Algoritmy pro rozsáhlá data

L06: Varianty R-stromů

Jan Konecny

18. listopadu 2021

Plán na dnešek (přednáška i cvičení)

- kd-stromy
- Varianty R-stromů: R^+-stromy, R^*-stromy, Hilbertovy stromy
- (megaúkol 2)

R^+-strom

R^+-stromy poprvé představeny v:

T. Sellis, N. Roussopoulos, C. Faloutsos.
The R+ -Tree: A Dynamic Index for Multi-Dimensional Objects.
1987

R-stromy poprvé představeny v:

A. Guttman,
R-Trees: A dynamic index structure for spatial searching,
in Proc. of ACM SIGMOD, June 1984, pp. 47–57

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

Chceme, aby pokrytí i překryv byly co nejmenší.

- Minimální pokrytí minimalizuje prázdný prostor pokrytý uzly.
- Minimální překryv je ještě důležitější – minimalizuje sestup do více větví.

Které z těch kritérií skutečně R-strom uvažoval, při vkládání?

Hlavní idea R^+-stromů:

- je zajistit prázdný překryv pro vnitřní uzly tím, že je umožněno splitovat obdélníky.
- R^+-stromy mohou být chápány jako rozšíření KDB-stromů (teď uděláme odbočku ke kd- a KDB-stromům).

Zpět k R^+-stromům

Definice:

1. pro každý záznam (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.

Všimněme si, že:

- nemáme minimální počet záznamů v uzlu; uvidíme, že záznamů může být i nula;
- indexový záznam se může nacházet ve více listech;
- na nelistových úrovních nedochází k žádným překryvům.
- **hlavní rozdíl:** obdélníky v nelistových úrovních se nemohou překrývat, takže to může vést k efektivnímu vyhledávání.

Vyhledávání v R^+-stromu:

- je podobné vyhledávání v R-stromu.

`R+-search`
Vstup: R^+-strom zakořeněný v uzlu R a vyhledávací okno (obdélník) W
Výstup: Všechna data překrývající W
Postup: Dekomponuj W 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, vyvolej `R+-search`($CHILD$, $W \cap I$), kde $CHILD$ je uzel na který ukazuje p.

S1 [Prohledej listové uzly]
Pokud R je list, testni I všech objektů v R a vrať ty, které se překrývají s W.

Vkládání do R^+-stromu:

- Vkládání nového obdélníku do R^+-stromu je provedeno prohledáním stromu a přidáním obdélníku do listových uzlů.

Rozdíl oproti R-stromu:

- vstupní obdélník může být vložen do více než jednoho listu z toho důvodu, že může být splitnut na pod-obdélníky podle existujícího rozkladu prostoru.
- přeplněné uzly mohou být splitnuty a splity jsou šířeny do rodičovských uzlů i do potomků.

`R+Insert`
Vstup: R^+-strom zakořeněný v uzlu R a vstupní obdélník IR
Výstup: Nový R^+-strom po vložení IR
Postup: Najdi, kam by I_R mělo být vloženo; 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: Vyvolej `R+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).

Mazání z R^+-stromu:

- Mazání 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.
- Mazání často vede k degeneraci stromu, proto se doporučuje jej občas přeuspořádat.
- Všimněme si, že nedochází k žádnému slévání/kondenzaci.

`R+Delete`
Vstup: R^+-strom zakořeněný v uzlu R a vstupní obdélník IR
Výstup: Nový R^+-strom po odstranění IR
Postup: 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 `R+Delete`($CHILD$, IR), kde $CHILD$ je uzel, na který ukazuje p.

D2 [Smaž z listových uzlů]
Pokud je R list, odstraň IR z R.

Rozlomení uzlu

- Pokud je uzel přeplněn, musíme jej splitnout a vytvořit dva nové uzly.
- Protože potřebujeme, aby ty dva nové uzly, pokrývaly disjukntní oblast, potřebujeme nejdříve najít dobré rozdělení (vertikální nebo horizontální).
- Všimněme si, že narozdíl od R-stromu se split může šířit i dolů.

`R+SplitNode`
Vstup: uzel R
Výstup: Nový R^+-strom
Postup: 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 je obdélník a p pointer uzlu R.
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_{k_1}, p_{k_1}) a (I_{k_2}, p_{k_2}) jsou dva uzly po splitu (I_k, p_k), kde I_{k_i} 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 P_R je uzel, který je předkem R.
Nahraď R v P_R uzly n_1 a n_2.
Pokud má nyní P_R více než M záznamů, vyvolej `R+SplitNode`(P_R).

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.

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

P1 [Nepotřebujeme pakovat]
Pokud $N = |S|$ je menší než ff, vytvoř kořen R^+-stromu a vrať ho.

P2 [Inicializace]
Nastav $AN = \emptyset$.
AN obsahuje množinu obdélníků další úrovně, které budou spakovány později.

P3 [Rozlož prostor]
$(R, S_0) = \texttt{Partition}(S, ff)$.
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.
$AN = \texttt{append}(AN, R)$.
Pokračuj krokem P3, dokud $S_0 = \emptyset$.

P4 [Rekurzivně zpakuj nelistové uzly] Vyvolej $\texttt{Pack}(AN, ff)$.

`Partition`
Vstup: Množina S obdélníků a fill-factor *ff* první podoblasti.
Výstup: Uzel R obsahující obdélníky první podoblasti a množina S_0 zbývajících obdélníků.
Postup: 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 nejmenší x a y souřadnice daných obdélníků.

PA3 [Sweep přes x]
$(Cx, xcut) = \texttt{Sweep}(x, Ox, ff)$. Cx je cena splitu přes rozměr x.

PA4 [Sweep přes y]
$(Cy, ycut) = \texttt{Sweep}(y, Oy, ff)$. Cy je cena splitu přes rozměr y

PA5 [Vyber bod rozkladu]
Vyber řez, který dává nejmenší z Cx a Cy.
Je vytvořen uzel R, který obsahuje všechny záznamy z první podoblasti.
Nechť S_0 označuje množinu všech obdélníků spadajících do druhé podoblasti.
Vrať $(R, S0)$.

`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

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 pakováním.
- Mazání a znovu-vložení uzlů ji umožňuje najít místo ve stromu, které může být vhodnější než jejich původní umístění.

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.

Vkládání do R*-stromu

- Procedura `choose-subtree` – 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

$$\mathrm{overlap}(E_k) = \sum_{i=1, i \neq k}^{p} \mathrm{area}(E_k.\mathrm{rectangle} \cap E_i.\mathrm{rectangle})$$

pro $1 \leq k \leq p$.

`choose-subtree`

CS1 [Inicializace]
Nastav N na kořen.

CS2 [Test na list]
Pokud je N list, vrať N.

CS3 [Výběr podstromu a sestup]

- 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.

Nastav N vybraného potomka záznamu a pokračuj krokem CS2.

Z před-předchozícho slajdu:

Nechť $E_1, \ldots, E_p$ jsou záznamy v aktuálním uzlu.
Pak

$$\mathrm{overlap}(E_k) = \sum_{i=1, i \neq k}^{p} \mathrm{area}(E_k.\mathrm{rectangle} \cap E_i.\mathrm{rectangle})$$

pro $1 \leq k \leq p$.

- V této verzi je složitost zjištění překryvu kvadratická vzhledem k počtu záznamů.
- 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.

- 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 — (p je předem zvolený parametr)
- 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.

Splity v R*-stromech

- Podle každé osy, záznamy jsou nejdříve seřízeny 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

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

(ii) hodnota obvodu

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

(iii) hodnota překryvu

$$\mathrm{area}[\mathrm{MBR}(\text{první oblast}) \cap [\mathrm{MBR}(\text{druhá oblast})]$$

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 té 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 rozdělení.
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.

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 záznamů z N a uprav MBR uzlu N.

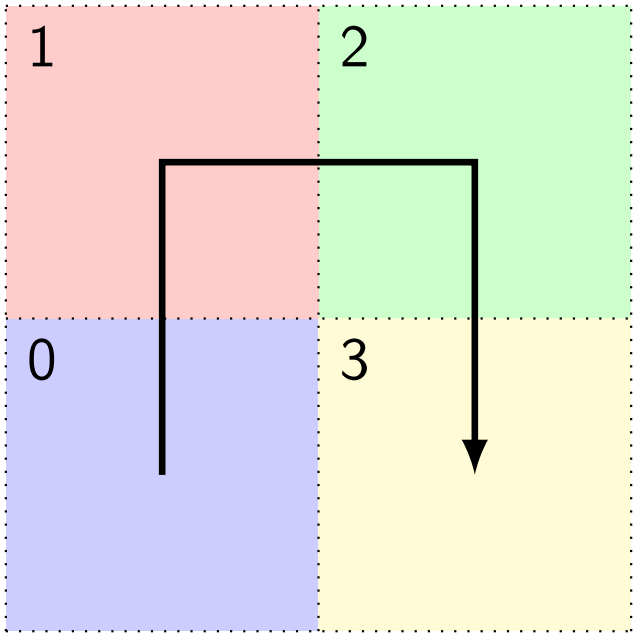
RI4 Podle pořadí z RI2, začínajíce maximální vzdáleností nebo minimální vzdáleností, vyvolej `Insert` ke znovu-vložení těchto uzlů.

Hilbertův R-strom

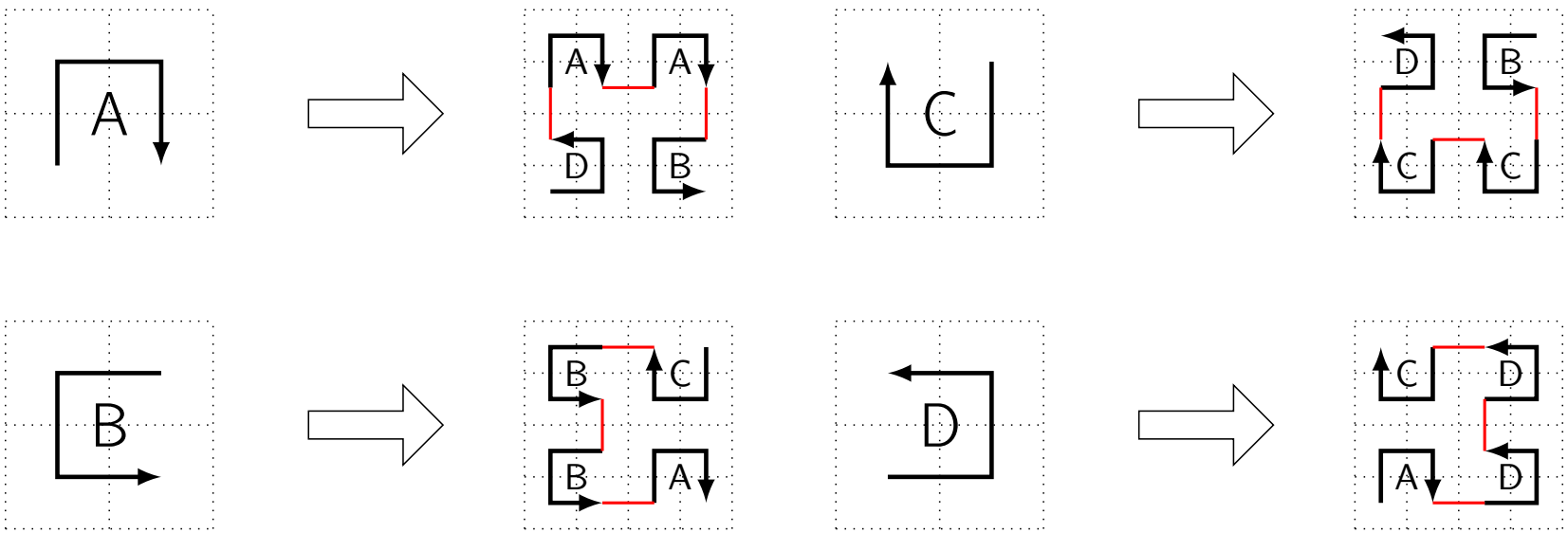
- Je hybridní struktura založená na R-stromu a B^+-stromu.
- Vlastně je to B^+-strom s geometrickými objekty charakterizovanými podle Hilbertovy hodnoty (H-hodnoty) jejich těžiště.
- Struktura je založena na Hilbertově křivce vyplňování prostoru – bylo ukázáno, že Hilbertova křivka dobře zachovává blízkost prostorových objektů.

Hilbertova křivka

Hilbertova křivka prvního řádu ($n = 1$) – jednoduchá křivka procházející prostor rozdělený do čtyř ($2^n \times 2^n$) oblastí, v pořadí
0 (modrý)–1 (červený)–2(zelený)–3(žlutý).



Hilbertovy křivku vyššího řádu $(n+1)$ vytvoříme rekurzivně z křivky nižšího řádu (n) pomocí násl. pravidel:



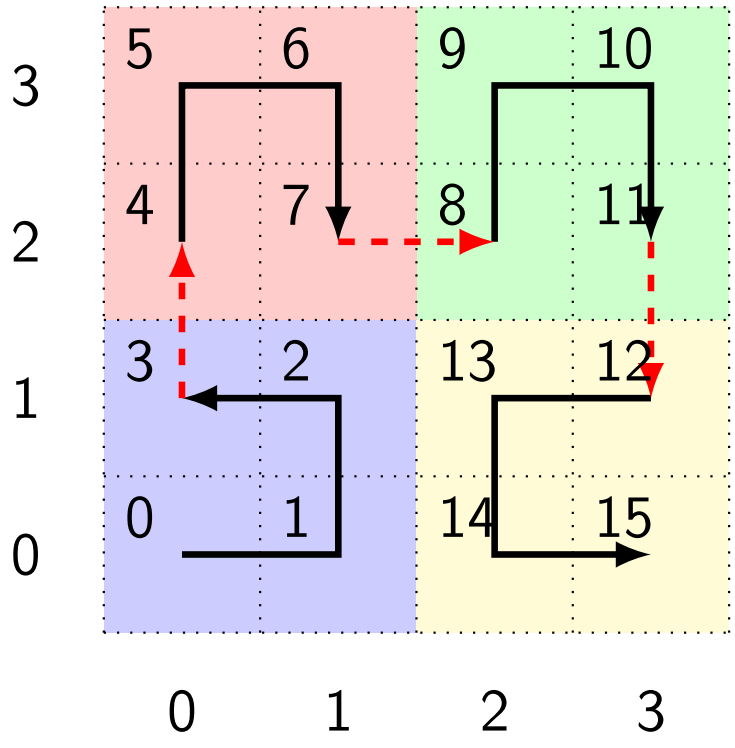
Příklad 1

Aplikací prvního pravidla získáme z Hilbertovy křivky prvního řádu Hilbertovu křivku druhého řádu.

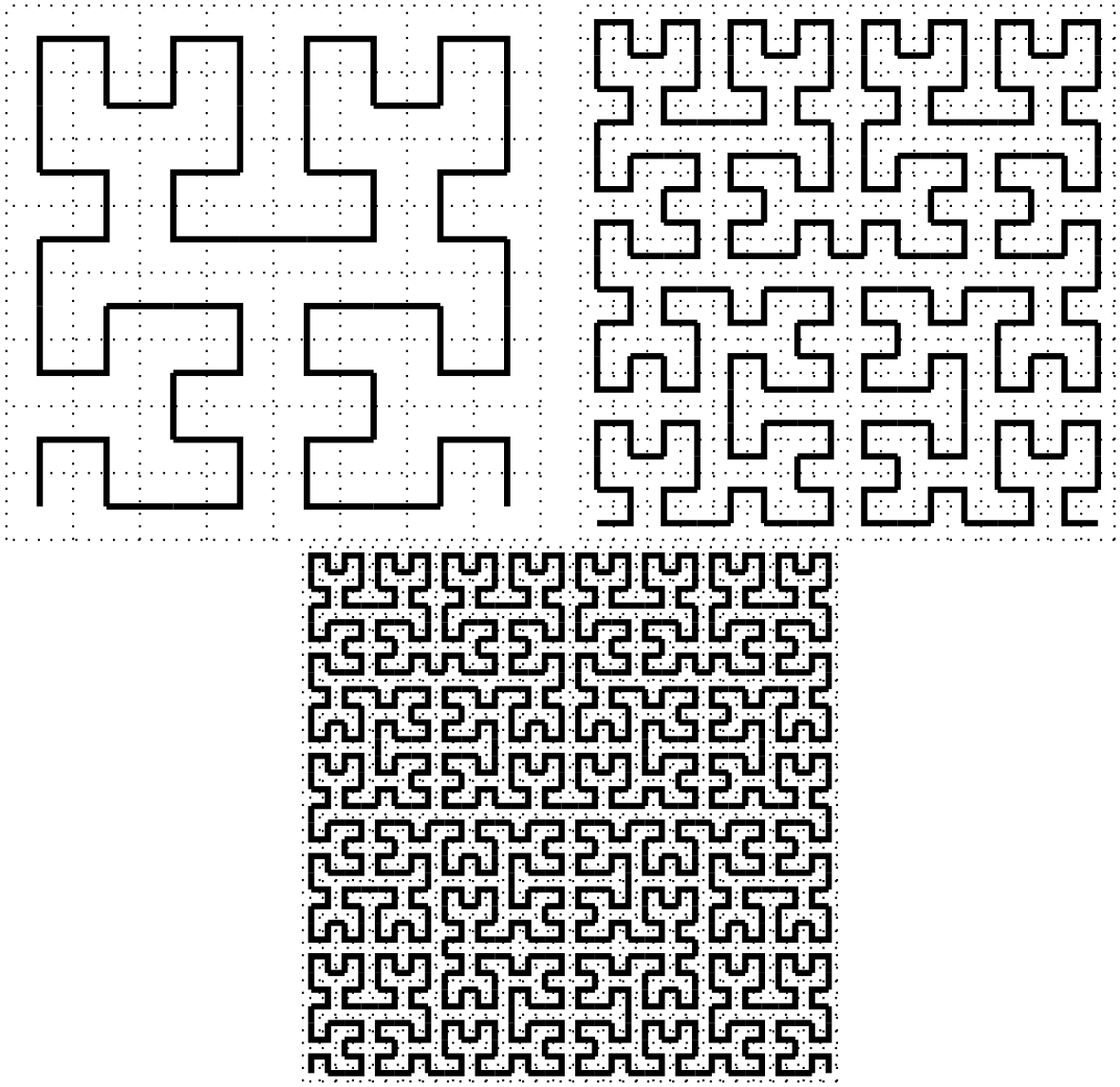
Všimněme si, že platí, že Hilbertova křivka řádu $n = 2, 3, \ldots$

- se skládá ze čtyř Hilbertových křivek řádu $n - 1$ (patřičně převrácenými);
- proto taky mapuje prostor rozdělený do čtyřikrát více oblastí;
- že prochází původní oblasti (oblasti křivky řádu $n - 1$;) ve stejném pořadí, jako Hilbertova křivka řádu $n - 1$.

Hilbertova křivka řádu 2



Hilbertovy křivky řádů 3, 4 a 5.



Zpět k Hilbertovým R-stromům: vkládání

- U Hilbertových R-stromů uvažujeme, že známe hranice oblasti, ve které se budou vkládaná data (případně jejich těžiště) vyskytovat.
- Tuto oblast proložíme dostatečně jemnou $2^n \times 2^n$ mřížkou a oblasti této mřížky namapujeme na Hilbertovu křivku řádu n.

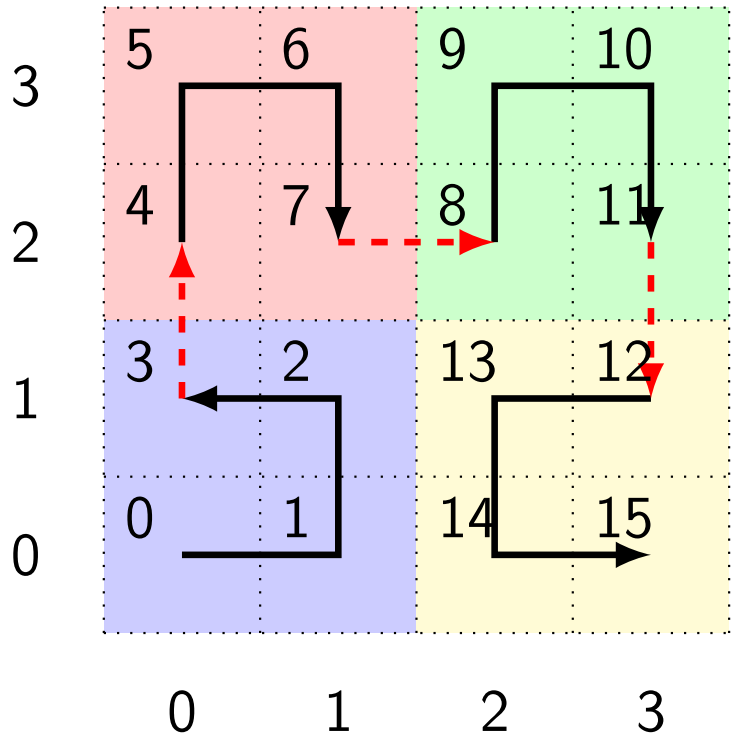
Při vkládání dat do Hilbertova R-stromu:

- Vypočteme H-hodnotu – pozici oblasti, do které padne těžiště, na Hilbertově křivce.
- Poté vložíme data stejně, jako bychom vkládali do B^+-stromu (uzly tedy oproti R-stromu musí navíc obsahovat H-hodnoty v nelistových uzlech).
- Upravíme obdélníky steně, jako u R-stromu.

Budeme potřebovat algoritmus, který pro argumenty

- souřadnice (x, y) oblasti,
- řád $n \in \mathbf{N}$ Hilbertovy křivky,

vrací H-hodnotu. Tedy například, pro $(x, y) = (2, 1); n = 2$ vrací 13 (viz obr. ??).



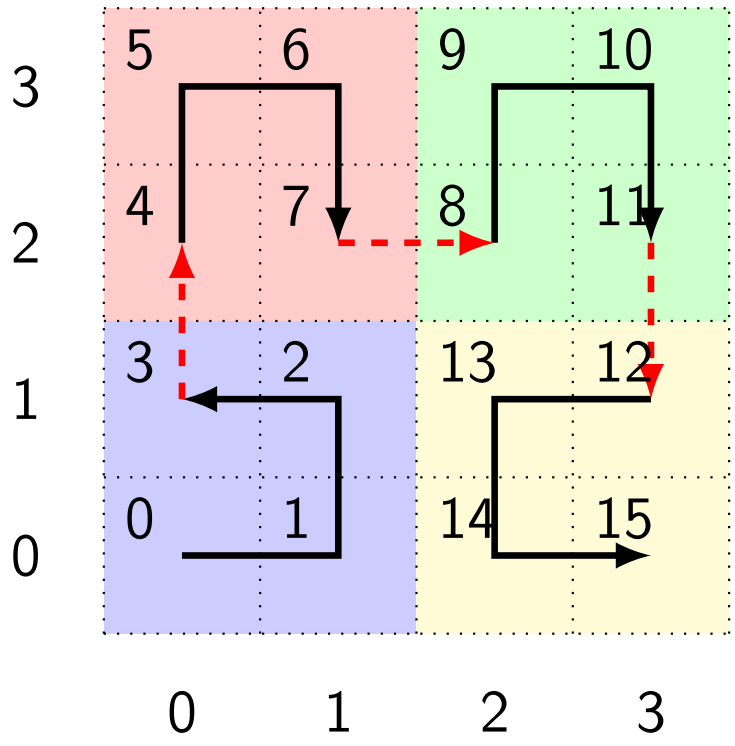
Tento algoritmus bude mít rekurzivní povahu, a bude asi takto:

- Pokud $n = 1$, zjistíme H-hodnotu přímo ze souřadnic (x, y):

$$(0,0) \mapsto 0 \qquad (0,1) \mapsto 1 \qquad (1,1) \mapsto 2 \qquad (1,0) \mapsto 3 \tag{1}$$

- Jinak z (x, y) zjistíme kvadrant, a rekurzivně řešíme pro $n - 1$ a upravené (x, y).

Např. pro $(x, y) = (2, 1); n = 2$, vidíme, že se nacházíme ve čtvrtém žlutém kvadrantu, to znamená, že výsledku rekurzivního volání přičteme $3 \cdot (2^{n-1} \cdot 2^{n-1}) = 12$ za tři kvadranty, které mu předchází.



To rekurzivní volání, provádíme pro Hilbertovu křivku, která je v tom žlutém kvadrantu. Musíme upravit (x, y):[1]

$$x \leftarrow 2^{n-1} - 1 - y \qquad y \leftarrow 2^n - 1 - x$$
$$= 2 - 1 - 1 = 0 \qquad = 3 - 1 - 2 = 1$$

Dle (1) tedy dostáváme 1. V součtu s tou 12 dostáváme očekávaných 14.

Poznámka: Případně můžeme proceduru ještě víc zjednodušit, když budeme uvažovat i $n = 0$, kdy křivka pokrývá jen jednu oblast, a vrací vždy 0.

Tento algoritmus domyslete a zpracujte jako domácí úkol.

[1] Toto bude vypadat pro každý kvadrant jinak.

Přeplnění uzlu

- Díky organizaci přes H-hodnoty může Hilbertův R-strom využívat stejné finty jako B^*-stromy.
- Tj. nelámat hned, jakmile dojde k přeplnění uzlu, ale vložíme do sousedního uzlu.
- Pokud je i sousední uzel plný, lámeme přeplněný uzel i jeho plného souseda na tři uzly.

Vyhledávání v Hilbertově stromu

- Vyhledávání probíhá stejným způsobem jako v R-stromech.
- H-hodnoty jsou tedy ignorovány a používají se pouze MBR stejným způsobem jako v R-stromech.