MAC 5747 – Geometria Computacional
Prof. José Coelho de Pina
Monitor: Fábio Henrique Viduani Martinez

LISTA DE EXERCÍCIOS 5 – RESPOSTAS

1. (Exercício 1.6.8.2 de [4]) Qual a complexidade de tempo do Algoritmo `Ototomia` quando restringimos a entrada a polígonos convexos?

   Se a entrada para o algoritmo OTOTOMIA é um polígono P convexo, então a linha 2 é executada em tempo $O(1)$, já que todo vértice $v_i \in P$ é uma orelha de P. As linhas 3 e 4 também são executadas em tempo $O(1)$. Dessa forma, devido ao laço na linha 1 envolvendo as linhas de 2 a 4, a complexidade de tempo do algoritmo OTOTOMIA quando P é convexo é $O(n)$.

2. (Ototomia em tempo $O(n^2)$) O algoritmo baseado em remoção de orelhas que foi visto em sala de aula tem complexidade de tempo $O(n^3)$. Nesta questão você desenvolverá um algoritmo de complexidade de tempo $O(n^2)$ que usa a mesma estrutura de dados e a função `Diagonal`.

   Uma chamada para a função `Diagonal` custa tempo $O(n)$. Logo, para obtermos um algoritmo de complexidade de tempo $O(n^2)$, a função `Diagonal` poderá ser chamada apenas $O(n)$ vezes. Imagine que você primeiro determina quais as potenciais diagonais de orelhas são de fato uma diagonal do polígono P, e marque o vértice v_i como 'ORELHA' se $v_{i-1}v_{i+1}$ é uma diagonal, caso contrário marque o vértice como 'NÃO-ORELHA'. Com isto, já gastamos $O(n^2)$ operações elementares.

   Seja P' o polígono de $n-1$ vértices resultante do polígono P após removermos uma de suas orelhas (digamos) $\triangle(v_{i-1}, v_i, v_{i+1})$. Responda as seguintes perguntas e justifique suas respostas:

   (a) É verdade que se o vértice v_j, $j \neq i$, está marcado como ORELHA em P então v_j continua sendo a ponta de uma orelha em P'?
   Não é verdade, como mostra a figura a seguir.

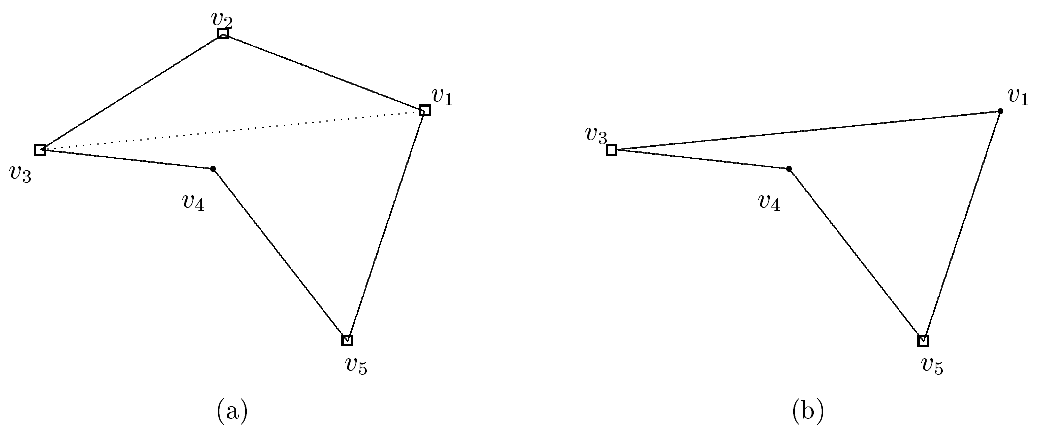
   

   Figura 1: (a) Um polígono $P = (v_1, v_2, v_3, v_4, v_5)$ com orelhas v_1, v_2, v_3 e v_5. (b) Um polígono $P' = (v_1, v_3, v_4, v_5)$, obtido a partir de P através do corte da orelha v_2, com orelhas v_3 e v_5. Note que v_1 não é mais orelha em P'.

   (b) Para quais valores de j a marca em v_j poderá ser alterada?
   Para $j = i - 1$ e $j = i + 1$ apenas. Mas devemos mostrar que isso é verdade! Uma forma é pensar por contradição.

(c) Quantas novas candidatas a diagonais precisam ser verificadas? (Ou seja, quantas novas chamadas da função `Diagonal` precisam ser feitas?)

Duas chamadas a `Diagonal` devem ser feitas, conforme 2(b), justamente para os vizinhos v_{i-1} e v_{i+1} de v_i.

(d) Em quais condições um vértice marcado como NÃO-ORELHA em P deverá ser marcado como ORELHA em P'?

Primeiro, conforme 2(b), os vértices que podem ter suas marcas alteradas de P para P' são apenas os vértices v_{i-1} e v_{i+1}. Suponha, que um deles, digamos v_{i-1}, esteja marcado como NÃO-ORELHA em P e sejam u e w os vizinhos de v_{i-1} em P'. Se `Diagonal`(P', u, w) = TRUE então v_{i-1} deve ser marcado como ORELHA em P'

(e) As marcas ORELHA e NÃO-ORELHA de P' herdadas por P podem ser atualizadas em tempo $O(n)$?

Seja P' o polígono de $n-1$ vértices resultante do polígono P após removermos uma de suas orelhas, digamos $\triangle(v_{i-1}, v_i, v_{i+1})$. Por 2(b) e 2(c), duas chamadas a `Diagonal` devem ser feitas: uma para os vizinhos de v_{i-1} e outra para os vizinhos de v_{i+1} em P'. As marcas de v_{i-1} e v_{i+1} são atualizadas conforme o resultado dessas chamadas, em tempo $O(n)$.

Se a resposta para a última pergunta é sim então um algoritmo de triangularização de complexidade de tempo $O(n^2)$ pode ser obtido simplesmente mantendo-se as marcas ORELHA e NÃO-ORELHA dos vértices do polígono corrente e escolhendo-se, em cada iteração, uma orelha do polígono corrente para ser removida.

Formule precisamente quais segmentos de P' precisam ser analisados para termos todas as orelhas de P' devidamente marcadas. Prove este lema, escreva o algoritmo resultante e a analise a sua complexidade de tempo.

Lema 1. Seja P um polígono em que todas as suas orelhas estão marcadas. Seja P' o polígono de $n-1$ vértices resultante do polígono P após removermos uma de suas orelhas, digamos $\triangle(v_{i-1}, v_i, v_{i+1})$. Para que as marcas dos vértices de P' estejam atualizadas, basta atualizar apenas as marcas dos vértices v_{i-1} e v_{i+1} de P'.

Prova. Diretamente de 2(b), 2(c) e 2(e).

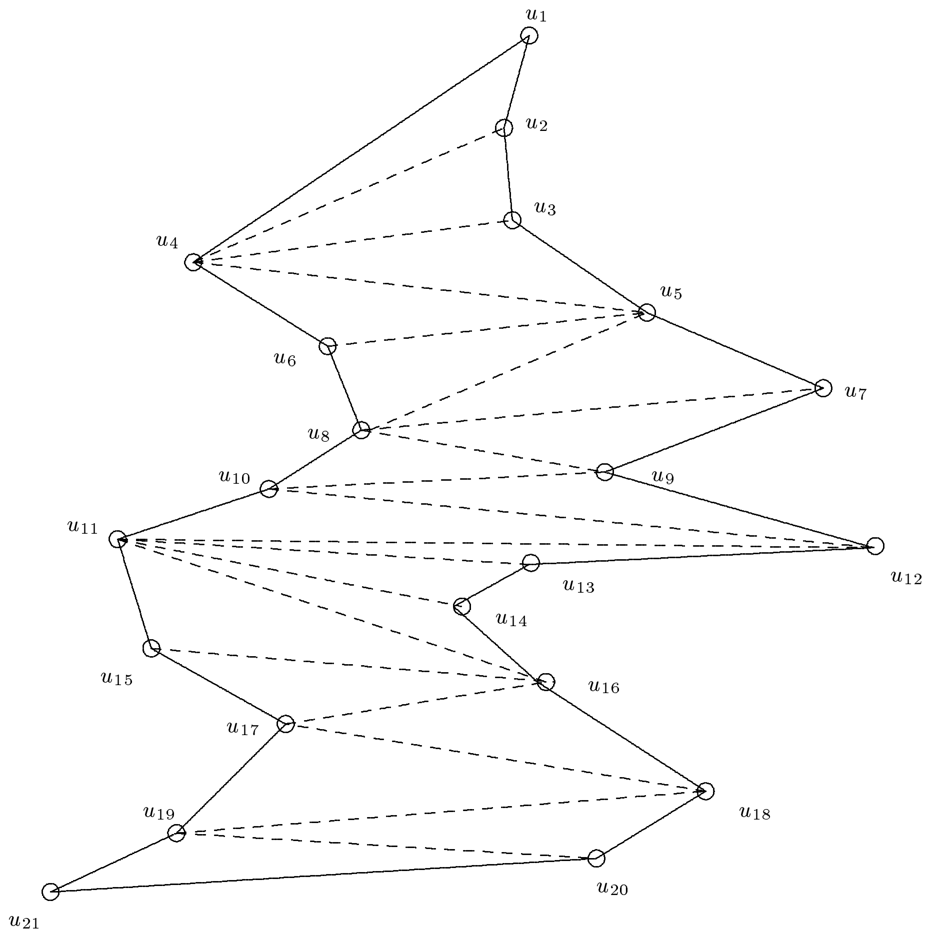
□

NOVO-OTOTOMIA(P)
Entrada: um polígono $P = (v_0, v_1, \ldots, v_{n-1})$.
Saída: uma triangularização de P.

para $i \leftarrow 0$ **até** $n-1$
  se `Diagonal`(P, v_{i-1}, v_{i+1})
    então `marca[`i`]` $\leftarrow$ ORELHA
    senão `marca[`i`]` $\leftarrow$ NÃO-ORELHA
enquanto $n > 3$
  Encontre v_i tal que `marca[`i`]` = ORELHA
  Trace a diagonal $v_{i-1}v_{i+1}$
  $P \leftarrow P - E$
  se `Diagonal`(P, v_{i-2}, v_{i+1})
    então `marca[`$i-1$`]` $\leftarrow$ ORELHA
    senão `marca[`$i-1$`]` $\leftarrow$ NÃO-ORELHA
  se `Diagonal`(P, v_{i-1}, v_{i+2})
    então `marca[`$i+1$`]` $\leftarrow$ ORELHA
    senão `marca[`$i+1$`]` $\leftarrow$ NÃO-ORELHA

O preprocessamento feito nas linhas de 01 a 04 gasta tempo $O(n^2)$. O laço das linhas 05 a 14 gasta tempo $O(n^2)$, já que as linhas 06, 09 e 12 têm tempo de execução $O(n)$ cada uma. Portanto, a complexidade de tempo do algoritmo Novo-Ototomia é $O(n^2)$.

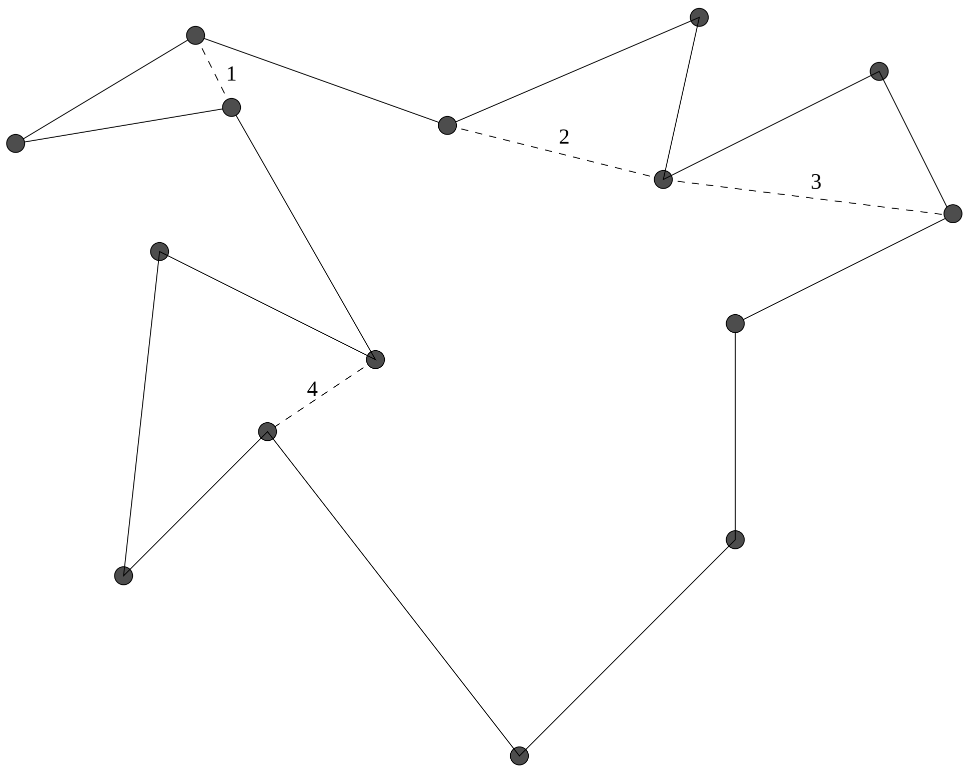
5. (Simulação: triangularização de polígonos monótonos) Simule (de alguma maneira clara) a execução do Algoritmo `Monótono` para triangularização de polígonos y-monótonos com o polígono da figura abaixo.



Supondo os vértices $\langle u_1, u_2, \ldots, u_{21} \rangle$ do polígono listados em ordem decrescente de y-coordenadas, a ordem de inclusão das diagonais feita pelo algoritmo `Monótono` é a seguinte: (u_4, u_2), (u_4, u_3), (u_5, u_4), (u_5, u_6), (u_8, u_5), (u_8, u_7), (u_9, u_8), (u_{10}, u_9), (u_{12}, u_{10}), (u_{11}, u_{12}), (u_{13}, u_{11}), (u_{14}, u_{11}), (u_{16}, u_{15}), (u_{16}, u_{11}), (u_{17}, u_{16}), (u_{18}, u_{17}), (u_{19}, u_{18}) e (u_{20}, u_{19}).

6. (Simulação: partição em polígonos monótonos) Simule (de alguma maneira clara) a execução do Algoritmo por linha-de-varredura `FazMonótono` que particiona um polígono em partes y-monótonos com o polígono da Figura 2.

   O algoritmo `FazMonótono` particiona o polígono conforme a figura seguir, onde as linhas tracejadas são as diagonais inseridas pelo algoritmo, e seus respectivos números indicam a ordem de inserção.


1
2
3
4