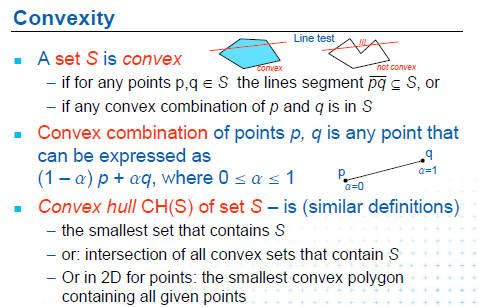
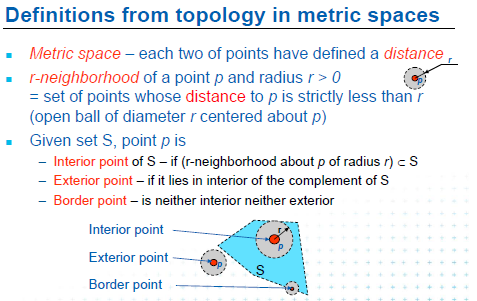
# 20. Konvexní množina, konvexní obálka množiny (definice). Reprezentace konvexní obálky ve 2D. Její výpočet pro množinu bodů: Grahamův algoritmus, Jarvisův algoritmus balení dárku, metoda rozděl a panuj. Výpočet konvexní obálky pro jednoduchý polygon. Výpočet a reprezentace konvexní obálky ve 3D.

1. **Konvexní množina, konvexní obálka množiny (definice).**

****

**(4)**

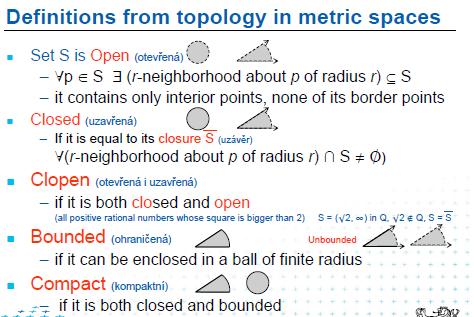
Aha...konvexni obalky lze delat asi jen do konvexnich tvaru ... (1 – a) p + aq, where .... to si pamatuji z optimalizace ... ale co to udava? konvexni kombinaci? WTF? to je ta usecka pq, a libovolny bod m, ktery na ni lezi je dan tim vztahem, pokud mohu najit takove alfa, aby ten vyraz se rovnal tomu m.

****

**(5)**

**metric space** - 2 body maji definovanou vzdalenost mezi sebou, r-neigh.. - polomer r>0, okoli je to neigh..

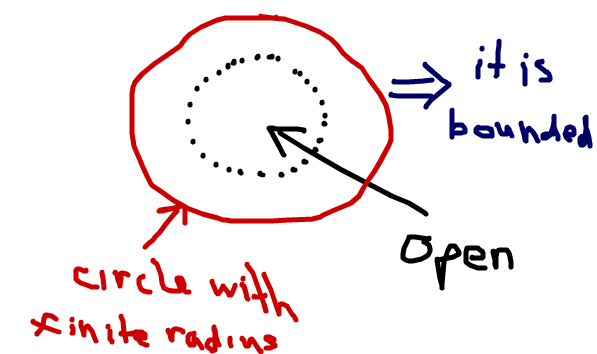
a muhou nastat 3 pripady, bud:  
**1) cele r je podmnozinou S  
2) cele r je mimo S  
3) r je castecne v S**

****

**(6)**

Ale plati, ze kazda otevrena je zaroven uzavrena? ano? pak tedy kazda otevrena je zaroven clopen. tak closed, nemuze byt open, no že všechna positivní racionální čísla jejichž druhá mocnina je vyšší 2 patří do clopen

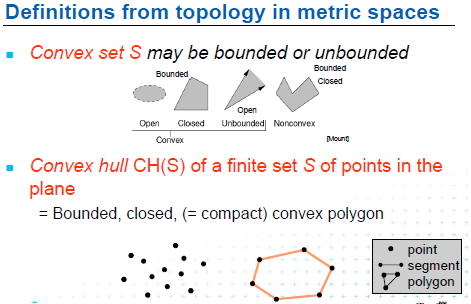
bounded muze byt i open, mas treba tu kruznici open, tak ji muzes obalit nejakym ball s finite radius, takze je bounded, kdyby jsi mel treba nejakou polorovinu nekonecnou, nebo neco takovyho nekonecne dlouhyho, tak to do finite kruznice nedas.



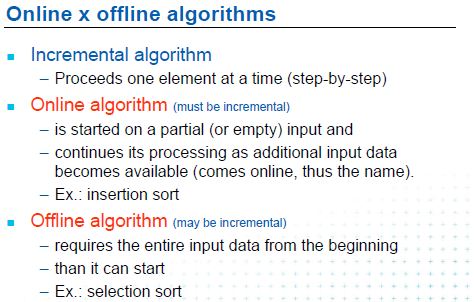
Open je nejaka oblast (kus plochy) a ten kus plochy mohu obalit nejakou kruznici, proto ten kus plochy je **bounded**

**unbounded** jsou nejake nekonecne plochy, ktere nedas do kruznice

**unbouded** = nekonecna plocha  
**bounded** = konecna plocha

****

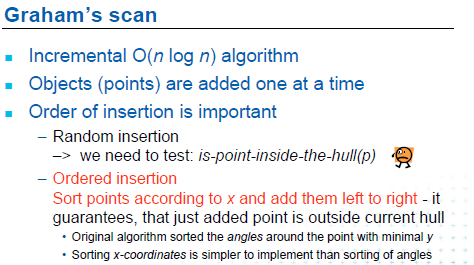
(7)



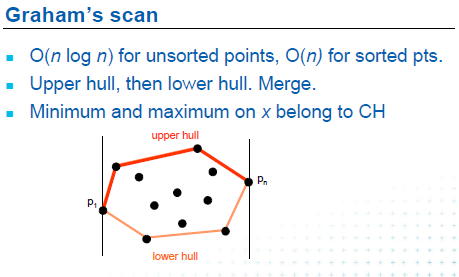
**(9)**

**inkremental** alg. jsou takove ze behem sveho behu zpracovavaji po jednom prvku, **step-by-step**

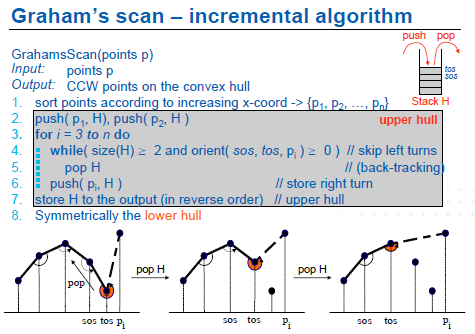
**online algoritmy** - dostavaji data postupne  
**offline algoritmy** - maji vsechna data na zacatku



(10)

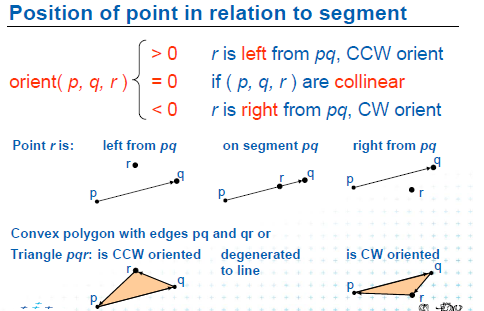


(11)



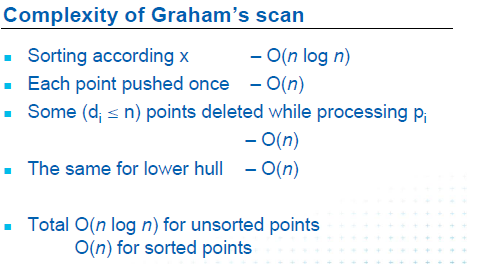
**(12) Graham’s scan – incremental alg.**

Takze **seradime** **podle x**, udelame upper hull, lower hull a spojime, takze zasobnik H a tam hazeme navstivene uzly, dame tam dva prvni uzly a koukame dal na treti, pokud je uhel 1 - 2 - 3 vetsi nez 180°, tak pop H a opakuj dokud je vetsi nez 180, pak udelame push ten treti bod a vysledek je **upper hull**, symetrikcky pro lower hull.



**(13)**

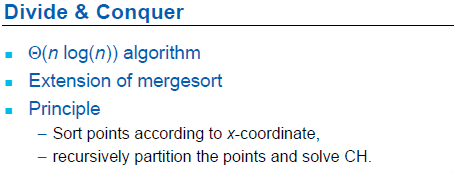
Tady se resi ta **orientace,** to je mozna lepsi nez uvazovat nad uhlem 180°, proste nalevo / napravo a ta fce orient ... háže co? vzdalenost toho bodu r od usecky pq, bud je 0, tak je na usecce nebo je na jednu stranu pak je kladna nebo na druhou pak je zaporna, ale vzdalenost jako vzdalenost se pocita obtizneji, pze tam musis normalizovat apod. takze ono staci jen nejakou vzdalenost, ktera treba neodpovida metrice, ale presto plati vetsi mensi.



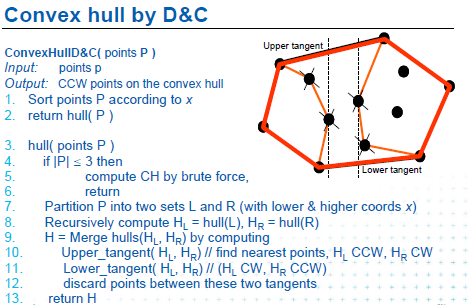
**(17)**

Pak prochazis kazdej bod a testujes orient, takze O(n), nekolik jich smazes, taky O(n) a udelas to 2x pro up a bottom

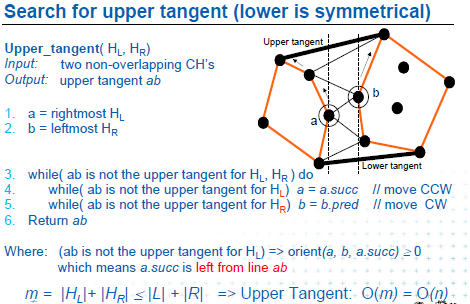
takze celkove **O(n log n)** (pro nesortovana data).



**(18)**



**(19)**

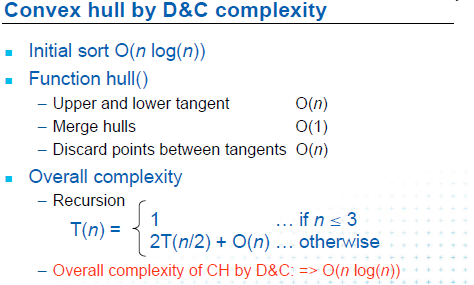
****

**(20) Convex hull by D&C**

OK, tak dejme tomu mame 2 hully, strana 20, potrebujem sestrojit upper a lower tangent a ty vnitrni hrany odstranit

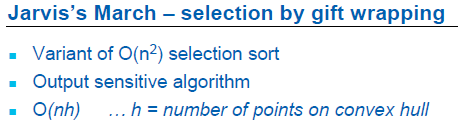
jo, tanget = asi tecna (myslim ze to nejak tak bylo v conics dual) a ty 2 hully jsme vyrobili taky timto alg? nebo třeba scanem? asi taky timhle algoritmem, proste rekurzivne zacnu delit na mensi casti az se dostaneme na body - co bod to hull a pak zpetne je spojujeme, aha, ty hully mohou byt posunute, pak by to nefungovalo, spojit maxima, ok

ok, takze najdeme a je **rightmost** a b je **leftmost** a pak jedeme nahoru a dolu, dokud to jde a mame upper a lower hrany.

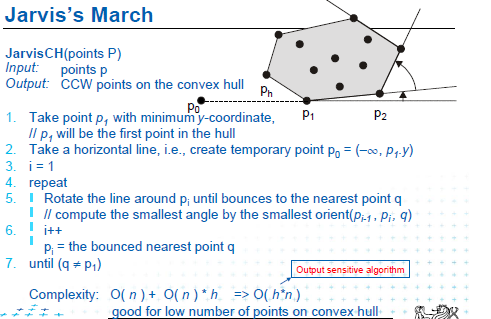
****

**(21)**

**sort** klasika O(*n* log(*n*)), upper a lower tangent k nim musim vyšplhat ... takže n, spojeni jen asi nějaká práce s pointerama, jj, proste napojim jednu horni a dolni, a pak mazu vnitrni, kterych muze byt n, celkem: **O(*n* log(*n*))**

****

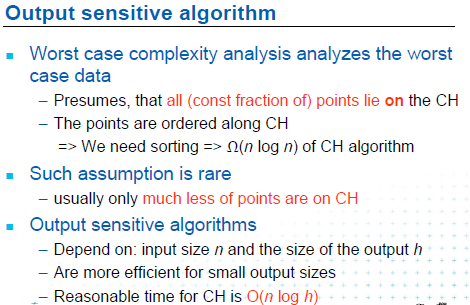
**(25)**

****

**(25, 26) Jarvis’s March – selection by gift wrapping**

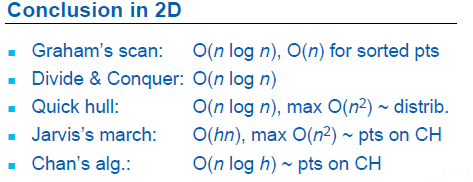
**Jarvis ... O(nh)** h point on CH

Najdeme nejnizsi a obalujeme okolo, takze **hledame body s nejmensim uhlem.**

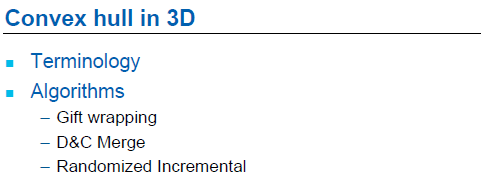
****

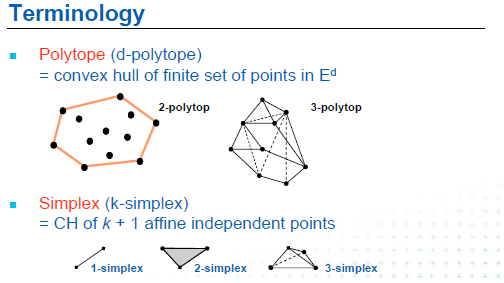
**(27)**

**rare –** ojedinělý, nejhorsi je to, kdyz vsechny body jsou na CH.

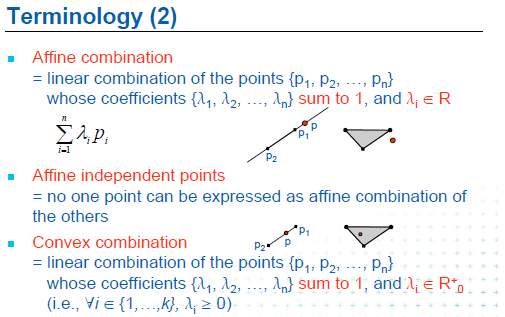
****

1. **Výpočet a reprezentace konvexní obálky ve 3D.**

****

****

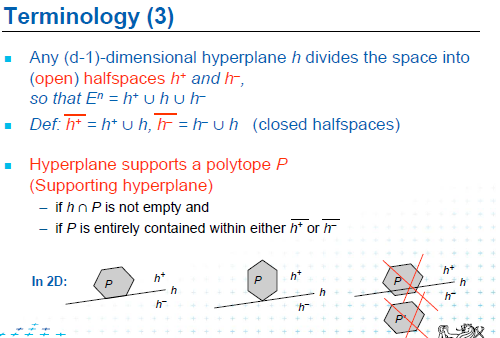
(13)

****

**(14)**

Jasne to bude takovy ten zapis: **alfa\*A + (1-alfa)\*B**

Takze timhle muzeme vyjadrit nejakou usecku, nebo trojuhelnik ci nejaky polygon. Jo, proste mas tri body a nejaky lambdy, tak s tema lambdama posouvat ten bod uvnitr. 3 body.

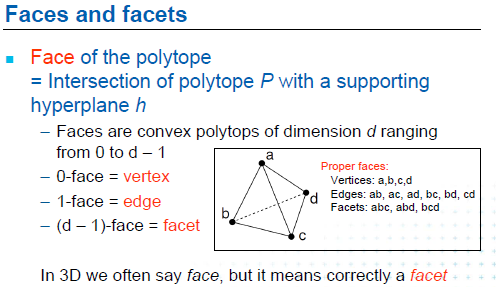
****

**(15)**

Co je hyperplane? nadrovina, jakože nadroviny jiné roviny? Mas treba prostor n-dimenzi, tak n-1 prostor je hyperplane, proste rozdeluje prostor na dve pulky

pro 2D je hyperplane 1D primka  
pro 3D je to rovina

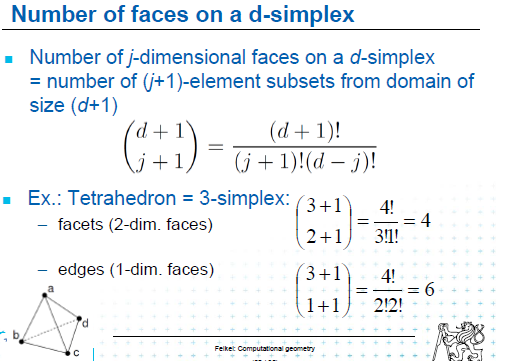
**Polytop** je asi nejaky zakladni tvar, pro 2D trojuheknik, pro 3D ctyrsten a tak dal.

****

**(16)**

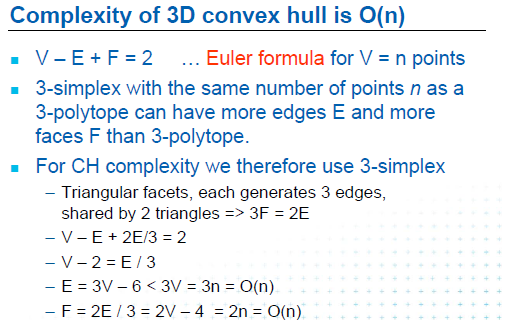
**Facet** je tedy stena, **face** je cokoliv s nizsi dimenzi

facet: n-1  
face: cokoliv mensi nez n, **face** je treba bod, usecka, stena; **facet** je stena.

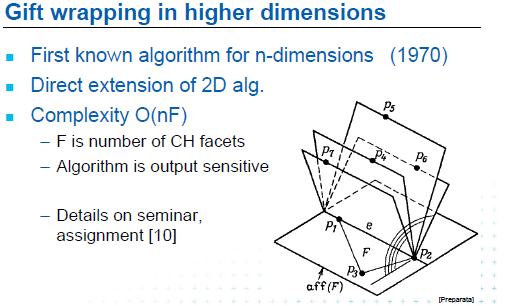


**(20)**

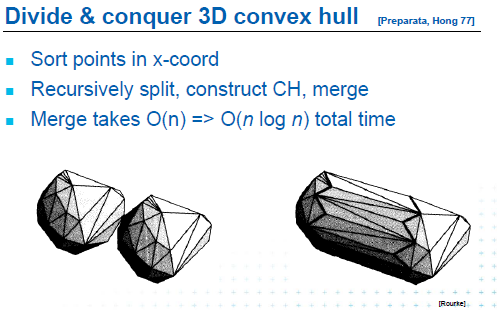
Tetrahedron - proste pocet facu, kdyz znas dimenzi.

****

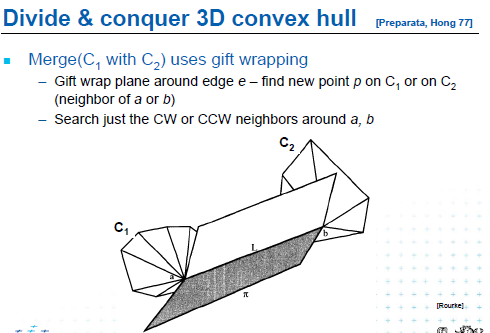
**(21)**

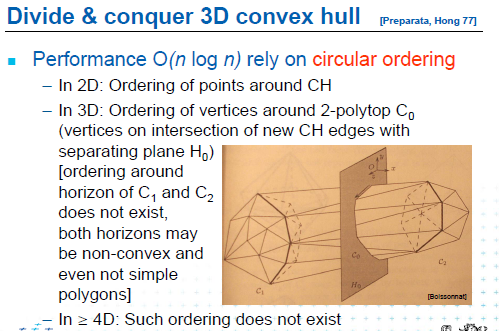
****

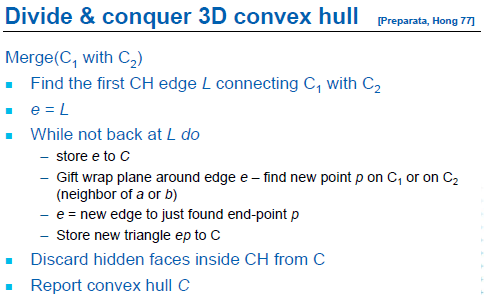
**(22)**

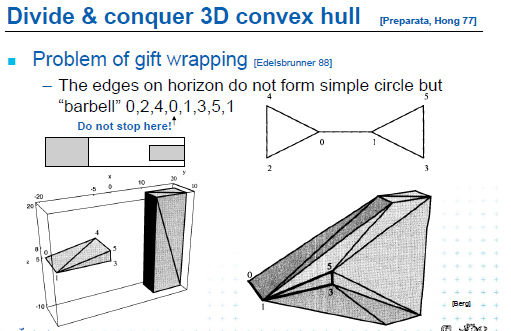
****

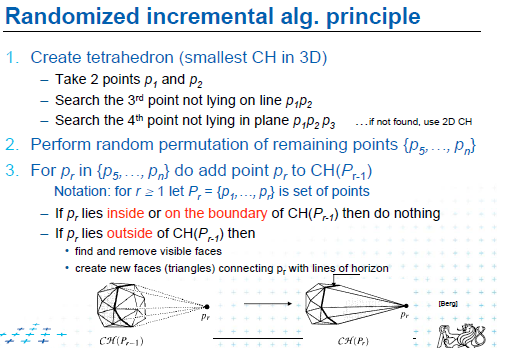
**(24)**

****

****

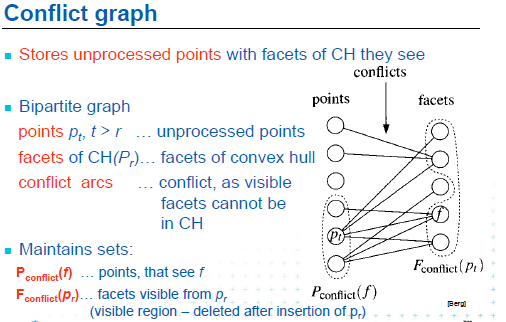
****

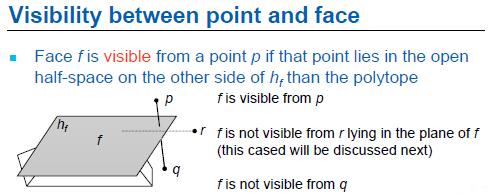
****

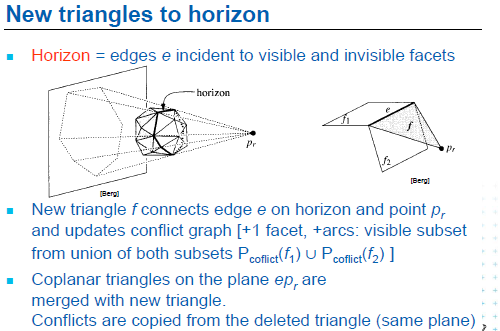
****

**(29)**

Takze sestavime ten ctyrsten, pak znahodnime body, tedy jejich poradi a pridavame do CH. Pri pridavani odstranujeme hrany a body, ktery jsou uvnitr, podle obrazku, no mozna ze nebude nahodna, stejne prolezes vsechny body nakonec.

****

****

****