2012 Spring CS504

Homework #3 Solution

9.13 The *Gabriel graph GG*

(a) Let e = pq be any Gabriel edge. Then, by definition, the circle C with diameter pq contains p and q on its boundary and does not contain any other point. So $e \in DG$.

(b) (=>) Suppose pq is a Gabriel edge and C is the circle with diameter pq. Let c be the center of C. Then, c must be on the Voronoi edge between V(p) and V(q). So pq and the Voronoi edge intersect on c.

(<=) Let x be the intersection between two edges. Since x is on the Voronoi edge, the distances px and qx are equal, and any other site, r, is far from x comparing to p and q. So x is actually in the middle of the line pq, and the circle C with diameter pq does not contain any other point. So $pq \in GG$

(c) Compute DT. Checking each edge whether it follows the condition in (b). Since the number of edges is linear, the time complexity is dominant by DT algorithm, which is $O(n \log n)$.

9.14 The *relative neighborhood graph RNG*

(a) (=>) Suppose a point x is inside the interior of lune(p,q). Then, $d(p,q) \leq \max\big(d(p,x), d(q,x)\big) < d(p,q)$, which leads to a contradiction.

(<=) obvious.

(b) Let pq be a relative neighborhood edge, and obviously lune(p,q) is not empty. Then, since lune(p,q) contains the circle with diameter pq, pq is a Gabriel edge, and also a Delaunay edge.

(c) Computing DT, and check each DT edge whether it satisfies the condition in (a).

9.15 $\mathrm{EMST} \subseteq \mathrm{RNG}$

Let pq be an edge in EMST. Suppose pq is not in RNG. Then, by definition, there is a point $\mathrm{r} \neq \mathrm{p}, \mathrm{q}$ such that $\mathrm{d}(\mathrm{p}, \mathrm{q}) > \max\big(d(p,r), d(q,r)\big)$. So, $\mathrm{d}(\mathrm{p}, \mathrm{q}) > d(p,r)$. By adding pr and deleting pq in EMST, we can get a new spanning tree that has smaller total edge length. It leads to a contradiction.

9.16 k-clustering

(a) Suppose $\mathrm{p_i}p_j$ is not a Delaunay edge. Consider the circle C with diameter $\mathrm{p_i}p_j$. By the property of DT, C must have at least one point of P in its interior. Let r be such a point. Then, including r to $\mathrm{P_i}$ makes the distance between clusters smaller. It contradicts to the assumption that the partition has minimum distance. So, $\mathrm{p_i}p_j$ must be a Delaunay edge.

(b) We first think each point as a single partition. Compute the DT of P, and sort Delaunay edges by its length in increasing order. Look at the edges in sorted order, and for the edge we merge two partitions that have endpoints of the edge. Repeat this process until we have k partitions.

In sorting process, $\mathrm{O}(\mathrm{n} \log \mathrm{n})$.

In merging process, we use Union-Find data structure, which takes $\mathrm{O}(\mathrm{n} \log \mathrm{n})$.