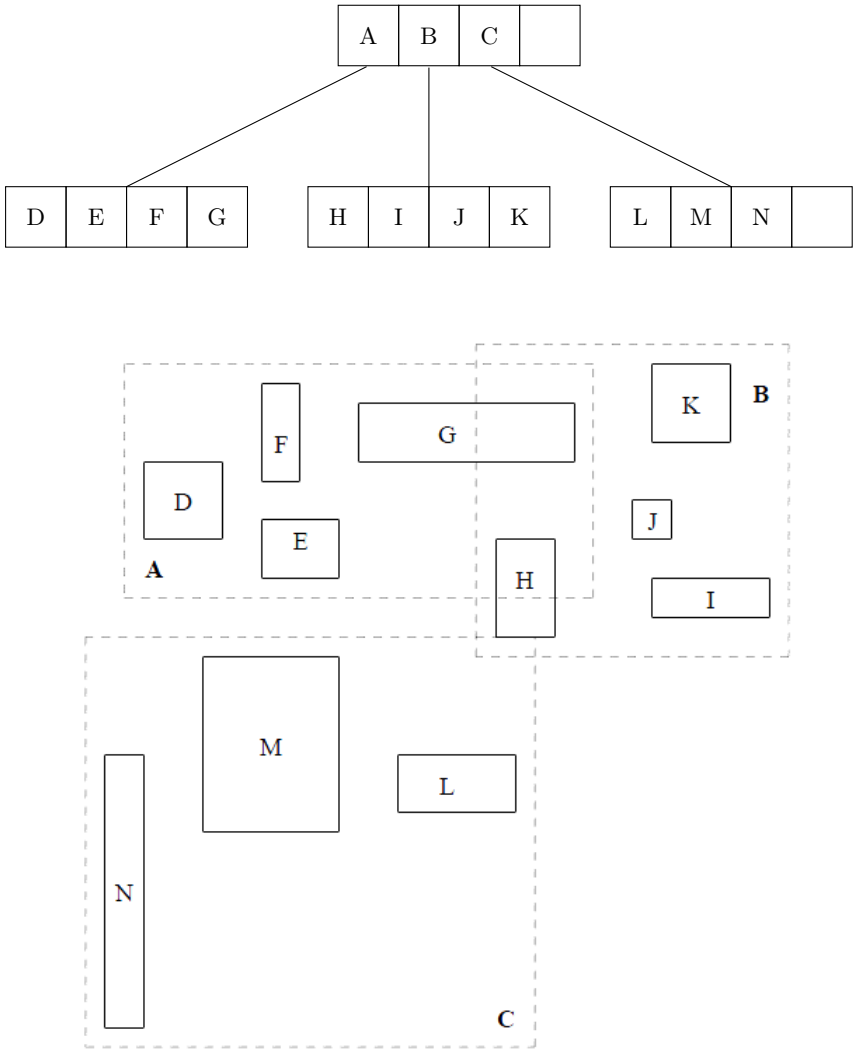
ALS1 – Přednáška 9

1 R+-stromy

Při vyhledávání v R-stromu jsou důležité dva koncepty: *pokrytí* (coverage) a *překryv* (overlap):

- pokrytí úrovně R-stromu je celková oblast všech MBR uzlů na té úrovni.
- překryv úrovně R-stromu je celková oblast obsažená ve dvou nebo více MBR uzlů na té úrovni.

Zjevně, efektivita vyhledávání v R-stromu vyžaduje, aby pokrytí i překryv byly co nejmenší. Minimální pokrytí minimalizuje prázdný prostor pokrytý uzly. Minimální překryv je ještě důležitější – minimalizuje sestup do více větví. Obrázek 1 ukazuje příklad R-stromu a rozložení jeho obdélníků.

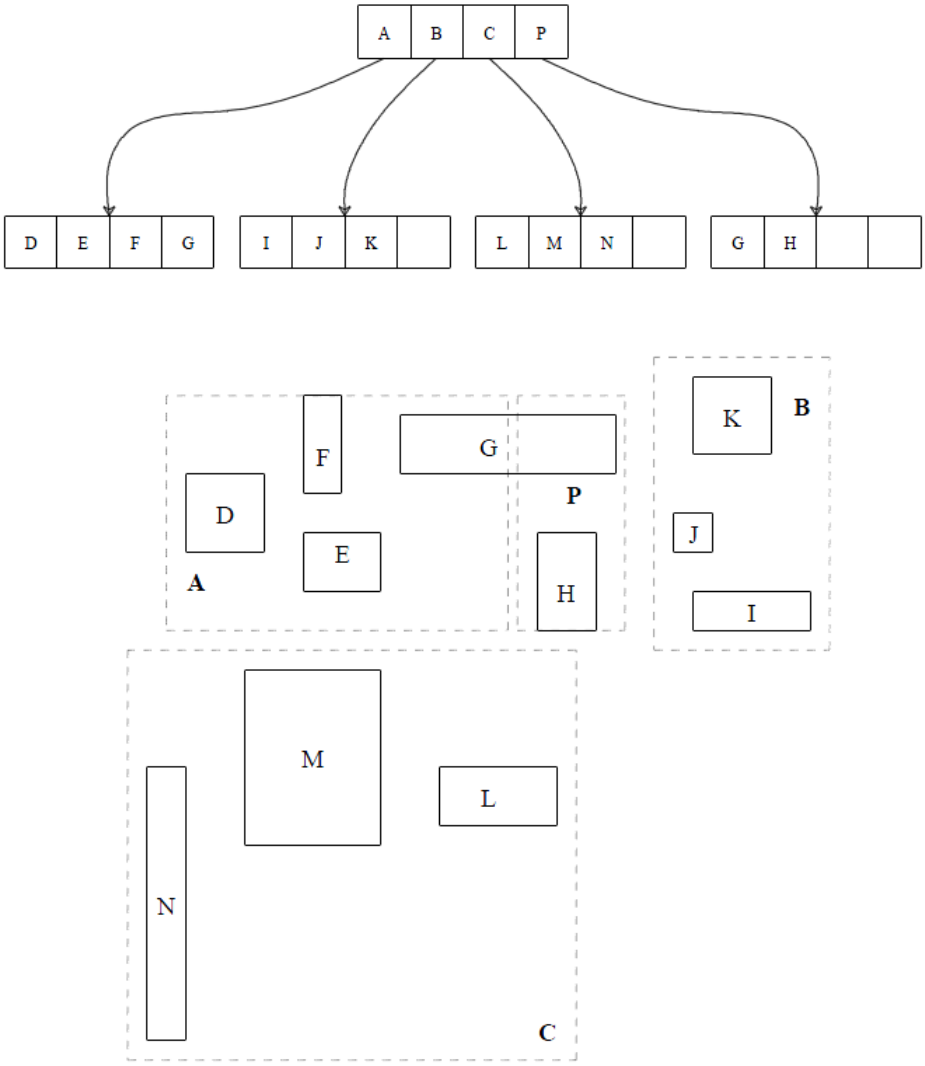


Obrázek 1. R-strom a rozložení jeho obdélníků

Hlavní idea R+-stromů je zajistit prázdný překryv pro vnitřní uzly tím, že je umožněno splitovat obdélníky (viz obr **??**).

R+-stromy mohou být chápány jako rozšíření KDB-stromů tak, aby zahrnovaly nenulové oblasti (tedy ne pouze body, ale i obdélníky). Vylepšení oproti KDB-stromům je snížené pokrytí; uzly dané úrovně nutně nepokrývají celý prostor. Oproti R-stromům vykazují R+-stromy velmi dobrý výkon při vyhledávání (obzvláště při vyhledávání bodu) na úkor paměti.

R^+-strom má následující vlastnosti:



Obrázek 2. R+-strom a rozložení jeho obdélníků (stejná data jako v obr. 1)

1. pro každý uzel (I, p) v nelistovém uzlu, podstrom zakořeněný v uzlu, na který ukazuje p, obsahuje obdélník I_{child} právě tehdy, když I_{child} je pokryt obdélníkem I.
   Výjimka je, když I_{child} je obdélník v listovém uzlu; v tom případě se I_{child} může pouze překrývat s I.
2. pro každé dva záznamy (I_1, p_1) a (I_2, p_2) nelistového uzlu je překryv mezi I_1 a I_2 nulový.
3. kořen má alespoň dva potomky, pokud to není list.
4. všechny listy jsou na stejné úrovni.
5. maximální počet záznamů v uzlu je M.

1.1 Vyhledávání

Algoritmus vyhledávání je podobný vyhledávání v R-stromu. Idea je rozdělit prohledávaný prostor na disjunktní pod-oblasti a pro každý z nich sestupovat až k datům. Hlavní rozdíl je, že v R+-stromu se nemohou MBR překrývat, takže to může vést k efektivnímu vyhledávání.

SEARCH
Vstup: R^+ strom zakořeněný v uzlu R a vyhledávací okno (obdélník) W
Výstup: Všechna data překrývající W
Postup: Dekomponuj prohledávaný prostor a rekurzivně prohledávej strom.

S1 [Prohledej vnitřní uzly] Pokud R není list, pak pro každý záznam (I, p) uzlu R testni, jestli se I překrývá s W. Pokud ano, Search($CHILD, W \cap I$), kde $CHILD$ je uzel na který ukazuje p.
S2 [Prohledej listové uzly] Pokud R je list, testni I všech objektů v R a vrať ty, které se překrývají s W.

1.2 Vkládání

Vkládání nového obdélníku do R+-stromu je provedeno prohledáním stromu a přidáním obdélníku do listových uzlů. Rozdíl oproti R-stromu je ten, že vstupní obdélník může být vložen do více než jednoho listu z toho důvodu, že může být splitnut na pod-obdélníky podle existujícího rozkladu prostoru. Navíc přeplněné uzly mohou být splitnuty a splity jsou šířeny do rodičovských uzlů i do potomků.

INSERT
Vstup: R^+ strom zakořeněný v uzlu R a vstupní obdélník IR
Výstup: Nový R^+ strom po vložení IR
Postup: Najdi, kam by IR mělo být; vlož ho do odpovídajících listových uzlů.

I1 [Prohledej nelistové uzly] pokud R není list, tak pro každý záznam (I, p) z R zkontroluj, jestli se I překrývá s IR. Pokud ano Insert($CHILD, IR$), kde $CHILD$ je uzel, na který ukazuje p.
I2 [Přidej záznam] Pokud R je list, přidej IR do R. Pokud poté, co je vložen nový obdélník, R má více než M záznamů, vyvolej SplitNode(R), přeuspořádej strom.

1.3 Mazání

Mazání obdélníku z R+-stromu se provádí stejně jako v R-stromu – nejdříve se najde obdélník, který má být smazán, a pak se odstraní z listových uzlů.

DELETE
Vstup: R^+ strom zakořeněný v uzlu R a vstupní obdélník IR
Výstup: Nový R^+ strom po odstranění IR
Method: Najdi, kde IR je, odstraň ho z odpovídajících listových uzlů.

D1 [Prohledej nelistové uzly] Pokud R není list, pak pro každý záznam (I, p) z R zkontroluj, jestli se I překrývá s IR. Pokud ano Delete($CHILD, IR$), kde $CHILD$ je uzel, na který ukazuje p.
D2 [Smaž z listových uzlů] Pokud je R list, odstraň IR z R a uprav rodičovský obdélník, který ohraničuje obdélníky zbývajících objektů.

Mazání často vede k degeneraci stromu, proto se doporučuje jej občas přeuspořádat.

1.4 Splity

Pokud je uzel přeplněn, je potřeba jej splitnout a vytvořit dva nové uzly. Protože potřebujeme, aby ty dva nové uzly, pokrývaly disjuknтní oblast, potřebujeme nejdříve najít dobré rozdělení (vertikální nebo horizontální).

Všimněme si, že narozdíl od R-stromu se split je potřeba šířit split i dolů.

SplitNode

Vstup: uzel R
Výstup: Nový R^+ strom
Method: Najdi rozklad uzlu, který má být splitnut; vytvoř nové uzly a, pokud je to potřeba, šiř split nahoru a dolu.

SN1 [Najdi rozklad] Rozlož R pomocí procedury Partition. Nechť I a p je obdélník a pointer asociované s uzlem R. A nechť S_1 a S_2 označují ty dvě podoblasti, které jsou výsledkem rozkladu.
Vytvoř $n_1 = (I_1, p_1)$ a $n_2 = (I_2, p_2)$ – nové uzly, které jsou výsledkem splitu R, kde $I_i = I \cap S_i$.

SN2 [Naplň nové uzly] Vlož do n_i všechny uzly (I_k, p_k) z uzlu R t.ž. I_k leží úplně v I_i, pro $i = 1, 2$.
Pro ty uzly pro které platí $I_k \cap I_i \neq I_k$:

- pokud R je list, vlož I_k do obou nových uzlů.
- jinak použij SplitNode pro rekurzivní split potomků přes rozklad. Nechť (I_{k1}, p_{k1}) a (I_{k2}, p_{k2}) jsou dva uzly po splitu (I_k, p_k) kde I_{ki} leží úplně v I_i, pro $i = 1, 2$. Vlož tyto dva uzly do odpovídajícího uzlu n_i.

SN3 [Šiř split nahoru] Pokud R je kořen, vytvoř nový kořen s dvěma potomky n_1 a n_2. Jinak, nechť PR je uzel, který je předkem R. Nahraď R v PR uzly n_1 a n_2. Pokud má nyní PR více než M záznamů, vyvolej SplitNode(PR).

Packing: Algoritmy Partition a Pack (pro 2D)
Partition rozdělí prostor obsahující N 2D obdélníků čarou paralelní s osou x (x-řez) nebo osou y (y-řez).
Výběr řezu:

- nearest neighbors,
- minimální celkový prostor těch dvou podoblastí,
- minimální počet splitů obdélníků.

(většinou exponenciální složitost, vhodnější je použít nějaké suboptimální kritéria)

Pack

Vstup: Množina obdélníků S, která má být utříděna a fill-factor ff stromu
Výstup: „dobrý“ R^+ strom
Method: Rekurzivně *spakuj* záznamy každé úrovně stromu

P1 [Nepotřebujeme pakovat] Pokud $N = |S|$ je menší než ff, vytvoř kořenový uzel R stromu a vrať ho.
P2 [Inicializace] Nastav $AN = \emptyset$. AN obsahuje množinu obdélníků další úrovně, které budou *spakovány* později.
P3 [Rozlož prostor] $(R, S') =$ Partition(S, ff) Pokud rozkládáme nelistové uzly a některé obdélníky byly splitnuty kvůli vybranému rozkladu, rekurzivně šiř split dolu a pokud je to nutné, šiř změny i nahoru.
$AN =$ append(AN, R).
Pokračuj krokem $P3$ dokud $S' = \emptyset$.
P4 [Rekurzivně *zpakuj* nelistové uzly] Vrať Pack(AN, ff).

Partition

Vstup: Množina S obdélníků a fill-factor ff první podoblasti.
Výstup: Uzel R obsahující obdélníky první podoblasti a množina S' zbývajících obdélníků.
Metoda: Rozlož celkový prostor do lokálně optimální (ve smyslu výkonu vyhledávání) první podoblast a zbývající podoblasti.

PA1 [Není potřeba rozklad] Pokud celkový prostor, který má být rozkládán, obsahuje méně nebo rovno jak ff obdélníků, není prováděna další dekompozice; je vytvořen uzel R s těmito záznamy a algoritmus vrací (R, nil)
PA2 [Vypočítej nejnižší x a y hodnoty] Nechť Ox a Oy jsou nemenší x a y souřadnice daných obdélníků.
PA3 [Sweep přes x] $(Cx, xcut) =$ Sweep(x, Ox, ff). Cx je cena splitu přes rozměr x.
PA4 [Sweep přes y] $(Cy, ycut) =$ Sweep(y, Oy, ff). Cy je cena splitu přes rozměr y.
PA5 [Vyber bod rozkladu] Vyber řez, který dává nejmenší z Cx a Cy, rozděl prostor, rozděl obdélníky a jejich splity. Je vytvořen uzel R, který obsahuje všechny záznamy z první podoblasti. Nechť S' označuje množinu všech obdélníků spadajících do druhé podoblasti. Vrať (R, S').

SWEEP

Vstup: Osa, na které je *sweep* prováděn, bod Oxy, na kterém *sweep* začíná, a fill-factor ff.
Výstup: Vypočtené vlastnosti první podoblastí a x-řez nebo y-řez.
Metoda: Skenuj od Oxy a počítej vlastnosti, dokud není dosaženo ff.

SW1 [Najdi prvních ff obdélníků] Začni od Oxy, vyber dalších ff obdélníků ze seznamu obdélníků setříděných podle vstupní osy.
SW2 [Vyhodnoť rozklady] Vypočítej celkovou cenu měřených vlastností použitých k utřiďování obdélníků.
Vrať cenu a největší x nebo y souřadnici z těch ff obdélníků.

2 R∗ stromy

Minimalizace pokrytí a překryvu je rozhodující pro výkon R-stromu.

R*-strom se pokouší redukovat obojí použitím kombinace přepracovaného algoritmu na split uzlu a koncepcí nuceného znovu-vložení při přeplnění. To je založeno na pozorování, že struktura R-stromu je velmi závislá na pořadí, ve kterém jsou data vkládána. Takže inkrementálně postavený strom bude spíše suboptimální, v porovnání s bulk-loaded. Mazání a znovu-vložení uzlů ji umožňuje najít místo ve stromu, které může být vhodnější než jejich původní umístění.

2.1 Možná optimalizační kritéria

O1 *Oblast pokrytá obdélníky by měla být minimalizovaná*, t.j. oblast pokrytá MBR, ale nepokrytá vloženými obdélníky (prázdný prostor) by měla být minimalizovaná. To zvýší výkon pri prohledávání, protože směr sestupu může být vybrán na vyšší úrovni.
O2 *Překryv mezi obdélníky by měl být minimalizován.* Také snižuje počet cest, kterými se sestupuje.
O3 *Obvod obdélníků by měl být minimalizován.* Pro fixní obsah, obdélník s nejmenším obvodem je čtverec. Snažíme se tedy o co nejčtvercovitější obdélníky – takové se snadněji pakují.
O4 *Využití paměti.* Vyšší využití paměti vede k menší výšce stromu a tím k rychlejšímu vyhledávání.

2.2 Algoritmus ChooseSubtree

Na rozdíl od R-stromů se nebere v potaz jen O1, ale O1,O2,O3, s tím, že překryv je definován následovně.
Nechť $E_1, \ldots, E_p$ jsou záznamy v aktuálním uzlu. Pak

$$\text{overlap}(E_k) = \sum_{i=1, i\neq k}^{p} \text{area}(E_k.\text{Rectangle} \cap E_i.\text{Rectangle}), \qquad 1 \leq k \leq p.$$

CS1 Nastav N na root.
CS2 Pokud je N list vrať N. Jinak
- pokud ukazatele na potomky v N ukazují na listy, vyber ten záznam v N, jehož obdélník potřebuje nejmenší zvýšení překryvu, aby obsáhl nová data. Remízy vyřeš výběrem obdélníku, který potřebuje nejmenší zvětšení obsahu, a poté záznamu s obdélníkem nejmenšího obsahu.
- pokud ukazatele na potomky v N neukazují na listy, vyber záznam v N jehož MBR potřebuje nejmenší zvětšení obsahu, zahrnul nová data. Remízy vyřeš výběrem záznamu s obdélníkem nejmenšího obsahu.

CS3 Nastav N potomka záznamy vybraného v CS2.

V této verzi je složitost zjištění překryvu kvadratická vzhledem k počtu záznamů. Ale pro velké velikosti uzlů můžeme snížit počet záznamů, pro které je výpočet proveden, protože pro velmi vzdálené obdélníky je nepravděpodobné, že by nastal minimální překryv. Takže pro snížení výpočetního času, tato část algoritmu může být modifikována následovně.

Setřiď obdélníky v N vzestupně vzhledem ke zvětšení jejich obsahu pro zahrnutí nových dat.
Nechť A je skupina prvních p záznamů.
Ze záznamů v A, uvažujíce všechny záznamy v N, vyber záznam, jehož obdélník potřebuje nejmenší zvětšení překryvu.
Remízy vyřeš, tak jak je to popsáno výše.

2.3 Splity v R* stromech

R* strom používá následující metodu k nalezení dobrého splitu. Podle každé osy, záznamy jsou nejdříve seřízeny podle podle nižší hodnoty, a pak podle vyšší hodnoty jejich obdélníku. Pro každou třídu je určeno $M - 2m + 2$ rozdělení $M + 1$ záznamů do dvou skupin. k-té rozdělení (pro $k = 1, \ldots, (M - 2m + 2)$) je popsáno následovně: první skupina obsahuje $(m - 1 + k)$ záznamů, druhá skupina obsahuje zbytek.

Pro každé rozdělení jsou vypočteny hodnoty vhodnosti. V závislosti na nich je vybráno konečné rozdělení záznamů.

(i) hodnota obsahu

$$\text{area}[\text{bb}(\text{první oblast})] + \text{area}[\text{bb}(\text{druhá oblast})]$$

(ii) hodnota obvodu

$$\text{margin}[\text{bb}(\text{první oblast})] + \text{margin}[\text{bb}(\text{druhá oblast})]$$

(iii) hodnota překryvu

$$\text{margin}[\text{area}(\text{první oblast}) \cap \text{margin}[\text{area}(\text{druhá oblast})$$

Možné metody zpracovnání těch hodnot jsou určit

- minimum podle jedné osy nebo třídy.
- minimum součtu hodnot jedné osy nebo třídy.
- celkové minimum.

Získané hodnoty mohou být aplikovány k výběru osy splitu, nebo konečného rozdělení (na vybrané ose splitu).

Split

S1 Vyvolej ChooseSplitAxis na určení osy, podle které bude split proveden.
S2 Vyvolej ChooseSplitIndex na určení nejlepšího rozdělení do dvou skupin podle te osy.
S3 Rozděl záznamy podle toho rozdělení.

ChooseSplitAxis

CSA1 Pro každou osu: setřiď záznamy podle dolní, a poté podle horní hodnoty jejich obdélníku a urči všechna rozděleni. Vypočítej S jako součet všech hodnot obvodu všech rozdělení.
CSA2 Vyber osu s minimálním S jako osu splitu.

ChooseSplitIndex

CSI1 Podle vybrané osy splitu vyber to rozdělení, které má minimální hodnotu překryvu. Remízy řeš výběrem rozdělení s minimální hodnotou obsahu.

2.4 Nucené znovuvložení

R-stromy i R*-stromy jsou nedeterministické v umisťování záznamů do uzlů, t.j. různé posloupnosti vložení vedou k různým stromům. Kvůli tomu R-stromům škodí staré záznamy vložené na začátku – v raných fázích budování stromu. Takové záznamy mohly vytvořit řídící obdélníky, které nemůžou zajistit dobrý výkon později. Velmi malá reorganizace řídících obdélníků je prováděna v rámci splitu. Je ale potřeba daleko mocnější nástroj k reorganizaci struktury.

InsertData

ID1 Vyvolej Insert počínajíce úrovní listů jako parametrem.

Insert

I1 Vyvolej ChooseSubtree, s úrovní jako parametrem k nalezení vhodného uzlu N, do kterého se umístí nový záznam E.
I2 Pokud N má méně než M záznamů, umísti E do N. Pokud N má M záznamů, vyvolej OverflowTreatment s úrovní uzlu N jako parametrem (pro znovuvložení nebo split).
I3 Pokud bylo vyvoláno OverflowTreatment a nastal split, šiř OverflowTreatment nahoru (pokud je to nutné).
Pokud OverflowTreatment způsobil split kořene, vytvoř nový kořen.
I4 Uprav všechny řídící obdélníky na cestě vkládání, tak aby to byly MBR uzavírající obdélníky v potomku.

OverflowTreatment

OT1 Pokud úroveň není úroveň kořene a toto je první vyvolání OverflowTreatment v dané úrovni během vkládání jednoho obdélníku, vyvolej ReInsert jinak vyvolej Split.

ReInsert

RI1 Pro všech $M + 1$ záznamů uzlu N, vypočítej vzdálenost mezi středy jejich obdélníků a středem MBR uzlu N.
RI2 Setřiď záznamy v klesajícím pořadí vzhledem k jejich vzdáleností vypočtených v RI1.
RI3 Odstraň prvních p záznamy z N a uprav MBR uzlu N.
RI4 Podle pořadí z RI2, začínajíce maximální vzdáleností nebo minimální vzdáleností, vyvolej Insert ke znovu-vložení těchto uzlů.

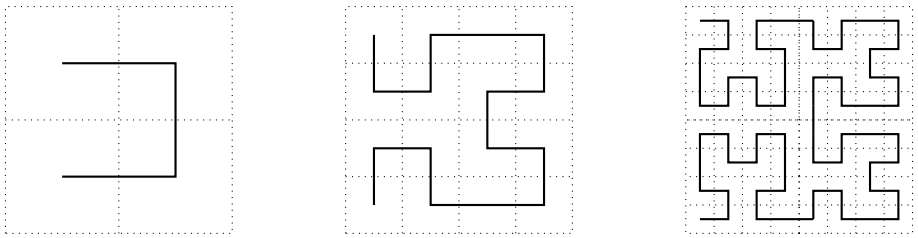
3 Hilbertův R-strom

– hybridní struktura založená na R-stromech a B-stromech.
– vlastně je to B-strom s geometrickými objekty, charakterizovanými *Hilbertovou hodnotou* jejich centroidu
– struktura je založena na *Hilbertově křivce*

Gramatika
axiom: x

$$x \to +yF - xFx - Fy+$$
$$y \to -xF + yFy + Fx-$$

Interpretace F – jeď dopředu, $+,-$ otočka o 90.



Obrázek 3. Hilbertova křivka pro $n = 1, n = 2, n = 3$.

Procedura na zjištění pozice na Hilbertově křivce:

```
dist(side, area, result, xx, yy)
{
if (side <= 0) return result;

if (xx < side && yy < side) {
   return dist(side div 2, area div 4, result, yy, xx);
}
else if (xx < side) { // && yy >= side
        return dist(side div 2, area div 4,
                    result + area, xx, yy - side);
}
else if (yy < side) { // && xx >= side
        return dist(side div 2, area div 4,
                    result + area * 3,
  side - yy - 1, side * 2 - xx - 1);
}
else { // xx >= side && yy >= side
        return dist(side div 2, area div 4,
                    result + area * 2, xx - side, yy - side);
}
```

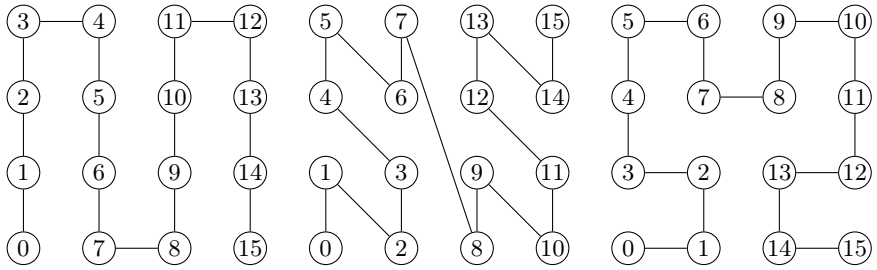
Insert(E, RN) E – vkládaný záznam, RN – uzel

HI1 (vnitřní uzel) pokud RN není list, mezi záznamy uzlu vyber záznam e, který má nejmenší H větší než H nového objektu. Vyvolej Insert$(E, e.p)$, Uprav MBR uzlu RN. Skonči.
HI2 (list, je tam místo) pokud je volné místo v RN. Přidej E do RN na pozici odpovídající H uspořádání. Skonči.
HI3 (list, není tam místo) Vyvolej HandleOverflow(E, RN), Rozšiř změnu nahoru.

HandleOverflow(E, RN), E – záznam, RN – uzel
vrací nový uzel, nebo **NULL**.

HO1 (inicializace) $\mathcal{E}$ – množina záznamů uzlu RN a jeho $s-1$ sousedních uzlů. $\mathcal{E} \leftarrow \mathcal{E} \cup E$
HO2 (je možné rozložit) pokud existuje uzel mezi $s-1$ sousedy, který není plný, rozlož záznamy v $\mathcal{E}$mezi s uzlů, podle H-uspořádání. Vrať **NULL**
HO3 (není možné rozložit) jinak: Vytvoř nový uzel NN. Rozlož všechny záznamy v $\mathcal{E}$ do $s+1$ uzlů. Vrať nový uzel NN.

Možno také vybrat jinou křivku vyplňující 2d-prostor (viz obr. 4).



Obrázek 4. křivka po sloupcích (vlevo), Peanova křivka (uprostřed), Hilbertova křivka (vpravo).