Datové štruktúry pre viacdimenzionálne vyhľadávanie

Ondrej Škopek
Matematicko-fyzikálna fakulta
Univerzita Karlova v Prahe
oskopek@matfyz.cz

29. mája 2015

Abstrakt

V dnešnej dobe „veľkých“ dát nadobúda viacdimenzionálne vyhľadávanie čoraz väčší praktický význam. V tejto práci uvádzame stručný prehľad a porovnanie existujúcich datových štruktúr vhodných na intervalové a podobnostné vyhľadávanie (anglicky Range Search, resp. Nearest Neighbor Search) v kontexte viacdimenzionálnych údajov.

1 Prehľad

1.1 Intervalové vyhľadávanie

Intervalové vyhľadávanie, anglicky Range Search, je formalizácia problému vyhľadávania všetkých bodov priestoru, ktoré patria danému podpriestoru, určenom intervalovým objektom. Intervalovým objektom môže byť ľubovolný objekt vymedzujúci nejaký „objem“ v k-dimenzionálnom priestore, typicky obdĺžniky, polpriestory alebo gule. Formálne zavedieme problém následovne [1]:

> Buď M množina bodov priestoru $\mathcal{P}$ a R intervalový objekt (vstup operácie Search).
> Nájdi všetky body z M také, ktoré patria intervalovému objektu R:
> $X' = \{x \in M : x \in R\}$

V rámci vyhľadávania bodov z priestoru vymedzeného daným intervalovým objektom sa dajú zadefinovať rôzne iné podotázky:

- optimalizačné – body so špeciálnou vlastnosťou (napr. maximalizácia prvej súradnice)
- počítacie – počet takýchto bodov
- prázdnosť – existuje taký bod?

Vzhľadom na konkrétnu podotázku sa väčšinou dá použiť jednoduchšia/rýchlejšia datová štruktúra.

1.2 Podobnostné vyhľadávanie

Podobnostné vyhľadávanie, anglicky Nearest Neighbor Search (NNS), je formalizácia optimalizačného problému nájdenia najbližších (alebo najpodobnejších) bodov k zadanému bodu. Formálne sa problém zavádza následovne [18]:

> Buď M množina bodov v priestore $\mathcal{P}$ a vstupný bod $x \in \mathcal{P}$.
> Buď $\sim$ podobnostná funkcia $\sim\colon \mathcal{P}^2 \to \mathbb{R}$.
> Nájdi najbližší bod $x' \in M$ ku x, teda:

$$x' = \min_{y \in M} \{\sim (x, y)\}$$

Funkcia $\sim (x, y)$, vyjadruje teda nepriamu mieru „podobnosti“ (čím podobnejšie body, tým menšia funkcia). Zovšeobecnenie tohto problému je hľadanie k-najpodobnejších bodov (k-nearest neighbor search). Cieľom tohto problému je nájdenie množiny bodov veľkosti k, ktoré sú z danej množiny M najbližšie ku x.

Existuje viacero algoritmov, ktoré tento problém riešia: lineárne prehľadávanie, rôzne metódy rozdeľovania na podpriestory, atď. My sa budeme zaoberať vlastnosťami datových štruktúr určených na zrýchlenie opakovaných použití takýchto algoritmov.

2 Datové štrukúry

V následujúcich sekciach stručne zavedieme najčastejšie používané datové štruktúry a zhrnieme ich výhody a nevýhody.

2.1 k-d strom

k-d strom (k-dimenzionálny strom) (Bentley [4]) je binárny strom, v ktorom každý list je bod v priestore. Každý interný vrchol reprezentuje nadrovinu, ktorá rozdeľuje priestor na dve časti. Body vľavo od nadroviny sú v ľavom podstrome, body vpravo v pravom. Rovina sa volí následovne: Zvolíme jednu z osí a hodnotu na nej. Tieto dva údaje asociujeme s daným interným vrcholom. Reprezentovaná nadrovina je kolmá na túto os, prechádzajúca danou hodnotou na osi, teda jednoznačne určená.

Z konštrukcie vyplýva, že operácie Insert a Delete majú zložitosť $O(\log n)$. Pamäťová zložitosť je triviálne $O(n)$. Liberty [20] rozoberá detailne zložitosť operácie Search s ohľadom na dimenzionalitu dát v k-d stromoch.

Z jeho výsledkov vyplýva, že použitie k-d stromov na podobnostné vyhľadávanie je vo veľkých dimenziach neoptimálne: počet bodov uložených v strome n by rozhodne mal byť $n \gg 2^k$, inak je zložitosť asymptoticky rovnaká ako hľadanie hrubou silou [17] (lineárna), respektíve, zložitosť je exponenciálna vzhľadom na počet dimenzií.

Ďalšia nevýhoda k-d stromov spočíva v tom, že na udržiavanie vlastnosti logaritmického vyhľadávania potrebujeme až $O(n \log n)$ operácií [4], teda asymptoticky rovnako ako keby sme strom vyvažovali vždy odznova.

Pre porovnanie, intervalové vyhľadávanie v k-d stromoch má najhoršiu zložitosť $O(kn^{1-\frac{1}{k}})$, teda polynomiálnu vzhľadom k počtu dimenzií [19]. k-d stromy sú preto veľmi výhodné na použitie pri intervalovom vyhľadávaní.

2.2 Range strom

Range stromy zaviedol Bentley [5] ako alternatívu ku k-d stromom určeným na intervalové vyhľadávanie s lepšou časovou zložitosťou na vyhľadávanie, ale horšou pamäťovou zložitosťou. Pre porovnanie, v Range stromoch trvá SEARCH $O(\log^d n + k)$ s pamäťou $O(n \log^{d-1} n)$, kde d je dimenzia daných dát.

Chazelle neskôr upravil pamäťovú zložitosť na $O\left(\left(\frac{\log n}{\log \log n}\right)^{d-1}\right)$[7] a časovú zložitosť vyhľadávania na $O(\log^{(d-1+\varepsilon)} n)$[8] pre pevné $\varepsilon > 0$.

2.3 R-strom

R-strom (rectangle tree, obdĺžnikový strom), ktorý naopak urýchľuje podobnostné vyhľadávanie zaviedol vo svojej práci Guttman [15]. Kľučovou myšlienkou je združovanie bodov do skupín a ich následná reprezentácia najmenším obdĺžnikom, ktorý ich všetky obsahuje. V listoch sa teda nachádzajú degenerované obdĺžniky obsahujúce iba jeden bod. Celkovo je strom vyvážený, teda všetky listy sa nachádzajú v rovnakej hĺbke.

Vyhľadávanie je založené na princípe pretínania vyhľadávacej otázky s obdĺžnikom v danom vrchole stromu: Ak sa nepretne vyhľadávaný obdĺžnik s obdĺžnikom vo vrchole, nemôže sa pretať ani so žiadnym z bodov, ktoré reprezentuje.

R-stromy nezaručujú vhodnú zložitosť v najhoršom prípade, ale na reálnych dátach fungujú preukázatelne [16] dobre. Existuje varianta R-stromov, tzv. prioritné R-stromy, ktoré majú dokazateľne optimálnu zložitosť v najhoršom prípade, ale v praxi sa nepoužívajú kvôli zložitosti implementácie.

Vzhľadom k viacdimenzionálnemu podobnostnému vyhľadávaniu, ako ukázal Brinkhoff et al. [6], k najpodobnejších bodov sa dá nájsť v R-stromoch rýchlo, pomocou tzv. spatial joins, teda „zjednotení v priestore“ (naprieč dimenziami), ktoré vedia R-stromy robiť dostatočne rýchlo. V tomto ohľade dokáže rýchlosť ešte viac vylepšiť následujúca podobná datová štruktúra, R^*-strom.

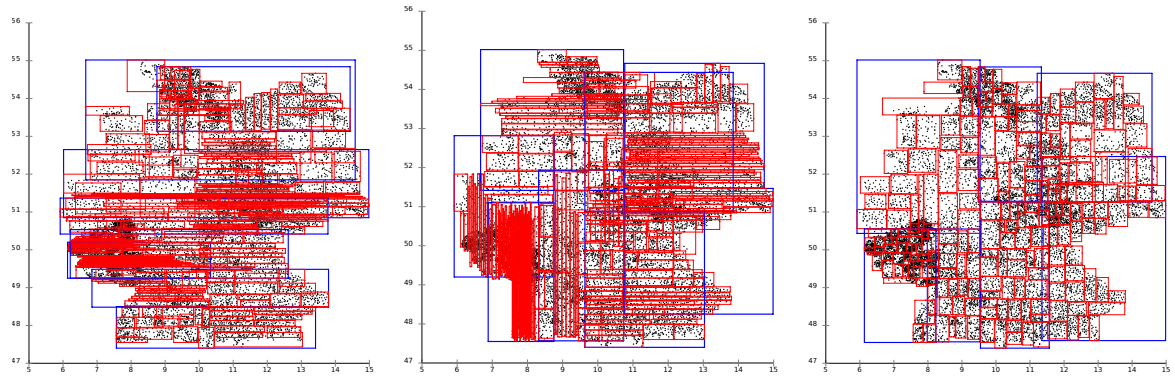
2.4 R^*-strom

Ako variantu R-stromov s lepšou zložitosťou vyhľadávania za cenu pomalšej výstavby ho zaviedli Beckmann et al. [3].

Skúmali vplyv rozdeľovania a pokrytia dát obdĺžnikmi v R-strome na rýchlosť následného vyhľadávania. Pomocou rôznych techník zmenšovania prekryvu jednotlivých obdĺžnikov vo vrcholoch zlepšili konštanty zložitosti operácie SEARCH v priemernom prípade (najhorší prípad je identický s R-stromom) na polovičku (približne, pre detailny rozbor viď záver práce originálnych autorov Beckmann et al. [3]).

Podľa pozorovania, ktoré hovorí, že štruktúra R-stromu veľmi závisí od poradia vkladania bodov zmenili prístup k pretečeniu vrcholu: opakovane vkladajú okrajové obdĺžniky z pretečeného obdĺžniku. Tento postup vedie postupne k optimálnejšiemu stromu (body sú viac združené do skupín, teda sa dajú pokryť menším počtom obdĺžnikov) a vznikajú pravidelnejšie obdĺžniky. Nevýhodou je, že opakovaným vkladaním sa zvyšujú časové nároky na údržbu stromu.

Rôzne rozdelenia na obdĺžniky sa dajú veľmi dobre vizualizovať v nižších dimenziach. Na obrázku 1 vidíme rôzne stratégie pre umiestnovanie bodov do obdĺžnikov vo vrcholoch R-stromu. Vidíme, že R^*-strom dokáže najlepšie zamedziť prekryvom, a preto je najvhod-



(a) Guttmanov R-strom [11] (b) Ang-Tan [2] a R-strom [9] (c) R^*-strom [10]

Obr. 1: Rozdelenia na obdĺžniky podľa rôznych heuristík na databáze nemeckých poštových smerovacích čísel

nejší pre praktické aplikácie, napriek vyššie spomenutým nevýhodám a implementačnej zložitosti.

2.5 R^+-strom

Autori Sellis et al. [21] ďalšej varianty R-stromu, tzv. R^+-stromu, úpravami znížili počet prístupov k súborovému systému pri vyhľadávaní súborov o 50%. Hlavný rozdiel oproti Guttmanovmu R-stromu je zníženie počtu prekryvov v interných vrcholoch stromu, čo má za následok o málo rýchlejšie vyhľadávanie pri veľkom objeme dát. Hlavná aplikácia tohoto prístupu je v databázach.

2.6 Pokrývací strom

Pokrývací strom (Cover tree) bol zavedený autorom Clarkson [13] špeciálne na zrýchlenie podobnostného vyhľadávania. Využíva predpokladu, že priestor $\mathcal{P}$ je metrický priestor $(\mathcal{P}, \text{dist})$ (pre definície viď sekciu 1.2 a Choudhary [12]). V listoch sa nachádzajú jednotlivé body priestoru. Každá hladina C stromu splňuje následujúce invarianty:

- Inkluzia: Hladina pod ňou je v inkluzií: $C_{i+1} \subseteq C_i$
- Pokrytie: $\forall p \in C_{i+1} : \exists q \in C_i : \text{dist}(p, q) \leq 2^i$ a práve jedno také q je „rodič“ p
- Oddelenie: Všetky body hladiny sú od seba ďaleko: $\forall p, q \in C_i : \text{dist}(p, q) > 2^i$

Prokývacie stromy majú lineárnu pamäťovú zložitosť, keďže štruktúra je uložená implicitne. Operácie Search a Insert majú obe logaritmickú zložitosť, avšak s rastúcou dimenzionalitou a priestorovou expanziou dát majú nezanedbatelné konštanty, preto pre viacdimenzionálne aplikácie nie sú cover stromy optimálne.

2.7 Intervalový/Segmentový strom

Veľmi dobrý popis intervalových a segmentových stromov dávajú autori de Berg et al. [14] na strane 220, resp. na strane 231. Tieto datové štruktúry sa sústreďujú hlavne na intervalové vyhľadávanie a sú si veľmi podobné.

Intervalový strom je jednoduchý binárny vyhľadávací strom, ktorý ako kľúče používa mediány daných intervalov. V listoch je uložený daný interval, prípadne zoznam jeho bodov.

Segmentový strom je zovšeobecnením intervalového, kde reprezentované úsečky nemusia byť paralelné k osiam priestoru a listy obsahujú projekciu intervalu na osi.

V jednej dimenzí majú intervalové stromy zložitosť operácie SEARCH $O(\log n + k)$ a pamäťovú zložitosť $O(n)$, kde k je počet vrátených bodov.

Segmentové stromy v jednej dimenzií majú zložitosť operácie SEARCH $O(\log n + k)$ a pamäťovú zložitosť $O(n \log n)$. Vo vyšších dimenziach je to kvôli udržiavaniu d rôznych úsečiek na každej osi $O(\log^d n + k)$ s pamäťou $O(n \log^d n)$.

V praxi je problém udržiavať vyváženosť, resp. vlastnosti a invarianty týchto datových štruktúr popri operáciach INSERT a DELETE.

3 Zhrnutie

Stručne sme opísali a zhrnuli výhody a nevýhody siedmich rôznych datových štruktúr používaných na vyhľadávanie vo viacdimenzionálnych údajoch. Implementačné a iné detaily sú veľmi dobre a do hĺbky rozobraté v článkoch ich originálnych autorov (viď literatúra a odkazy z príslušných sekcií).

Ako najvhodnejšia datová štruktúra pre intervalové vyhľadávanie z porovnania vzišly Range stromy 2.2 s úpravami od Chazella. Na podobnostné vyhľadávanie sa najviac hodia R-stromy, konkrétne s úpravou od Beckmanna, teda R^*-stromy 2.4. Pre menšie dimenzie je najvhodnejšia klasická volba, k-d strom od Bentleyho 2.1, hlavne kvôli svojej jednoduchosti.

Rýchle viacdimenzionálne vyhľadávanie ostáva napriek využitiu týchto datových štruktúr stále nevyriešeným problémom, hlavne z dôvodu výskytu rôznych nežiadúcich fenoménov vo vyšších dimenziach, ako je napríklad Curse of dimensionality[1], a iné.

Literatúra

[1] Pankaj K. Agarwal. Range Searching. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, 1997. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.8179`.

[2] Chuan-Heng Ang and T. C. Tan. New linear node splitting algorithm for R-trees. In *Advances in Spatial Databases*, pages 337–349. Springer, 1997. URL `http://link.springer.com/chapter/10.1007/3-540-63238-7_38`.

[3] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, SIGMOD '90, pages 322–331, New York, NY, USA, 1990. ACM. ISBN 0-89791-365-5. doi: 10.1145/93597.98741. URL `http://doi.acm.org/10.1145/93597.98741`.

[4] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM*, 18(9):509–517, September 1975. ISSN 0001-0782. doi: 10.1145/361002.361007. URL `http://doi.acm.org/10.1145/361002.361007`.

[1] `https://en.wikipedia.org/wiki/Curse_of_dimensionality`

[5] Jon Louis Bentley. Decomposable searching problems. *Information Processing Letters*, 8(5):244–251, 1979. ISSN 0020-0190. doi: 10.1016/0020-0190(79)90117-0. URL http://www.sciencedirect.com/science/article/pii/0020019079901170.

[6] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient Processing of Spatial Joins Using R-Trees. In Peter Buneman and Sushil Jajodia, editors, *SIGMOD Conference*, pages 237–246. ACM Press, 1993. URL http://dl.acm.org/citation.cfm?doid=170035.170075.

[7] Bernard Chazelle. Lower Bounds for Orthogonal Range Searching: I. The Reporting Case. *Journal of the ACM (JACM)*, 37(2):200–212, 1990. URL http://www.cs.princeton.edu/~chazelle/pubs/LBOrthoRangeSearchReporting.pdf.

[8] Bernard Chazelle. Lower Bounds for Orthogonal Range Searching: II. The Arithmetic Model. *Journal of the ACM (JACM)*, 37(3):439–463, 1990. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.3411&rep=rep1&type=pdf.

[9] Chire. R-tree on ZIP codes for Germany, constructed using the linear split heuristic of Ang and Tan. The tree has many long and narrow pages, which is suboptimal for typical window queries in map data, October 2011. URL https://en.wikipedia.org/wiki/File:Zipcodes-Germany-AngTanSplit.svg.

[10] Chire. R*-tree on ZIP codes for Germany, constructed using the R*-Tree heuristics of Prof. Kriegel. The pages overlap only very little and have a good spatial extend for window queries as used in map applications, October 2011. URL https://en.wikipedia.org/wiki/File:Zipcodes-Germany-RStarTree.svg.

[11] Chire. R-tree on ZIP codes for Germany, constructed using the quadratic split heuristic of Guttman. The tree pages overlap a lot, which is not beneficial for the performance, October 2011. URL https://en.wikipedia.org/wiki/File:Zipcodes-Germany-GuttmanRTree.svg.

[12] B Choudhary. *The Elements of Complex Analysis*. New Age International, 1992. URL http://goo.gl/C05fUI.

[13] Kenneth L. Clarkson. Nearest-Neighbor Searching and Metric Space Dimensions. In Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, 2006. ISBN 0-262-19547-X.

[14] Mark de Berg, Otfried Cheong, Marc Kreveld, and Mark Overmars. *More Geometric Data Structures*, chapter 10, pages 219–241. Springer, 2nd edition, 2008. ISBN 978-3-540-77974-2. doi: 10.1007/978-3-540-77974-2_10. URL http://dx.doi.org/10.1007/978-3-540-77974-2_10.

[15] Antonin Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In *Internation Conference on Managment of Data*, pages 47–57. ACM, 1984. URL http://www-db.deis.unibo.it/courses/SI-LS/papers/Gut84.pdf.

[16] Sangyong Hwang, Keunjoo Kwon, Sang K. Cha, and Byung S. Lee. *Performance Evaluation of Main-Memory R-tree Variants*. Springer, 2003. ISBN 978-3-540-45072-6. doi: 10.1007/978-3-540-45072-6_2. URL http://dx.doi.org/10.1007/978-3-540-45072-6_2.

[17] Piotr Indyk. Nearest Neighbors In High-Dimensional Spaces. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*. CRC Press LLC, Boca Raton, FL, 2nd edition, April 2004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.3826.

[18] Donald Ervin Knuth. *Sorting and searching*, volume 3 of *The Art of Computer Programming*. 1973.

[19] D. T. Lee and C. K. Wong. Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. *Acta Informatica*, 9(1):23, 1977. ISSN 1432-0525. doi: 10.1007/BF00263763. URL http://dx.doi.org/10.1007/BF00263763.

[20] Edo Liberty. Nearest Neighbor Search and the Curse of Dimensionality. In *Algorithms in Data Mining*. 2013. URL http://www.cs.yale.edu/homes/el327/datamining2013aFiles/11_nearest_neighbor_search.pdf.

[21] Timos Sellis, Nick Roussopoulos, and Christos Faloutsos. The R+-Tree: A Dynamic Index For Multi-Dimensional Objects. pages 507–518, 1987. URL http://repository.cmu.edu/cgi/viewcontent.cgi?article=1563&context=compsci.