7. Um algoritmo quadrático

Nesta seção estudaremos um algoritmo quadrático para construir o diagrama de Delaunay de um dado conjunto $S = \{p_1, \dots, p_n\}$ de sítios. Note que dado Del(S) podemos, em tempo linear, construir Vor(S).

7.1. **A idéia do algoritmo.** A idéia básica, análoga a do Algoritmo Jarvis, é usar as arestas do diagrama de Delaunay (*arestas de Delaunay*) dos triângulos que já foram determinados para encontrar os outras triângulos. Cada arestas do diagrama de Delaunay é compartilhada por um ou dois triângulos (*triângulo de Delaunay*—triângulos da triangularização de Delaunay). Uma aresta de Delaunay tem apenas um triângulo do diagrama incidente a ela quando está na fronteira do fecho convexo de S. Chamaremos uma aresta de Delaunay de *livre* se algum dos triângulos que compartilham esta aresta ainda não foi determinado pelo algoritmo. Dada uma aresta livre p_ip_j de um triângulo de Delaunay podemos determinar o triângulo que compartilha esta aresta da seguinte maneira: encontre o ponto p_k, do lado apropriado do segmento p_ip_j, para o qual o $\angle(p_ip_kp_j) > \angle(p_ip_sp_j)$ para todo ponto p_s que esta do mesmo lado de p_k (cf. Teorema 9). Este é o passo básico do algoritmo e pode ser feito usando-se a primitiva `InCircle`.

7.2. **O Algoritmo.** Como fizemos no Algoritmo Jarvis uma estrutura de adjacência (como *winged-edge*) pode ser utilizada para armazenarmos o diagrama de Delaunay. A medida que as faces são introduzidas no diagrama suas arestas são examinadas para verificarmos se elas já fazem parte da estrutura (se são livres ou não). Em caso da aresta examinada não fazer parte da estrutura (i.e. a aresta é livre) ela deverá ser explorada mais tarde.

Como começar o algoritmo já que no início não temos nenhuma face (triângulo) de Del(S)? Basta procedermos de maneira análoga ao que fizemos no Algoritmo de Jarvis: determinar uma aresta de conv(S). Esta aresta é com certeza uma aresta de Del(S) (para a qual ainda resta contruirmos o triângulo de Del(S) incidente a ela—i.e., esta aresta é livre). Com esta aresta podemos, usando o Teorema 9, encontrar uma face de Del(S).

O algoritmo empregará uma fila Q para armazenar as faces que vão sendo encontradas ao longo da execução do algoritmo.

Algoritmo `Delaunay`(S)
`Entrada:` Um conjunto finito S de pontos em E^2.
`Saída:` O diagrama de Delaunay dos pontos em S.
1. $Q \leftarrow \emptyset$; [Fila com as faces encontradas.]
2. $f \leftarrow$ uma face inicial de conv(S);
3. Insira f na fila Q e na winged-edge estrutura T;
4. **while** $Q \neq \emptyset$ **do**
5. $\quad f \leftarrow$ primeira face na fila Q;
6. $\quad$ **para** cada aresta livre e de f **do**
$\quad$ [Passo correspondente ao Teorema 9.]
7. $\quad\quad f' \leftarrow$ face do diagrama de Delaunay que compartilha e com f;
8. $\quad\quad$ Insira f' na fila Q;
9. $\quad\quad$ Insira f' na estrutura winged-edge T, conectando-a com as faces já

geradas que lhe são adjacentes e determinando suas arestas livres.

10. **return** T.

7.3. **Análise do algoritmo.** O algoritmo pára somente quando toda a triangularização de Delaunay tiver sido obtida. Observe que se as faces geradas (inclusive a face externa) não cobrissem o plano, alguma aresta de uma das faces estaria livre, o que não ocorre já que o algoritmo só termina quando todas as arestas livres forma exploradas. O Passo 2 pode ser feito em tempo $O(n)$. Utilizando a primitiva `InCircle` podemos executar o Passo 7 em tempo $O(n)$. Logo a complexidade de tempo do Algoritmo `Delaunay` é $O(n^2)$.

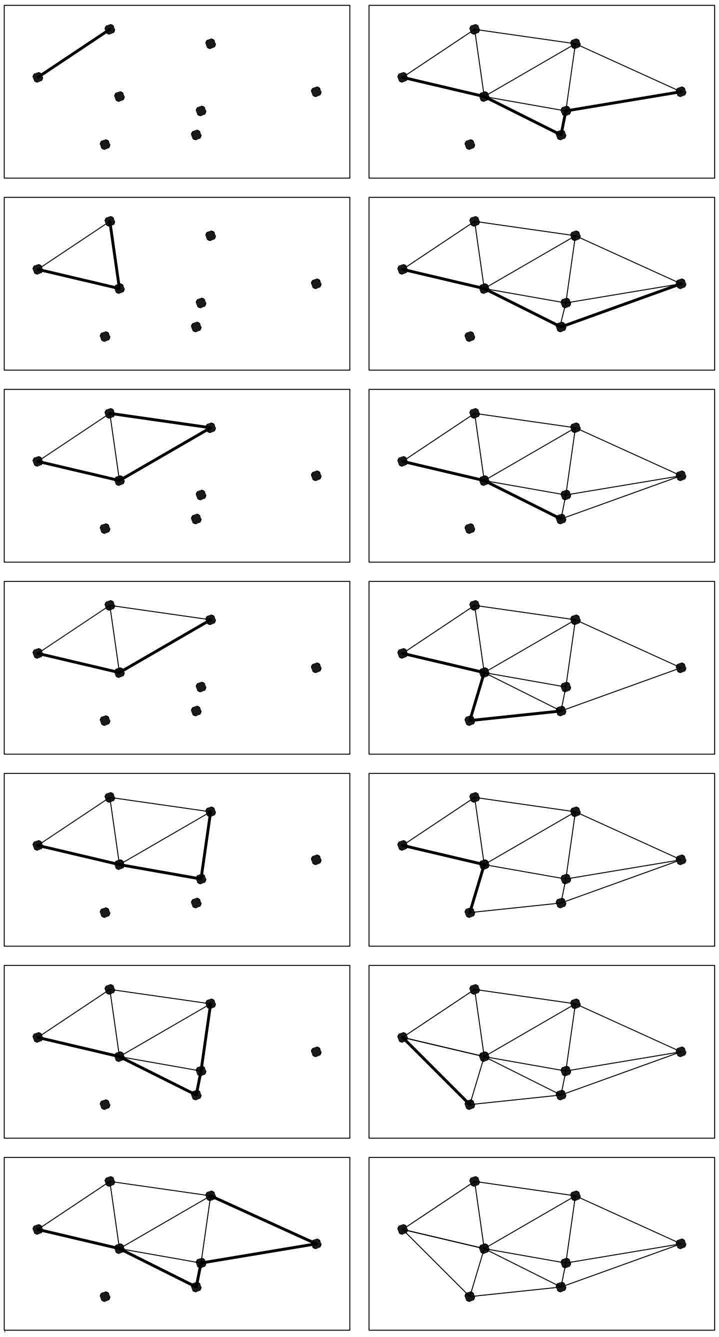

FIGURA 18. Ilustração do Algoritmo `Delaunay` em ação..

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