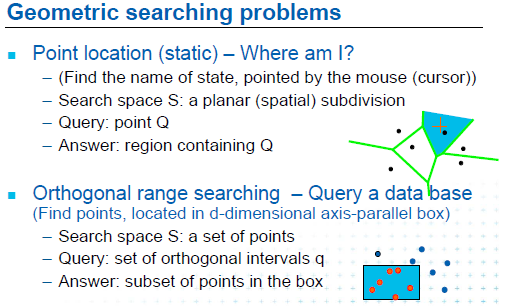
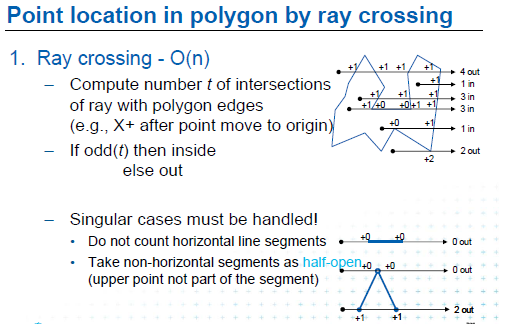
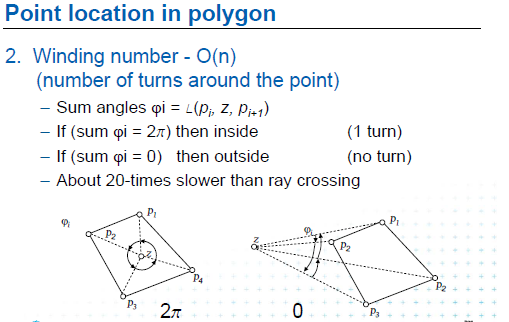
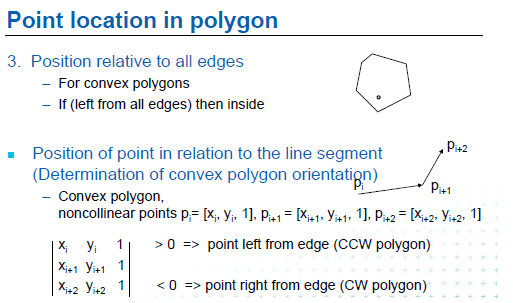
# 21. Test příslušnosti bodu k polygonu a k oblasti v planárním dělení (metoda pásů, strom monotónních řetězů). Reprezentace planárního dělení (DCEL), výpočet překrytí planárních dělení (průsečík, sjednocení, rozdíl) modifikovaným Plane-sweep algoritmem pro průsečíky množiny úseček.

1. **Test příslušnosti bodu k polygonu a k oblasti v planárním dělení.**

****

****

****

****

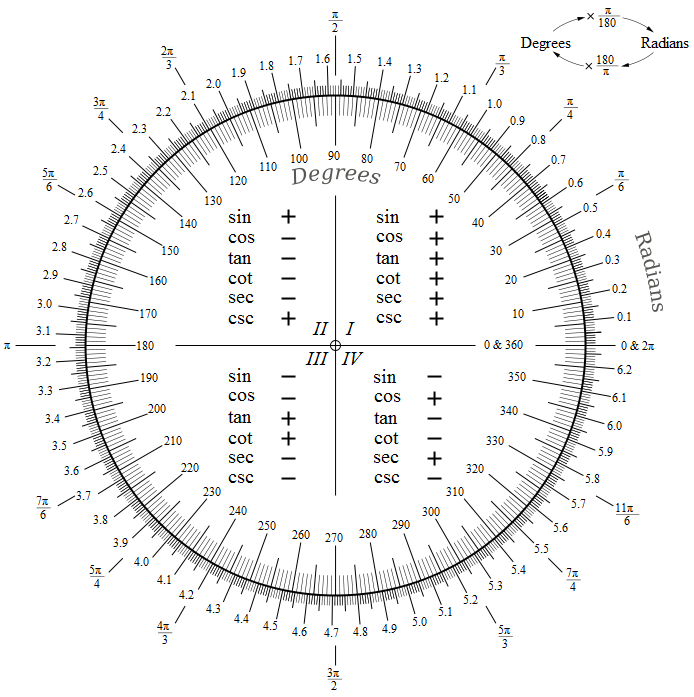
(6)

**Druhy mnohoúhelníků (polygonu)**

Kromě mnohoúhelníků lišících se počtem vrcholů , se mnohoúhelníky dělí na:

* **pravidelné** (všechny strany i vnitřní úhly jsou [shodné](http://cs.wikipedia.org/wiki/Shodnost)) a *nepravidelné*,
* **konvexní** (všechny vnitřní úhly jsou menší než 180°) a **nekonvexní, konkávní** (alespoň jeden vnitřní úhel je vetší než 180°),
* **pravoúhelníky** (všechny vnitřní úhly jsou pravé, příp. 270°`) a *nepravoúhelníky* (aspoň jeden vnitřní úhel se nerovná pravému úhlu).

|  |  |
| --- | --- |
|  |  |



hm :) staci vedet **rad = alfa\*pi/180,** alfa = uhel ve stupnich

u **ray crossing**, ta první metoda mají O(n)

a u **winding number** mají taky O(n) a mají tam, že je to 20x pomalejší ... než ray ... já vím, je to asymptotická složitost .. ale sejně na obrovských datech to musí jít znát...

tak plati asi u windowing neco jako **20\*n**

**noncollinear** znamena, ze **nelezi na primce**

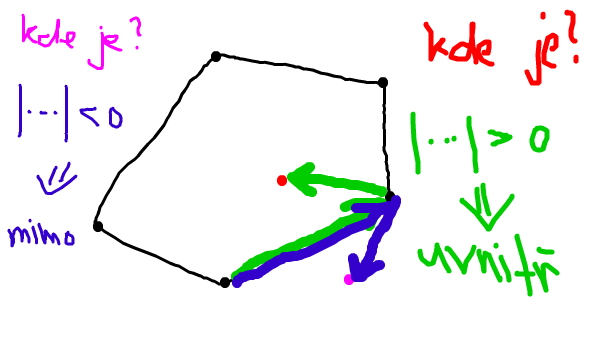
vzdycky vyberes jen 3, protoze to je **konvexni**

tak z toho, kam se to staci na kterou stranu lze urcit, jestli je bod uvnitr nebo ne

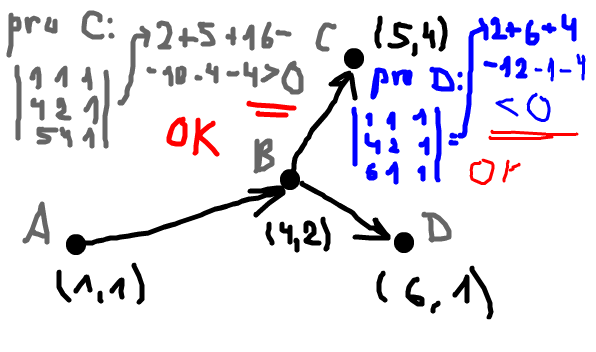
zkusme tam dosadit 3 body na primce, bude to nula? jo asi bude

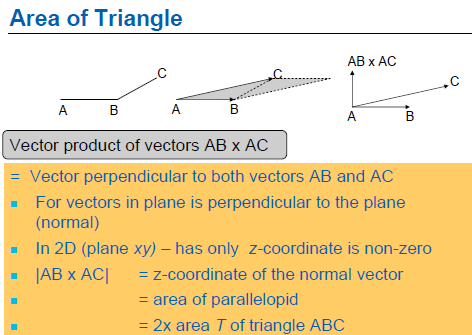
vyberes asi takove body (hrany), ktere jsou napravo od bodu

kdyz to otocis cele, tak v tom pripade si vyberes jine 3 body, aby byly napravo a asi by to platilo a na vzdalenosti nezalezi



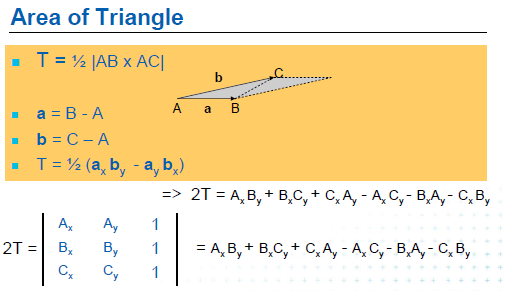
**Výpočet**



****

(7)

Tak 7 je jasna, vektor. soucin je **kolmy vektor, zajimave je, ze jeho velikost je dvojnasobek obsahu trojuhelnika ABC.**

****

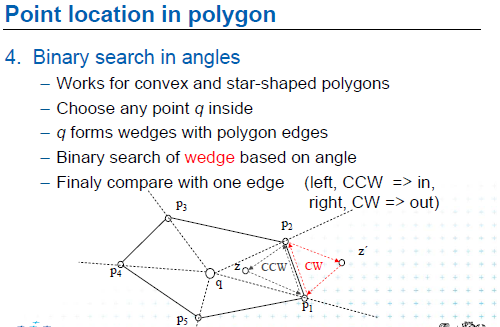
**(8)**

muzeme tedy z **vektr. soucinu urcit plochu trojuhelnika**, co ten deter. tam? asi je to jenom jinak zapsane s determinantem, rekl bych, ze staci pamatovat to v zlutem

a to **2T je tedy už plocha celého obdélníku**, tedy 2 trojuhelniku, ju? jo, a nemusi byt pravouhle

na 7 písou perpendicular ... to je ten novy vektor AB x AC, ale AB nemusi byt perpendicular na AC

jj, ta **co vysla je kolma na prvni vektor i na druhy**, ale ty dva vektory navzajem mezi sebou byt kolme nemusi

****

**(9)**

mame nekde **bod q,** udelame ty trojuhelniky mezi vrcholy

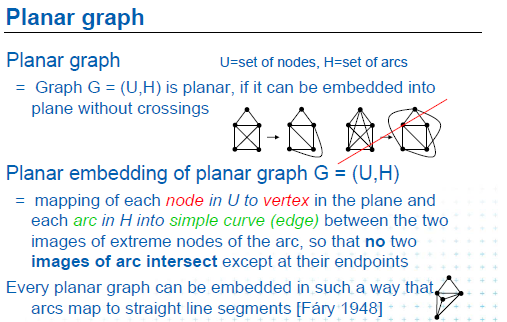
**wedge angle** - úhel klínu; úhel břitu; úhel hřbetního břitu

takze bod q mame asi vzdy uvnitr, to tam píšou ... jj, ten nam rozdeluje polygon na trojuhelniky, jj

pak testujeme bod z a pokud CCW tak uvnitr, pokud CW, tak je vne

**CCW** - proti smeru hod. rucicek, counter-clockwise direction  
**CW** - ve smeru rucicek, clockwise direction

q je jen **pomocny**, ktery je vzdy uvnitr

****

**Rovinný graf** (též planární graf) je [graf](http://cs.wikipedia.org/wiki/Graf_%28teorie_graf%C5%AF%29), pro který existuje takové rovinné nakreslení, že se žádné dvě hrany nekříží.

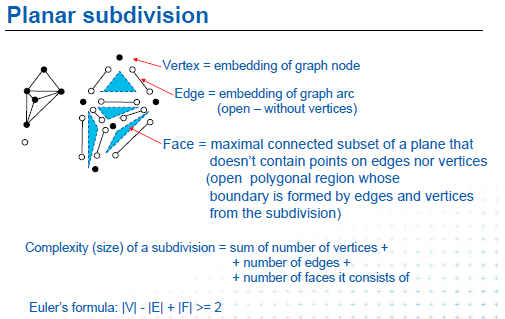
jj, ale ten prvni domecek je taky planarni i kdyz se prekryvaji - protoze ho lze nakreslit tak, aby se neprekryvali

jo to je asi jen vysvetleni toho, ze ten domecek se zkrizenymi hranami lze prekreslit na graf bez prekrizenych hran

to je ta poslední věta ... a grafy U (puvodni) a H (novy) jsou tedy **izomorfni** (ze uzly a hrany jsou stejne napojeny mezi sebou, nezalezi jak jsou umistene, prohnute apod..)

ta to říká, že každý planární graf lze překreslit na ryze segmentovaný

****

****

**(12)**

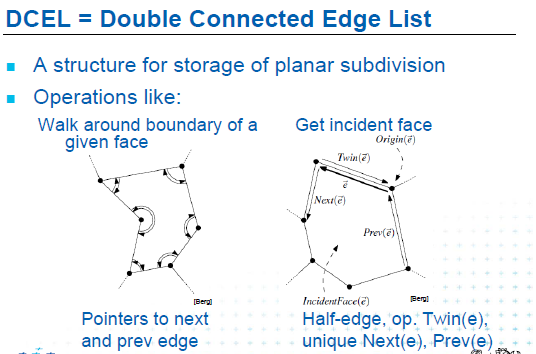
**Euler's formula**: **|V| - |E| + |F| >= 2**

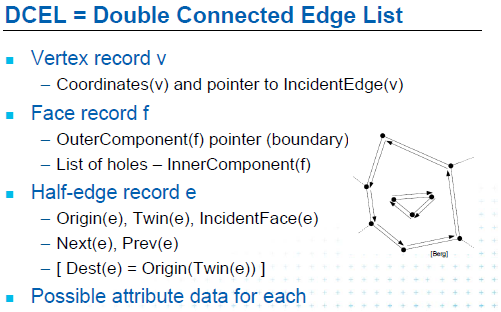
no a jeste tady ukazuji to size ... ze je suma vseho ...

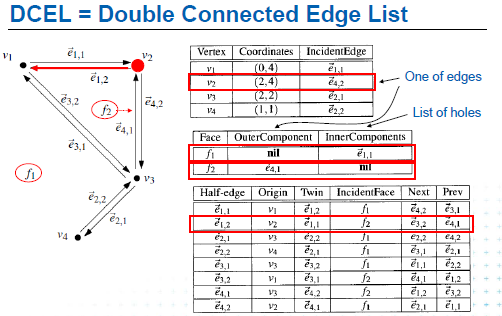
ale pozor, zatim jedna hrana, je jedna hrana ... pozdeji to budou vzdy dve, ale pro tenhle size jedna

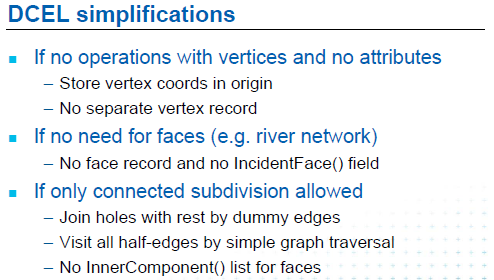
jasne, neorientovane grafy

ok, a ta EF platí pro všechny planární grafy? jj, tedy s **max. jednou hranou mezi uzly**

****

****

****

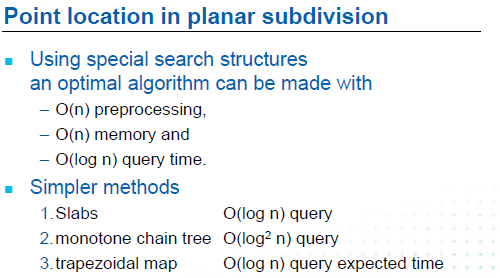
****

**(16)**

tak to první říká, když se v daném alg nebude pracovat s vertexama a zadnýma tributama vetexu, pak

můžeme **první** **tabulku vynechat** a ukládat jen coordinace do tabulky Half-edge do sloupce origin

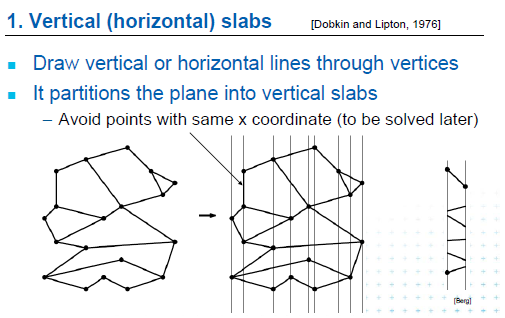
to druhe, když nebudeme potřebovat pracovat s facema, tak **tabulku face zahodim** a incidentFace v tabulce Half-edge zahodim tez

****

**(17)**

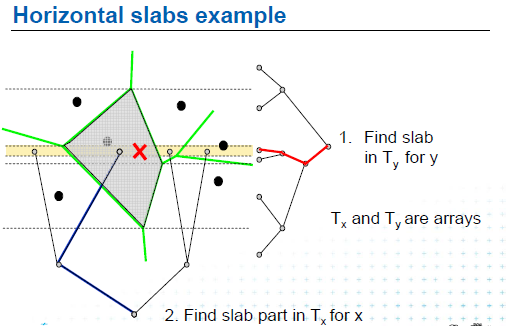
tzn. asi by sis měl představit, že máš tedy graf naházený do struktury DCEL jak je na obr například 15 (slidu)

a proto nemůžeme využít ty první alg, co jsme viděli na začátku, ray crossing, winding, a to další .-D, ale musíme použít tyto 3, které neznáme, ale dál jsou popsané

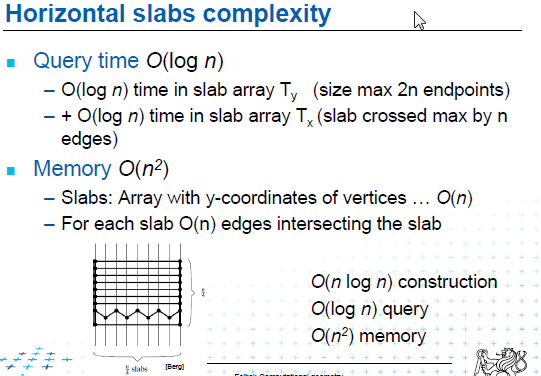
****

**(18)**

Tak to je jasny, v kazdem uzlu to svisle oddelim na **řízky**



**(19)**  
v kazdem bode grafu, zde jest VD (tedy vertexy), udelame horizontalni primku - tim dostaneme **slabs** a Ty a kazdý slab je zde rozdelen VD a to definuje Tx, takze mame 2 stromy podle x a podle y, **takže strom x je co list, to rozdělaný řízek VD a co list y stromu to 1 řízek**



**(20)**

tak **query** **time** je jasny, je to binarni strom, tedy dva, log n + log n = **O(log n)**

tak tam je o první, to mluví o tom Ty stromečku

a to plus další stejná složitost mluví o tom Tx stromečku

takze 2\*log n, takze **O(log n)**

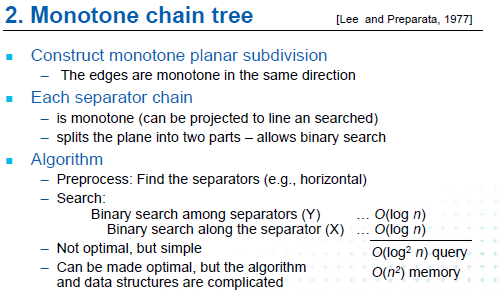
jo a ty stromy slouží pouze k nalezení řízku ... endpoint???

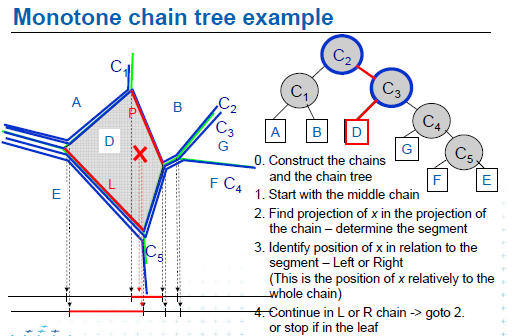
jinak rizku je tolik kolik je uzlu, tedy n a pro kazdy rizek potrebujeme si pamatovat dalsi strom, souhlas...

a kazdy strom ma n dalsich hodnot jo? ze je to n^2

jasne to je nejake horni omezeni O(n^2)

no nevim, to je jedno, proste n^2





**(21, 22) Monotone chain tree**

takže zřejmě jde o **hledání bodu v grafu**, respektive, v jaké face daný bod je, **faces** jsou A,B,D,G,F,E a C jsou jednotlivé **řetězy**, tedy ty modré lajny, co si pamatuji, tak těma řetězama se má ten graf rozjebat takovým způsobem, že mohu po některých lajnách projet vícekrát, ale musím hlavně projet každou z nich ... každý ten řetěz, **rozděluje graf na dvě spíš části**, protože to nejsou rovnocenné pulky

nejvíce dělalo lidem problém pobrat, jak se to dělí těmi řetězi, ale myslím, že to je zřejmé ... žádný z těch řeťezů se nesmí zastavit uvnitř grafu, ale vždy třeba grafem projit až z něj jakoby zase vylezu

to že každý ten chain rozdělí plane na **2 části** vyúsťuje v možnosti **binárnoho prohledávání** ...

a to je asi ten strom ...

vypadá to zase na nějaké horizontální i vertikální dělení ...

to C2 je kořenem, neboť jím začínáme stavět ten strom a proč? no protože chceme aby ten strom byl co nejvíce rozvážený, tak vezmeme nějaký prostřední chain, takže třeba ten C2, jak vidno, C3 by byla asi lepší volba ...

takže 0. jak na ten strom?

proč jsou rozestavené zrovna takhle ty řetězi?

tak začneš C1, řízneš si graf tak, že od nej oddělíš tu horní face A

pak C2, k A přídáš B, řízneš

pak C3, přidáš D

a pak C4, přidáš přidáš G

a C5 E

tedy, C3 dělí plochu na A,D,B a E,F,G

odkud vezmu tu modrou caru C1? nebo caru C2?

to je jedno od kud ji vezmeš, ale hlavne jí musíš rozdělít tu rovinu ...

a použiješ jich tolik, kolik je třeba, prostě aby jsi oddlělil všechny face

BTW. můžeš si jeden chain představit uplne stejne jako ten řez, kterým dělíš plochu na řízečky

C2, odděluje A, B a zbytek, tedy EDGF

prostě musíš projet každou stěnou ... vše od sebe odsekat ...

takovej char. znak je, že třeba začínáš vždy ze stejné hrany ... ale asi to není podmínka...

proto je tam tolik řezu a u pismene C5, jak je ta hrana je jen jeden chain

takže, vyberu nějakej chain, kerej mi tu plochu děli zhruba na pulky ... tak třebaten C2

to proč je toto důležité píšu nahoře

proste podle toho jak ten chain oddeluje roviny

no a pak se díváš na co ten C2 rozdělil tu rovninu, vidím, že v horní pulce je další chain a to C1, takže jde nalevo ... .

napravo umistim C3, C4 a C5, proč? protože spodní jsou obsaženy ve spodni části toho dělní pomocí C2 ok?

no pak přiřazuji listy ...

ted jak v nem hledat, tak najdu projekce na x vertexu ... takže na ose x, budu mit bloky, neco jako rizky ...   
takže já si myslím, že uděláš ty projekce na x a pak začneš upřesnovat pozici bodu X

no ale mělo by se to hledat pomoci toho stromu ... aby to bylo rychle ... kcemu pak vyuziju ty projekce?

asi zjistis, ze ten bod je **mezi** **chainy C2 a C3,** takze se ve strome posunes tou hranou

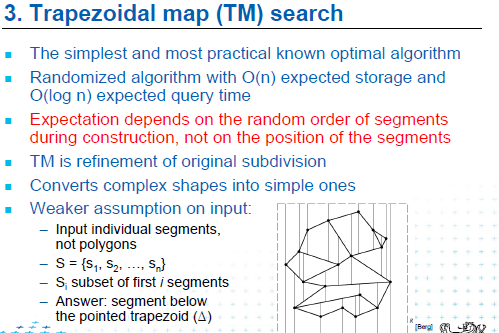
jak zjistim ale ze je mezi C2 a C3??? ty dvě osy tam dole ... rekl bych, ze ta nahore je C2 a dole C3

ale pravda, opravdu to je C2 a C3, máš pravdu ... ale jak to zjistil? že zrovna tyhle dva? tak **zacnes od korene**

to je C2 a koukas kam jit nalevo nebo napravo, **cerveny bod** nema projekci v te krivce C1 a ma projekci v krivce C3

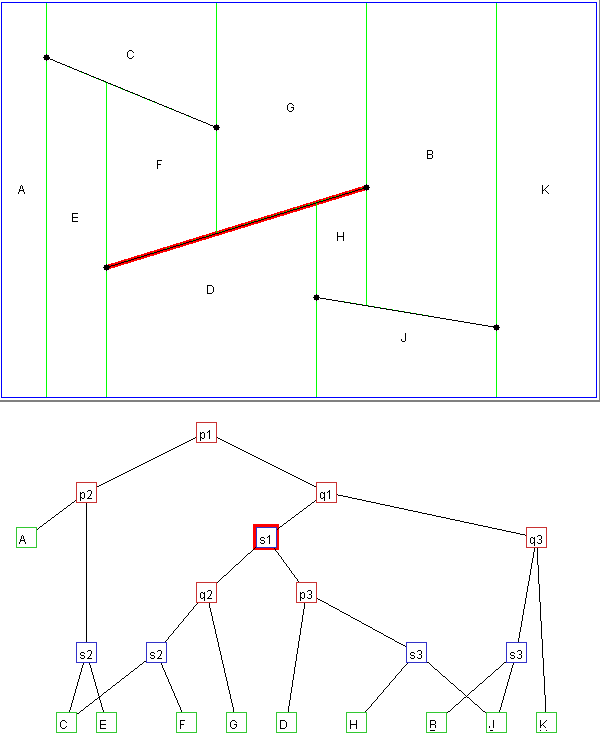
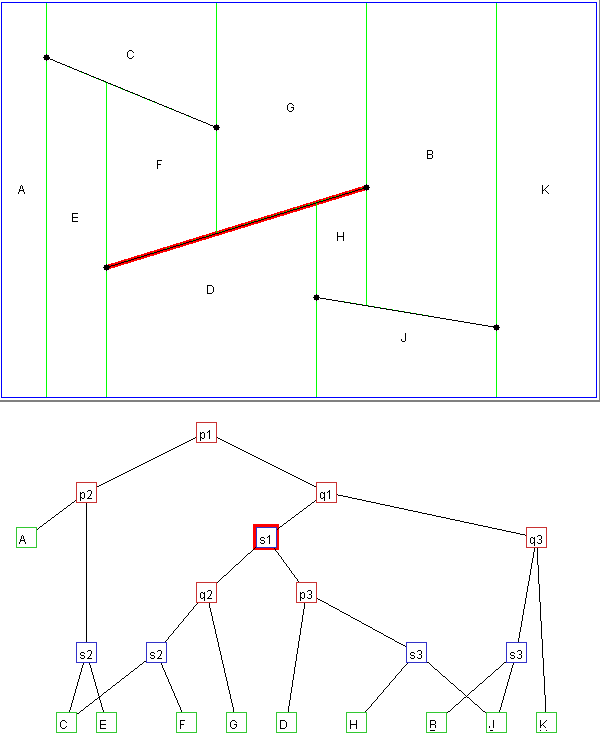
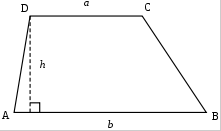
takze jdu k C3, v C4 a C5 má taky projekci, C2 mi to rozdeluje na 2 casti a vidim, ze ten bod lezi **napravo**, tak jdu doprava na C3, pak se kouknu a od C3 je bod **nalevo**, tak jdu do listu D

napravo se **myslí dolu** ... zřejmě ... nebo jako by jsi udělal šipku u každého řetězu a podle směru pak můžeš určit kde je R a kde L od ní ...

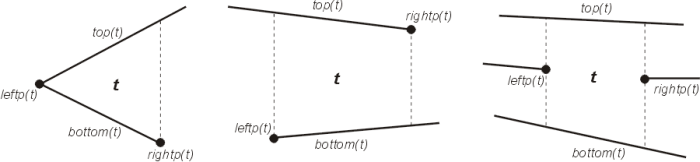


**(23) Trapezoidal map**

takže jedná se o tu 3. metodu hledání bodu v grafu (pro DCEL??)

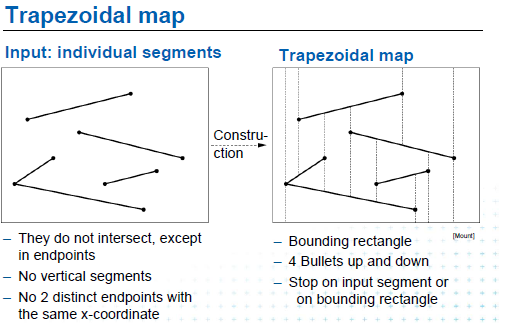
In [geometry](http://en.wikipedia.org/wiki/Geometry), a four-sided figure with one pair of parallel sides is referred to as a **trapezoid (lichoběžník).**

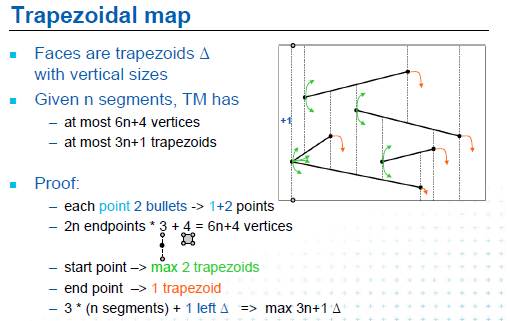
<http://cg.scs.carleton.ca/~cdillaba/comp5008/applet/applet.html>



Possible trapezoid configurations

<http://cg.scs.carleton.ca/~cdillaba/comp5008/trapezoid.html>



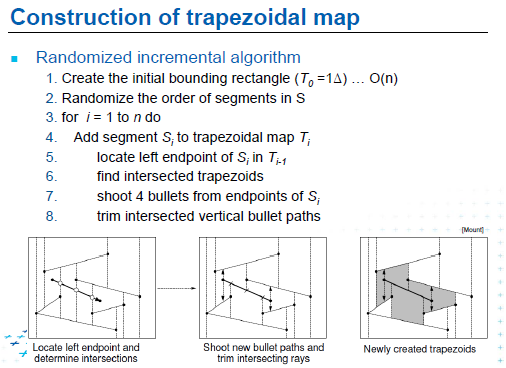


**(25)**

TM has at most **6n + 4 vertices** ... takže n je počet segmentu ... proč 6? protože 2 **endpoints** na 1 segment a každý segment má 3 **trapezoidy,** max 2 ve startu a 1 v koncovým, to jsou ty zelené a oranžové šipečky

a **3n + 1 trapezoidu (lichoběžníků)** ... tak zase 3, n je taky jasne a ta 1 je vždycky uplne vlevo ...

ty šipky jsou tedy lichobezniky, ktere jsou urceny body, takze **pocet lichobezniku** je pocet sipek +1, jj



**(26) Randomized incremental alg. of constructing TM**

tak na 26 se objevila uprostred nejaka silna cara, ktera predtim nebyla, to je **další přidávaný segment** ...

a jeste otazka - proc musi byt **randomizovane ty segmenty**? no já myslim, ze jde o vybirání segmentu, tak jak to bylo v tom appletu, ten vybiral segmenty v takovem poradi, v jakem jsem je pridaval, ale kdyz neznas toto poradi, vybiras je nahodne ... treba

1. takze udelas ohranicujici obdelnik

2. randomizujes vyber segmentu

3. vybiras jeden za druhym ...

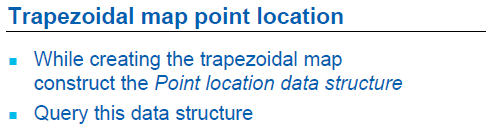
4. vybrany vkladam do TM

5. najdes na ni levy end p.

6. najdes prd, nebot mluvime o prvni pridane, ale kdyby jsi mluvil o x-te, pak by jsi mohl najit najit nejake pruniky trapezoidu, to jsou ty 3 čary jak ji sekaj asi ...

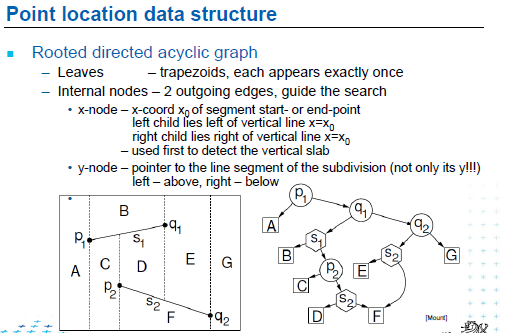
7. v tech end pontech zacnes strilet

8. rozsekas ty cesty, jak frcely kulky, proste aby jsi nepretahoval pres jine segmenty ...



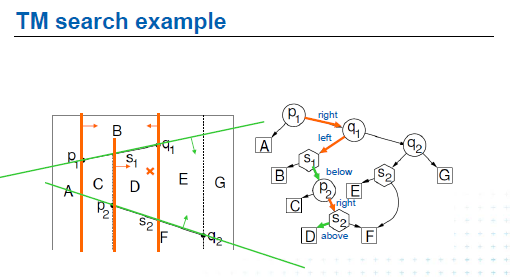
**(27)**

takže ten query time bude nejspis time, pri dotazovani te struktury ... na hledani bodu



**(28)**

to **neni cyklus**, když ty hrany jsou proti sobe u toho listu F? ne



**(28-29)**

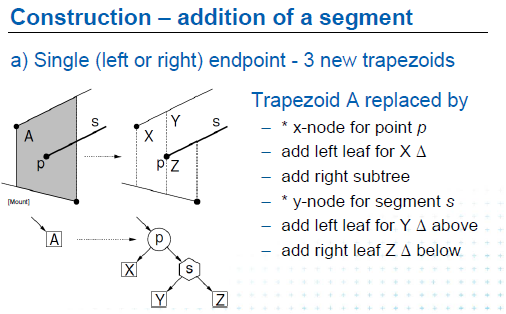
2 vycházejici hrany, řídí vyhledávání ... zacneme od p1, **nalevo** mame A (od úsečky p1), tak to dame nalevo do stromu, **napravo** je zbytek, v tom zbytku vybereme q1, ten nam to zase rozdeli na **levou a pravou cast**

no a proč zrovna q1? protože to je **druhý bod te line s1**, to s1 a s2 jsou lines? jj, dělí to na **upper a lower** část

proste **nalevo** od q1 je cosi s1 a na **pravo** od q1 je q2, tak to same do grafu, s1 jako lajna nam to deli na nahore a dole (**nahore - dame v strome doleva**), takze doleva dame B, do prava zbytek - tedy C, D, p2, s2 a část z F

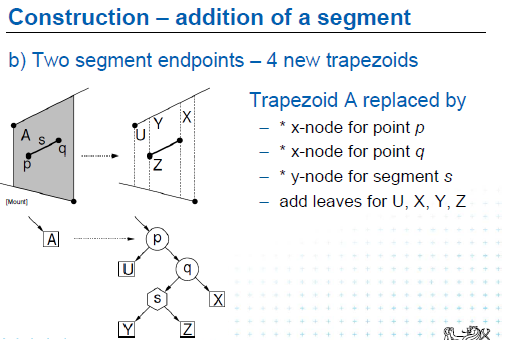
vybereme p1 a ten to zase oddeluje na levo a vpravo, C v levo, zbytek vpravo, pritom zbytek tvori: D, S2 a část z F

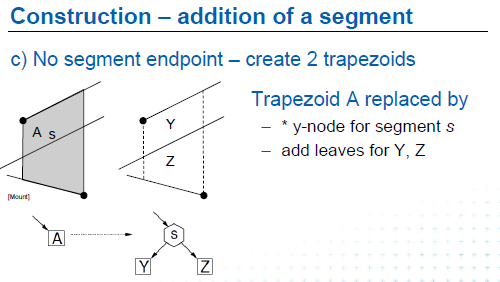
s2 nam to zase oddeluje na nahore a dole, takze dostaneme D a F jako potomky s2, q2 nam to rozdeluje na vpravo a vlevo, vlevo je: E, s2, část z F, vpravo je jen G, tak G dame jako pravy list a vlevo dame s2, ktery nam to rozdeluje na nahore a dole, kde mame E a F, je třeba si pojmenovat koncove body a segmenty ... proto s jako segment

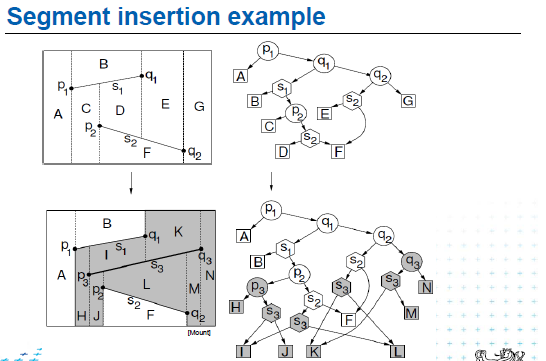


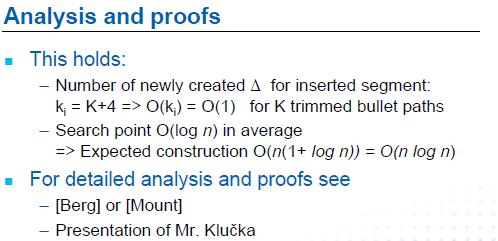
**(30)**

co je to X? to je lichoběžník misto toho A, viď?









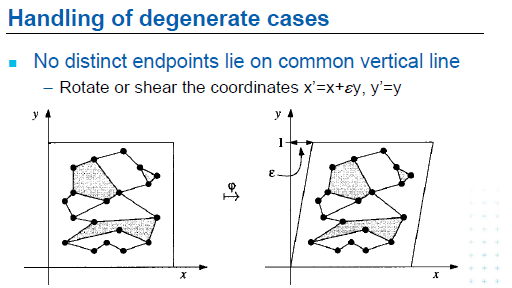
**(34)**

asi ze novy **segment pridas se slozitosti 1**

je to asi počet nových věcí (uzlu) po vlozeni segmentu

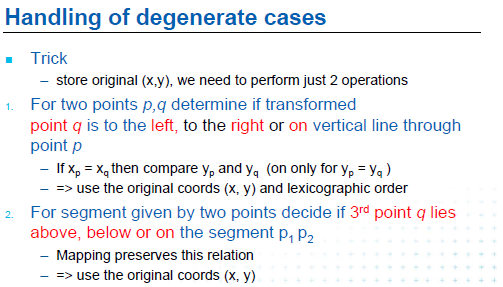
protoze tam jen upravis omezeny pocet tech kulek, to ki je tedy počet nově vytvořených trapezoidů?

pro **vyhledání bodu**, o to tady jde především, je pak **O(logn)**



**(35)**

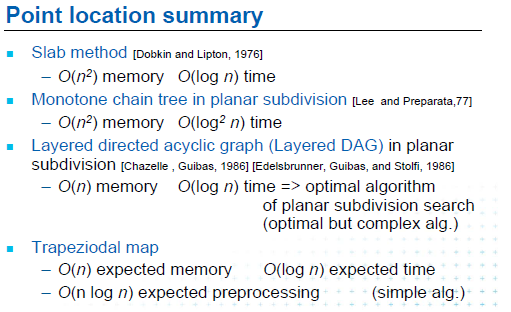
to je pekne, ale co kdyz prave tou rotaci nebo tim shear mi vzniknou body, ktere budou nad sebou?



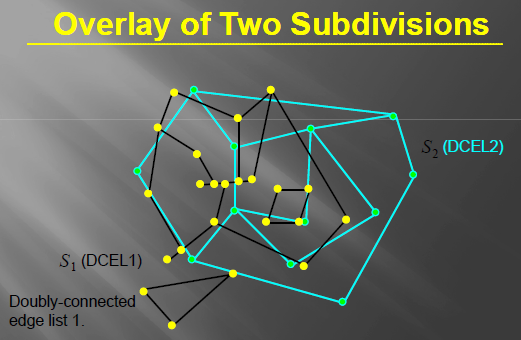
**(36)**

1 - to je asi to urceni, jestli jsou body nad sebou nebo ne, if(xp == xq) tak jsou nad sebou i kdyz taky nevim co s tim y, ne jenom nad sebou ... mas tam i vlevo vpravo ... tak to bych urcil lehce xq < xp, xq > xp, mluvis o tech novych souřadnicich jo? jj k cemu si ukladam ty puvodni? ja si to posunu jen proto abych mohl toto určit, ale jinak používám ty originální souřadnice .... že ano?

2 - je to o tom, ze chceme zjistit jestli treti bod lezi nad, pod ci na primce?



1. **Výpočet překrytí planárních dělení (průsečík, sjednocení, rozdíl) modifikovaným Plane-sweep algoritmem pro průsečíky množiny úseček.**



The overlay is a **new planar subdivision**.

