzującej się globalnym porządkiem, w R-drzewie Hilberta można zastosować proste algorytmy utrzymania drzewa wzorowane na indeksach jednowymiarowych.

Do generowania etykiet komórek przestrzeni zastosowano krzywą Hilberta. Wartością Hilberta danego MBR jest wartość Hilberta środka regionu.

Regiony składowe przepełnionych węzłów są dzielone na podstawie ich wartości Hilberta.

Struktura indeksu:

- liście zawierają rekordy postaci: (`R`, `oid`)

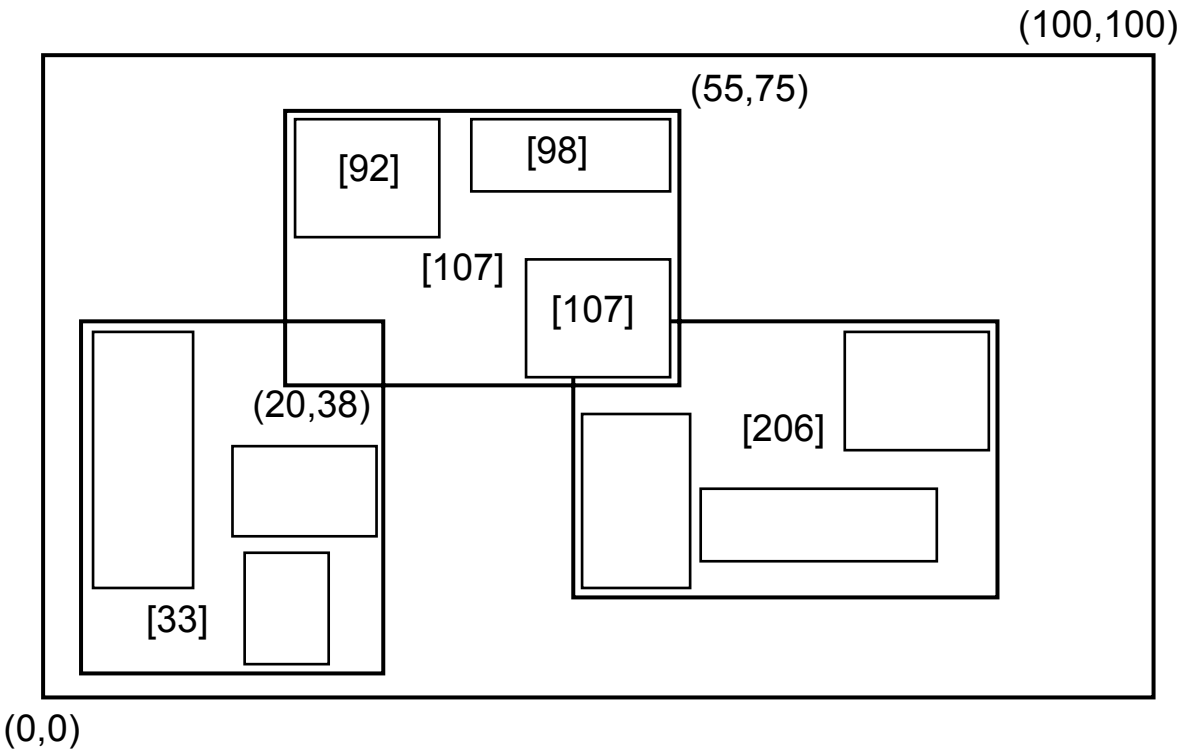
  gdzie: `R` jest regionem MBR obiektu wielowymiarowego, a `oid` wskaźnikiem na blok zawierający ten obiekt;

- węzły pośrednie zawierają rekordy postaci: (`LHV`, `R`, `ptr`)

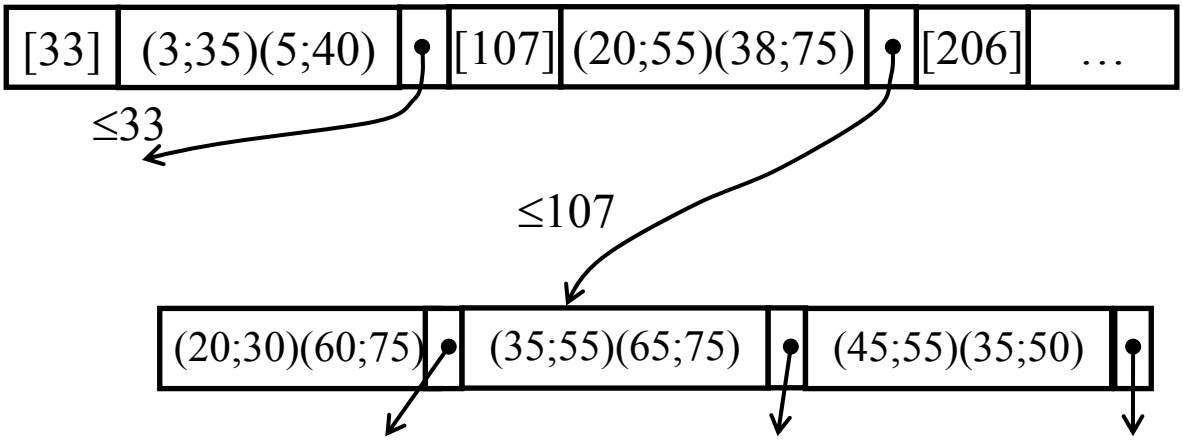
  gdzie: `R` jest regionem MBR regionów zdefiniowanych w węźle potomnym, `ptr` wskaźnikiem na ten węzeł, a `LHV` jest największą wartością Hilberta regionów zawartych w `R`;

Przykład R-drzewa Hilberta

Organizacja przestrzeni za pomocą krzywej Hilberta



Struktura indeksu



Algorytm wstawiania

Algorytm `Insert(node Root, Region R)`

I1 **Znajdź właściwy liść**:
Wyznacz wartość Hilberta h dla R;
`L := ChooseLeaf(R, h);`

I2 **Wstaw R do liścia `L`**:
`if` (wolne miejsce w L)
`then` wstaw R do L zgodnie z porządkiem Hilberta
`else` **`HandleOverflow`**(L, R)

I3 **Propaguj modyfikacje**:
Utwórz zbiór S, który zawiera L, sąsiednie węzły i ewentualnie nowy węzeł;
`AdjustTree`(S);

I4 **Zwiększ wysokość drzewa**:
`if` (podział korzenia);
`then` utwórz nowy korzeń, adresujący dwa uzyskane węzły;

Wybór ścieżki wstawiania

Algorytm `ChooseLeaf(Region R, int h)`

CL1 Zmiennej `N` przypisz korzeń drzewa

CL2 `if` (N jest liściem)
`then return N`

CL3 `if` (N nie jest liściem)
Znajdź rekord (R_i, ptr_i, LHV_i) z najmniejszą wartością LHV większą od h;

CL4 Pod `N` podstaw węzeł wskazywany przez ptr_i;
Wróć do CL2;

Algorytm wyszukiwania węzła do wstawienia nie ma nic wspólnego z algorytmem wyszukiwania danych. Wyszukiwanie danych za pomocą R-drzewa Hilberta jest analogiczny jak w innych R-drzewach!

Podział węzła

Algorytm `HandleOverflow(node N,Region R)`
/* jeżeli wystąpi podział zwróć nowy węzeł */

H1 Utwórz zbiór `E`, który będzie zawierał wszystkie rekordy węzła `N` i jego sąsiadów;

H2 dodaj region `R` do zbioru `E` uwzględniając porządek wartości Hilberta;

H3 `if` co najmniej jeden z sąsiadów `N` nie jest pełny
`then` rozprosz zbiór `E` między przepełnionym węzłem i jego sąsiadami zgodnie z porządkiem Hilberta;

H4 `if` wszyscy sąsiedzi `N` są pełni
`then` utwórz nowy węzeł `N'`;
rozprosz zbiór `E` między wszystkie węzły zgodnie z porządkiem Hilberta;
`return N'`;

Wyrównywanie drzewa

Algorytm `AdjustTree(set S)`

/* S zawiera zmodyfikowany węzeł, jego sąsiadów i ewentualnie nowo utworzony węzeł N'

Algorytm przebiega w górę drzewa modyfikując kształty regionów i wartości LHV węzłów, których regiony obejmują regiony węzłów ze zbioru S */

A1 `if` osiągnięto korzeń indeksu `then` zakończ;

A2 /* propagacja podziału lub modyfikacji węzłów */
NP := `parent`(N);
`if` wystąpił podział węzła N;
`then` wstaw rekord opisujący nowy węzeł N' do NP w miejscu odpowiednim dla jego wartości Hilberta;
`if` wystąpiło przepełnienie węzła NP
`then` **`HandleOverflow`**(NP, N');
`if` wystąpił podział węzła NP
`then` PP nowy węzeł;

A3 /* wyrównywanie MBR i LHV */
P := `parent`(S);
Zmodyfikuj MBR i LHV w węzłach ze zbioru P;

A4 /* przejście o poziom wyżej */
S := P;
`if` wystąpił podział węzła NP
`then` N' := PP;
wróc do kroku A1;

Pakowanie R-drzew

Wszystkie algorytmy utrzymywania R-drzew są niedeterministyczne. Kształty R-drzew są zależne od historii operacji ich modyfikowania. Optymalizacja wyboru ścieżki wstawienia nowego rekordu oraz optymalizacja podziału węzła są realizowane lokalnie - dla ograniczonego zbioru MBR.

Dla części zastosowań struktur wielowymiarowych zbiór danych wielowymiarowych jest statyczny.

Na przykład:

- dane geograficzne;
- dane OLAP przechowywane w tabelach zagregowanych.

Dla takich zastosowań jest sensowne stosowanie algorytmów pakowania indeksu, to jest konstruowania indeksu dla istniejącego zbioru danych. Pakowany indeks powinien się charakteryzować lepszymi parametrami, niż indeks powstały w wyniku dynamicznego utrzymania związanego z historią operacji: `insert`, `delete` i `update`.

Algorytm Pakowania R-drzewa

Algorytm pakowania R-drzewa PACK jest funkcją rekurencyjną, której argumentem jest lista MBR. Funkcja zwraca wskaźnik na korzeń R-drzewa indeksującego dane z listy DLIST. Dany jest rząd R-drzewa – p.

```
PACK(DLIST)
if count(DLIST)<p then
      przydziel nowy węzeł i wstaw do niego wszystkie
      elementy z DLIST;
      zwróć wskaźnik węzła;
else
      uporządkuj elementy DLIST ze względu na ich cha-
      rakterystykę przestrzenną;
      utwórz listę NLIST = ( ); //
// first zwraca i usuwa pierwszy element listy
      while (DLIST niepusta)
            Przydziel nowy węzeł indeksu N;
            Element = first(DLIST);
            Dodaj Element do N;
            for (i=1; i<=p; i++)
            // NN zwraca element listy najbliższy danemu
                  Element= NN(DLIST, Element);
                  Dodaj Element do N;
            end loop;
            NLIST = append(NLIST, N);
            Return (PACK(NLIST));
      end loop;
else if;
end PACK;
```

Wyniki eksperymentu

GUTTMAN'S INSERT						PACK ALGORITHM				
J	C	O	D	N	A	C	O	D	N	A
10	68483	43731	1	4	2 217	39590	0	1	3	1 424
25	74577	124311	2	12	4 800	31230	144	2	9	2 249
50	70718	177809	3	28	7 775	37421	1295	2	16	2 282
75	74561	220949	3	39	9 379	36152	1329	3	26	3 431
100	75234	235078	4	60	12 955	38271	994	3	35	3 645
125	77578	246084	4	73	14 024	36476	1318	3	42	3 658
150	77342	255692	4	86	14 894	40145	2729	3	51	3 784
175	79869	255523	4	103	16 277	36432	2532	3	58	3 820
200	80034	295091	4	117	17 870	33959	1394	3	68	3 873
250	79117	293730	4	142	18 585	40069	1946	3	83	3 867
300	78891	376731	4	167	20 838	38438	1527	4	102	5 397
400	82116	553650	5	233	28 935	37558	965	4	135	5 418
500	85290	698248	5	302	36 132	39820	1688	4	168	5 466
600	85253	749874	5	368	40 799	39542	2106	4	202	5 276
700	86225	852205	5	438	45 924	37016	1252	4	234	5 604
800	87418	1002339	6	507	55 462	38614	1522	4	268	5 730
900	87640	1164809	6	573	63 595	38808	1512	4	302	6 071

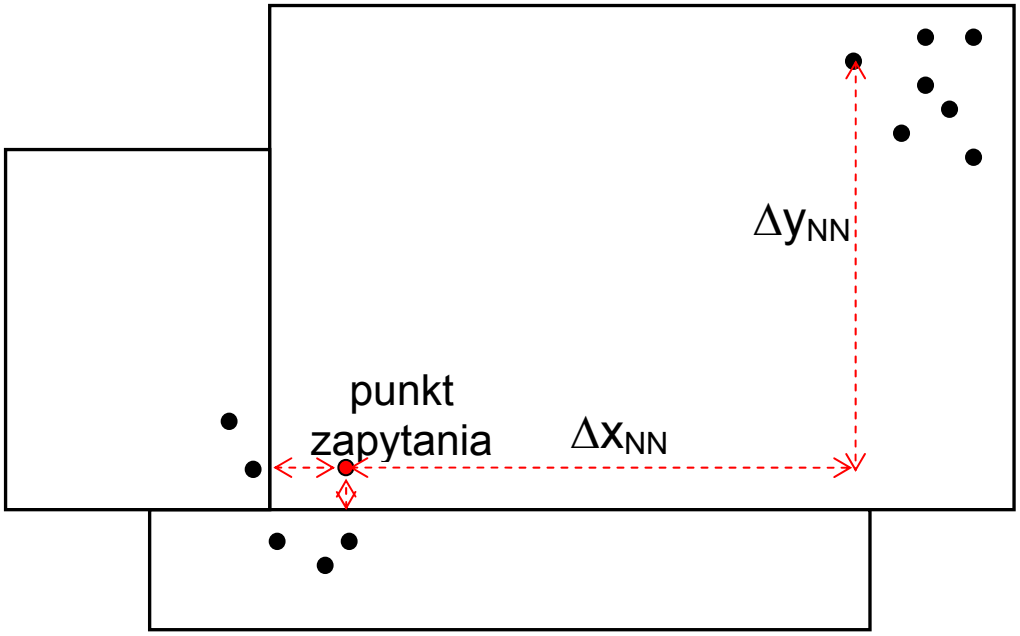
Oznaczenia:

J = Liczba obiektów w bazie danych
C = Globalna objętość R-drzewa
O = Globalne pokrywanie się R-drzewa
D = Wysokość R-drzewa
N = Liczba węzłów R-drzewa
A = Średnia liczba odczytywanych węzłów dla stu losowych zapytań

Szukanie najbliższego sąsiada (NN) za pomocą indeksów wielowymiarowych

Indeksy stosujące podział przestrzeni (pliki kratowe, kd-drzewa, kdB-drzewa, hB-drzewa, quad-drzewa) charakteryzują się znacznym pokryciem pustej przestrzeni. Kształt regionów nie odzwierciedla rozkładu danych.

Region zawierający punkt zapytania nie musi zawierać najbliższego sąsiada.



Po znalezieniu NN w regionie zawierającym punkt zapytania należy zweryfikować dla każdego wymiaru, czy odległość punktu zapytania od NN w regionie, jest większa niż odległość punktu zapytania od odpowiednich powierzchni regionu. Jeżeli tak, należy przejrzeć regiony przylegające do każdej takiej powierzchni.

Operacja jest w całości wykonywana w indeksie.

Szukanie najbliższego sąsiada (NN) za pomocą R-drzew

Złe wiadomości:

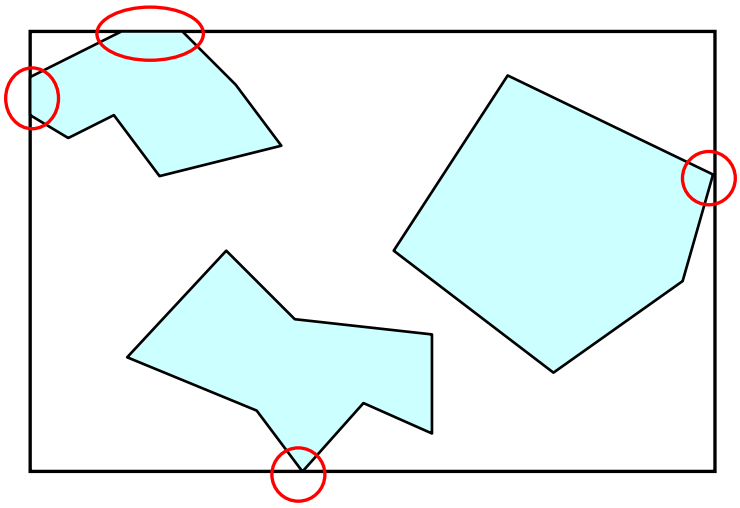
1. Ponieważ w R-drzewach regiony MBR mogą się pokrywać, więc może istnieć wiele regionów zawierających punkt zapytania.
2. Nie istnieje relacja przylegania regionów.

Dobre wiadomości:

W R-drzewach kształt regionów MBR lepiej odzwierciedla rozkład zawartych w nim danych.

Własność regionów MBR R-drzew

Każda powierzchnia (odcinki w 2-d, prostokąty w 3-d, prostopadłościany k-1 wymiarowe w k-d) regionu MBR, na dowolnym poziomie R-drzewa, zawiera, co najmniej jeden punkt indeksowanych obiektów przestrzennych bazy danych. Własność tę można wykorzystać dla optymalizacji wyszukiwania NN.



Metryki regionów MBR

Dla danego punktu **P**=(p_1, p_2, ..., p_n) i regionu **R**=(S,T), gdzie: S=(s_1, ..., s_n) i T=(t_1, ..., t_n) i $s_i \leq t_i$ dla $1 \leq i \leq n$.

MinDist(P,R) metryka optymistyczna (najlepszy przypadek) - reprezentuje najmniejszą możliwą odległość punktu **P** od obiektów przestrzennych pokrytych przez MBR - **R**.

Dla dowolnego punktu **P** i regionu MBR **R**, który pokrywa zbiór obiektów **O**={o_i}:

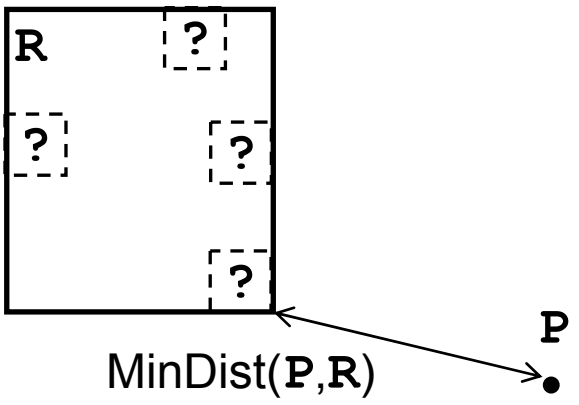
$$\forall o_i \in \mathbf{O}\ \text{MinDist}(\mathbf{P},\mathbf{R}) \leq |(\mathbf{P},o_i)||,$$

gdzie: ||(**P**,o_i)|| jest odległością między punktem **P** i obiektem przestrzennym o_i.

$$\mathbf{MinDist(P,R)}^{*} = \sum_{i=1}^{n} |p_i - r_i|^2$$

gdzie:

$$r_i = \begin{cases} s_i \text{ jeżeli } p_i < s_i; \text{ // mniejsze od min} \\ t_i \text{ jeżeli } p_i > t_i; \text{ // większe od max} \\ p_i \text{ // w pozostałych przypadkach.} \end{cases}$$



* dla uproszczenia odległości euklidesowe zostały zamienione na ich kwadraty

Metryki regionów MBR

MinMaxDist(P,R) metryka pesymistyczna (najgorszy przypadek, który może zajść) - reprezentuje **największą** możliwą odległość punktu **P** do **najbliższego** z obiektów przestrzennych pokrytych przez region **R**.

Dla dowolnego punktu **P** i regionu MBR **R**, który pokrywa zbiór obiektów **O**={o_i}:

$$\exists o_i \in \mathbf{O}\ \text{MinMaxDist}(\mathbf{P},\mathbf{R}) \geq ||(\mathbf{P},o_i)||.$$

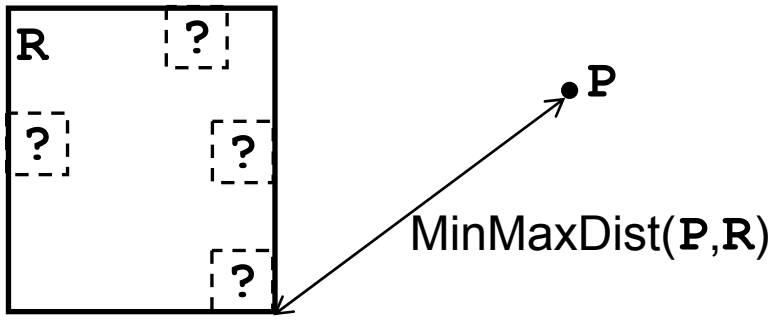
$$\mathbf{MinMaxDist(P,R)} = \min_{1\leq k\leq n}\left(|p_k - rm_k|^2 + \sum_{i=1;i\neq k}^{n} |p_i - rM_i|^2\right)$$

gdzie:

$$rm_k = \begin{cases} s_k \text{ jeżeli } p_k \leq (s_k+t_k)/2; \\ t_k \text{ w przeciwnym wypadku.} \end{cases}$$

$$rM_i = \begin{cases} s_i \text{ jeżeli } p_i \geq (s_i+t_i)/2; \\ t_i \text{ w przeciwnym wypadku.} \end{cases}$$

Dla każdego wymiaru **k** najdalszy punkt rM_i, na bliższej ścianie rm_k

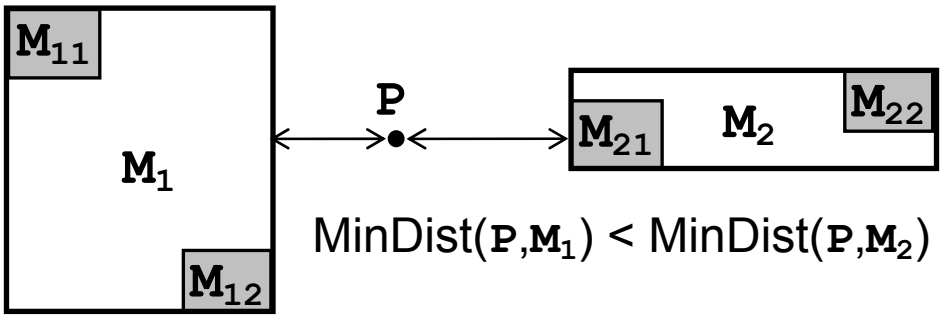


Optymalizacja szukania NN

Metryki **`MinDist`** i **`MinMaxDist`** są używane do porządkowania i ograniczania zbioru regionów MBR i skojarzonych z nimi węzłów indeksu w celu minimalizacji liczby odczytywanych bloków indeksu.

Porządkowanie

Węzły indeksu są odczytywane w kolejności odpowiadającej ich porządkowi utworzonemu ze względu na jedną ze zdefiniowanych metryk.



Ograniczanie

1. Ze zbioru regionów MBR, które mogą zawierać NN, należy usunąć takie regiony M, dla których istnieje region M' taki, że `MinDist`(P,M) > `MinMaxDist`(P,M'). One nie mogą zawierać NN.
2. Usuń znaleziony obiekt o, dla którego odległość od punktu P jest większa od `MinMaxDist`(P,M), ponieważ region M musi zawierać obiekty, który są bliższe punktowi P.
3. Ze zbioru regionów MBR należy usunąć takie regiony M, dla których `MinDist`(P,M) jest większy od odległości między P i znalezionym już punktem o. Regiony takie nie mogą zawierać NN.

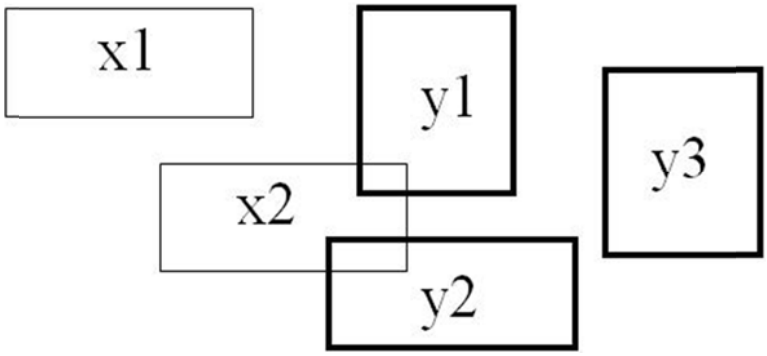
Algorytm szukania NN w R-drzewach

```
Procedure NNSearch(Node, Point, Nearest)
if Node.type=LEAF then
   for i=l to Node.count
// Na poziomie liści – wyznacz odległości do obiektów
      dist=objectDIST(Point,Node.ABL[i].rect)
      if dist<Nearest.dist
         Nearest.dist = dist
         Nearest.rect = Node.ABL[i].rect
      endif
    endfor
else
  // Na poziomach pośrednich - porządkuj, eliminuj i czytaj węzły
   // Generuj listę aktywnych gałęzi ABL
   ABL = genBranchList()
   // Sortuj ABL odpowiednio do wartości metryk
   sortBranchList(ABL)
   // Eliminuj gałęzie za pomocą reguł 1 i 3 (może wszystkie)
   last = pruneBranchList(Node,Point,Nearest, ABL)
   // Wyzeruj NN za pomocą reguły 2
   if areNear(dist, ABL) dist = null
   // Wykonaj iterację po liście ABL
   for i=l to last
      newNode = Node.branch[ABL[i]]
      // Rekurencyjnie odczytuj węzły potomne
      NNSearch(newNode,Point,Nearest)
      // Eliminuj za pomocą reguły 3
      last=pruneBranchList(Node,Point,Nearest, ABL)
   endfor
endif
End procedure
```

Łatwa adaptacja do znajdowania k-NN

Połączenie przestrzenne (ang. Spatial Join)

Dla dwóch zbiorów obiektów przestrzennych A i B, operacja przestrzennego połączenia wyznacza zbiór par $\{<a, b> \mid a \in A$ i $b \in B$ takich, że między a i b zachodzi dana relacja przestrzenna, np. $a \cap b \neq \varnothing) \}$.



Wynik zapytania: (x2, y1), (x2, y2)

Ze względu na brak globalnego porządku dla danych przestrzennych (linearyzacja przestrzeni jest w tym wypadku niestosowalna), niedostępne są szybsze metody łączenia, takie jak: `sort-merge` i `hash-join`.

Zastosowanie R-drzew pozwala przyśpieszyć operację łączenia przestrzennego poprzez ograniczenie zbioru porównywanych obiektów. Przyspieszenie łączenie jest uzyskiwane w wyniku unikania porównywania obiektów pokrywanych przez MBR, dla których nie jest spełniona zadana relacja przestrzenna.

Algorytmy wykorzystujące R-drzewa są wieloprzebiegowe.

Inne algorytmy przestrzennych połączeń:

- Seeded trees
- Hash join na pliku kratowym
- Sort-merge w pamięci operacyjnej

Zastosowanie R-drzew

Algorytm łączenia przestrzennego klasy depth-first:

```
SpatialJoin(Node: R, Node: S)
  for(all Es∈S) do
    for(all ER∈R with MBR(ER)∩MBR(Es)≠∅)do
    // jeżeli MBR się nie pokrywają dalsze schodzenie
    // w głąb drzewa jest zbędne
       if (R and S are leaf pages) then
         output(ER,Es);
       elseif (R is a leaf page) then
         SpatialJoin(ER,Es.ref);
       elseif (R is a leaf page) then
         SpatialJoin(ER.ref,Es);
       else
         SpatialJoin(ER.ref,Es.ref);
       end
    end
  end
end
```

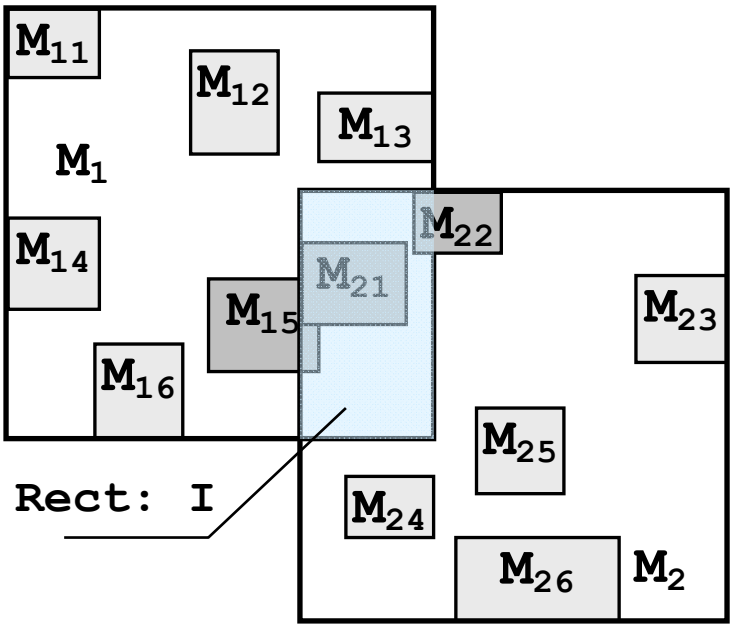
Ulepszenia poprawiające złożoność obliczeniową:

- Sortowanie elementów w węzłach
- Wstępne odfiltrowywanie elementów w węzłach do elementów, które mają część wspólną z przecięciem rodzicielskich regionów MBR

Optymalizacja czasu procesora

Modyfikacja algorytmu *połączenia przestrzennego* w celu ograniczenia liczby porównywanych MBR.

```
SpatialJoin(Node: R, Node: S, Rect: I)
```



Dla obniżenia złożoności obliczeniowej porównywane są tylko te składowe regiony, które mają część wspólną z przecięciem regionów nadrzędnych. Na rysunku powyżej są to regiony M_{15} oraz M_{21} i M_{22}.

Przykłady komercyjnych zastosowań R-drzew

Indeksowanie danych geograficznych

- Opcja Spatial w systemie Oracle obejmuje możliwość przyśpieszenia wykonywania zapytań za pomocą Quad-Tree z funkcją linearyzacji przestrzeni opartej na krzywej Peano (Z-ordering) oraz R*-drzew.

Przetwarzanie OLAP

- W systemie Informix pakowane R-drzewa są używane dla przyśpieszenia operacji utrzymywania tabel agregacji oraz przyśpieszenia analiz danych. Zamiast zbioru zagregowanych tabel system oferuje rozszerzony model kostki wielowymiarowej **Cubetree**, która modeluje dane magazynu danych jako punkty w przestrzeni wielowymiarowej. Zbiór tabel agregacji jest zastąpiony pojedynczym zapakowanym R-drzewem.