**PROXIMITY GRAPHS**

- **NNG**: NEAREST NEIGHBOUR GRAPH
  - For each point: connect to closest point.

```
not necessarily connected
```
PROXIMITY GRAPHS

• **NNG**: NEAREST NEIGHBOUR GRAPH
  - For each point: connect to closest point.

  ![Diagram of a NNG graph]

  Assuming we're happy with breaking ties arbitrarily.

• Any edge in NNG must be in MST
  (Prove by contradiction)
  - Compute NNG in \( O(n) \) after MST.

• \( \Omega(n \log n) \) (Even for closest pair problem)

  Element uniqueness

**NOTE**: \( k \)-NNG also exists
RNG: RELATIVE NEIGHBORHOOD GRAPH

- JOIN \( \bullet \rightarrow \bullet \) IF LUNE IS EMPTY.

GG: GABRIEL GRAPH

- JOIN \( \bullet \rightarrow \bullet \) IF \( \bigcirc \) IS EMPTY.

WHICH OF THE 2 HAS MORE EDGES?

RNG: MORE RESTRICTIVE \( \Rightarrow \) FEWER EDGES

RNG \( \subseteq \) GG

BUT HOW MANY EDGES? \( f(n) \)
Any edge in GG must also be in D.T.

\[ GG \subseteq DT \implies O(n) \text{ edges.} \]

and clearly the opposite is not true.

\[ \text{Claim: } MST \subseteq RNG \]

\[ \text{Suppose not true: } \overline{ab} \text{ in MST but not in RNG.} \]

Then \( \exists c \in \text{LUNE}. \)

Now, fact:

\[ \frac{ac}{bc} < \frac{ab}{bc} \]

So \( ac \) or \( bc \) each cannot be in MST.

(we could replace \( ab \) in MST)

So \( \exists \) path from \( c \) to some \( x \) & then to \( \bullet \rightarrow \bullet \).

- If \( \bullet \) is a leaf, use \( ac \) instead of \( ab \). (symmetric for \( \bullet \))

- If neither a leaf, replace

CONCLUSION: MST can be IMPROVED! X
Thus \( \text{NNG} \subseteq \text{MST} \subseteq \text{RNG} \subseteq \text{GG} \subseteq \text{DT} \)

**Algorithms:**

\( \text{GG} \rightarrow \text{keep edge } \overline{a,b} \) of \( \text{DT} \) if it crosses the Voronoi diagram at \( \text{cell}(a) \cap \text{cell}(b) \)

\( O(n) \) after DT.

- RNG can also be computed in \( O(n) \) time \([\text{Lingas'94}]\)

starting from DT and deleting.
I'm leaving the following (very incomplete) notes on Fortune's algorithm for computing the Voronoi diagram.

... just in case it helps at all if you ever want to read about it.

You are NOT responsible for any of this for the final.
FORTUNE'S LINE-SWEEP ALGORITHM FOR VORONOI DIAGRAMS

- Regular line-sweep cannot work in the sense that everything "behind" the line is final. Everything (above) behind the beachfront is final.
- Every node on the beachfront represents the propagation of a Voronoi edge.

- Parabolas change shape as sweep line continues.
- When passes through a new point, create a parabola.

It intersects the beachfront at an existing parabola.

\[ d(\bullet \bullet) = d(\bullet \bullet) \implies \bullet \text{ is on Voronoi edge}. \]

- As continues, \( \bullet \) grows left & right following intersection \( \vee \).

This is the only way to add to \( \bullet \) - see textbook.

\( \Rightarrow \text{beachfront has size } O(n) \).
Creation of Voronoi Vertex:

$V \& U$ meet and eliminate $U$

3 parabolas intersect at $U U U U$: equal dist. to $O$

Thus $O$ are on $O$

Now that $U$ is gone,
$U \& U$ continue formation of Voronoi edge (merged)

Creation Type II