Андреј Ивашковић

Геометријски алгоритми и структуре података

НЕДЕЉА ИНФОРМАТИКЕ У МАТЕМАТИЧКОЈ ГИМНАЗИЈИ

kd стабло

Претпоставимо да нам је дат скуп тачака у простору. Тада постоји једна класа геометријских упита који су сличног типа и са великим бројем примена:

- Одређивање тачке из овог скупа која је најближа некој датој — ***nearest neighbour*** (NN) упит. Основна примена се јавља у организацији служби са хитним реакцијама: ватрогасна служба, полиција, хитна помоћ. Може да се користи и у савременим *online* апликацијама за одређивања најближег локала неког типа. Овај упит се јавља и у неким хеуристикама и оптимизационим проблемима (проблем трговачког путника).
- Одређивање највећег могућег правоугаоника који не садржи ниједну тачку. Ово може да буде корисно уколико се планира изградња неког објекта што је могуће веће површине.
- *Range searching* упити: одређивање броја тачака у некој области, неког екстремалног својства у области...

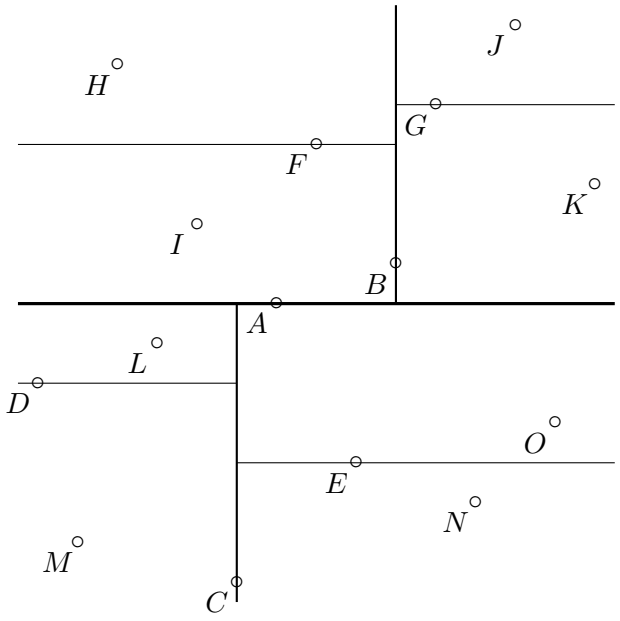
Од читаве ***spatial query*** класе упита (просторни упити), применљивим при раду са базама података у којима су посматрани објекти геометријске природе, посебно смо издвојили ове. Очекујемо да индексирање свих тачака преко низа неће дати задовољавајуће резултате по питању ефикасности, те нам је неопходан некакав **просторни индекс**. За овакве потребе користи се **kd стабло**[1].

Суштина структуре kd стабла се заснива на сегментацији простора: најпре се све тачке поделе у два мање-више једнако бројна подскупа које раздваја један услов: однос са неком координатом (све тачке чија је одговарајућа координата мања од те референтне налазе се у једном скупу, оне чија је већа налазе се у другом скупу, а у случају једнакости постоје разлике у зависности од приступа). Након тога се рекурентно врши подела ових подскупова на исти начин, али се сада врши раздвајање по некој другој координати (ако је прво раздвајање било по x координати, даље иде по y, па по z итд...када се дође до последње, поново се дели по x координати).

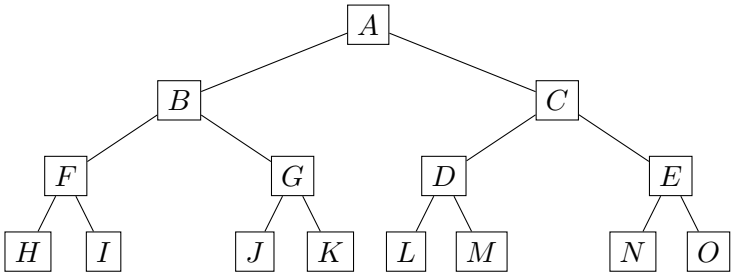
Реч је, дакле, о стаблу над k-димензионим подацима. У случају $k = 1$ се памте тачке које су у сортираном поретку и ово је онда бинарно стабло претраге. У случају $k = 2$ се оперише са једном равни, у случају $k = 3$ са простором итд.

Испоставиће се да је овако формирано стабло врло подесно за задатке наведеног типа. Да би се стабло креирало, неопходно је одредити ту границу која се повлачи ради поделе на два подскупа. Једноставно би било узети аритметичку средину, али је то незгодно у великом броју случајева јер тачке углавном нису «равномерно» распоређене (наравно, уколико тежимо прављењу балансираног стабла). Да би била извршена подела на два скупа приближно једнаке бројности, неопходно је одредити **медијану** (при томе ћемо игнорисати уобичајену дефиницију ако је број елемената паран: аритметичка средина два средња елемента).

[1] Користи се и запис **k-d** стабло у литератури. При пут када је уведено («Multidimensional binary search trees used for associative searching», Бентли, 1975.) се говорило о «уопштењу бинарног стабла претраге».



Слика 1: Подела простора на сегменте



Слика 2: kd стабло формирано над сегментисаним простором

Quickselect алгоритам

У наивном случају, медијана неког низа се одреди тако што се најпре изврши сортирање, па се узме средњи елемент. Ипак, овакав поступак захтева $O(n \log n)$ време, где је n дужина низа. Од осталих метода издвајамо, због једноставности, ***Quickselect*** алгоритам.

Quickselect има сличну идеју као *Quicksort* алгоритам сортирања. Посматрајмо низ A који има n елемената. Нека се тражи елемент са позицијом k у сортираном низу тј. k-ти најмањи. Најпре издвајамо неки елемент x (који зовемо **пивот**) и испитујемо који су елементи у A мањи, а који већи од њега (у случају једнакости је потпуно свеједно у коју групу га стављамо). Уколико преуредимо низ тако да су сви елементи који су мањи од x пре њега у низу, а они који су већи од x после њега, тада се x налази на истом месту у ком би се налазио у сортираном низу — означимо ту позицију са p. На основу овога можемо да закључимо да ли је он мањи, већи или баш једнак траженом k-том најмањем елементу, те раздвајамо случајеве:

1° уколико је баш $p = k$, тада је x решење;

2° уколико је $p > k$, тада се тражени елемент налази у делу низа закључно са елементом на позицији $p - 1$;

3° уколико је $p < k$, тада се тражени елемент налази у делу низа почевши од елемента на позицији $p + 1$.

У случају да k-ти најмањи елемент није одређен, понавља се поступак, с тим што се пивот бира из допустивог региона. Овај поступак се понавља (дакле, мењају се лева и десна граница у којој се тражени елемент налази) и сигурно ће се коректно завршити.

Посматрајмо рад овог алгоритма на примеру низа $(1, 6, 2, 9, 4, 7, 5)$, где тражимо четврти најмањи елемент:

$$
\begin{gathered}
(|1, \mathbf{6}, 4, 9, 2, 7, 5|), \\
(|1, 4, 2, 5, \mathbf{6}, 7, 9|), \\
(|1, \mathbf{4}, 2, 5|, 6, 7, 9), \\
(|1, 2, \mathbf{4}, 5|, 6, 7, 9), \\
\left(1, 2, 4, \left|\boxed{\mathbf{5}}\right|, 6, 7, 9\right).
\end{gathered}
$$

У најгорем случају, *Quickselect* ради у $O\left(n^2\right)$ времену (на пример, ако стално бирамо први елемент за пивот, а низ је већ сортиран). Ипак, очекивано и просечно време рада овог алгоритма је $O(n)$ јер је очекивани број итерација за налажење медијане врло мали (исти аргумент правда узимање да је временска сложеност *Quicksort* алгоритма дата са $O(n \log n)$). Предлаже се насумично бирање пивота да би се постигли овакви ефекти.

Принцип функционисања kd стабла

Ради праћења рада kd стабла неопходно је најпре набројати све операције које је над њим могуће вршити. За неке је оптимизовано (креирање, *NN search*), а за неке уопште није (убацивање новог елемента, брисање).

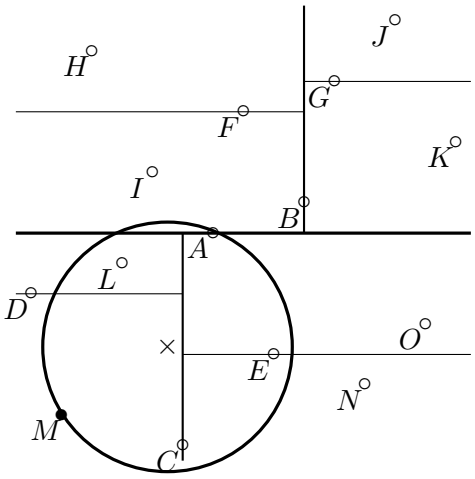
Најпре наведимо помоћне податке у структури стабла. Поред пуких показивача на децу и података о тачки (могуће имплементирати такође преко показивача), требало би памтити и извесне помоћне податке: све оне координате које говоре о ком се делу простора ради (пошто неки сегменти простора имају границе облика $\pm\infty$, постоје различити начини превазилажења овог проблема у зависности од конкретне ситуације: или се зада неки број који излази из предвиђених ограничења за улазне податке или се уводни помоћна променљива логичког типа).

Креирање. За креирање стабла се користи рекурзиван поступак. Реч је о креирању стабла над неким делом A, низа тачака. Уколико не постоји ниједна тачка у издвојеном делу низа A, реч је о *null* чвору. У супротном, користимо поменути *Quickselect* алгоритам, где се тражи $\lfloor n/2 \rfloor$-ти најмањи елемент. Тада се он придружује актуелном чвору и разматрају се деца која се добијају из низа који је модификован након примене *Quickselect*, врши се подела по одговарајућој хиперравни. Критеријум по ком се уреде елементи зависи од висине на којој се чвор налази (испитује се остатак при дељењу са бројем димензија).

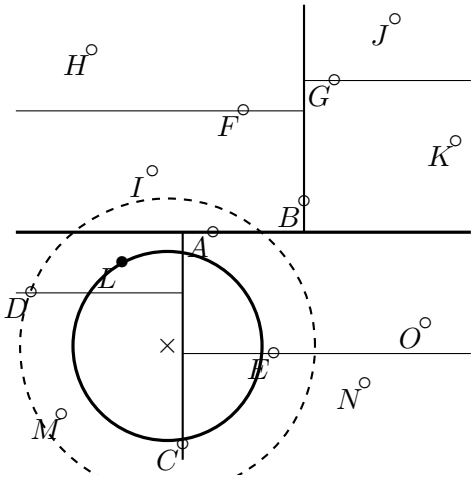
Сложеност креирања kd стабла је $O(n \log n)$ (реч је о очекиваној сложености!). Следи оправдање које се заснива на грубој процени (не и доказ!). Наиме, постоји око $n/2$ листова у стаблу и они су на висини $\log_2 n$. Да би

се стабло конструисало, било је неопходно применити *Quickselect* одређен број пута: једном за низ дужине n, два пута за низ дужине $n/2$, четири пута за низ дужине $n/4$ итд. Дакле, за сваку висину се врши $O(n)$ операција. Следи да је сложеност изградње стабла баш $O(n \log n)$. Интересантно је да овде k уопште не учествује изузев различитих k критеријума по којима се врши сортирање, па су ефекти клетве димензионалности овде минимални.

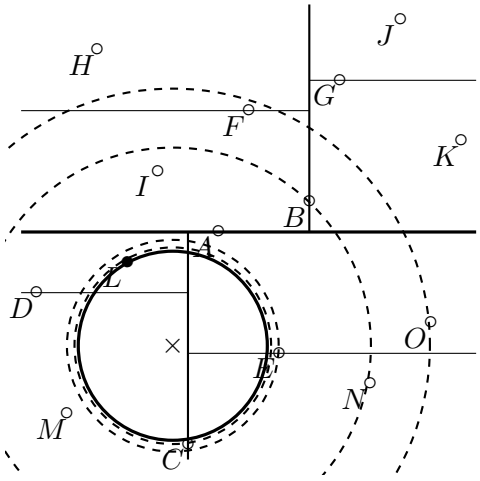
Одговор на NN упит. Најпре се установи у ком се тачно сегменту простора налази поменута тачка — претрага стабла која одређује одговарајући лист је једноставна (уз уобичајене проблеме у случају једнакости). Тачка уочена у том сегменту простора је сигурно један од кандидата и актуелни минимум, најближи сусед. Докле год постоји неки чвор такав да део простора који описује има удаљеност мању од актуелне минималне (тј. уколико одговарајућа кугла пресеца граничну хиперраван), тада је он кандидат за новог најближег суседа.



Слика 3: Први корак у раду претраге: тачка из упита се налази сегменту у ком је M, па је M при избор за најближег суседа



Слика 4: Други и трећи корак у раду претраге: круг сече сегмент у ком је D, али то је много веће од растојања до M; L је следећи избор и то је нови најближи сусед



Слика 5: Преостали кораци у претрази: кандидати су и C, E, N и O, али је на крају баш L тачка која је NN; остатак простора не долази у обзир при претрази

Асимптотска анализа оваквог поступка за одређивање најближег суседа је проблематична. Уколико су тачке насумично генерисане, тада се очекује $O(\log n)$ време. Фридман, Бентли и Финкел у раду из 1977. године доказују да је могућа претрага у логаритамском времену.

Ефекти клетве димензионалности често доводе до претраге много више чворова, чиме се сложеност увећава. У случајевима када је k упоредиво са n рад kd више нема толико предности над линеарном претрагом. Ипак, у пракси се овакви случајеви не разматрају, те узимамо да су ефекти клетве димензионалности минимални.

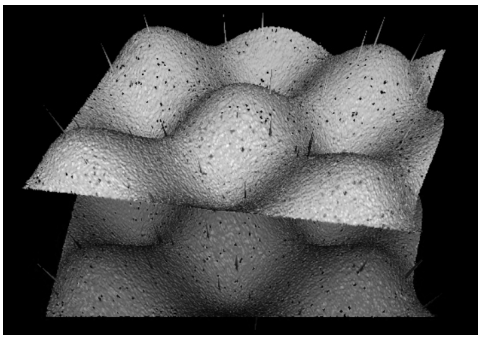
Додавање новог чвора. Убацивање новог чвора је велики проблем за бинарно стабло претраге јер нарушава балансираност, те није неочекивано да се то дешава и овде. Конкретни задаци у којима долази до примене kd стабла углавном «статични» (нема ни додавања ни брисања тачака). Пошто је проблем ребаланса иначе врло компликован, овде му неће бити посвећена нарочита пажња. Постоје нека **уопштења** која заобилазе овај проблем: адаптивно kd стабло, kd B стабло, псеудо kd стабло...

Брисање задатог чвора. Важи исти коментар као и за додавање чвора.

Примене

Најчешће примене kd стабала се заснивају на NN упиту. У складу са тим се јављају као предложено решење у великом броју проблема у којима се траже некакве хеуристике — нека NN спаривања се природно намећу у пракси. Размотрићемо ***iterative closest point*** (ICP) хеуристику, применљиву у различитим проблемима, као и веома актуелан ***ray tracing*** проблем.

Претпоставимо да су нам дата нека два скупа тачака (у 2d, 3d или неком другом простору), A и B, који су на неки начин корелисани: то су два скупа једнаке кардиналности, свака тачка из првог скупа одговара тачно једној тачки из другог и обратно. За скуп тачака се користи назив ***point cloud***. Дакле, постоји нека трансформација (тражимо бијекцију) која пресликава A у B — ова операција се зове ***point cloud stitch***. Посматрајмо мало конкретније:

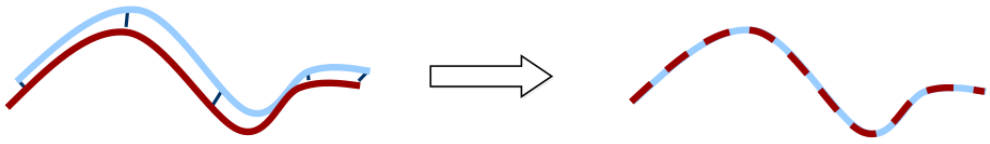


Слика 6: Која изометрија слика први рељеф у други?

реч је о неким истакнутим тачкама у простору у два различита временска тренутка које је посматрач уочио, при чему се он у међувремену кретао. Тада је неопходно одредити ону ригидну трансформацију (директну изометријску трансформацију) која што боље пресликава први скуп у други. Овде ћемо тачке приказивати у наредном облику: поред стандардних координата ћемо додати још једну «помоћну» која је увек једнака јединици. У случају 2d простора пишемо тачке у облику $\begin{pmatrix} x & y & 1 \end{pmatrix}^T$, а у случају 3d користимо **кватернионски**[2] **запис**: $\begin{pmatrix} x & y & z & 1 \end{pmatrix}^T$.

ICP је хеуристика која се користи да би се та трансформација τ пронашла и подсећа на познате итеративне нумеричке методе за решавање једначина (како оних облика $f(x) = 0$, где је f позната реална функција, тако и матричних, диференцијалних, система диференцијалних...). На тренутак оставимо по страни чињеницу да се тражи бијекција, већ за сваку тачку из A тражимо ону из B у коју желимо да се пресликa. За ово ћемо користити њој најближу тачку[3] из B (обично се користи стандардна Еуклидска метрика) — ово је корак који се ефикасно обави применом kd стабала и временска сложеност овог корака је $O(n \log n)$. Оправдајмо овај избор: уколико смо већ одредили тражену трансформацију, тада долази до мање-више поклапања свих тачака из $\tau(A)$ и B. Најпре дефинишемо неке критеријуме конвергенције (нпр. да је просечно растојање између упарених тачака довољно мало). Уколико радимо у 3d простору, трансформацији τ придружујемо матрицу која има следећи облик:

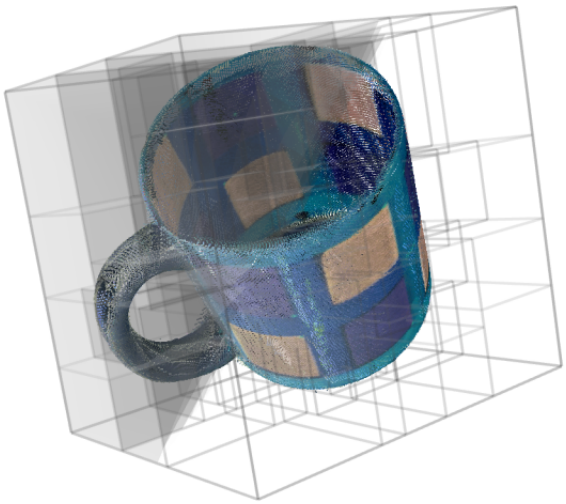
$$\mathbf{I} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$



Слика 7: Узима се најближа тачка из другог point cloud-а

[2] **Кватерниони** су проширење скупа комплексних бројева. Могу да се запишу у облику уређене четворке (a, b, c, d) или у алгебарском облику $a + bi + cj + dk$, при чему $i^2 = j^2 = k^2 = -1$, а правила сабирања и множења подсећају на правила при раду са комплексним бројевима.

[3] Ово је *point to point* варијанта, постоји и *point to plane* приступ који је ефикаснији.



Слика 8: За одређивање најближе тачке користе се kd стабла

Приметимо да матрица ове трансформације (помножена са кватернионом даје кватернион) садржи две подматрице: **T** и **R**. Подматрица **T** одговара транслацији за вектор $\begin{pmatrix} T_x & T_y & T_z \end{pmatrix}^T$. Подматрица **R** одговара некој ротацији у Еуклидском простору и добије се као производ неке три матрице ротације око различитих оса, односно:

$$\mathbf{R} = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{pmatrix}.$$

Важно је напоменути да τ, којој придружујемо **I**, представља трансформацију која се састоји од композиција једне транслације и једне ротације, при чему се прво примењује ротација, па онда транслација. Овако може да се представи свака ригидна трансформација.

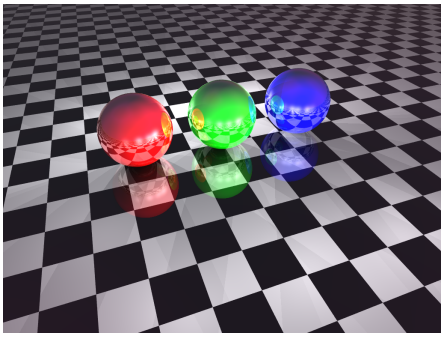
Након што се упаре тачке, неопходно је оценити матрицу која врши претпостављено пресликавање. За критеријум најбоље оцене може да се користи метод најмањих квадрата, али се некада, због потреба брзине, користе бржи и нешто непрецизнији критеријуми. При раду са 3d координатама је овај корак нешто компликованији, те се нећемо њиме бавити. У сваком случају, тако се добије нека оцењена трансформација τ_1 са матрицом $\mathbf{I_1}$. Уколико је задовољен критеријум конвергенције, тада је $\tau = \tau_1$. У супротном, поново се групишу тачке користећи исти критеријум (сада је најближа тачка можда нека друга!) и добије се трансформација τ_2, па се поступак можда опет понови и добије се τ_3, τ_4, ..., τ_n и ту се достигне конвергенција. Тада је $\tau = \tau_n \circ \ldots \tau_2 \circ \tau_1$, односно $\mathbf{I} = \mathbf{I_n} \cdot \ldots \cdot \mathbf{I_2} \cdot \mathbf{I_1}$. Ово је комплетан опис ICP алгоритма.

Постоји могућност да не дође до конвергенције. Да би се избегла бесконачна петља, обично се постави горња граница за n (у бесплатно доступној и обимној PCL библиотеци је *default* вредност једнака 50). Некада се, додуше, јави проблем тако што алгоритам не конвергира решењу, већ тек неком локалном минимуму (функције одступања). Практичне примене овог поступка се уочавају у проблемима који се тичу локализације посматрача (нпр. робота) или мапирању простора (може и оба: ***simultaneous localization and mapping*** (SLAM)). За ове потребе се користе специјализовани сензори који процењују удаљености објеката, праве «дубинске слике» (*grayscale* слика на којој интензитет сиве

одређује удаљеност), након чега се уоче одговарајуће значајне тачке у простору. ICP хеуристика се користи и у неким другим ситуацијама које захтевају поређење два *point cloud*-а (проблеми класификације и препознавања).

Слика 9: Слика у боји и њена дубинска варијанта

Ray tracing (праћење зрака) је алгоритам који се јавља у рачунарској графици и применљив је углавном уколико се ради о систему који не захтева превелику брзину или није интерактиван (*real time* системи, попут рачунарских игара, углавном не користе овако детаљне ефекте). Реч је о проблему генерисања реалистичног приказа полазећи од 3d модела, тачке у којој се налази посматрач и извора светлости — кључни проблем се тиче ефеката геометријске оптике попут сенки, одбијања (рефлексије) и преламања (рефракције) светлости.



Слика 10: Једна генерисана слика

Нећемо улазити у детаље овог алгоритма, али ћемо се осврнути на део у ком се посматра «први пресек» једног зрака (математички моделованог као полуправа) са објектима који се налазе на сцени. У оваквим ситуацијама се користи *surface area* хеуристика (SAH) и захтева коришћење некакве ефикасне структуре података за анализу. Данас ту улогу све чешће имају баш kd стабла јер имају асимптотски оптимално $O(n \log n)$ време, при чему је n број троуглова на које је простор подељен применом триангулације објеката.