OPPA European Social Fund
Prague & EU: We invest in your future.

Průnik rovnoběžníků

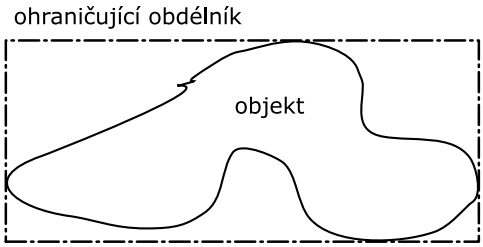
Tomáš Drtina*
FEL ČVUT Praha
Katedra Počítačů

1 Anotace

V tomto článku se budeme zabývat vztahem definovaným na množině rovnoběžníků, kterým je průnik rovnoběžníků. Uvedeme význam důvodu řešení tohoto problému, rozebereme různé metody řešení a jednu z nich popíšeme podrobně včetně její paměťové i operační složitosti.

2 Motivace

Často se v počítačové grafice setkáváme s problémem, kdy potřebujeme řešit kolize jednotlivých objektů ve scéně. Obvykle se tento problém řeší tak, že se jednotlivé objekty vloží do ohraničujícího obdélníku. Poté už stačí pouze detekovat, zda mají tyto obdélníky průnik, což nám v kladném případě značí přibližnou kolizi objektů. Ukázka takového obdélníku (Bounding Box) je na obrázku 1.



Obrázek 1: Objekt v ohraničujícím obdélníku.

3 Specifikace problému

Problém 3.1 *U zadané množiny N isotetických rovnoběžníků zjistěte všechny jejich páry průniků.*

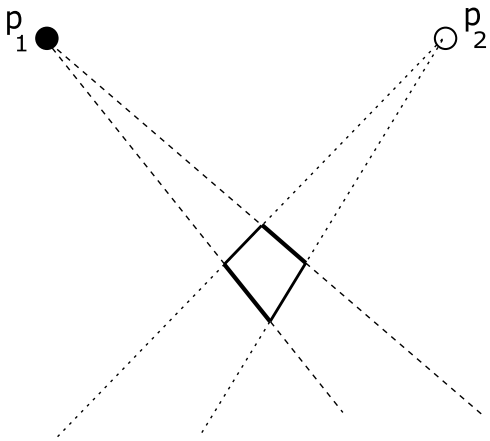
Rovnoběžník je isotetický, pokud jeho hrany leží na dvou soustavách přímek vycházejících ze dvou bodů jako na obrázku 2.

3.1 Definice průniku

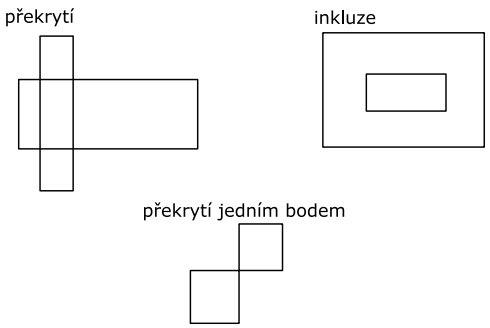
Dva isotetické rovnoběžníky mají průnik tehdy a jedině tehdy, pokud sdílejí alespoň jeden bod. Existují dvě možnosti průniků rovnoběžníků jaké mohou nastat. Buď inkluze, což znamená, že jeden z rovnoběžníků je celý uvnitř druhého nebo překrytí, kdy jeden z rovnoběžníků překrývá svou částí druhý. Tyto dva případy jsou zobrazeny na obrázku 3.

Nejjednodušší případ průniku nastává v jedné dimenzi, kde jsou „rovnoběžníky“ intervaly na přímce. V d-dimenzionálním prostoru vznikne rovnoběžník kartézským součinem na d intervalech, přičemž každý je na jiné souřadnicové ose. Dva $d-$dimenzionální

*e-mail: drtint1@fel.cvut.cz



Obrázek 2: Isotetický rovnoběžník



Obrázek 3: Dva typy průniků obdélníků. Vlevo překrytí, vpravo inkluze a uprostřed dva obdélníky překrývající se v jednom bodě, opět případ překrytí.

rovnoběžníky mají průnik tehdy a jedině tehdy jestliže se jejich x_j-zobrazení (dva intervaly) protínají, pro $j = 1,2,\ldots,d$. Vidíme, že jednorozměrný případ hraje zásadní roli a proto si jej rozebereme podrobněji. Mějme dva intervaly $R' = [x'_1,x'_2]$ a $R'' = [x''_1,x''_2]$. Podmínka $R' \cap R'' \neq \varnothing$ je ekvivalentní jedné z následujících vzájemně se vylučujících podmínek:

$$x'_1 \leq x''_1 \leq x'_2; \quad (1)$$
$$x''_1 < x'_1 \leq x''_2. \quad (2)$$

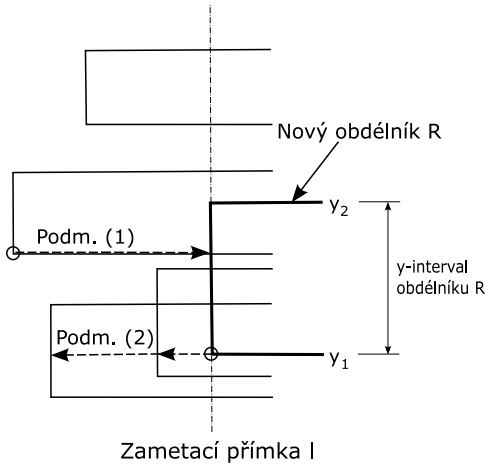
Čtyři možné konfigurace krajních intervalů odpovídající situaci $R' \cap R'' \neq \varnothing$ jsou zahrnuty v (1) a (2). Tudíž testujeme-li jestli R' a R'' mají průnik, zjišťujeme, jestli levý okraj R' zasahuje do R'' nebo levý okraj R'' zasahuje do R'.

4 Možná řešení

Dále budeme pro zjednodušení uvažovat pouze obdélníky. Problém můžeme řešit naivním způsobem tak, že budeme hledat průniky obdélníků každý s každým, což vede na operační složitost $O(N^2)$. Existuje ovšem lepší způsob uvedený v [Preparata and Shamos 1985], který vede na operační složitost $O(s+N\log N)$, kde s je počet nalezených párů průsečíků. Tento způsob využívá zametací techniky a intervalového stromu.

5 Zametací technika

Využijeme metodu zametací techniky, kde body události[1], budou úsečky vertikálních stran obdélníků. Stav zametací přímky je dán průsečíkem obdélníků se zametací přímkou. Obdélníky, které mají průsečík se zametací přímkou se nazývají *aktivní obdélníky*. Procházíme-li zametací přímkou zleva doprava a uvažujeme, že současný bod události je levá strana (nového) obdélníku R víme tedy, že obdélník R může mít průsečík se všemi právě *aktivními obdélníky*. Jediné co potřebujeme zjistit je, který z *aktivních obdélníků* splňuje podmínku (1) nebo (2) pro y-interval. Takovéto obdélníky, které splňují jednu z těchto podmínek, mají průnik s obdélníkem, na jehož levé straně se právě nachází zametací přímka. Průchod zametací přímky je naznačen na obrázku 4.



Obrázek 4: Situace která nastane, pokud zametací přímka protne levou stranu obdélníku R. Všechny zobrazené obdélníky jsou aktivní, protože je protíná zametací přímka. Zdroj [Preparata and Shamos 1985].

Toto se jeví jako dvojí prohledávání, jedno pro podmínku (1) a druhé pro podmínku (2). Společná povaha těchto dvou hledání by mohla podněcovat použití dvou různých datových struktur a tím i komplikovanější algoritmy. Vhodného řešení lze v tomto případě docílit použitím intervalového stromu.

6 Intervalový strom

Označme $[b^{(i)},e^{(i)}]$ jako y-interval obdélníku $R^{(i)}$, necháme seřadit sekvenci okrajů N y-intervalů v posloupnost $(y_1,y_2,\ldots,y_{2N})$.

[1] nebo též *postupový plán*, jedná se o body, ve kterých se mění stavy řešení

Intervalový strom bude mít kostru (též primární struktura) definovanou staticky pro danou množinu bodů (v našem případě sekvence $(y_1,y_2,\ldots,y_{2N})$) a bude moci skladovat libovolnou podmnožinu (*aktivní podmnožinu*) intervalů, jejíž okraje jsou v množině $(y_1,y_2,\ldots,y_{2N})$.

Intervalový strom T bude pro posloupnost $(y_1,y_2,\ldots,y_{2N})$ a množinu intervalů $I \subseteq \{[b^{(i)},t^{(i)}] : i=1,\ldots,N\}$ definován následovně:

1. Listy stromu bude tvořit posloupnost $(y_1,y_2,\ldots,y_{2N})$ nad níž bude postaven vyvážený binární strom (primární struktura). Hodnotu uzlu budeme označovat $H(v)$. Ohodnocení uzlů binárního stromu je provedeno tak, že hodnota levého syna uzlu je menší nebo rovna hodnotě uzlu otce. Hodnota uzlu pravého syna je větší než hodnota uzlu otce.
2. Každý primární uzel v intervalového stromu T obsahuje ukazatele na dva sekundární seznamy. Levý sekundární seznam $\mathscr{L}(v)$ a pravý sekundární seznam $\mathscr{R}(v)$. Levý sekundární seznam $\mathscr{L}(v)$ obsahuje vzestupně seřazenou posloupnost levých koncových bodů patřící do I. Pravý sekundární seznam $\mathscr{R}(v)$ obsahuje sestupně seřazenou posloupnost pravých koncových bodů patřících do I.
3. Každý primární uzel T je klasifikován jako „aktivní“ nebo „neaktivní“. Uzel je aktivní buď pokud jsou jeho sekundární listy neprázdné nebo obsahuje aktivní uzly v obou jeho podstromech.

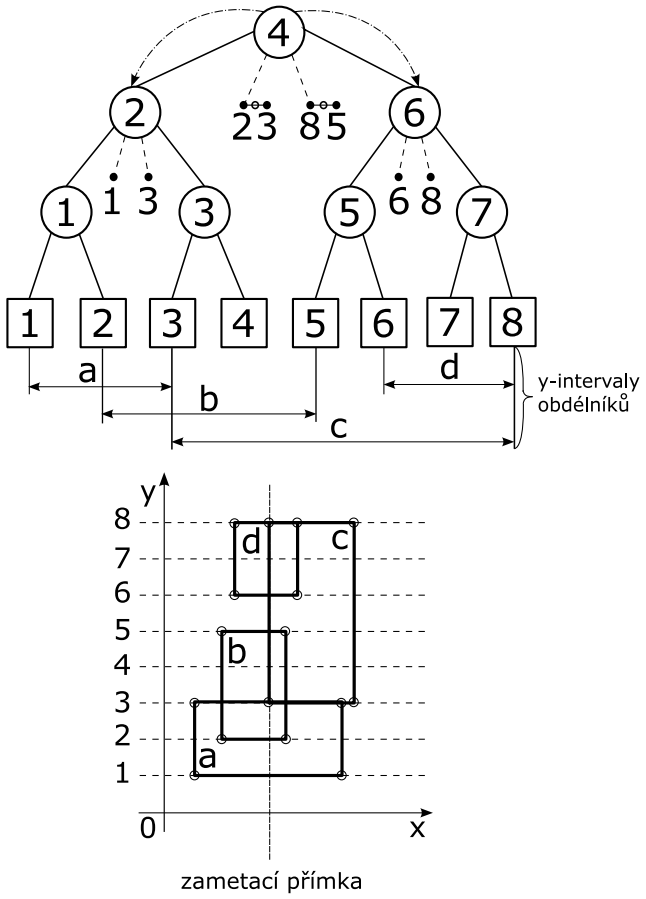
Intervalový strom bude tedy mít několik následujících zajímavých vlastností:

1. Při inorder průchodu primární strukturou stromu T, dostaneme z hodnot uzlů seřazenou posloupnost $(y_1,\ldots,y_{2N})$.
2. Sekundární seznamy $\mathscr{L}(v)$ a $\mathscr{R}(v)$, pro nelistový uzel v, musí podporovat vkládání a mazání. Jsou obyčejně realizovány jako stromy vyvažované do výšky nebo do šířky.
3. Aktivní uzly mohou být spojeny jako binární strom $\mathscr{T}$. Každý z uzlů leží na cestě vycházející z kořene stromu T. Tudíž každý primární uzel v má dva nové ukazatele LPRT (levý ukazatel) a RPTR (pravý ukazatel), které jsou použity při tvorbě stromu $\mathscr{T}$. Jestliže je v neaktivní, potom LPRT$[v] = 0$ a RPTP$[v] = 0$. Jestliže je v aktivní, potom LPRT$[v] \neq 0$ jedině v tom případě, že jsou v levém podstromu v aktivní uzly. Podobně RPRT$[v] \neq 0$ jedině v tom případě, jestliže jsou v pravém podstromu v aktivní uzly. Všimněme si, že více jak polovina aktivních uzlů má neprázdné sekundární seznamy.

Ukázku intervalového stromu vidíme na obrázku 5.

6.1 Vkládání a mazání v intervalovém stromu

Budeme uvažovat interval vkládání $[b,e]$. Při vkládání intervalu budeme postupovat takto: jdeme od kořene stromu T dokud nenarazíme na uzel v^* z T takový, že je splněna podmínka $b \leq H(v^*) \leq e$. V tomto uzlu je souřadnice b vložena do $\mathscr{L}(v^*)$ a souřadnice e do $\mathscr{R}(v^*)$. Pokud uzel nebyl do doby před vložením intervalu aktivní, stává se aktivním a je třeba aktualizovat LPTR či RPTR předchůdce. Analogicky může být implementováno mazaní. Následující pseudokód ukazuje vkládání intervalu $[b,e]$ do T:



Obrázek 5: Příklad intervalového stromu. V dolní části stromu jsou zobrazeny y-intervaly obdélníků. Nad nimi vidíme vzestupně seřazeny jejich okraje, které jsou umístěny v listech stromu. Primární struktura je znázorněna plnými čarami, sekundární seznamy čárkovanými čarami a strom $\mathscr{T}$, který je tvořen ukazateli na aktivní uzly, čerchovanými čarami. Na spodním obrázku vidíme rozložení obdélníků se zametací přímkou protínající levou stranu obdélníka c. Stav intervalového stromu odpovídá situci po vložení y-intervalu obdélníka c.

```
if (H(v) < b < e) {

      postupuj do pravého syna;

} else if (b < e < H(v)) {

      postupuj do levého syna;

} else {  // b = H(v) < e || b < H(v) = e

       vlož interval;
}
```

6.2 Hledání v intervalovém stromu

Ve stromě T máme najít intervalovou shodu s intervalem $[b,e]$. Intervalová shoda v tomto případě znamená průnik s jiným obdélníkem. Při prohledávání mohou nastat tři případy:

1. $b < H(v) < e$ - všechny intervaly umístěné v sekundárních seznamech listu v splňují intervalovou shodu s hledaným intervalem $[b,e]$ a tudíž ohlásíme průnik se všemi obdélníky, které odpovídají intervalům umístěným v sekundárních seznamech tohoto uzlu. Prohledávání dále pokračuje do pravého i levého podstromu, pokud obsahuje aktivní uzel.

2. $b = H(v) < e$ nebo $H(v) < b < e$ - v tomto případě postupně prohledáváme pravý sekundární seznam $\mathscr{R}$ uzlu v dokud je splněna podmínka, že prvky seznamu jsou větší nebo rovny b. Všechny tyto prvky opět značí průnik. Hledání dále pokračuje pouze v pravém podstromě, pokud obsahuje aktivní uzel.

3. $b < H(v) = e$ nebo $b < e < H(v)$ - totéž jako v předchozím případě, ale s tím rozdílem, že prohledáváme levý sekundární seznam $\mathscr{L}$ uzlu v a hledáme prvky, které jsou menší nebo rovny e. Hledání dále pokračuje v levém podstromě, pokud obsahuje aktivní uzel.

Následující pseudokód znázorňuje vyhledávání v intervalovém stromě T. Hledání iniciujeme `hledej(root)`.

```
hledej (UZEL* v) {

    if (b < H(v) < e) {

        i = 0;
        while (R(v).[i] >= b) {
            průnik; i++;
        }
        if (RPTR(v)) hledej(RPTR(v));
        if (LPTR(v)) hledej(LPTR(v));

    } else if (b = H(v) < e || H(v) < b < e) {

        i = 0;
        while (R(v).[i] >= b) {
            průnik; i++;
        }
        if (RPTR(v)) hledej(RPTR(v));

    } else {     // b < H(v) = e || b < e < H(v)

        i = 0;
        while (L(v).[i] <= e) {
            průnik; i++;
        }
        if (LPTR(v)) hledej(LPTR(v));

    }
}
```

7 Algoritmus vyhledání všech párů průsečíků na množině N isotetických rovnoběžníků

Nyní si popíšeme algoritmus, který řeší problém 3.1.

1. Vložíme do postupového plánu zametací přímky všechny svislé hrany všech N obdélníků. Hrany jsou v takovém pořadí, v jakém bychom na ně postupně naráželi při průchodu po ose x směrem zleva doprava.

2. Vezmeme všechny y-intervaly všech N obdélníků a seřadíme okraje těchto intervalů ve vzestupnou posloupnost.

3. Nad touto posloupností postavíme primární strukturu intervalového stromu, přičemž posloupnost bude tvořit listy tohoto stromu.

4. Procházíme zametací přímkou její postupový plán, tj. všechny svislé hrany obdélníků zleva doprava po ose x. Při tomto průchodu zametací přímkou mohou nastat dva případy:

   - Zametací přímka protíná levou svislou hranu některého obdélníku - v tomto případě se provádí v jednom kroku vyhledávání v intervalovém stromě, jak je popsáno v kapitole 6.2 a vkládání intervalu jak je popsáno v kapitole 6.1. Vyhledáváme a vkládáme interval, který odpovídá y-intervalu obdélníku, jehož levou hranu protíná zametací přímka. Co se týče pořadí, tak nejprve daný uzel prohledáme a až poté do jeho sekundárních seznamů interval vložíme, pokud splňuje podmínku pro vložení popsanou v 6.1.
   - Zametací přímka protíná pravou svislou hranu některého obdélníku - y-interval tohoto obdélníku je odebrán ze sekundárních seznamů uzlu ve kterém se nachází způsobem, který je popsán v kapitole 6.1.

Uvědomme si, že všechny obdélníky, které protíná zametací přímka jsou *aktivní obdélníky* a jejich y-intervaly jsou umístěny v sekundárních seznamech aktivních uzlů intervalového stromu a tedy prohledáváme pouze intervaly těchto právě *aktivních obdélníků*.

8 Paměťová a operační složitost algoritmu

Nyní analyzujeme složitost metody. Statický intervalový strom s N intervaly má paměťová složitost $O(N)$. Je zde $2N-1$ primárních uzlů a nejvíce $2N$ položek uložených v sekundárních seznamech. Kostra primární struktury je konstruovaná v čase $O(N \log N)$ s předběžným setříděním úseček. Každý interval je vkládán nebo vymazáván v čase $O(\log N)$. Prohledávání primární struktury stromu se děje v čase $O(\log N)$. Celková operační složitost tedy bude $O(N \log N + s)$ s $O(N \log N)$ časem na předzpracování, kde s znamená počet nalezených párů průsečíků.

Reference

KAPOOR, V., AND WOLFF, A. 1998. Reference manual for generic data structure for the rectangle intersection problem. http://www.jouy.inra.fr/unites/miaj/public/vigneron/cs4235/l5cs4235.pdf

PREPARATA, F. P., AND SHAMOS, M. I. 1985. *Computational Geometry*. Springer-Verlag, Berlin.

VIGNERON, A. Segment trees and interval trees. http://i11www.iti.uni-karlsruhe.de/map-labeling/design/old/interfaces/gen_rec_inters.ps

OPPA European Social Fund
Prague & EU: We invest in your future.