8. Um algoritmo de divisão-e-conquista

Neste seção estudaremos um algoritmo de divisão-e-conquista que constrói o diagrama de Delaunay de um dado conjunto S de n pontos no plano em tempo $O(n \log n)$ (i.e. um algoritmo ótimo para o problema). O método de divisão-e-conquista já foi visto várias vezes durante o andamento desta disciplina: algoritmo de complexidade de tempo $O(n \log n)$ para o Problema do Par Mais Próximo; Algoritmo `QuickHull`; e Algoritmo `MergeHull`.

O algoritmo que veremos é recursivo e terá três passos (os típicos passos de um algoritmo por divisão-e-conquista) em cada nível da recursão

Dividir: O problema é dividido em subproblemas de tamanho menor.

Conquistar: Os subproblemas são resolvidos recursivamente. Entretanto, se algum subproblema é suficientemente pequeno ele é resolvido diretamente.

Combinar: As soluções dos subproblemas são combinadas para obtermos uma solução do problema original.

Observação. O material desta seção foi extraído do Capítulo 4 de [5] e do artigo de Guibas & Stolfi [7]. O algoritmo que veremos é devido a Lee & Schachter [10] e é dual do algoritmo de Shamos & Hoey [13] (veja também [12] páginas 211–220).

8.1. **A descrição do algoritmo.** Para facilitar a nossa exposição do algoritmo assumiremos que qualquer subconjunto de quatro pontos de S são não-cocirculares e que qualquer dois pontos não têm a mesma x-coordenada. O algoritmo está descrito logo abaixo. Note que para que o o diagrama de Delaunay de S seja contruído em tempo $O(n \log n)$ necessitamos que a fase `Combinar` seja feita em tempo $O(n)$.

Algoritmo `Delaunay2`(S)
`Entrada`: Um conjunto S de sítios em E^2.
`Saída`: O diagrama de Delaunay $\mathrm{Del}(S)$.
1. **if** $|S| \leq 3$ construa $\mathrm{Del}(S)$ e **return**.
2. (`Dividir`) Particione o conjunto S em conjuntos S_1 com $\lceil n/2 \rceil$ pontos e S_2 com $\lfloor n/2 \rfloor$ pontos.
3. (`Conquistar`) Recursivamente construa os diagramas de $\mathrm{Del}(S_1)$ e $\mathrm{Del}(S_2)$.
4. (`Combinar` ou `Merge`) Tendo os diagramas $\mathrm{Del}(S_1)$ e $\mathrm{Del}(S_2)$
   construa $\mathrm{Del}(S_1 \cup S_2)$.

8.2. **Pré-processamento: 'sorting step'.** Para facilitar os passos `Dividir` e `Conquistar`, antes de aplicarmos o algoritmo `Delaunay2`, ordenaremos o conjunto de pontos dado em relação as suas x-coordenadas.

8.3. **Dividir.** O conjunto de pontos S será particionado em dois subconjunto em relação a mediana das x-coordenas. Graças ao pré-processamento este passo pode ser feito rapidamente. Em outras palavras, o conjunto S será divido com relação a uma reta vertical: seja L o conjunto formado pelos sítios de S que têm x-coordenada menor ou igual a mediana das x-coordenadas dos pontos e seja R o conjunto formado pelos demais pontos de S (pontos à direita de reta vertical passando pela mediana das x-coordenadas). (Escreveremos L (*left*) e R (*right*) ao invés de S_1 e S_2, respectivamente.) Note que esta escolha de conjuntos L e R juntamente com a nossa hipótese de que S não tem dois pontos com a mesma x-coordenada garante que os diagramas Del(L) e Del(R) são disjuntos.

8.4. **Conquistar.** O passo conquistar corresponde as chamadas recursivas onde cada subproblema será resolvido independentemente. A base da recursão é atinginda quando o conjunto de sítios em questão tem no máximo três sítios. Na base da recursão o diagrama de Delaunay desses (no máximo três) sítios pode ser facilmente construído. Por exemplo, o diagrama de Delaunay de três sítios corresponde a um triângulo que tem esses sítios como vértices ou a dois segmentos ligando os sítios (no caso do sítios serem colineares).

8.5. **Combinar: 'the merging step'.** A maior parte do trabalho deste algoritmo de divisão-e-conquista está neste passo (o mesmo fenômeno se verificou quando estudamos outros algoritmos de divisão-e-conquista). Precisamos ser cuidadosos para realizarmos este passo em tempo linear: um algoritmo ingênuo pode atingir a complexidade de tempo $O(n^2)$ para este passo, o que não nos permitiria obter a complexidade de tempo de $O(n \log n)$ para todo o algoritmo.

Chamaremos as arestas de Del(L) e Del(R) de L-L-arestas e R-R-arestas. As arestas de Del(S) que têm um sítio de L e o outro em um sítio de R será dita uma L-R-aresta (*crossing edge*). O passo `Combinar` consistirá em removermos algumas L-L e R-R-arestas e incluirmos algumas L-R-aresta.

Todas as L-R-arestas cruzam uma reta vértical l que separa o conjunto L e o conjunto R. Estes pontos de interseção de l com as L-R-arestas de Del(S) induzem uma ordem total no conjunto das L-R-arestas. Chamaremos esta ordem total de y-ordem. A L-R-arestas mais 'baixa' (menor na y-ordem) e mais 'alta' (maior na y-ordem) corresponde a aresta tangente inferiores e superior a conv(L) e conv(R), respectivamente. (Veja Figura 19.) O seguinte teorema nos fornece mais informações sobre a estrutura das L-R-arestas.

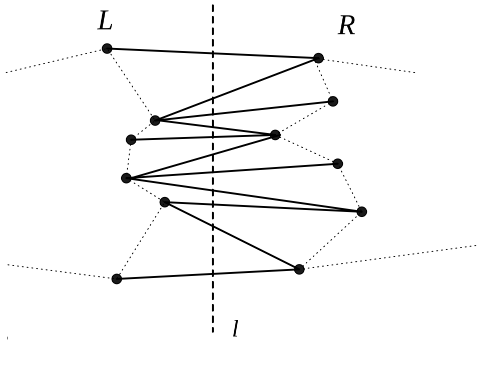

FIGURA 19. As L-R-arestas de Del(S).

Teorema 15. *Qualquer duas L-R-arestas consecutivas na y-ordem têm um vértice em comum. Além disso, o terceiro lado do triângulo que estas arestas determinam é uma L-L-aresta ou uma R-R–aresta.*

Prova. A primeira parte do teorema segue do fato de Del(S) ser uma triangularização dos pontos em S (cf. Teorema 11). Logo, o terceiro lado do triângulo tem ambos os seus vértices em L ou em R. Seja p, q e r os sítios do triângulo em questão e suponhamos, sem perda de generalidade, que $p \in L$, $q, r \in R$ (i.e. pq e pr são as L-R-arestas consecutivas na y-ordem). Precisamos mostrar que qr é uma aresta de Del(L). Como o triângulo $\triangle(p, q, r)$ é um triângulo de Del($L \cup R$) temos então que qr é uma aresta de Del($L \cup R$). Pelo Teorema 4 existe um círculo C contendo q e r tal que todos os demais sítios de S são exteriores a C. Logo, em particular, C é um círculo que contendo q e r que não contém nenhum outro sítio de R. Portanto, novamente pelo Teorema 4, temos que qr é uma aresta de Del(R). ∎

Suponha que tenhamos encontrado todas as L-R-arestas na y-ordem entre a tangente inferior a conv(L) e conv(R) até uma aresta $p_l p_r$, $p_l \in L$ e $p_r \in R$. (Veja Figura 20.) Pelo Teorema 15 sabemos que a próxima L-R-aresta na y-ordem é incidente a p_l ou p_r. Para determinarmos esta aresta basta examinarmos os sítios de L adjacentes a p_l em Del(L) e os sítios de R adjacentes a p_r em R. Por outro lado, pelo Teorema 9, sabemos que a próxima L-R-aresta na y-ordem tem uma ponta em p_l ou p_r e a outra ponta em um sítios p tal que o ângulo $\angle(p_l p p_r)$ é máximo entre todos os sítios que estão à esquerda do segmento (orientado) $p_l p_r$.

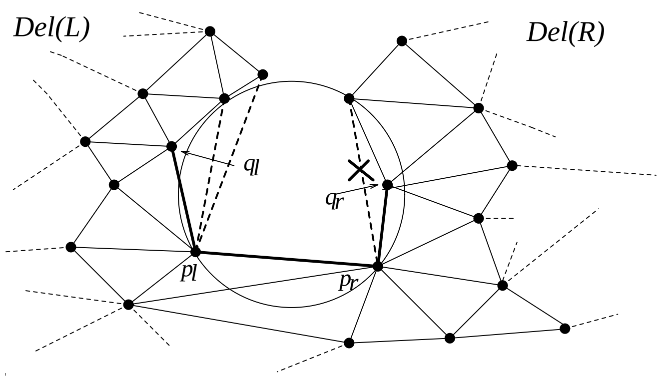

FIGURA 20. Determinando a próxima L-R-aresta de Del($L \cup R$).

Abaixo encontra-se uma descrição do Algoritmo `MergeDelaunay` que encontra as L-R-arestas e remove as L-L e R-R-arestas que deixam de ser arestas do diagrama Del($L \cup R$). Em cada iteração o algoritmo começa com uma L-R-aresta $p_l p_r$ e trabalha no sentido de encontrar a próxima L-R-aresta (próxima em relação a y-ordem). Dentre os sítios adjacentes a p_l (p_r, respectivamente) o algoritmo determina um sítio q_l (q_r, respectivamente) candidato a ponta da próxima L-R-aresta. (Veja Figura 20.) No final da iteração, o algoritmo determina, entre q_l e q_r qual sítio que será a ponta da próxima L-R-aresta (a próxima L-R-arestas será $p_l q_r$ ou $p_r q_l$).

Para encontrar q_l o algoritmo procederá como segue. Sejam $q_1, q_2, \ldots, q_k$ os sítios de L adjacentes a p_l em Del(L) listados no sentido anti-horário. Se o círculo determinado pelo triângulo $\triangle(p_l, p_r, p_1)$ contém o ponto p_2 concluímos imediatamente que $p_l p_1$ não pode ser uma aresta de Del($L \cup R$).

Neste caso, eliminamos a L-L-aresta $p_l p_1$ e repetimos o mesmo teste com os sítios p_2 e p_3 no lugar de p_1 e p_2, respectivamente. O sítios procurado q_l é o primeiro sítios q_t tal que o círculo determinado por $\triangle(p_l, p_r, p_t)$ não contém p_{t+1}. (Por que isto funciona?) A determinação de q_r é análoga a única diferença é que examinamos os sítios adjacentes a p_r em $\mathrm{Del}(R)$ no sentido horário ao ínves de anti-horário.

Para garantir que a determinação de q_l e q_r seja feita em tempo constante por aresta incidente a p_l e p_r, admitiremos que $\mathrm{Del}(L)$ e $\mathrm{Del}(R)$ sejam dadas através de suas estruturas *winged-edge*.

Notação. Se u e v são vértices de um diagrama planar H então denotaremos por $\texttt{esq}(u, v)$ o vértice de H adjacente a u que está imediatamente à esquerda do segmento (orientado) uv no sentido anti-horário. Analogamente, denotaremos $\texttt{dir}(u, v)$ o vértice de H adjacente a u que está imediatamente à direita do segmento uv no sentido horário. (Veja Figura 21.) Notemos que se H é representado por uma estrutura *winged-edge* então as operações `esq` e `dir` podem ser feitas em tempo constante (i.e. $O(1)$).

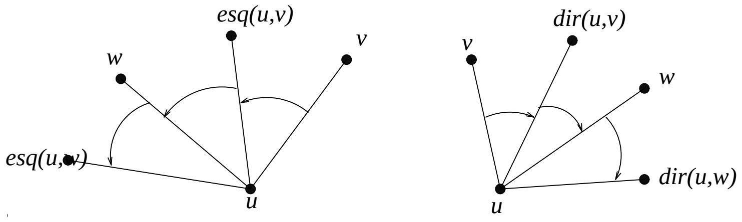

FIGURA 21. Ilustração da definição de $\texttt{esq}(uv)$ e $\texttt{dir}(u, v)$.

Algoritmo `MergeDelaunay`($\mathrm{Del}(L)$,$\mathrm{Del}(R)$)
`Entrada:` Dois diagramas de Delaunay $\mathrm{Del}(L)$ e $\mathrm{Del}(R)$ tal que os sítios de L estão à esquerda dos sítios de R.
`Saída:` O diagrama de Delaunay $\mathrm{Del}(L \cup R)$.
1. $T \leftarrow \mathrm{Del}(L) \cup \mathrm{Del}(R)$;
 [T é uma estrutura winged-edge.]
2. Determine o segmento $p_l p_r$ tangente inferior a $\mathrm{conv}(L)$ e $\mathrm{conv}(R)$.
 [Veja o Algoritmo `Tangente-Inferior`.]
3. **while** $p_l p_r$ não é a tangente superior a $\mathrm{conv}(L)$ e $\mathrm{conv}(R)$ **do**
 [**Determinação do candidato** q_l.]
4. $q_l \leftarrow \texttt{esq}(p_l, p_r)$; $q \leftarrow \texttt{esq}(p_l, q_l)$;
5. **while** $\texttt{Left}(p_l, p_r, q_l)$ e q pertence ao círculo determinado por p_l, p_r e q_l **do**
6.   Remova a L-L-aresta $p_l q_l$ de T;
7. $q_l \leftarrow q$; $q \leftarrow \texttt{esq}(p_l, q)$;
8. **if** $\texttt{Left}(p_r, p_l, q_l)$
9.   **then** não temos candidado a q_l.
 [**Determinação do candidato** q_r.]
10. $q_r \leftarrow \texttt{dir}(p_r, p_l)$; $q \leftarrow \texttt{dir}(p_r, q_r)$;
11. **while** $\texttt{Left}(p_l, p_r, q_r)$ e q pertence ao círculo determinado por p_l, p_r e q_r **do**
12.   Remova a R-R-aresta $p_r q_r$ de T;

    $q_r \leftarrow q$; $q \leftarrow$ `dir`(p_r, q);
  if `Left`(p_r, p_l, q_r)
    then não temos candidado a q_r.
[**Determinação da próxima L-R-aresta.**]
  Seja p o sítio entre os candidatos cujo ângulo $\angle(p_l p p_r)$ é máximo.
  [Se não existirem candidatos, então $p_l p_r$ é tangente superior.]
  if $p \in L$
    then Insira $p_r p$ em T;
    else Insira $p_l p$ em T;
Return T.

As operações primitivas usadas pelo algoritmo são `Lefton` e `InCircle`. Nós podemos intuitivamente ver o algoritmo como uma bolha (bidimensional) que vai subindo entre Del(L) e Del(R) e que a cada instante ocupa a maior área possível. Cada vez que a bolha atinge um novo sítio a próxima L-R-aresta é determinada (e talvez algumas L-L- ou R-R-arestas tenham que ser removidas).

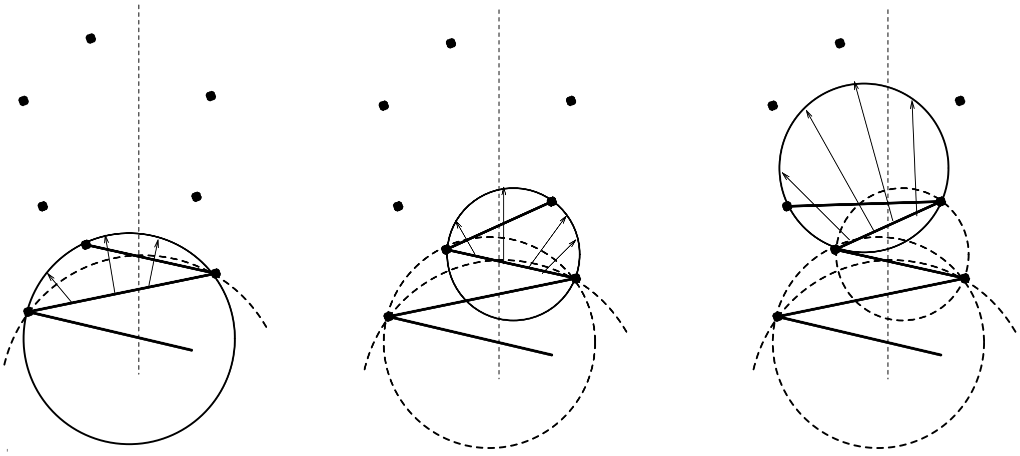

FIGURA 22. As bolhas subindo.

8.6. **Corretude do Algoritmo `MergeDelaunay`.** Para mostrarmos a corretude do algoritmo precisamos verificar que:

1. o algoritmo produz todas as L-R-arestas de Del$(L \cup R)$;
2. as L-L e R-R-arestas removidas pelo algoritmo não são arestas de Del$(L \cup R)$; e
3. ao final do Algoritmo `MergeDelaunay` o diagrama resultante corresponde a triangularização de Delaunay de $L \cup R$.

Teorema 16. *Seja S um conjunto de sítios em E^2 e considere um sítio particular p e uma reta l passando por p. Assuma por conveniência que l é uma reta horizontal. Sejam $p_1, p_2, \ldots, p_k$ $(k \geq 1)$ os sítios de S acima de l que são vizinhos de p em Del(S) listados em ordem anti-horária. Seja q um ponto em l à direita de p e seja $\alpha_i := \angle(pp_iq)$ $(i = 1, \ldots, k)$. Então existe um índice j, $1 \leq j \leq k$ tal que*

1. $\alpha_i < \alpha_{i+1}$ *para* $i = 1, \ldots, j-1$; *e*
2. $\alpha_i > \alpha_{i+1}$ *para* $i = j, \ldots, k$.

Prova. Primeiro mostraremos que se q_i denota a intersecção da circunferência determinada por triângulo $\triangle(p, p_i, p_{i+1})$ ($i = 1, \ldots, k-1$) com a reta l então a seqüência $q_1, q_2, \ldots, q_{k-1}$ se move monotonicamente para a esquerda. (Veja Figura 23.)

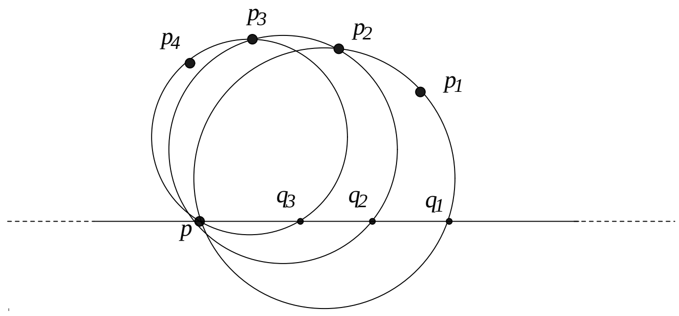

FIGURA 23. A seqüência $q_1, \ldots, q_{k-1}$ se move monotonicamente para a esquerda.

Para $i = 1, \ldots, k-3$ temos que o sítio p_{i+2} não pertence ao círculo determinado por $\triangle(p, p_i, p_{i+1})$ (cf. Teorema 2). Logo, $\angle(pp_{i+2}p_{i+1}) + \angle(pq_ip_{i+1})$ Portanto, q_i não pertence ao círculo determinado por $\triangle(p, p_{i+1}, p_{i+2})$ o que implica que q_{i+1} está à esquerda de q_i.

Para terminarmos a prova do teorema basta notarmos que se q_i está à direita de q então q pertence ao círculo determinado por $\triangle(p, p_i, p_{i+1})$ (o que implica que $\angle(pqp_i) + \angle(pp_{i+1}p_i) > \pi$, ou, equivalentemente, $\angle(p_{i+1}pq) + \angle(p_{i+1}p_iq) < \pi$). Logo p_{i+1} pertence ao círculo determinado por $\triangle(p, q, p_i)$. Analogamente, se q_i esta à esquerda de q então p_{i+1} não pertence ao círculo determinado por $\triangle(p, q_i, q)$. Com isto completamos a prova do teorema. ∎

A primeira L-R-aresta determinada pelo algoritmo na Linha 2 é certamente uma aresta de Del$(L \cup R)$ (cf. Teorema 4). Os Teorema 15 e 16 implicam que as Linhas 5–9 do Algoritmo `MergeDelaunay` encontra o sítios $q_l \in L$ tal que o ângulo $\angle(p_lq_lp_r)$ é máximo. Analogamente, os Linhas 10–15 determinam o sítio $q_r \in R$ tal que o ângulo $\angle(p_lq_rp_r)$ é máximo. A Linha 16 decide qual sítios $p \in \{q_l, q_r\}$ máximiza o ângulo $\angle(p_lpp_r)$. Logo, pelo Teorema 9, a L-R-aresta inserida na estrutura T pelas Linhas 17–19 é a aresta de Del$(L \cup R)$ sucessora da L-R-aresta p_lp_r na y-ordem. Isto implica que o Algoritmo `MergeDelaunay` encontra todas as L-R-arestas de $L \cup R$.

Teorema 17. *As L-L e R-R-arestas removidas pelo algoritmo não são arestas do diagrama de Delaunay de* $(L \cup R)$.

Prova. Uma L-L-aresta p_lq_l é removida pelo algoritmo na Linha 6 somente se o círculo determinado por q_l está à esquerda do segmento p_lp_r e o sítio q imediatamente à esquerda do segmento p_lq_l no sentido anti-horário pertence ao círculo determinado por $\triangle(p_l, p_r, q_l)$. Como p_lp_r é uma aresta da triangularização de Delaunay de $L \cup R$ temos que se p_lq_l é uma aresta de Del$(L \cup R)$ então $\triangle(p_l, p_r, q_l)$ é um triângulo de Del$(L \cup R)$. Isto contradiz o

Teorema 3, já que q é um sítio contido no interior do $C(v)$ onde v é o vértice de Voronoi correspondente ao triângulo $\triangle(p_l, p_r, q_l)$.

Analogamente mostra-se que as L-L-arestas removidas pela Linha 12 do algoritmo não são arestas do diagrama de Delaunay de $L \cup R$. ∎

Teorema 18. *Ao final do Algoritmo* `MergeDelaunay` *o diagrama resultante corresponde a triangularização de Delaunay de* $L \cup R$

Prova. Primeiro mostraremos que o diagrama obtido é uma triangularização de $L \cup R$. No ínicio do algoritmo $p_l p_r$ corresponde a tangente inferior a $\text{conv}(L)$ e $\text{conv}(R)$ e depois de cada iteração o segmento $p_l p_r$ terá uma de uma extremidade mais alta (maior y-coordenada). O laço das Linhas 3–19 termina quando $p_l p_r$ alcança a tangente superior a $\text{conv}(L)$ e $\text{conv}(R)$. As faces interiores do diagrama serão ou faces de $\text{Del}(L)$ ou $\text{Del}(R)$ ou faces formadas por duas L-R-arestas e um outra L-L- ou R-R-aresta. Em ambos os casos as faces do novo diagrama formam triângulos.

Precisamos verificar agora que a triangularização obtida de $L \cup R$ é de fato uma triangularização de Delaunay, ou seja, que as arestas do diagrama resultante são arestas de $\text{Del}(L \cup R)$. Devido aos Teoremas 16 e 17 precisamos apenas verificar que arestas do diagrama resultante que são L-L- ou R-R-arestas são arestas de $\text{Del}(L \cup R)$. Seja pq uma L-L-aresta que não foi removida pelo algoritmo. Se pq não é uma aresta de $\text{Del}(L \cup R)$ então existe uma aresta rs de $\text{Del}(L \cup R)$ que tem intersecção não vazia com pq. Já que pq é uma L-L-aresta então um dos sítios r ou s deve ser um sítios de R. Mas isto implica que rs é uma L-R-aresta e portanto foi corretamente encontrada pelo algoritmo. Logo, se pq não é uma aresta de $\text{Del}(L \cup R)$ ela deve ter sido removida pelas Linhas 5–7 do algoritmo. Analogamente se verifica que toda R-R-aresta do diagrama resultante ao final do algoritmo é uma aresta de $\text{Del}(L \cup R)$. ∎

Estes teoremas completam a prova da corretude do Algoritmo `MergeDelaunay`—o que implica a corretude do Algoritmo `Delaunay2`.

8.7. **Análise do algoritmo.** Seja n o número de sítios de S. A complexidade de tempo do pré-processamento (ordenação dos sítios) que é $O(n \log n)$.

No Algoritmo `MergeDelaunay` (passo `Combinar`) a Linha 2 pode ser executada em tempo $O(n)$ (veja o Algoritmo `Tangente-Inferior`). O laço entre as Linhas 3–19 são executados $O(n)$ vezes. Em cada iteração do laço das Linhas 5–7 uma L-L-arestas é removida. Logo, esse laço será executado $O(n)$ vezes. Analogamente, o laço das Linhas 11–13 também será executado $O(n)$ vezes. Como a execução de cada Linha i, $(i = 3, \ldots, 19)$ gasta tempo $O(1)$ concluímos que a complexidade de tempo do Algoritmo `MergeDelaunay` é $O(n)$.

Se $T(n)$ é a função complexidade de tempo do Algoritmo `Delaunay2`, sem contar o custo do pré-processamento, temos que

$$T(n) \leq T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + cn = O(n \log n),$$

onde c é uma constante. Logo, a complexidade de tempo do Algoritmo `Delaunay2` é $O(n \log n)$.