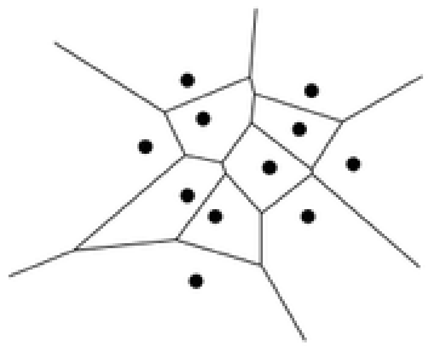
Диаграмма Вороного

Постановка задачи

Представим, что у нас есть N почтовых отделений и мы хотим для каждой точки на плоскости найти близжайшее отделение.
То есть нужно разбить плоскость на N областей по признаку того, какая из данных точек близжайшая.



Терминология

Исходные точки на плоскости будем называть **сайтами** (*sites*).

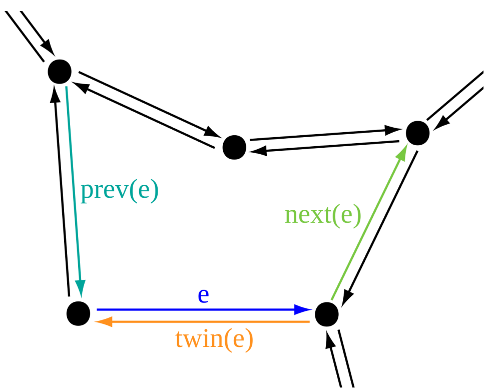
Определение. $P \coloneqq \{p_1, p_2, ...p_n\}$ - множество сайтов, а $V(p_i) \coloneqq \{x \mid dist(x, p_i) \leq dist(x, p_j) \forall j\}$ – ячейка диаграммы Вороного.

Определение. $Vor(P)$ (Диаграмма Вороного) – это совокупность всех ячеек.

DCEL

Определение. DCEL – это структура данных, которая состоит из вершин, полурёбер и граней и позволяет хранить планарный граф. В нашем случае диаграмму Вороного.

- *Вершина.* Содержит свои координаты и указатель на любое полуребро, исходящее из этой вершины.
- *Полуребро.* Содержит указатель на вершину-исток, указатель на грань, лежащую справа от ребра, и указатели на следующее, предыдущее ребро в обходе грани, указатель на ребро-близнец.
- *Грань.* Содержит указатель на любое полуребро, составляющее её границу.



Наивный алгоритм

Очевидно, что два сайта можно разделить их серединным перпендикуляром. Получится две полуплоскости. Введём ещё одно обозначение $h(p_i, p_j)$ – такая полуплоскость, содержащая p_i. Заметим, что $V(p_i) = \bigcap_{\{1 \le j \le n, i \ne j\}} h(p_i, p_j)$. Тогда для каждого сайта нужно посчитать пересечение $n - 1$ плоскости. Занимает это $O(n^2 \log(n))$.

Наблюдение. Заметим, так как каждая ячейка диаграммы Вороного является перечечением полуплоскостей, она будет представлять собой обобщённый выпуклый многоугольник.

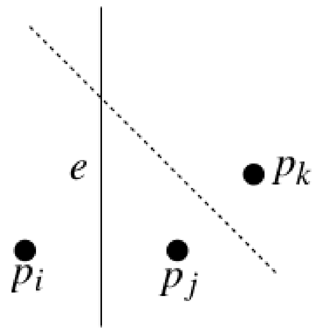
Структура диаграммы

Теорема.

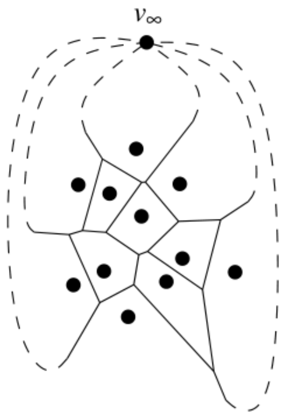
1. Если все сайты лежат на одной прямой, то $Vor(P)$ состоит из $n - 1$ параллельных прямых.
2. В противном случае:
   - Все рёбра (в творческом смысле) либо отрезки, либо лучи.
   - Диаграмма связна.
   - $n_v \le 2n - 5$, где n_v – количество вершин.
   - $n_e \le 3n - 6$, где n_e – количество рёбер.

Доказательство.

1. Очев.
2. - Пусть существует три сайта p_i, p_j, p_k, не лежащих на одной прямой. Тогда два серединных перпендикуляра a и b, которые определяют соответственно границы между точками p_i и p_j, p_j и p_k пересекаются в точке X. Понятно, что a не может целиком входить в диаграмму, потому что точки на "дальнем" луче будут ближе к p_k, чем к p_j, а значит не могут лежать в $V(p_j)$.

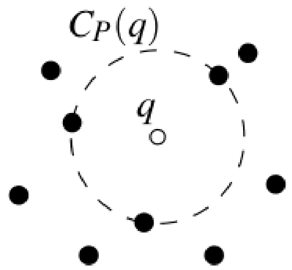


   - Елси граф несвязный, то его разделяет одна ячейка. Так как ячейка выпуклая, она может состоять только из двух прямых, что противоречит условию.
   - Формула Эйлера утверждает, что $n_v - n_e + n_f = 2$. Однако наш граф не является планарным, потому что может содержать рёбра-лучи. Добавим фиктивную вершину v_∞ и соединим её со всеми лучами. Это можно сделать, если нарисовать достаточно большой *bounding box* вокруг диаграммы и пересечения с лучами соединить с v_∞. Тогда $n_v - n_e + n = 1$. Заметим, что на каждую вершину приходится хотя бы 3 ребра, иначе по выпуклости ребро может быть только прямой. v_∞ тоже имеет хотя бы 3 ребра, потому что в диаграмме хотя бы 3 луча (это можно легко показать). Отсюда $2n_e \ge 3(n_v + 1) \Rightarrow n_v - \frac{3}{2}(n_v + 1) + n \ge 1 \Rightarrow n_v \le 2n - 5$.
   - $\Rightarrow (2n - 5) - n_e + n \ge 1 \Rightarrow n_e \le 3n - 6$

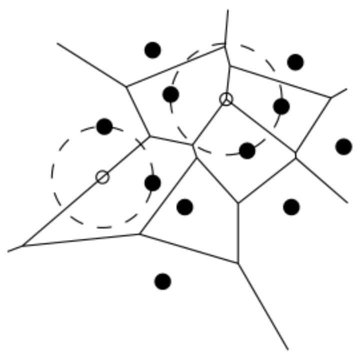


Вывод. Число вершин и рёбер $Vor(P)$ линейно относительно количества сайтов.

Опеределение. $C_p(q) := \bigcup_r \left(\overline{B_r(q)} : \forall j\ p_j \notin B_r(q)\right)$ – максимальный шарик в точке q, который не содержит сайтов.



Теорема.
1. Точка q является вершиной $Vor(P) \Leftrightarrow |\{p_i \mid p_i \in C_p(q)\}| \geq 3$.
2. Часть серединного перпендикуляра a между сайтам p_i и p_j является ребром $Vor(P) \Leftrightarrow$ $\exists q \in a : p_i, p_j \in C_p(q) \wedge p_k \notin C_{p(q)} \forall k \neq i, j$.



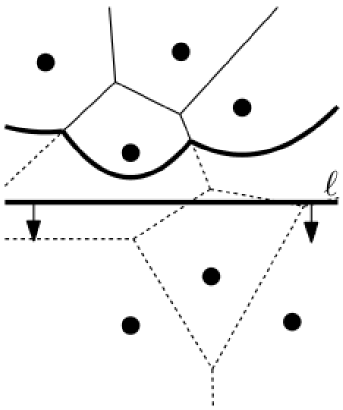
Доказательство.
1. $\Rightarrow$ q содержит хотя бы 3 ребра $\Rightarrow$ она равноудалена хотя бы от 3 сайтов p_i, p_j, p_k и $q \in V(p_i) \cap V(p_j) \cap V(p_k)$, потому что лежит на 3 серединных перпендикулярах. Образуется круг с центром в нашей вершине и радиусом, равным расстоянию до сайтов. Внутри круга не будет сайтов, потому что если сайт p_l есть, то существует $B_r(q)$, такой что $\forall t \in B_r(q)\ dist(t, p_l) < dist(t, p_i) \Rightarrow q \notin V(p_i)$. Противоречие.

   $\Leftarrow$ $q \in V(p_i) \cap V(p_j) \cap V(p_k)$, многоугольники могут пересекаться только по одной точке $\Rightarrow$ q – вершина.

2. $\Rightarrow$ Пусть $q \in a$, где a – перпендикуляр. Пусть иначе, тогда аналогично п.1 целая окрестность $qB_r(q) \in V(p_l)$ $\Leftarrow$ Пусть существует такой $C_p(r)$, тогда если немного подвинуть q, то $p_i, p_j \in C_p(q')$. Таким образом, $qq' \in V(p_i) \cap V(p_j)$.

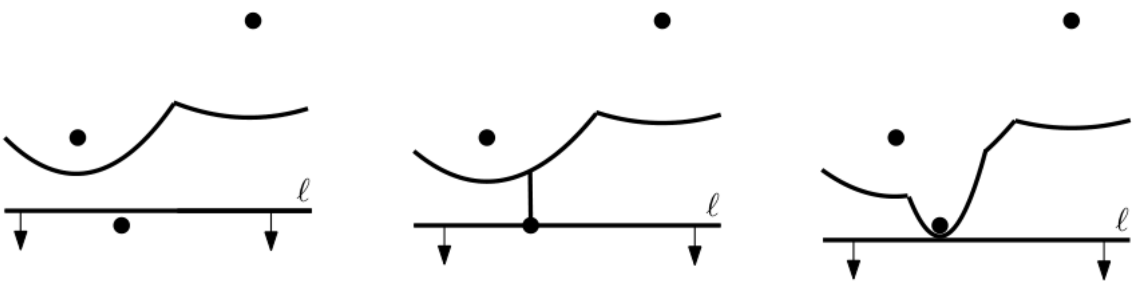
Алгоритм Форчуна

Алгоритм будет представлять из себя подобие *scanline*, то есть будет прямая l, которая двигается сверху вниз, есть множество обработанных сайтов, оно выше прямой, есть множество ещё не обработанных. Нужно для каждого сайта определить область, которая точно на ходится с его ячейке. Заметим, что ГМТ, равноудалённых от прямой и точки – это парабола. В данном случае сайт будет фокусом, а l – директрисой.



Определение. Береговая линия (*beech line*) — граница объединения всех парабол уже рассмотреных точек. Точки пересечения соседних парабол будем называть *breakpoints*.

Береговая линия состоит из частей парабол, которые можно упорядочить по правым концам. Будем хранить эти кусочки в упорядоченном множестве (*set*). Сами элементы множества будем хранить неявно: чтобы вычислить параболу, нам нужно знать только фокус (сайт) и директрису.

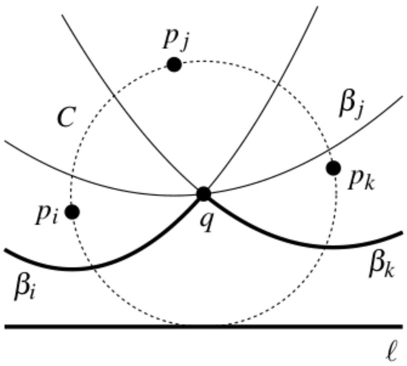


Все точки, которые выше береговой линии, мы уже знаем, к какому сайту отнести. Когда l пересекает новый сайт, появляется вырожденная парабола, которая по факту является лучом. Далее она расширяется по мере удаления l. Точка пересечения двух соседних частей парабол равноудалена от соответствующих сайтов, поэтому эта точка будет вычерчивать ребро между сайтами. вершины появляются, когда 3 подряд идущие параболы схлопываются в одну, то есть когда встречаются соседние *breakpoint'ы*.

Наблюдение. Каждой x-координате соответствует одна точка береговой линии.

Определение. *Site event* – появление нового сайта на сканирующей линии.

Определение. *Circle event* – момент, когда сканирующая линия касается окружности, на которой лежат фокусы трёх подряд идущих частей парабол береговой линии. Понятно, что три параболы будут пересекаться в одной точке в центре окружности, а после центральная парабола выродится.



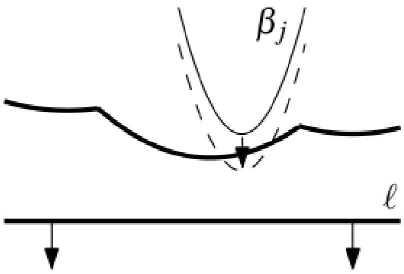
Заметим, что у нас есть ровно два явления: появление новой параболы и удаление старой. Давайте более подробно рассмотрим случаи.

Теорема.

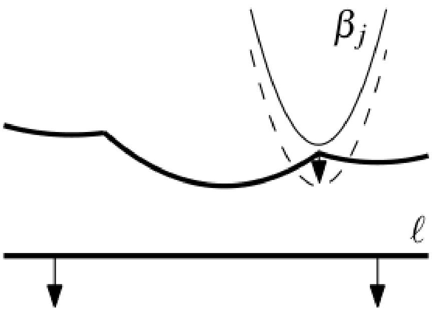
1. Появление новой параболы может быть только через *site event.*
2. Исчезновение старой параболы может быть только через *circle event.*

Доказательство.

1. • Заметим, что не может быть случая, когда парабола не входящая в береговую линию "догнала" параболу с более низким фокусом и прорезалась где-то внутри неё.



   • Так же она не может прорезаться через стык двух парабол.



   • Остаётся только *site event.*
2. • Следует из вышеизложенного.

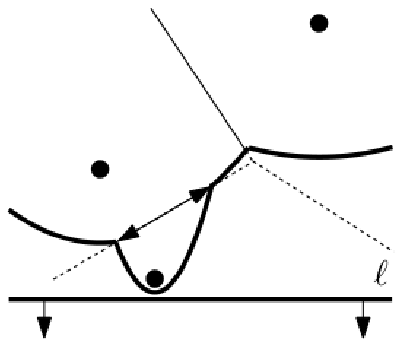
Обработка события

Понятно, что события нужно хранить по возрастанию их времени, то есть по y-координате. Поэтому добавим в наш алгоритм ещё одну структуру данных. Например, очередь с приоритетами. Ключ очереди – y-координата точки, значение – x-координата.

1. *Circle event.* Нам нужно удалить параболу из сета. Для этого в каждом таком событии можно хранить указатель на удаляемую дугу. Также нужно удалить из очереди *circle event'ы* (если они присутствуют), которые образуются тройками дуг, где удаляемый был левой или правой дугой (*false alarm*).
2. *Site event.* Если у нас нет элементов в сете, просто добавляем. Иначе бинарным поиском найдём часть параболы, которая находится над новым сайтом. Тогда нужно расщепить этот элемент на два и между ними вставить элемент, соответствующий дуге новой параболы. У нас пропадёт одна тройка и появятся две новые (если изначально было хотя бы 3 элемента). Нужно проверить, есть ли в очереди соответствующий *circle event* и удалить, если нужно. Также проверить, будут ли сходиться *breakpoint'ы* новых двух троек и добавить *circle event'ы*, если да. Конечно, перед запуском алгоритма все *site event* уже известны и их можно добавить в очередь.

Замечания.

1. Нужно отдельно обработать ребро, которое получается после вставки новой параболы. Когда её ветви расходятся, *breakpoint'ы* идут в противоположных направлениях и задают ребро.



2. Отдельно нужно рассмотреть случай, когда самые первые сайты находятся на одной высоте. Их надо добавить в сет по порядку.

3. Другие события с одинаковой y-координатой можно обрабатывать в любом порядке.

4. Если *circle event'ы* совпадают, то у нас получится вершина, из которой выходит больше трёх рёбер. Специально обрабатывать такой случай не нужно, алгоритм просто добавит несколько вершин с тремя рёбрами в одно место. На этапе постобработки просто склеим их в одну.

5. Новая парабола может попасть на *breakpoint.* Здесь возникает дуга нулевой длины, её можно удалить позже.

6. Если есть три сайта на одной высоте, то они не образуют *circle event.*

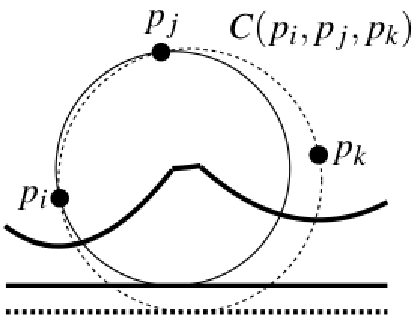
7. Когда очередь событий опустошается, береговая линия не пропадает. *breakpoint'ы* образуют полубесконечные рёбра диаграмы Воронова.

8. DCEL не может содержать полубесконечные рёбра, поэтому их нужно прикрепить к какому-то достаточно большому *bounding box'у*.

Корректность

Теорема. Каждая вершина диаграммы Вороного отлавливается в каком-то *circle event'е*.

Доказательство. Пусть есть вершина q, которая лежит на границах нескольких ячеек. Докажем для трёх: p_i, p_j, p_k. Пусть p_j между ними. Верно, что $C_p(q)$ проходит через все три сайта. Докажем, что мы добавили в очередь *circle event*, которое замечает удаление дуги параболы p_j. Возьмём момент *circle event'а* и немного подвинем нашу l вверх так, что в шарик, образованный p_i, p_j и l не содержал другие сайты. Очевидно, что ещё будет жить парабола p_j, так как в какой-то окрестности центра нового шарика каждая точка ближе к p_j, чем к p_k. Аналогично, для шарика, на образованного p_j, p_k и l. Короче в этот момент времени дуга параболы p_j действительно жива $\Rightarrow$ *circle event* будет пойман.



Время работы

Теорема. Алгоритм работает за $O(n \log(n))$ времени и использует $O(n)$ памяти.

Доказательство. Операции с сетом и очередью занимают $O(\log(n))$ времени. Операции с DCEL занимают $O(1)$. Время обработки события получается $O(\log(n))$. Всего n *site event'ов*. Каждый *circle event* образует новую вершину. Ложные *circle event'ы* удаляются из очереди до их обработки. Таким образом, количество *circle event'ов* в худшем случае $2n - 5$.