

Triangulace polygonu

Petr Buchal, Martin Ivančo

Fakulta informačních technologií

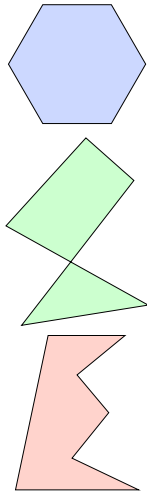
Vysoké učení technické v Brně

{xbucha02, xivanc03}@stud.fit.vutbr.cz

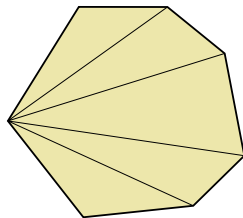


6. května 2020

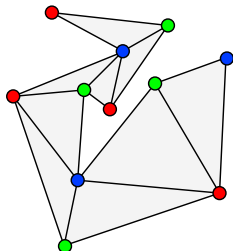
- Polygon je část roviny vymezená konečnou sekvencí navazujících úseček.
- Základní vlastnosti polygonů:
 - počet stran
 - **konvexnost** – jakákoliv přímka protínající polygon ho protíná maximálně dvakrát
 - **jednoduchost** – hranice polygonu neprotíná sebe sama
 - **konkávnost** – nekonvexní a zároveň jednoduchý polygon
 - **monotónnost** k přímce p – každá přímka kolmá na p protíná polygon nejvíce dvakrát



- Dekompozice polygonu na trojúhelníky.
- Konvexní polygon můžeme jednoduše triangulovat v lineárním čase pomocí **vějířové triangulace** – vybereme jeden vrchol a přidáváme diagonály od tohoto vrcholu ke všem ostatním.
- Vějířová triangulace se může také použít na konkávní polygon s jediným konkávním vrcholem – ten ale musíme použít jako bod, od kterého budeme přidávat diagonály.
- Počet všech možných triangulací konvexního polygonu s n stranami je výjádřen $(n - 2)$. **Catalanovým číslem:**
$$C_n = \frac{(2n)!}{(n+1)!n!}.$$



- Metoda triangulace pro jednoduché polygony založena na **teorému dvou uší**, podle kterého má každý jednoduchý polygon s více než 3 vrcholy alespoň 2 uši. Ucho polygonu je definováno jako vrchol v takový, že úsečka mezi jeho sousedy leží plně uvnitř polygonu.
- Podle tohoto teorému tedy můžeme vytvořit algoritmus, který postupně vyhledává uši a odstraňuje je, až z polygonu zůstane pouze trojúhelník. Ačkoli je tento algoritmus jednoduchý na implementaci, jeho složitost je poměrně neoptimální – $O(n^2)$.



Algorithm 1: Triangulace jednoduchého polygonu

Input: List vrcholů `polygon`

Output: List trojic vrcholů `triangles`

if `clockwise(polygon)` **then**

`reverse(polygon);`

end

for `v` **in** `polygon` **do**

if `is_ear(v, polygon)` **then**

`append(v, ears);`

end

end

while not `empty(ears)` **and** `length(polygon) >= 3` **do**

`append(remove_ear(polygon, ears), triangles);`

end

Algorithm 2: remove_ear

Input: List vrcholů **polygon**, List vrcholů **ears**

Output: Trojice vrcholů **triangle**

$v = \text{pop}(\text{ears});$

$\text{triangle} = \{\text{previous}(v), v, \text{next}(v)\};$

if $\text{length}(\text{polygon}) > 3$ **then**

$\text{neighbors} = \{\text{previous}(v), \text{next}(v)\};$

for n **in** **neighbors** **do**

if not n **in** **ears** **and** $\text{is_ear}(n, \text{polygon})$ **then**

$\text{append}(n, \text{ears});$

end

if n **in** **ears** **and** not $\text{is_ear}(n, \text{polygon})$ **then**

$\text{remove}(n, \text{ears});$

end

end

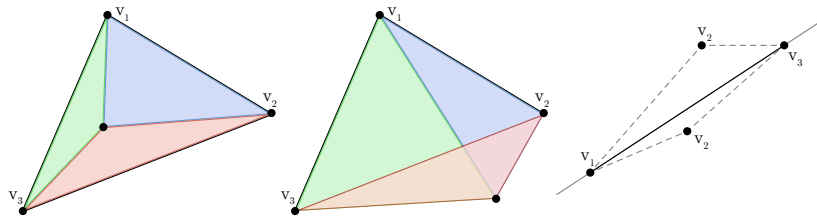
end

Algorithm 3: is_ear

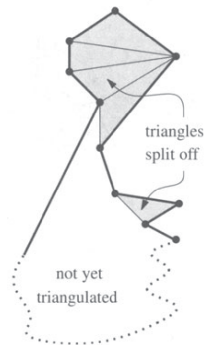
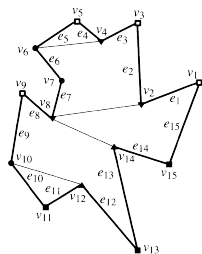
Input: Vrchol v , List vrcholů polygon

Output: Boolean ear

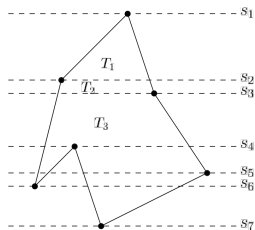
$\text{ear} = \text{contains_no_points}(\text{prev}(v), v, \text{next}(v), \text{polygon})$ and
 $\text{is_convex}(\text{prev}(v), v, \text{next}(v))$ and $\text{area}(\text{prev}(v), v, \text{next}(v)) > 0$;



- V knize *Computational Geometry* (de Berg et al., 2008) je uvedena složitější, nicméně také rychlejší metoda.
- Tato metoda také pracuje nad jednoduchými polygony a dělí se na dvě části – **rozdělení polygonu na monotónní části** (se složitostí $O(n \log n)$) a **triangulace monotónního polygonu** (se složitostí $O(n)$).
- Jelikož celkový počet vrcholů monotónních částí je n , celý algoritmus pracuje se složitostí **$O(n \log n)$** .
- Plný algoritmus je ve zmíněné knize podrobně vysvětlen.



- Jak bylo ukázáno, monotónní polygon umíme triangulovat v lineárním čase. Pokud tedy dokážeme nalézt dekompozici jednoduchého polygonu na monotónní části v lineárním čase, dokážeme celou triangulaci provést v lineárním čase.
- Jednou z technik pro takovouto dekompozici je tzv. *trapezoidace*.
- Princip trapezoidace je založen na vyslání paprsků ve vertikálním nebo horizontálním směru z každého vertexu, které se zastaví při kolizi s jiným segmentem polygonu. Tyto paprsky efektivně rozdělí polygon na lichoběžníky.

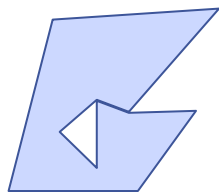
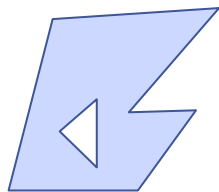


- Algoritmus provádějící trapezoidaci a tedy i triangulaci polygonu v lineárním čase byl prvně představen v článku *Triangulating a simple polygon in linear time* (Chazelle, 1990).
- Tento algoritmus je však velmi komplexní a obecně považován za "dostatečně beznadějný pro implementaci, takže slouží spíš jako existenční důkaz" (Skiena, 2008).¹
- Algoritmus také obsahuje velké množství konstant, jejichž načítání by pravděpodobně reálnou implementaci zpomalovalo a algoritmus by tak nedosáhl výrazného zrychlení oproti ostatním $O(n \log n)$ algoritmům.

¹volný překlad – v originálu "This algorithm is sufficiently hopeless to implement that it qualifies more as an existence proof."

- Článek *A Randomized Algorithm for Triangulating a Simple Polygon in Linear Time* (Amato et al., 2001) proto představil randomizovanou metodu pro výpočet trapezoidace jednoduchého polygonu s předpokládanou lineární složitostí, čímž představil další metodu pro triangulaci jednoduchého polygonu v (předpokládaném) lineárním čase.
- Metoda je založena na trapezoidaci postupně zpřesňovaných verzí polygonu.
- Ačkoliv je tento algoritmus na rozdíl od Chazelleovy metody prakticky implementovatelný, i tak je nad rámec této prezentace.

- Polygony s dírami se většinou definují tak, že vrcholy ohraničující díry jsou seřazeny opačným směrem oproti vnější hranici polygonu.
- Takovéto polygony pak umíme převést na polygony bez děr tak, že přidáme dvě hrany směřující do a ven z každé díry, přičemž tyto hrany spojují stejnou dvojici vrcholů a leží tedy "na sobě".
- Sebe protínající polygony nejsou až tak často používané a triangulační algoritmy pro tyto typy polygonů jsou relativně komplexní, nicméně existují.²



²Například FIST:

<http://www.cosy.sbg.ac.at/~held/projects/triang/triang.html>



Amato, N. M., Goodrich, M. T., & Ramos, E. A. 2001.
A Randomized Algorithm for Triangulating a Simple Polygon
in Linear Time.
Discrete Computational Geometry, **26**(2), 245–265.



Chazelle, B. 1990.
Triangulating a simple polygon in linear time.
*Pages 220–230 vol. 1 of: Proceedings (1990) 31st Annual
Symposium on Foundations of Computer Science*, vol. 1.
IEEE Comput. Soc. Press.



de Berg, Mark, Cheong, Otfried, van Kreveld, Marc, &
Overmars, Mark. 2008.
Computational Geometry: Algorithms and Applications.
Third edition edn.
Berlin, Heidelberg: Springer Berlin Heidelberg.



Skiena, Steven S. 2008.
Computational Geometry.
Pages 562–619 of: The Algorithm Design Manual, 2 edn.
London: Springer London.

Děkujeme za pozornost!