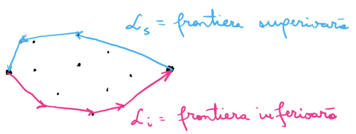
Partea 1

Partea 2

Cursul 8 - Acoperiri convexe

- Proprietate necesara ca A, B si C sa fie coliniare (in $\mathbb{R}^2$):
  - $\begin{vmatrix} x & y & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} = 0$
- Proprietati ca A, B si C sa fie coliniare (in $\mathbb{R}^n$):
  - A, B si C sunt coliniare $\Leftrightarrow$ cea mai mare dintre distantele AB, AC si BC este egala cu suma celorlalte doua (ceva de calculat)
  - A, B si C sunt coliniare $\Leftrightarrow$ AB si AC sunt proportionale
    - Unde $\overrightarrow{AB} = B - A = (x_B - x_A, y_B - y_A, \dots)$
  -
- Distanta dintre doua puncte:
  - $\mathbb{R}^2 : d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
  - $\mathbb{R}^3$: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$
- Raport
  - Pentru a determina $r(A, B, C)$, trebuie sa determinam $r = ?$ a.i. $\overrightarrow{AB} = r\overrightarrow{BC}$.
  - $\overrightarrow{AB} = B - A = (x_B - x_A, y_B - y_A) = (1, 1)$
  - $\overrightarrow{BC} = C - B = (x_C - x_B, y_C - y_B) = (5, 5)$
  - Deci $\overrightarrow{AB} = \frac{1}{5} \cdot \overrightarrow{BC}$
    - Unde $r(A, B, C) = \frac{1}{5}$
- Testul de orientare
  - Se refera la pozitia relativa a unui punct fata de o dreapta
  - Determinarea pozitiei punctului R raportat la dreapta PQ:
    - Calculam determinantul:
      - $\Delta(P, Q, R) = \begin{vmatrix} 1 & 1 & 1 \\ p_1 & q_1 & r_1 \\ p_2 & q_2 & r_2 \end{vmatrix}$
    - R e situat:
      - **pe dreapta** $PQ \Leftrightarrow \Delta(P, Q, R) = 0$ ("ecuaţia dreptei")
      - **"în dreapta" segmentului** orientat $\overrightarrow{PQ} \Leftrightarrow \Delta(P, Q, R) < 0$
      - **"în stânga" segmentului** orientat $\overrightarrow{PQ} \Leftrightarrow \Delta(P, Q, R) > 0$
  - Asa putem determina daca un punct e in dreapta/stanga, viraje, natura unui poligon (convex/concav), daca doua puncte sunt de-o parte si de alta a unei drepte

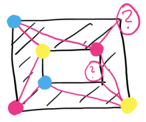
- Multime convexa:
  - O mulţime $M \subset \mathrm{R}^m$ este convexă dacă oricare ar fi $p, q \in M$, segmentul $[pq]$ este inclus în M.
- Problema: forma determinata de mai mult de 2 puncte nu e convexa => vrem sa gasim acoperirea convexa
- Acoperirea convexa:
  - Conventie: sensul de parcurgere a frontierei este cel trigonometric
  - Idee algoritm lent (O(n^3)):
    - Un punct este pe frontiera daca toate celelalte puncte se afla in stanga lui (folosind testul de orientare)
      - (Aka pentru fiecare dreapta AX, parcurgem toate punctele Y != {A, X}. Daca gasim un punct in dreapta, valid = false)
  - Graham's scan, varianta Andrew (O(n log n)):
    - Sortam si renumerotam punctele lexicografic (dupa x).



    - Pentru cele doua frontiere, ne asiguram ca ultimele 3 puncte gasite nu formeaza un viraj la dreapta
  - Javis' march (O(nh), unde h=nr puncte acoperire convexa):
    - Nu e necesara sortarea
    - Incepem cu un punct sigur de pe frontiera (cum ar fi cel mai din stanga)
    - Pentru ultimul punct X de pe frontiera gasit, alegem un pivot random Y
    - Luam toate punctele Z. Daca Z e in dreapta dreptei XY, atunci pivotul Y devine Z
    - Ne oprim cand punctul gasit este cel de la care am plecat

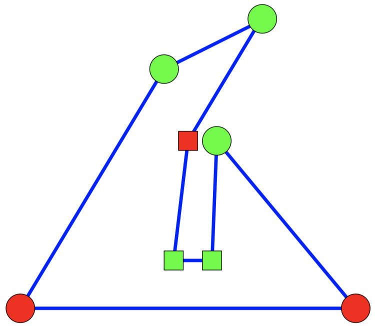
Cursul 9 - Triangularea poligoanelor

- Problema galeriei de arta = plasarea de camere (360°) intr-o galerie de arta (=un poligon) care sa poata vedea toata suprafata (orice punct poate fi unit cu o camera a.i. segmentul sa fie in interiorul poligonului)
  - Intrebari: cate camere? unde sa fie puse?
- Natura poligonului:
  - Daca e poligon convex: o camera (oriunde)
  - Daca e poligon concav: mai greu
- Vrem sa exprimam numarul de camere in functie de un n

- Principiu: descompunem poligonul in triunghiuri (=triangulare)
- Definitii:
  - Diagonala a poligonului: segment ce uneste doua varfuri ale acestuia si e situat in interiorul poligonului
  - Triangulare: descompunere in triunghiuri data de numarul maximal de diagonale ce nu se intersecteaza
- n varfuri $\Leftrightarrow$ $n - 2$ triunghiuri
- Idee rezolvare problema:
  - Facem triangularea poligonului
  - Facem o 3-colorare a acestuia (astfel incat sa nu fie muchii monocrome)
  - Numaram culorile. Punem camera in varfurile de culoarea care apare de cele mai putine ori (ca sa minimizam)
  - Problema:
    - Nu merge pentru "contururi" (nu putem 3-colora):
    - 

- Teorema galeriei de arta:
  - $\left[\frac{n}{3}\right]$ camere sunt **uneori necesare** si **intotdeauna suficiente** (pentru a rezolva problema)
    - Uneori necesare: (probabil) atunci cand cele 3 culori apar de un numar egal de ori
- Triangulare poligon
  - **Varf convex** $\Leftrightarrow$ are acelasi viraj ca cel mai din stanga varf
    - Aka marchezi nodul din stanga ca fiind convex, apoi incepi sa parcurgi nodurile in sens trigonometric.
    - Virajul unui nod x = virajul dat de [predecesor x] x [succesor x]
  - **Varf principal** x $\Leftrightarrow$ [predecesor x] - [succesor x] e diagonala
    - Aka triunghiul [predecesor x] - x - [succesor x] nu contine niciun alt varf
  - **Ear (E)** = varf principal convex (orice graf cu >= 4 noduri are >=2 ears)

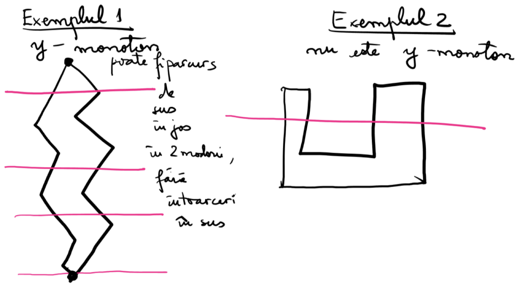
- **Mouth (M)** = varf principal concav
  - Cerc = convex; patrat = concav
  - Verde = principal; rosu = neprincipal
  - Ear = cerc verde; mouth = patrat verde



- Algoritmul ear clipping este O(n^2)
  - (banuiesc ca doar tai triunghiurile date de "urechi")

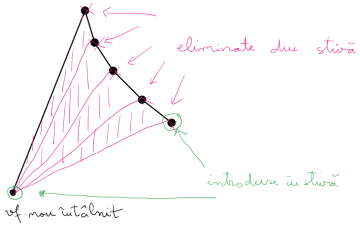
- Algoritmi de triangulare:
  - O(n) pentru poligoane y-monotone
  - O(n log n) pentru poligoane oarecare

- Poligon y-monoton:
  - Poate fi parcurs de sus in jos in 2 parcurgeri, fara intoarceri in sus

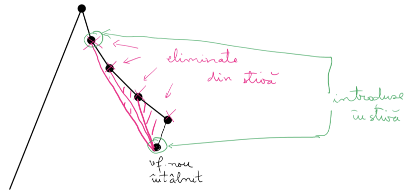


  - Metoda: prin dreapta de baleire (line sweep):
    - Sortam nodurile de sus in jos, dupa y (si apoi dupa x)
    - Avem o stiva, in care la inceput punem primele 2 noduri
      - (din stiva asta eliminam un varf cand e trasa o diagonala mai jos de acesta)

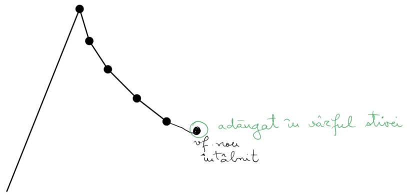
- Avem doua cazuri:
  - Cazul 1 (nodul curent v_j se afla pe lantul opus ultimului nod din stiva => extragem toate nodurile din stiva in afara de ultimul, facand triunghiuri cu ele, si punem in stiva v_{j-1} si v_j):



  - Cazul 2a (nodul v_j nou intalnit se afla pe acelasi lant ca ultimul nod din stiva => elimina ultimul nod din stiva, daca diagonala este in interior si verifica din nou; la final adauga nodul nou):



  - Cazul 2b:

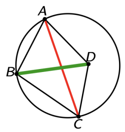


- Tipuri de varfuri:
  - Start (respectiv end): nod al carui grad interior e $\leq 180°$ si cele doua muchii adiacente cu el sunt sub (respectiv deasupra) lui
  - Split (respectiv merge): nod al carui grad interior e $\geq 180°$ si cele doua muchii adiacente cu el sunt sub (respectiv deasupra) lui
  - Standard: restul nodurilor

- Elimina nodurile split/merge, tragand diagonale din ele, pentru a imparti un poligon in poligoane y-monotone

Cursul 10 - Triangularea multimilor de puncte

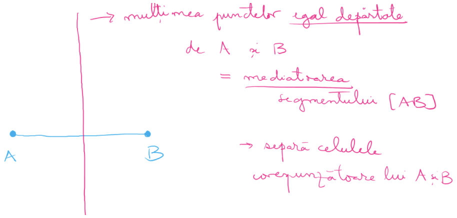
- Pentru orice triangulare a unei **multimi de puncte**:
  - Notam k numarul de puncte de pe frontiera acoperirii convexe
  - Vom avea $(2n-k-2)$ triunghiuri si $(3n-k-3)$ muchii.
- Relatia lui Euler:
  - $n-m+f=2$
    - n = punctele initiale
    - m = laturile triunghiurilor
    - f = fetele; adica numarul de triunghiuri + fata exterioara
  - Deci vom avea:
    - 2 *[numar muchii] = 3* [numar triunghiuri] + k (fata exterioara)
    - ^ numarul de fete din perspectiva muchiilor
- Vrem o triangulare a punctelor cat mai buna (daca avem puncte dintr-un radar, sa putem obtine o harta care sa aiba sens, sa le legam bine intre ele)
  - Aka vrem triangulari legale ("naturale")
  - Intre doua triangulari diferite, cea legala este cea care are cel mai mic unghi dintr-un triunghi cea mai mare valoare
- Muchii ilegale
  - Intr-un patrulater (convex) ABCD, muchia AC este ilegala daca min(T_{AC}) < min (T_{BD}).
    - Unde min(T_x) este unghiul minim din triangularea cu diagonala respectiva
  - Criteriu geometric:
    - **Criteriu geometric** pentru a testa dacă o muchie este legală: muchia AC, adiacentă cu triunghiurile ΔACB și ΔACD este ilegală dacă și numai dacă punctul D este situat în interiorul cercului circumscris ΔABC.

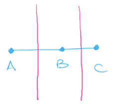


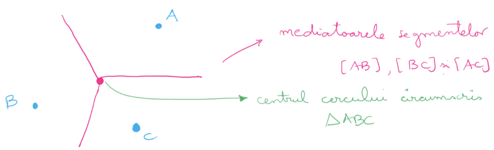
  - Criteriu numeric:
    - Calculam:
      - $$\Theta(A,B,C,D) = \begin{vmatrix} x_A & y_A & x_A^2+y_A^2 & 1 \\ x_B & y_B & x_B^2+y_B^2 & 1 \\ x_C & y_C & x_C^2+y_C^2 & 1 \\ x_D & y_D & x_D^2+y_D^2 & 1 \end{vmatrix}$$
    - $\Theta(A,B,C,D)=0 \Leftrightarrow$ punctele sunt conciclice (poti alege orice diagonala)
    - $\Theta(A,B,C,D)>0 \Leftrightarrow$ punctul D este situat in interiorul cercului circumscris $\triangle ABC$ (aka muchie ilegala) - daca ABC e viraj la stanga
    - Daca muchia AC e ilegala, o inlocuim cu BD.
  - Exista o singura triangulare legala (daca punctele nu sunt conciclice).

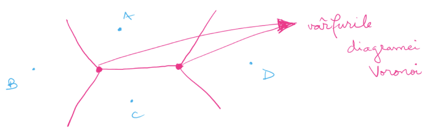
- Triangulare legala:
  - Nu are muchii ilegale
  - Putem folosi un algoritm de timp O(n log n) (si memorie O(n)) pentru a determina triangularea legala a unei multimi cu n puncte
  - 2*7 -2

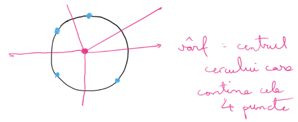
Cursul 11 - Diagrame Vornoi

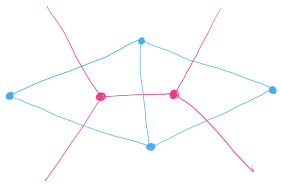
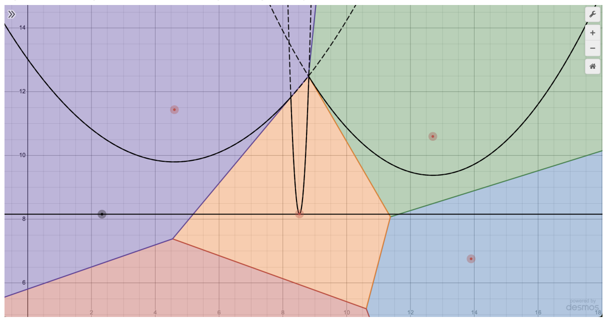
- Avem mai multe magazine (=puncte). Ce zone vor deservi acestea? (aka daca ai fi in punctul X, la ce magazin te-ai duce? - normal, cel ce ar fi cel mai aproape)
  - Delimitam in "zone de influenta"
- 2 puncte distincte:
  - 

- 3 puncte:
  - Distincte, coliniare:
    - 

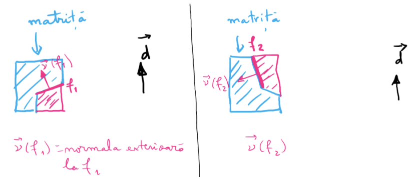
  - Necoliniare:
    - 

- 4 puncte:
  - Necoliniare, neconciclice:
    - 

  - Conciclice:
    - 

- 2 celule adiacente au in comun o muchie sau un varf
- Pentru n situri (= "magazinele", punctele carora vrem sa gasim "zonele de influenta") avem:
  - $n_v \leq 2n - 5$ (n_v = numar de varfuri in diagrama)
  - $n_m \leq 3n - 6$ (n_m = numar de muchii in diagrama)
- Numar de semidrepte din diagrama = numar de drepte pe acoperirea convexa a siturilor
- Obtinere triangulare din diagrame Voronoi:
  - 
  - Punctele albastre: cele pentru care am determinat diagrama Voronoi (in rosu)
  - Facem un nou graf:
    - Noduri = siturile (punctele albastre)
    - Muchii = intre celulele cu o muchie comuna
    - => Triangulare $\mathcal{T}_P$ **(triangulare Delaunay)**
  - O triangulare este legala ⇔ este o triangulare Delaunay.
- Algoritmul lui Fortune (O(n log n) timp si O(n) spatiu)
  - = metoda clasica (& eficienta) de determinare a diagramei Voronoi pentru o multime de puncte din $\mathbb{R}^2$
  - Principiu = dreapta de baleiere (sweep line)
    - 
    - La un moment dat, vom sti partea diagramei care nu mai poate fi influentata (= partea de sus), indiferent de ce ar veni mai jos
  - Beachline = curba parabolica. Un punct de pe curbe e situat la distanta egala de situl ce determina arcul si dreapta de baleiere
    - Este x-monotona
  - Curba parabolica se modifica la:
    - Site event:
      - Cand sweep line-ul trece printr-un sit
    - Circle event:
      - Cand trei arce consecutive ajunga sa se intersecteze intr-un punct (cel din mijloc reducandu-se la un punct) - aka dispare o curba parabolica
      - Determina punctele de pe diagrama Voronoi
  - Deci triangularea a unei multimi de puncte se poate face in timp O(n log n) si spatiu O(n), prin triangulare Delaunay

Cursul 12, 13 - Elemente de programare liniara

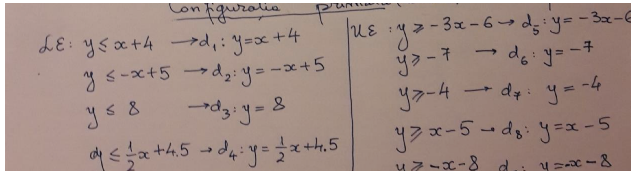
- Turnarea pieselor in matrite
  - Vrem sa turnam piese in matrite si sa le extragem, fara sa distrugem matrita
  - Unele obiecte pot ramane blocate - nu pentru toate obiectele exista o matrita adecvata
- Diverse orientari ale obiectelor pot genera diverse matrite (unele valide, altele nu)
- Conventii:
  - Toate fetele au macar o fata superioara (orizontala). Celelalte sunt fete standard
  - Directie admisibila = orientarea obiectului in care poate fi turnat si extras
- Proprietatea de a putea extrage un obiect:
  - Pentru fiecare fata, trebuie sa ne asiguram ca unghiul dintre directia fetei (="nornala exterioara") si directia de extragere este $\geq 90°$
    - 

  - Adica pentru fiecare vector trebuie sa verificam:
    - $\nu_x \cdot d_x + \nu_y \cdot dy + \nu_z \leqslant 0$
      - Sau, mai simplu de notat: $\nu_x \cdot x + \nu_y \cdot y + z \leqslant 0$
      - (si determinam x si y)
    - ^ e o ecuatie a unui semiplan
      - Deci, practic, trebuie sa facem intersectia tuturor semiplanelor. Daca e nevida, atunci putem extrage obiectul (directia fiind buna).
- Rezultate:
  - Algoritm pentru intersectia de n semiplane = timp O(n log n), memorie O(n)

- In medie, e nevoie de O(n) pentru a determina daca o intersectie de semiplane e nevida
- Poti determina daca un obiect poate fi turnat intr-o matrita in timp O(n^2) si spatiu O(n)
  - In acelasi timp si spatiu determini si matrita + directia necesara
- Dualitate:
  - E nevoie de 2 informatii pentru a indica un punct in plan (x si y)
  - Tot de 2 informatii e nevoie pentru a indica si o dreapta in plan (panta si a)
  - Corespondenta intre ele? Prin dualitate
  - Punctul $p = (p_x, p_y)$
    - Rezulta dreapta (duala lui p): $p^* : y = p_x x - p_y$
  - Dreapta $d : y = m_d x + n_d$
    - Rezulta punctul (dualul lui p): $d^* = (m_d, -n_d)$
- Proprietati dualitate:
  - Pastreaza incidenta:
    - Daca un punct p e pe dreapta d, atunci si duala d *va di pe dreapta p*
  - Pastreaza "ordinea":
    - p e situat deasupra dreptei d (neverticale) => d e *situat deasupra dreptei p*

Plan primal	Plan dual
Punct p	Dreaptă neverticală p^*
Dreaptă neverticală d	Punct d^*
Dreaptă determinată de două puncte	Punct de intersecție a două drepte
Punctul p deasupra dreptei d	Punctul d^* deasupra dreptei p^*
Segment	Fascicul de drepte (*wedge*)

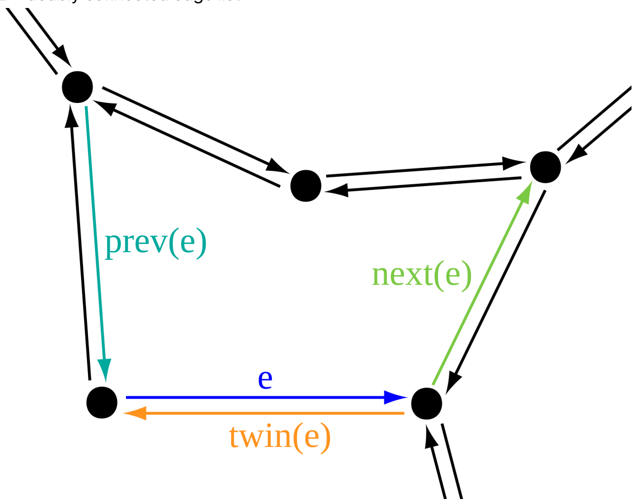
- Semiplane inferioare si semiplane superioare:
  - Avem $ax + by + c \leq 0$
  - Superior: $b < 0$
  - Inferior: $b > 0$
- Cand determinam intersectie de semiplane inf/sup, nu sunt toate relevante
- Un segment [pq] participa la frontiera superioara a acoperirii convexe a lui P ⇔ toate celelalte puncte sunt dedesuptul dreptei d = pq
- Determinarea intersectiilor:
  - Pentru intersectia de **semiplane inferioare**, luam frontiera superioara a acoperirii convexe a multimii de puncte din planul dual
  - Pentru intersectia de **semiplane superioare**, luam frontiera inferioara a acoperirii convexe a multimii de puncte din planul dual
- I.e. asa transformi semiplanele in drepte:



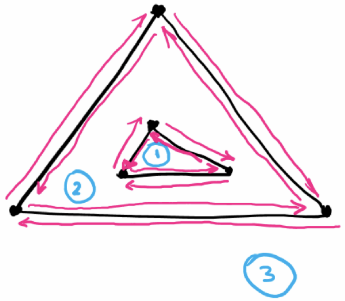
- Pe care mai apoi le transformi in puncte, prin dualitate, si aplici ce e scris mai sus pentru a vedea care sunt intersectiile pentru inferior si superior
- $\alpha\varepsilon$ = semiplanele inferioare
- $u\varepsilon$ = semiplanele superioare
- Abordare calitativa:
  - Vrem sa gasim cel mai bun punct ce minimizeaza functia f(x, y), date fiind niste constrangeri
  - Pentru asta, gasim regiunea fezabila (= intersectia semiplanelor date de constrangeri)
    - Regiunea fezabila poate fi marginita, nemarginita, vida

Cursul 14 - Probleme de localizare

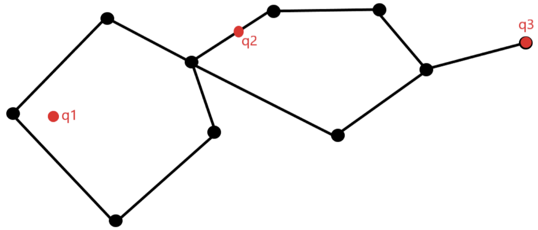
- DCEL = doubly connected edge list



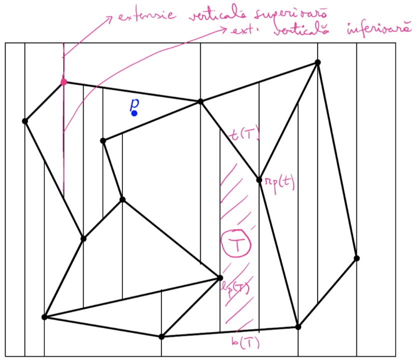
- Poligon cu goluri:



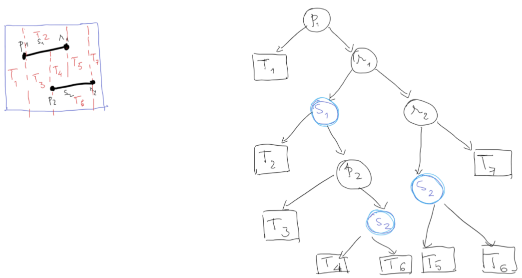
- Avem 3 fete
- Avem 4 frontiere:
  - 2 exterioare - parcurse cu ajutorul semi-muchiilor, a.i. poligonul sa fie la stanga frontierei, iar virajele sa fie la stanga
  - 2 interioare - poligonul e tot la stanga, dar virajele in varfurile convexe sunt la dreapta
- Problema: vrem sa spunem in ce zona (= pe ce fata/pe ce varf/pe ce muchie) se afla un punct pe o harta (cum ar fi pe Google Maps):



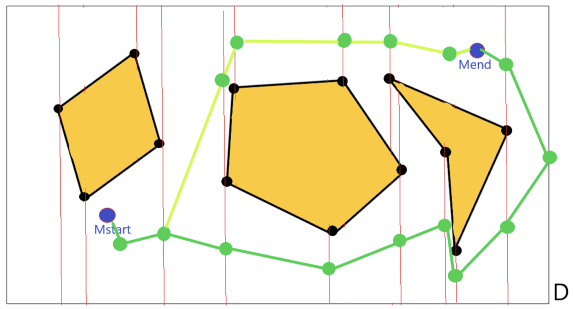
- Putem impartii harta in benzi verticale, sa vedem in ce banda se afla punctul, si apoi sa vedem pe ce zona din banda se afla mai exact
  - Nu foarte eficient - posibil memorie O(n^2)
- Optimizare: facem o descompunere in trapeze (trapezoidal map)



- Practic, dintr-un varf tragem o linie verticala (in sus si in jos) cat se poate (limitandu-ne, totusi, la un dreptunghi in care punem forma noastra)
- Fiecare fata va fi un trapez, un dreptunghi sau un triunghi
  - Mereu vor fi doua margini neverticale si 1/2 margini verticale
- Harta trapezoidala:
  - Va contine cel mult $6n + 4$ varfuri si cel mult $3n + 1$ trapeze (unde n e numarul de segmente)
  - Un trapez e adiacent cu cel mult 4 trapeze (ignorand trapezele de sus/jos?)
  - Structura de cautare asociata:



  - Constructie harta = timp O(n log n), spatiu O(n). Cautare punct arbitrar p: O(log n).
  - Gasire drum de la A la B. Gasim centrul de greutate al trapezului in care se afla A si al trapezului in care se afla B. Gaism mijloacele liniilor verticale. Construim un drum.



- Un drum liber de coliziuni intre doua puncte poate fi gasit in O(n) (daca exista).
