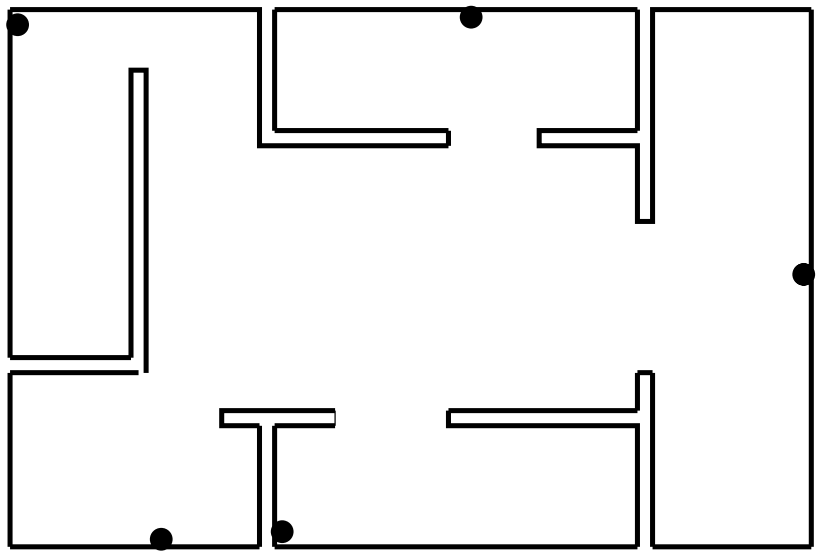
# Hálózattervezés Alapjai
# 2006

## 8: Művészeti Galéria Probléma – Őrzési / Megvilágítási problémák

# A művészeti galéria probléma

A művészeti galéria probléma (art galery problem):
A “művészeti galéria” megfigyelése kamerákkal / őrökkel.

- A galéria tervét egy síkbeli sokszöggel (polygon) modellezük.
- Feltesszük, hogy ez egy egyszerű sokszög, azaz
  - egy önmagát nem metsző poligonális lánc határolja és
  - nem tartalmaz lyukat.



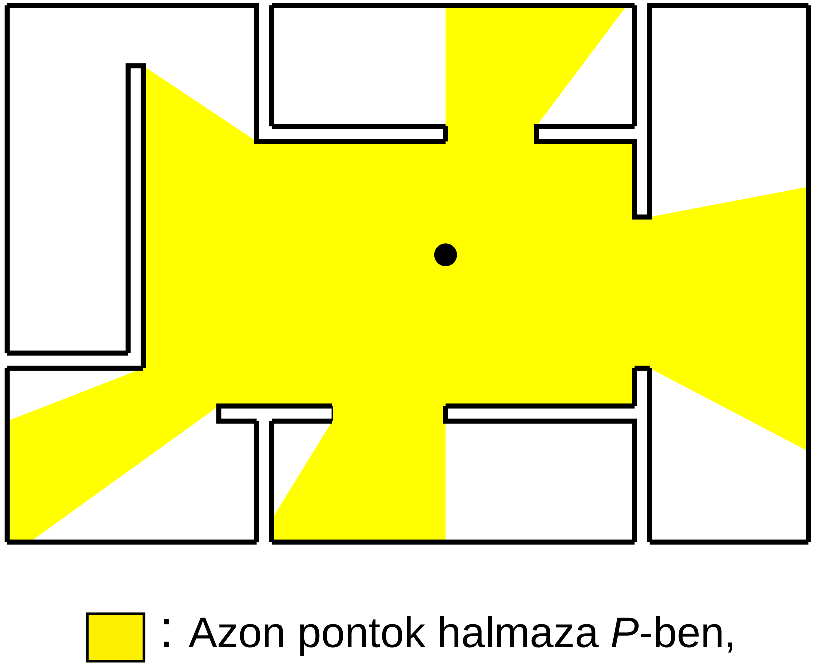
# A művészeti galéria probléma

- Legyen *P* egy egyszerű sokszög és *p* és *q* két pont *P*-ben.
- Azt mondjuk, hogy *p* és *q* látják egymást, ha *P* teljes egészben tartalmazza a *pq* szakaszt.

A művészeti galéria probléma:

- Adott: Egy egyszerű sokszög *P.*
- Feladat: Hol és hány pontot (őrt) kell elhelyezni *P*-ben, úgy hogy minden *P* beli pontot legalább egy őr lásson.

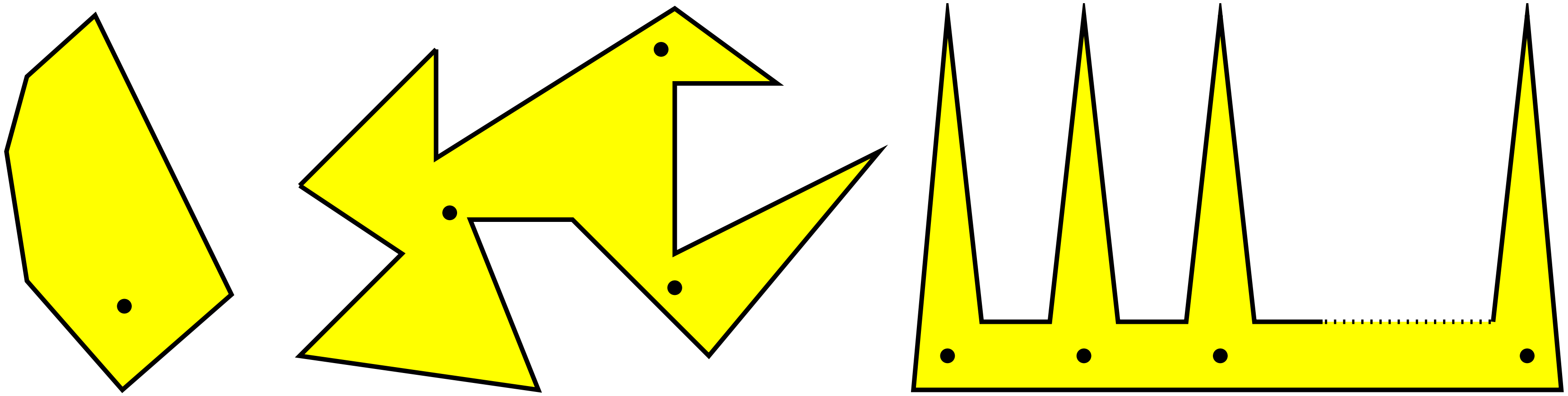
(őr: mozgásérzékelő szenzor, kamera, fényforrás,…)



: Azon pontok halmaza *P*-ben, melyeket *p* lát

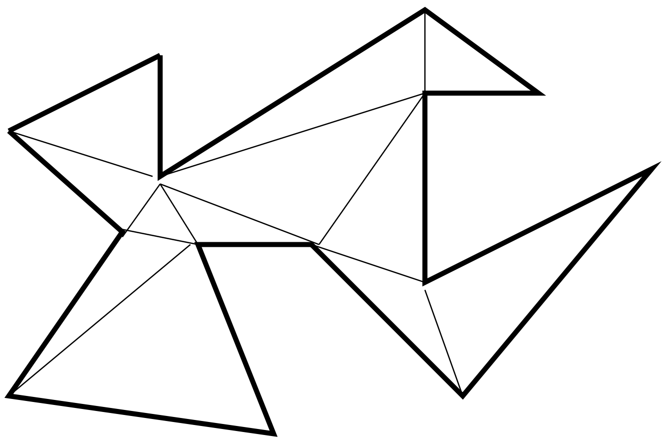
# A művészeti galéria probléma

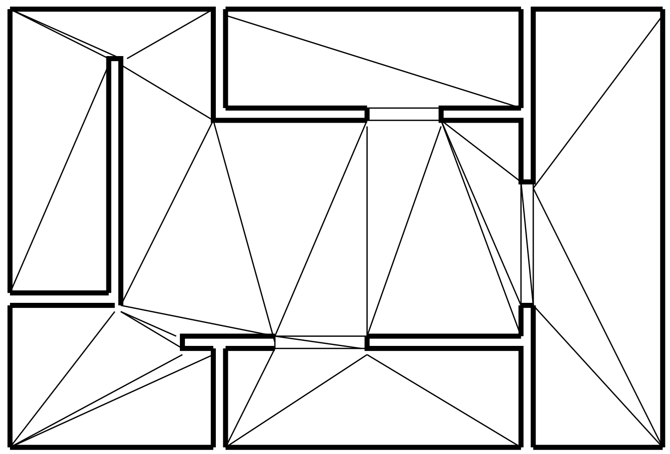
- Az őrök minimális számának kiszáítása NP-nehéz.
- De: Megmutatjuk, hogy $\lfloor n/3 \rfloor$ őr elegendő, ahol $n$ a csúcsok száma P-ben.
- Néhány esetben $\lfloor n/3 \rfloor$ őr szükséges is.



## Sokszögek háromszögelése

- Egy *P* sokszög háromszögelése ($\Delta$-elése) egy planár felosztása *P*-nek, melynek
  - csomópontjai *P* csúcsai,
  - élei egyenes szakaszok a csúcspárok között (az éleket átlóknak is nevezik),
  - felületei pedig háromszögek.





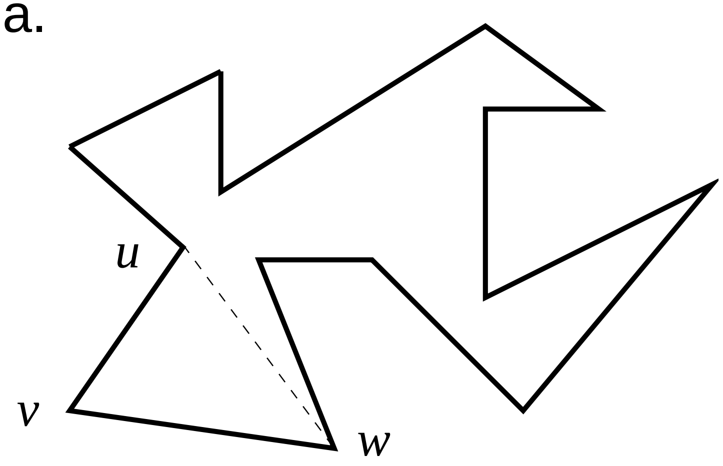
# Sokszögek háromszögelése

**Lemma 1:** Minden *n* csúcsú egyszerű sokszög *P* háromszögelésében pontosan *n*-2 háromszög van.

**Biz.:** Indukció *n* szerint.

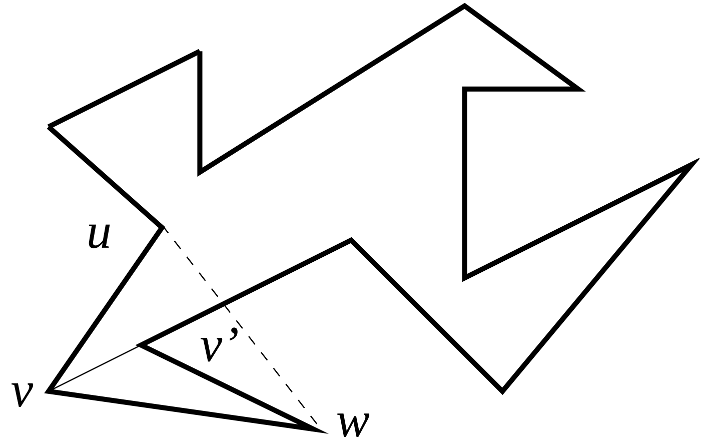
- *n*=3 esetén igaz, *P* egy háromszög.
- Legyen *n*>3. Feltétel, a lemma állítása igaz minden *m*<*n* esetén.
  Először megmutatjuk, hogy *P*-nek van egy átlója.
  Ezután alkalmazzuk az indukciós feltételt a két sokszögre, amire ez az átló osztja *P*-t.

  Legyen *v* az a csúcsa *P*-nek, melynek koordinátái lexikografikusan minimálisak.
  Legyen *u* és *w* a két *v*-vel szomszédos csúcs.
  - Ha az *uw* szakasz teljesegészben *P*-ben van, akkor *uw* egy átló.



# Sokszögek háromszögelése

- Ha *uw* nincs teljesegészen *P*-ben, akkor *uvw* Δ tartalmazza *P* egy vagy több csúcsát. Legyen ezen csúcsok közül *v´* az a csúcs, melynek távolsága *uw*-től maximális. Ekkor a *vv´*, szakasz nem metszheti *P* egy oldalát sem, mert egy ilyen oldal egyik végpontja az *uvw* Δ belsejében lenne és a távolsága *uw*-től nagyobb lenne mint *v´*-nek. Ezért *vv´* egy átló.

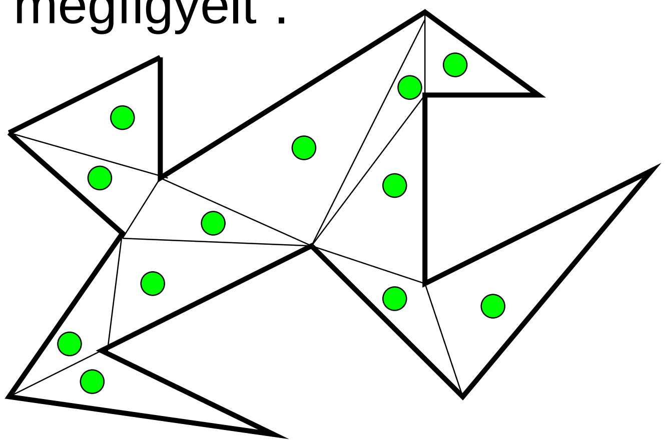


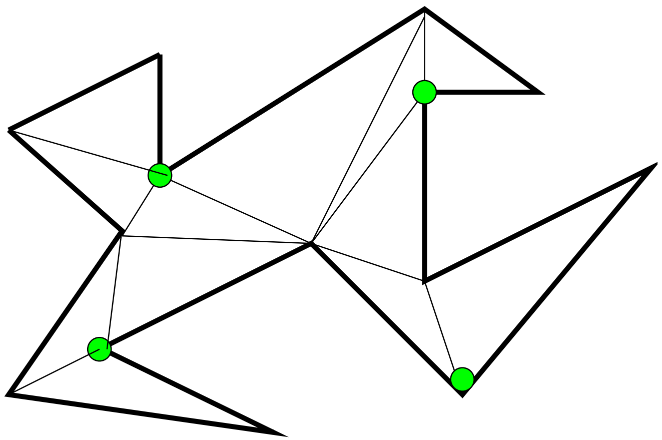

Minden átló *P*-t két sokszögre $P_1$-re és $P_2$-re osztja. Legyen $m_1$ és $m_2$ a csúcsok száma $P_1$-ben és $P_2$-ben. Mivel $m_1 < n$ és $m_2 < n$, az indukciós feltétel szerint $P_1$ és $P_2$ Δ-elhető, és ezzel *P* is.

Mivel $P_1$-nek és $P_2$-nek pontosan két közös csúcsa van, $m_1+m_2=n+2$. $P_1$ és $P_2$ Δ-elése az indukciós feltétel szerint $m_1$-2 és $m_2$-2 Δ-t tartalmaz. Ezért *P* Δ-elése *n*-2-t. □

# Sokszögek háromszögelése – műgaléria probléma

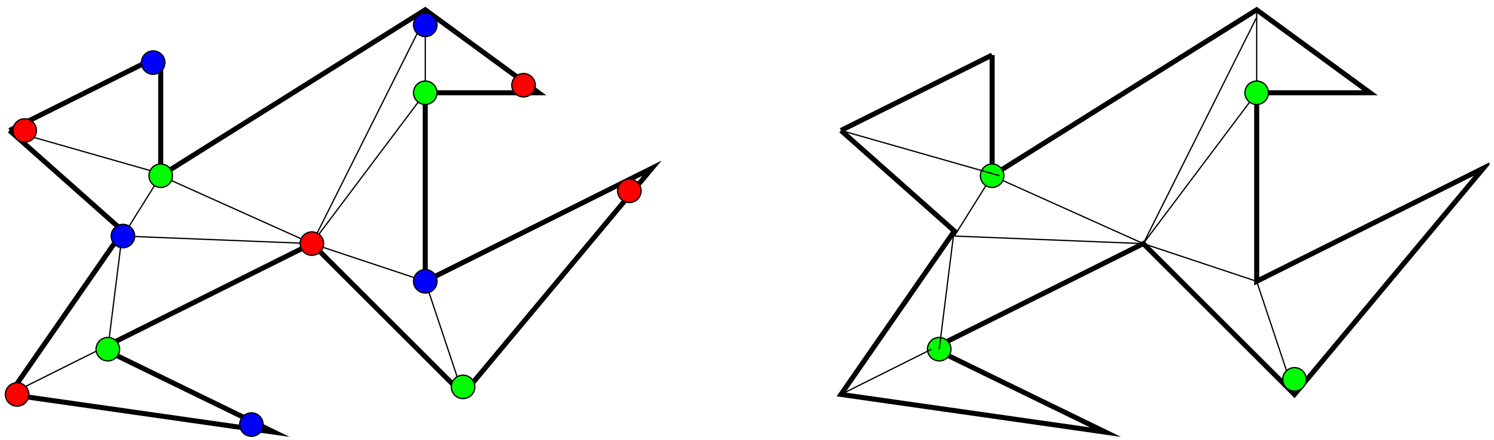
- Lemma 1 implikálja, hogy minden *n* csúcsú egyszerű *P* sokszögben elegendő *n* - 2 őr, Δ-enként 1. Ez azonban túl „nagyvonalú“.
  Ha az őröket bizonyos jól kiválasztott csúcsokban helyezzük el, akkor kevesebb őr is elegendő.
- A „stratégia“:
  - Válasszunk ki a csúcsok egy (lehetőleg kicsi) részhalmazát, úgy hogy a Δ-elés minden Δ-ében legalább egy kiválasztott csúcs van.
  - Helyezzük el az őröket ezekben a csúcsokban. Ekkor az egész *P* “megfigyelt”.





## Sokszögek háromszögelése – műgaléria probléma

- Szinezzük a $\Delta$-elés csúcsait 3 színnel, úgy hogy a csúcsok, melyek egy éllel össze vannak kötve, különböző színűek legyenek.
  - Ekkor minden $\Delta$ csúcsai 3 különböző színt kap.
  - Ez 3 színosztályra osztja a csúcsok halmazát.
- Kiválasztjuk a legkisebb színosztályt és az őröket ezekben a csúcsokban helyezzük el.
  - Meg kell mutatni, hogy a „$\Delta$-elés 3 színnel szinezhető”



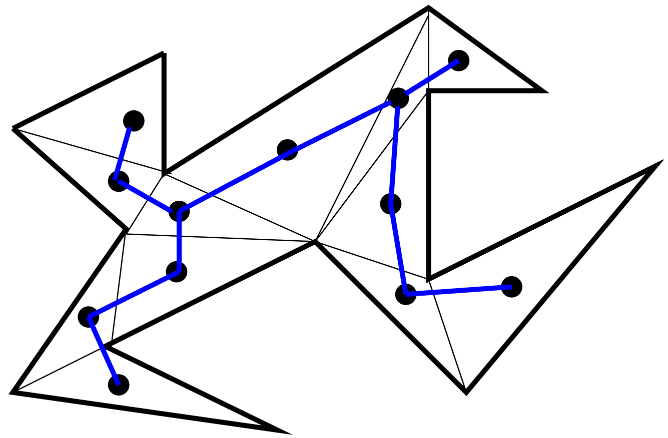
## Sokszögek háromszögelése – műgaléria probléma

**Lemma 2:** Tekintsük egy sokszög Δ-elését *T*-t egy gráfként. Ekkor *T* 3 színnel szinezhető.

**Biz.:**

Egy planár gráf *G* duál-gráfja *G** az a gráf, melynek csomópontjai *G*-nek a felületei. Két csomópont *G**-ben pontosan akkor van összekötve egy éllel, ha a megfelelő felületek *G*-ben egy él által szomszédosak.

Tekintsük *T* duál-gráfját *T**-t (a külső felület kivételével).

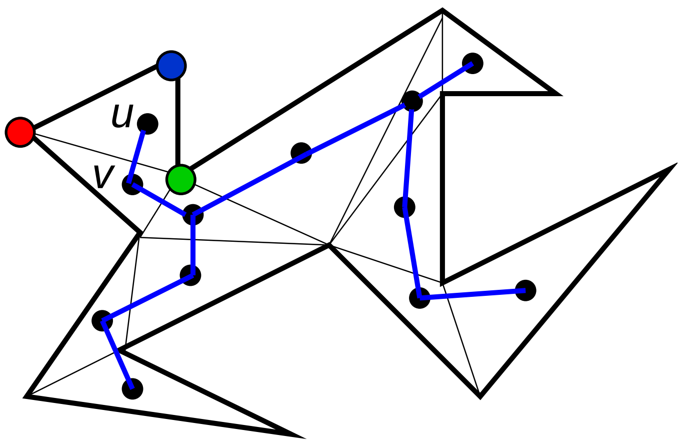


Mivel minden átló P-t két sokszögre osztja, *T** bármelyik élének törlése két komponensre osztaná *T**-t. Tehát *T** egy fa. (Ez nem lenne érvényes olyan sokszögre, amelyben lyuk van.) Továbbá, *T** foka 3, mivel minden Δ-höz *T*-ben legfeljebb 3 szomszédos Δ van.

# Sokszögek háromszögelése – műgaléria probléma

*T* szinezéséhez *T** bejárását használjuk mélységi keresés (DFS) által.

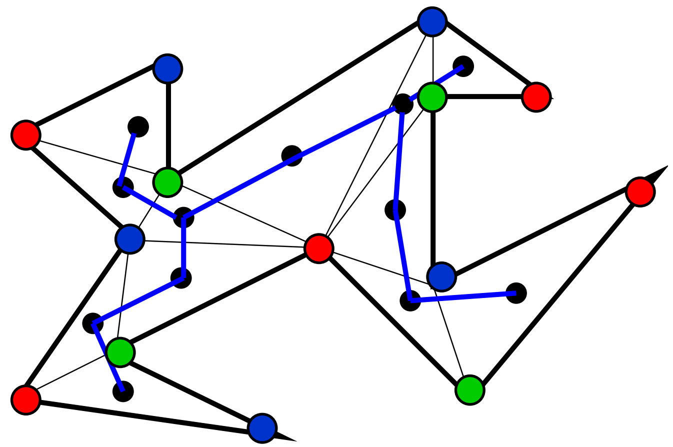
- Egy tetszőleges *T** beli csomóponttal kezdünk.
- *T* megfelelő Δ-ének a csúcsait 3 különböző színnel szinezzük.
- Megőrizzük azt az invariánst, hogy *T* minden elért Δ-ének csúcsai helyesen vannak szinezve.
- Amikor *T**-ben bejárunk egy (*u*,*v*) élt, az *u*-nak és *v*-nek megfelelő Δ-eknek $t_u$-nak és $t_v$-nek *T*-ben van egy közös oldala.
- Ekkor a közös oldal két végpontja már szinezett.
- A harmadik csúcs szinezéséhez $t_v$-ben a 3-adik színt használjuk.



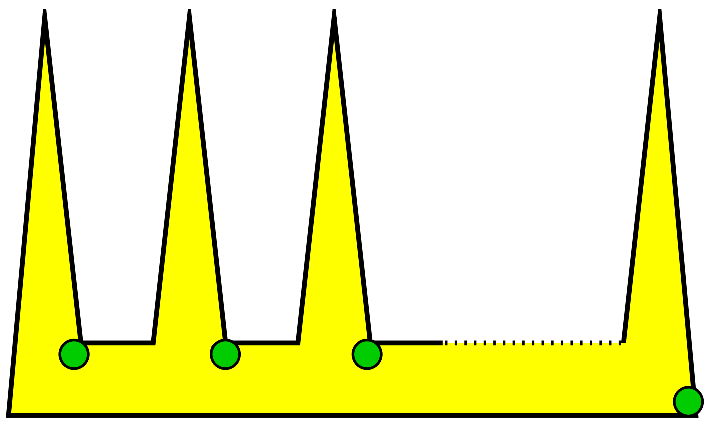

A DFS után *T*-ben egy helyes szinezést kapunk 3 színnel. □

## Sokszögek háromszögelése – műgaléria probléma

- Mivel a szinezés után a 3 színosztály közül legalább egy $\leq \lfloor n/3 \rfloor$ csúcsot tartalmaz és minden $\Delta$-ben pontosan egy csúcs ezzel a színnel van szinezve, a következő teljesül:

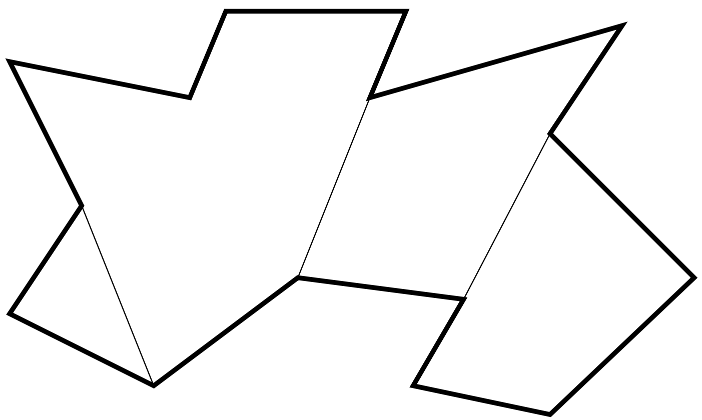


**Tétel 1:** Legyen *P* egy egyszerű sokszög *n* csúccsal. Akkor *P* legfeljebb $\lfloor n/3 \rfloor$ őrrel mindig megfigyelhető.
Vannak olyan sokszögek, melyekben $\lfloor n/3 \rfloor$ őr szükséges is. □



# A háromszögelés kiszámítása

- Először *P*-t u.n. monoton sokszögekre bontjuk.
- Ezután ezeket a monoton sokszögeket Δ-eljük.
- Akkor mondjuk, hogy egy egyszerű sokszög *P* egy *h* egyenes szerint monoton, ha minden *h*-ra ortogonális *h´* egyenesre $P \cap h'$ összefüggő, azaz $P \cap h'$ vagy egy pont, vagy egy szakasz.
- *P* *x*-monoton, ha az *x*-tengely szerint monoton.

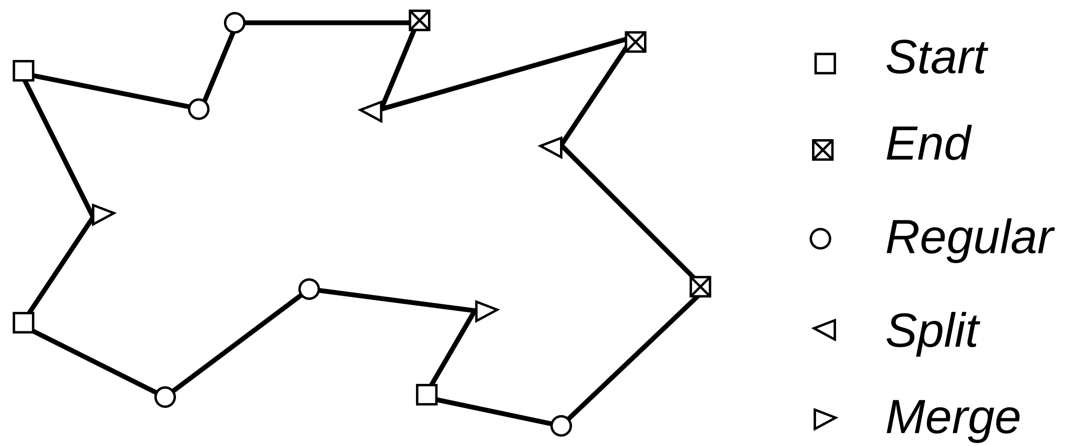


## *P* felbontása *x*-monoton sokszögekre

5 csúcstipust különböztetünk meg:

Legyen $v$ egy csúcs és $\alpha(v)$ belső szög $v$-nél $P$-ben. Ekkor $v$ egy

- Start-csúcs: $v$ mindkét szomszédja jobbra van $v$-től és $\alpha(v)<\pi$,
- Split-csúcs: $v$ mindkét szomszédja jobbra van $v$-től és $\alpha(v)>\pi$,
- End-csúcs: $v$ mindkét szomszédja balra van $v$-től és $\alpha(v)<\pi$,
- Merge-csúcs: $v$ mindkét szomszédja balra van $v$-től és $\alpha(\mathrm{v})>\pi$,
- Regular-csúcs: különben.

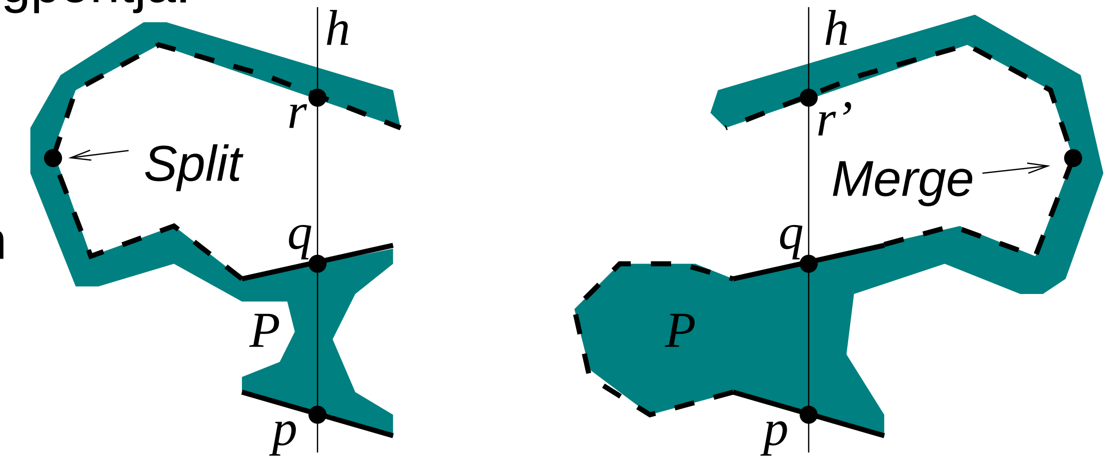


# *P* felbontása *x*-monoton sokszögekre

**Lemma 3:** Egy *P* sokszög akkor *x*-monoton, ha nem tartalmaz se split-csúcsot se merge-csúcsot.

**Biz.:** Tegyük fel, hogy *P* nem *x*-monoton, akkor létezik egy vertikális egyenes *h*, melyre $h \cap P$ több mint 1 összefüggő komponenst tartalmaz. Válasszuk *h*-t úgy, hogy a legmélyebben lévő komponens egy szakasz nem egy pont. Legyen *p* és *q* a szakasz két végpontja.

Kövessük *P* határát *q*-ból balra indulva addig, amíg egy $r \in P \cap h$ metszéspontot nem találunk. Ha $p \neq r$, akkor útközben kellett lenni egy split-csúcsnak. Különben kövessük *P* határát *q*-ból jobbra indulva addig, amíg egy $r' \in P \cap h$ metszéspontot nem találunk. Ekkor útközben kellett lenni egy merge-csúcsnak. □

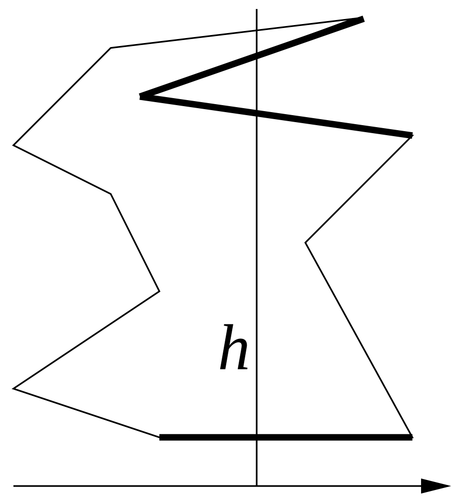


## *P* felbontása *x*-monoton sokszögekre – plane swep

- Így tehát a split- és a merge-csúcsokat kell *P*-ben kezelni. A sokszögeknek *P* felbontása után nem szabad ilyen csúcsokat tartalmazniuk.
- *P* felbontásához egy plane-sweep eljárást használunk.

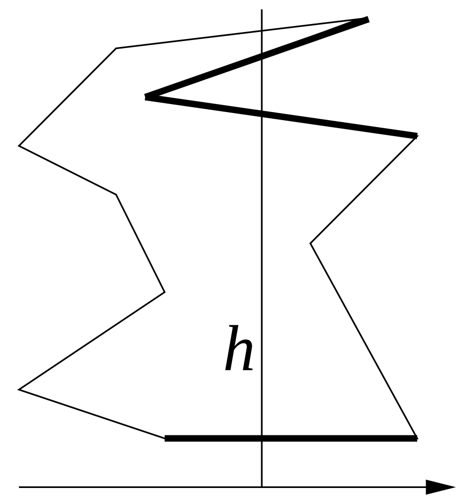
### Plane sweep

- Ötlet:
- csúsztassunk egy függőleges egyenest (sweep-line) balról jobbra a síkon
- Egy sweep-status adatstruktúrában tároljuk a sweep-line és a szcenárió metszetét.
- A sweep-status adatstruktúrát csak bizonyos eseményeknél (event point) kell aktualizálni (amikor az egyenes és a szcenárió metszete változik).



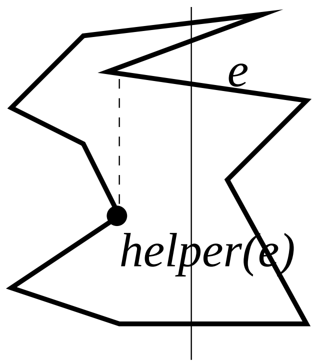
## *P* felbontása *x*-monoton sokszögekre – plane swep

- Sweep-Status *T*:  egy bináris keresőfa.
  Az algoritmus futása alatt *T* mindig *P*-nek azon oldalait (éleit) tárolja, melyek *h*-t metszik, a metszéspontok *y*-koordinátáinak megfelelően rendezve.
- Eseménypontok: *P* csúcsai. Ezeket egy *Q* várakozási sorban tároljuk, ahol *Q* egy Priority-Queue.
  - Egy csúcs prioritása a csúcs *x*-koordinátája (pontosabban: a csúcs koordinátái lexikorafikusan).
  - *Q*-t rendezett listaként is implementálhatjuk, mivel minden esemény kezdettől fogva ismert.

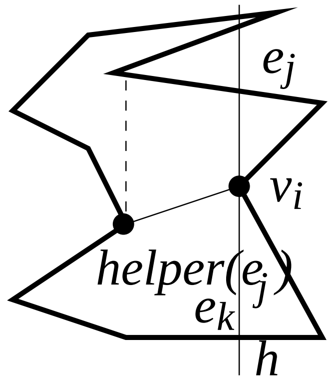


# *P* felbontása *x*-monoton sokszögekre – plane swep

- *P* Minden e oldalához, amelyet a Sweep-Status *T* tartalmaz, tárolunk egy pointert: *helper(e)*. Ez *P*-nek arra a legjobboldalibb *v* csúcsára mutat,
  - amely balra van *h*-tól és
  - *P* belsejében *v* és *e* egy függőleges szakasszal összeköthető.
  - Ha nincs ilyen csúcs, akkor *helper(e)* a bal végpontja *e*-nek.




- Ha *h* elér egy $v_i$ split-csúcsot, akkor meg kell találni *P*-nek azokat az $e_j$ és $e_k$ éleit, amelyek a *T*-ben direkt $v_i$ felett illetve direkt $v_i$ alatt vannak. Ekkor a szakasz $v_i$-től *helper*($e_j$)-hez (és *helper*($e_k$)-hoz) nem metszheti *P* egy másik élét és így teljesen *P*-ben van.



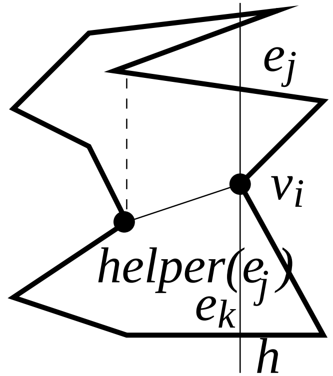

- Miután *h* elért egy $v_i$ split-csúcsot, $v_i$ lesz $e_j$ és $e_k$ új *helper()*-je és a két $v_i$-hez incidens élé.
- Elegendő a *helper()* mutatót csak azoknál az éleknél tárolni, amelyek direkt az újonnan elért csúcs fölött vannak *T*-ben.

# *P* felbontása *x*-monoton sokszögekre – plane swep

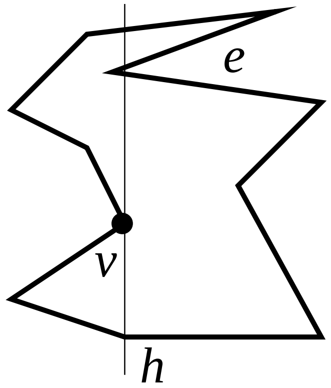
Az események feldolgozása:

Legyen *v* az aktuális csúcs, amit *h* éppen elér.

- *v* split-csúcs: Keressük meg *P* azon *e* és *e´* élét, ami direkt *v* felett és direkt *v* alatt van *T*-ben. Fűzzük be az átlót, amely *v*-t és *helper(e)*-t összeköti. Fűzzük be *T*-be *P*-nek a *v*-hez incidens két élét. Ezután az *e* és a *v*-hez incidens alsó oldal *helper()* mutatóját állítsuk *v*-re.




- merge-csúcs: Töröljük a *v*-hez incidens két élt *T*-ből. Keressük meg azt az *e* élt, ami direkt *v* fölött van *T*-ben. Legyen *helper(e)*:=*v*. (Később még visszatérünk erre az esetre)

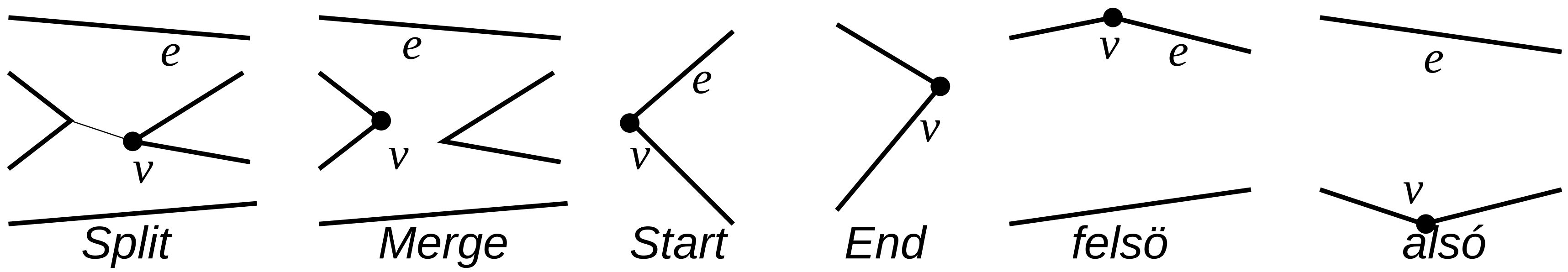



- start-csúcs: Fűzzük be a két incidens élet *T*-be. A felső él *helper()* mutatója legyen *v*.

- end-csúcs: Töröljük a két incidens élt *T*-ből.

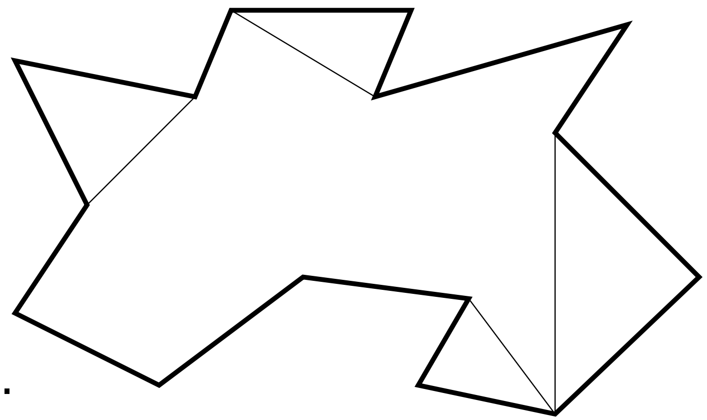
## *P* felbontása *x*-monoton sokszögekre – plane swep

- <u>felső reguláris-csúcs</u>: Cseréljük ki *T*-ben az élt, ami balról incidens *v*-hez a jobbról incidens *e* élre. Legyen *helper(e)*:=*v*.
- <u>alsó reguláris-csúcs</u>: Cseréljük ki *T*-ben az élt, ami balról incidens *v*-hez a jobbról incidens *e* élre. Legyen *e* az az él, ami *T*-ben direkt *v* felett van. Legyen *helper(e)*:=*v*.



# *P* felbontása *x*-monoton sokszögekre – plane swep

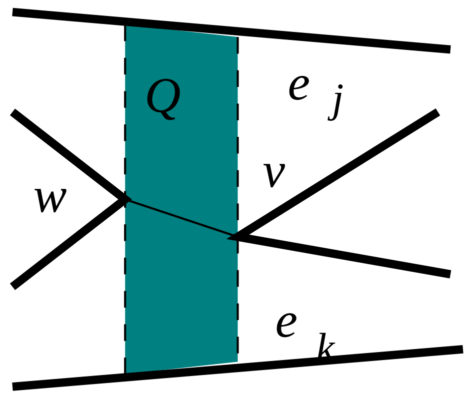
- A split-csúcsokat teljesen feldolgoztuk.
  Átlókat fűztünk be mindegyiknél. Az új sokszögekben már nincs split-csúcs.
  A merge-csúcsokat kell még teljesen feldolgozni.
- <u>1. lehetőség</u>: Végrehajtunk egy másik *plane sweep*-et hátrafele.
- <u>2. lehetőség</u>: Csak egy *plane sweep* (csak balról jobba).
  - Minden alkalommal, amikor egy *e* élnél *helper(e)* megváltozik, teszteljük, hogy a régi *helper()*-csúcs egy merge-csúcs-e. Ha igen, akkor fűzzük be az átlót a régi és az új *helper()*-csúcs között.
  - Ugyanígy, mindig, amikor *h* egy e élt elhagy, teszteljük, hogy *helper*(*e*) egy merge-csúcs-e. Ha igen, fűzzük be az átlót *e* jobb oldali végpontja és *helper*(*e*) között.
  - Ennek ugyanaz a hatása mint a „hátrafele plane sweep“-nek.



# *P* felbontása *x*-monoton sokszögekre – plane swep

Helyesség:

- Az új sokszögekben nyilvánvalóan nincs se split- se merge-csúcs.
- Azt kell még megmutatni, hogy nem fűzünk be olyan átlót, amely *P* valamelyik élét vagy egy másik átlót metsz.
- Egyszerűség végett tegyük fel, hogy nincs két egyenlő *x*-koordinátájú csúcs (a kiterjesztés az általános esetre egyszerű: a lexikografikus sorrend segítségével). Legyen *wv* egy átló, amit akkor fűztünk be, amikor egy split-csúcsot elértünk. A *Q* tartomány a*T*-ben *v*-hez szomszédos élek és a függőleges szakaszok között *w*-n és *v*-n keresztül nem tartalmaz csúcsot a helper() definíciója miatt. Ezért a *wv* átló nem metszhet se másik átlót se *P*-nek egy élét.




- A merge-csúcsoknál befűzött átlókra hasonló érvek ismételhetők.

## *P* felbontása *x*-monoton sokszögekre – plane swep

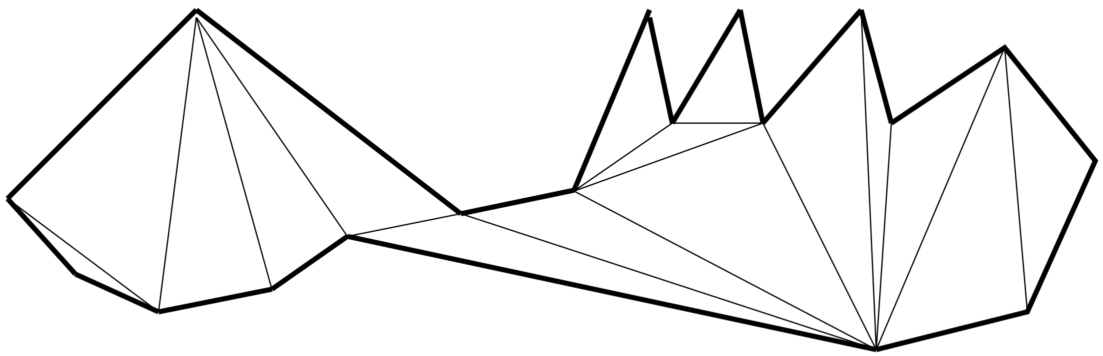
Futási idő és tárigény:

- Minden esemény (csúcs) feldolgozása $O(\log n)$ időt igényel.
- Összesen: $O(n \log n)$ idő. Tárigény $O(n)$.

**Lemma 5**: Egy egyszerű sokszög felbontható *x*-monoton sokszögekre $O(n \log n)$ idő alatt $O(n)$ tárigénnyel. □

## Monoton sokszögek háromszögelése

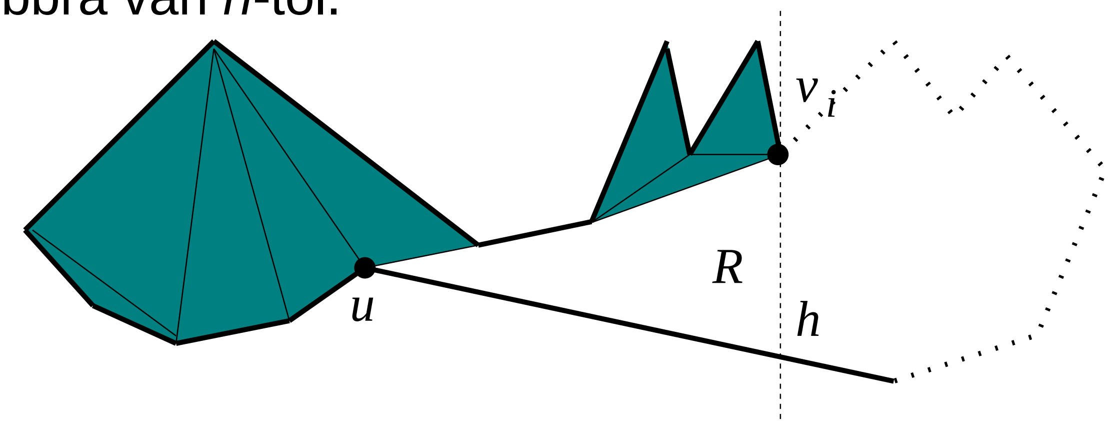
- Legyen *P* egy *x*-monoton sokszög.
- Egyszerűség kedvéért tegyük fel, hogy *P* nem tartalmaz egyenlő *x*-koordinátájú csúcsokat.
- Egy plane sweep-et hajtunk végre balról jobbra. Ennek során *P*-be átlókat fűzünk be, amikor csak lehetséges.



## Monoton sokszögek háromszögelése

Invariáns: Legyen $v_i$, $i \geq 2$, az a csúcs *P*-ben, amelyet a *h* sweep line éppen elért. Legyen *R* a nem-háromszögelt tartomány *P*-ben *h*-tól balra. Legyen *u* a legbaloldalibb csúcs *R*-ben. Ekkor

- *R*-t két *x*-monoton lánc határolja, a felső lánc és az alsó lánc. Mindkét lánc legalább egy élt tartalmaz.
- Ha a lánc $v_i$-től *u*-hoz több mint egy élt tartalmaz, akkor ez a lánc egy u.n. reflex-lánc, azaz a lánc minden belső csúcsánál a belső szög legalább π.
- A másik lánc csak egy élt tartalmaz, melynek bal végpontja *u* és jobb végpontja jobbra van *h*-tól.

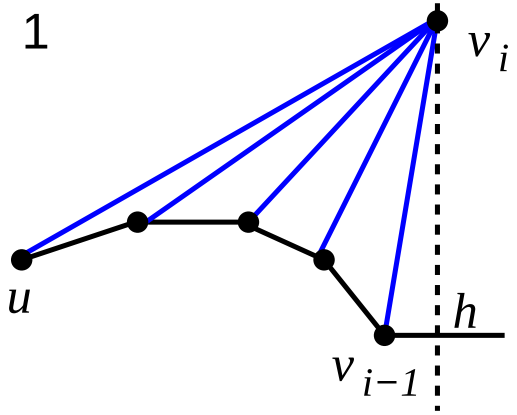


## Monoton sokszögek háromszögelése

$i$=2: az invariáns $u=v_1$-gyel teljesül. A $v_2v_1$ lánc csak egy élt tartalmaz, a másik lánc pedig abból a másik élből áll, amely $v_1$-hez incidens.

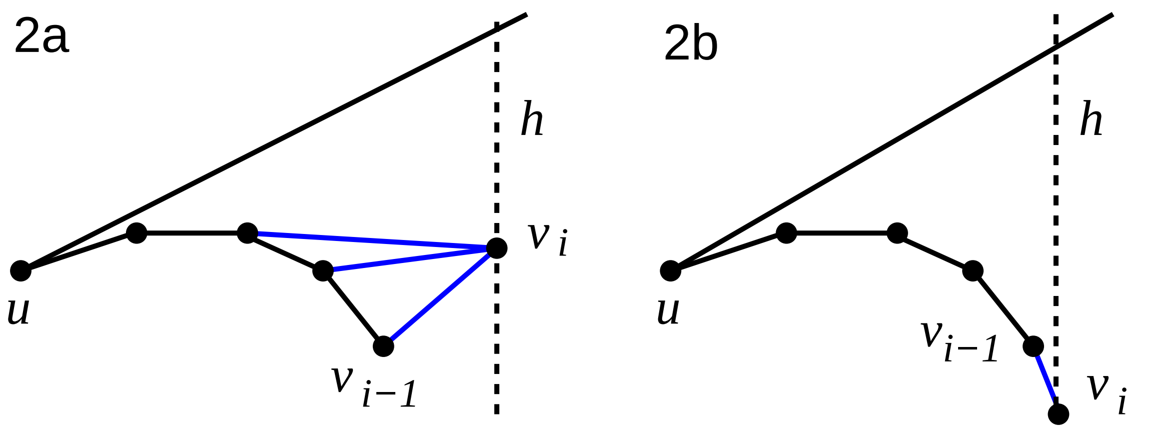
i>2: Tegyük fel, hogy az invariáns teljesül $v_{i\text{-}1}$-re. Az algoritmusnak a következő eseteket kell kezelni:

- 1. eset: $u,...,v_{i\text{-}1}$ egy reflex-láncot alkot és $v_i$ a másik láncon van. Ekkor fűzzünk be egy átlót $v_i$-től a reflex-lánc minden csúcsához $u$-ig (exkluzív $u$). Ezután legyen $u:=v_{i\text{-}1}$. Most a reflex-lánc egyetlenegy élt tartalmaz $v_iu$-t.



## Monoton sokszögek háromszögelése

- 2. eset: $v_i$ ugyanazon a láncon van mint $v_{i-1}$.
  Ekkor menjünk a láncon $v_i$-től hátrafele és fűzzünk be minden látható csúcshoz egy átlót, amíg el nem érjük az első csúcsot, ami $v_i$-ből már nem látható. (Lehetséges, hogy egy átlót se fűzünk be. (2b. Eset).)
  Ezután a $v_i$-ből látató csúcsokat töröljük a láncból.
  Ekkor az új lánc $v_i$-től $u$-ig egy reflex-lánc.



# Monoton sokszögek háromszögelése

Implementálás:

- A reflex-lánc csúcsai egy verembenn tárolhatók.
- Egy flag adja meg, hogy a verem a felső vagy az alsó láncot tárolja.
- Tegyük fel, hogy minden csúcshoz tudjuk, hogy az alsó vagy a felső láncon van.

Elemzés:

- Ha *P* csúcsai sorba vannak rendezve balról jobbra, akkor a $\Delta$-eléshez $O(n)$ idő szükséges.
- *P* csúcsainak balról jobbra rendezet sorrendje $O(n)$ idő alatt kiszámolható a csúcsok óramutatóval ellentétes sorrendjéből.
- A $\Delta$-elés során összesen $O(n)$ pop-, push-operációt iránytesztet ($v_j$ a láncon pontosan akkor látható $v_i$-ből, $j+1<i$, ha $\angle v_i v_{j+1} v_j < \pi$) hajtunk végre, $O(1)$ idő alatt operációnként.
- Az adatstruktúrák tárigénye $O(n)$.

## Sokszögek háromszögelése – műgaléria probléma

**Lemma 6:** Legyen $P$ egy $x$-monoton sokszög $n$ csúccsal. Akkor $P$ egy háromszögelése $O(n)$ idő alatt $O(n)$ tárigénnyel kiszámítható. □

**Tétel 2:** Legyen $P$ egy egyszerű sokszög $n$ csúccsal. Akkor $P$ egy háromszögelése $O(n \log n)$ idő alatt $O(n)$ tárigénnyel kiszámítható. □

**Következmény:** Legyen $P$ egy egyszerű sokszög $n$ csúccsal. Akkor a $\lfloor n/3 \rfloor$ őr elhelyezése, amelyek $P$-t megfigyelik, $O(n \log n)$ idő alatt $O(n)$ tárigénnyel kiszámítható. □

# Irodalom

[1]: Joseph O´Rourke: **Art Galery Theorems and Algorithms**. *Oxford University Press*,1987.

[2]: Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf: **Computational Geometry, Algorithms and Applications**. *Springer-Verlag,* 1997.