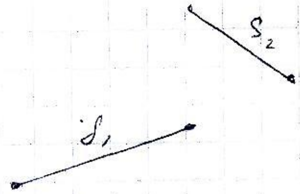
Антон.

> куча много книжек на тему вычислительной геометрии

Пересечение отрезков



Отрезок — хз))) в Евклиде
Пересечение — имеют общую точку

$S_1 : p_{11} p_{12}$
$S_2 : p_{21} p_{22}$

Координаты в Евклиде неудобно задавать

Пересечение отрезков — можем проверить пересечение прямых и проверить точку пересеч. на принадл. отрезку.

Концы точки мы поставим в разных полуплоскостях.

Аффинное пр-во — пр-во сост. из точек.

1. $\exists$ хотя бы 1 точка
2. $A, B \Longleftrightarrow \overrightarrow{AB}$
3. Точка + вектор = точка + еще 10 аксиом из векторного

$\exists$ d-мерного аф. пр-ва

$\sum \alpha_i v_i \neq 0 \Longleftrightarrow \sum \alpha_i^2 \neq 0$. Мы можем взять любую точку и сделать координаты

и

$\nexists$ $d+1$ - эл набор векторов ЛЗ.

1. Ориентация
↓
2. Объем

$\Big\} \rightarrow$ гиперпл-ть делит пр-во на 2 св. вып. области.

$\measuredangle$ гиперплоскость $\cdot$ $\{d-1$ ЛНЗ векторов$\}$ — d точек их задающих

$\overrightarrow{AP_1}$

Если $A \in$ плоскости, то этот набор ЛЗ и наоборот

$\overrightarrow{AP_d}$

Можем построить систему координат еще с точки B

Посмотрим как меняются координаты при переходе из A в B (система координат)

$$X = X_A^1 \overrightarrow{AP_1} + X_A^2 \overrightarrow{AP_2} + \dots + X_A^\partial \overrightarrow{AP_\partial}$$ $\cdot X \cdot \partial\partial\partial Mok \cdot$

$$= X_B^1 BP_1 + X_B^2 BP_2 + \dots + X_B^\partial BP_\partial$$

$$\overrightarrow{AP_1} = \alpha_1^1 \overrightarrow{BP_1} + \alpha_1^2 \overrightarrow{BP_2} + \dots + \alpha_1^\partial \overrightarrow{BP_\partial}$$

$$\overrightarrow{AP_2} =$$

подставим в

$$\overrightarrow{AP_\partial} = \alpha_\partial^1 \overrightarrow{BP_1} + \alpha_\partial^2 \overrightarrow{BP_2} + \dots + \alpha_\partial^\partial \overrightarrow{BP_\partial}$$

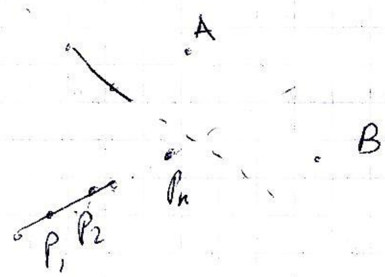
$$X = X_A^1 (\Sigma \alpha_1^i \overrightarrow{BP_i}) + X_A^2 (\Sigma \alpha_2^i \overrightarrow{BP_i}) + \dots + X_A^\partial (\Sigma \alpha_\partial^i \overrightarrow{BP_i}) =$$

$$= \overrightarrow{BP_1} \cdot (\Sigma \alpha_i^1 X_A^i) +$$

$$+ \overrightarrow{BP_2} \cdot (\Sigma \alpha_i^2 X_A^i) +$$

$$\vdots$$

$$+ \overrightarrow{BP_\partial} (\Sigma \alpha_i^\partial X_A^i)$$



$\Rightarrow$

$$(X_B^1, X_B^2, \dots X_B^\partial) = (X_A^1 \dots X_A^\partial) \begin{pmatrix} \alpha_1^1 & \dots & \alpha_1^\partial \\ \vdots & \ddots & \\ \alpha_\partial^1 & & \alpha_\partial^\partial \end{pmatrix} + (\overrightarrow{BA}^1 \dots \overrightarrow{BA}^\partial)$$

Если нам дано O : $\vec{e}$, то

матрицу A можно получить: $A = \begin{pmatrix} P_1 - A \\ P_2 - A \\ \vdots \\ P_\partial - A \end{pmatrix}$ ← типа строки

Тогда мы можем разбить всё пр-во на 3 класса согласно тому, какой знак перехода из O в A: <0, >0, $=0$, ЛЗ

То есть $O\{e_1 \dots e_\partial\}$ и $O'\{e_1' \dots e_\partial'\}$

A - матр. перех. от O к O' - $\{+, -, 0\}$

При переходе $O' \to O$ sign det A - const

sign det A = sign det A^{-1}

Ориентация - св-во точки отн. $O\{e_1 \dots e_n\}$ и симпл-ов: если $\det > 0$, то ориент. O и A совп.

$$\measuredangle \left| \begin{pmatrix} \overline{P_1}, & 1 \\ & \\ \overline{P_\partial}, & 1 \\ \overline{A}, & 1 \end{pmatrix} \right|^{(\partial+1)\times(\partial+1)} = \left| \begin{pmatrix} P_1 - A \\ P_2 - A \\ \vdots \\ P_\partial - A \end{pmatrix} \right|^{\partial\times\partial}$$

$\{\vec{A}t+\vec{B}(1-t)\}$ — мн-во точек отрезка

$$\begin{vmatrix} P_1 & 1\\ P_2 & 1 \\ \vdots & \\ P_d & 1 \\ At+B(1-t) & Pt+1(1-t)\end{vmatrix} = t\begin{vmatrix}P_1&1\\ \vdots & \\ P_d&1\\A&1\end{vmatrix}+(1-t)\begin{vmatrix}P_1&1\\ \vdots & \\P_d&1\\B&1\end{vmatrix}$$ — это д-т связности и выпуклость

$[0,1]$

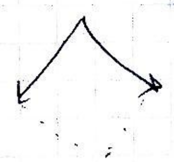
возьмем f $t \to A$, $f(0)<0$ $f(1)>0$, тогда $f=0$ достижимо $\Rightarrow$

$$V = \int\limits_W dx_1\, dx_2\, dx_3 \dots dx_d$$

$\downarrow$

$$V' = \int\limits_W dx_1'\, dx_2'\, dx_3' \dots dx_d' = \int\limits_W \left(\left|\frac{\partial(x_1' \dots x_d')}{\partial(x_1 \dots x_d)}\right| dx_1 \dots dx_d\right)$$

$\left|\frac{\partial \dots W}{\partial \dots}\right| =$

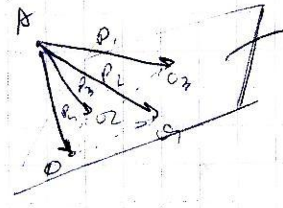


$$\int_0^1\int_0^1 dx_1\, dx_2$$

перегибаем в $O \Rightarrow$

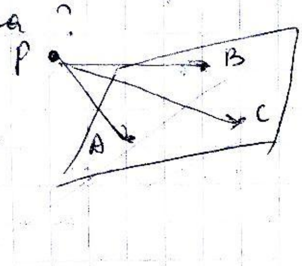
26.02.2015

Как доказать, что $P_1 \dots P_n$ ЛНЗ?



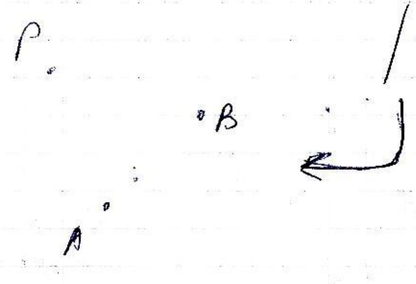
— гиперплоскость. Возьмем в качестве т. отсчета одну из точек в гиперпл-ти, тогда $n-1$ ЛНЗ векторов по опр. гиперпл-ти, значит OA нельзя выразить через $a_1 \dots a_3$, тогда $\{a_n\}$ ЛНЗ $\Rightarrow \{P_n\}$ ЛНЗ.

У нас есть мн-во и точка, с какой стороны нах-ся точка?



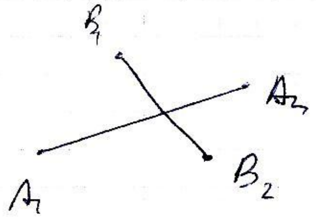
Знак от $\begin{vmatrix}A-P\\B-P\\C-P\end{vmatrix}$ зависит от того, с какой стороны нах. точка.

$$\begin{vmatrix}A & 1\\B & 1\\C & 1\\P & 1\end{vmatrix} = \begin{vmatrix}A-P\\B-P\\C-P\end{vmatrix}$$



$\left|\frac{1}{2}\begin{vmatrix} A-P \\ B-P \end{vmatrix}\right|$ — площадь Δ ABP

Итак, пересеч. отрезки:



Мы можем хранить в дробке P, Q, определяющие рациональное или $\frac{P}{Q}$ - рац. опред.
Либо gmp — mpq-class

С другой стороны, это все 4 порядка медленнее double

fp - floating-point-operations

$a \oplus b = c$ ← floats
$a + b = c$ ← rational

$a \oplus b = (a+b)(1+\delta)$

$\otimes \ominus \oslash$ $\quad |\delta| < \varepsilon_M$

$< 2\varepsilon_m$

Давайте оценим погрешность для $|2\times 2|$

$$E = \left|\frac{1}{2}\begin{vmatrix} A-P \\ B-P \end{vmatrix}\right| = (a_x-p_x)(b_y-p_y) - (a_y-p_y)(b_x-p_x)$$

$$\tilde{E} = (a_x \ominus p_x) \otimes (b_y \ominus p_y) \ominus (\dots)$$

$$\varepsilon(a,b,p) = |E - \tilde{E}|$$

Заменим fp на op:

$$\varepsilon(a,b,p) = \Big|E - (1+\delta_7)\Big[(a_x-p_x)(b_y-p_y)(1+\delta_1)(1+\delta_2)(1+\delta_3) - (a_y-p_y)(b_x-p_x)(1+\delta_4)(1+\delta_5)(1+\delta_6)\Big]\Big| \le$$

$$\le \big|(a_x-p_x)(b_y-p_y)(1-(1+\delta_1)(1+\delta_2)(1+\delta_3)(1+\delta_7))\big| + \big|(a_y-p_y)(b_x-p_x)(1-(1+\delta_4)(1+\delta_5)(1+\delta_6)(1+\delta_7))\big| =$$

$$= |(a_x-p_x)(b_y-p_y)|\,\underbrace{|\delta_1\delta_2\delta_3\delta_7 + \dots + \delta_3 + \delta_7|}_{\le 4{,}5\,\varepsilon_M} + \dots$$

то есть

$$|E - \tilde{E}| \le 4.5\,\varepsilon_M \left(|(a_x-p_x)(b_y-p_y)| + |(a_y-p_y)(b_x-p_x)|\right),$$

$E \in [\tilde{E}-k; \tilde{E}+k]$, где k – $4{,}5\varepsilon_M(|\cdot| + |\cdot|)$.

тогда если $[\tilde{E}-k, \tilde{E}+k] \not\ni 0$, знак $\tilde{E}$ совп. с знаком E

Но как считать k? Δ первую скобку

$$|(a_x-p_x)(a_y-p_y)| = |(a_x \ominus p_x) \otimes (a_y \ominus p_y)| \underbrace{\frac{1}{(1+\delta_1)(1+\delta_2)(1+\delta_3)}}_{\le (1+4\varepsilon_M)}$$

$(1-\delta_3)$

A_1 второе

$\le (1+4\varepsilon_M) \to 4{,}5\varepsilon_M$ (раскладываем по Тейлору, мажорируем)

$$|E-\tilde{E}| \le 4{,}5\varepsilon_M((1+4\varepsilon_M)(A_1 \oplus A_2)) = (4{,}5\varepsilon_M \circ (1+4{,}5\varepsilon_M))(A_1 \oplus A_2) \le$$

$$\le 8 \cdot \varepsilon_M \,(*)\, (A_1 \oplus A_2) = 8\varepsilon_M \otimes (A_1 \oplus A_2)$$

Тогда теперь если $|\tilde{E}| \ge \tilde{k}$

Короче считаем знак, проверяем точность, иначе считаем в рациональных.

$$fp \rightarrow mpq \rightarrow ret$$

– дорого переходить.

→ Boost/interval

Интервальная арифметика: $a+b \le a \overline{\oplus} b$

$a+b \ge a \underline{\oplus} b$

На каждом шаге считаем, что если имеем $v \in [\underline{v}, \overline{v}]$, сможем в итоге вычислить $E \in [\underline{E}, \overline{E}]$

$[\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] = [\underline{a} \underline{\oplus} \underline{b}; \overline{a} \overline{\oplus} \overline{b}]$

$[\underline{a}, \overline{a}] - [\underline{b}, \overline{b}] = [\underline{a} \underline{\ominus} \overline{b}; \overline{a} \overline{\ominus} \underline{b}]$

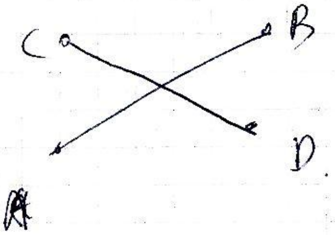
$[\underline{a}, \overline{a}] * [\underline{b}, \overline{b}]$ – посчитать руками, выбрать min, max

$[\underline{a}, \overline{a}]^2$ – хорошо точнее, чем $[\underline{a}, \overline{a}] \times [\underline{a}, \overline{a}]$ (min ≥ 0 хотим)

Тут точнее, потому что оценка в fp была очень грубая и недостижимая (особенно $|a-b| \le |a|+|b|$)

+ еще профит – компилятор может поменять это в код исполняемый на сопроцессоре. ($10б > 8б$) Точность лучше

– Интервальная медленная – округление менять попарно дорого. (не $a \overline{\oplus} b = -((-a) \underline{\oplus} (-b))$)



Если C отн. AB лежит с другой стороны от D и аналогично для другого отрезка, тогда отрезки пересекаются.

Вычисление точки пересечения.

Можно использовать интервальную арифметику

$\mathcal{Q}\ k(x-x_0)+b(y-y_0)=0$

Выпуклые оболочки convex hull

C $\mathbb{R}^d$, n точек

C $CH(A) = \bigcap \tilde{A}$, где $A \subset \tilde{A}$ — мн-во п-пий сод. точки из A

Многогранник $H = \bigcap P_i$ + H-огр. — полупространств

O $B \subset \mathbb{R}^d$ огр. если $B \subset \Pi[a_i, b_i]$

Строим многогранник — берем все гиперпл-ти, в полупр-ве оф. точки с одной стороны — C_n^d — берем точки и, находим все в 1 стор.

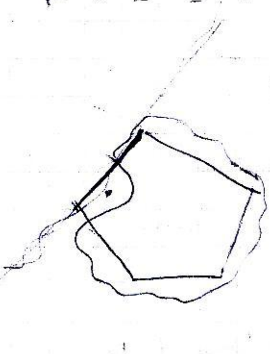
Л $\tilde{CH}(A) = CH(A)$, /CH - выпукл. $\tilde{CH}$ - многогр/

1) $\tilde{CH}(A) \supseteq CH(A)$ - очевидно

2) $\mathbb{R}^1$. $\tilde{CH}(A) \subseteq CH(A)$

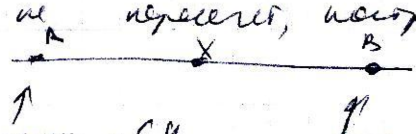
Инд-е $\mathbb{R}^{d-1} \Rightarrow \mathbb{R}^d$

$\leftarrow \mathcal{Q}$: $\exists q \in \tilde{CH}(A)$
$q \notin CH(A)$



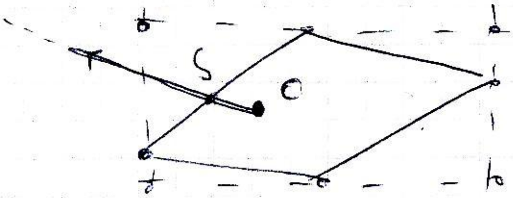
4) Находим пр, пересек. CH

Если не пересечет, построим прямую



$AB \subset Q$
$X \in [A, B]$ — входит в отр на-во

б) Хотим доказать, что луч из O пересекает гиперпл-ть.



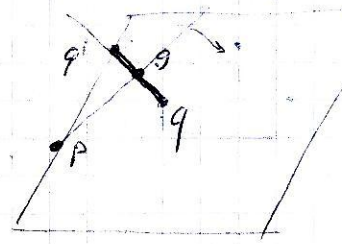
строим куб ограничивающий конструкцию, если совпас хотим найти какую-нибудь огр. гиперпл-ть.

Хотим найти S.

ДЗ

Заливс

Имеем и точки.



Построим вокруг них куб

$g \in [q' q]$

Мы хотим двигать Pq' к Pq пока она не встретит какую-нибудь $(\cdot) \in A$

типа f — объем по мн. вз точек?

В $\mathbb{R}^d$ просто такой

$(\underbrace{P_1, P_2,}_{A}\ \underbrace{P_d}_{g})$ Заменим $P_d \notin A$ на g, подвинем, пока не найдем первую точку

g движ. по $q' \to q$ линия $t q' q$ $t \in [0,1]$

<u>Повторение</u> д-ва с нуля $\quad Ch(A) = \cap Conv \,/\, Conv \supset A$

1. База очевидна ТеО

2. $d-1 \to d \quad S^{d-1} \to S^d$

1) $\exists A' \subset A \Rightarrow Ch(A') \subseteq Ch(A)$

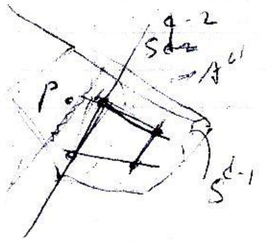
$\triangle \ \exists A$, Conv $\supset A$, тогда Conv $\supset A' \Rightarrow Ch(A') \subset Ch(A)$

$\leftarrow$ 1. $\exists p \in \cap H$, $p \notin Ch(A)$
$\quad \in$ halfspace

а) $p \in HP$, (лежит на грани)
б) $p \notin HP$

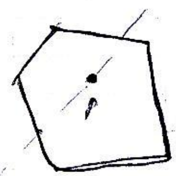
⊙ Она (гп-ть) содержит d точек из $A' \subset A$, значит $Ch(A') \subset Ch(A)$, $p \notin Ch(A) \Rightarrow p \notin Ch(A') \Rightarrow$

Е нар-ре грани ∃ гиперпл-ть, отд. ее от всех остальных точек



Мы хотим грань из S_n, что она содержит грань из S_{n-2}.
Мы будем поворачивать S_B^{d-1} вокруг S^{d-2} пока она не коснется точки из A, тогда A в одной стороне, A с другой. Значит оно не построение и в S_n

(б) Типа выборы



Почему (б) → (а).
Пусть прямая через p. Она делит на 2 луча, один из них не пересечет CH (иначе оба принадл. CH, по опр. выпукл. мн-ва весь отрезок должен входить в $CH(A)$, но p не входит в)

Нетривиальные вещи:

1) пр пересекает грани
2) как вращать гиперпл-ть?

(2) F_1 – грань, кот. принадл. точки

$A' \subset A \mid \#A' \geq d$

Сложность – кол-во пар точек

$A'' \subset A'$, E_1 – пр-во S^{d-2}, $A'' \subset E_1$, $A' \subset F_1$

все с одной стороны кроме p

$\forall a \in A', a \notin E_1 \quad orient(a, E_1) \neq orient(p, E_1)$

Хотим показать, что $\exists a \in A \;\; a \notin F_1 \mid F(a, E_1)$
$\forall b \in A$
$b \notin F$
$orient(b, F) \neq orient(p, F)$

Возьмем $F(a, E_1)$ – пл-ть, $a \in A \setminus A'$
она делит $\tilde{A}$ на $\tilde{A}$ и $\hat{A}$, → и !
тогда если $\hat{A} = \varnothing$, то все хорошо, если нет, то будем брать новые точки из $\hat{A}$, $\#\hat{A} \to 0$

$\measuredangle \; orient(a', F') = \cancel{det} \; sign\left(det \begin{vmatrix} E_1 & 1 \\ & 1 \\ & 1 \\ a_1 \dots a_d & 1 \\ a_1^d \dots a_d^d & 1 \end{vmatrix}\right) \Big\} d-1 \leftarrow a$

Если $\measuredangle \; orient(a, F)$, то sign = sign пр.

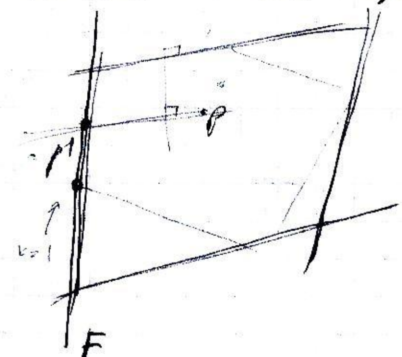
Хотим показать, что если $a' \in \hat{A}$, то в итоге $\#\hat{A} \to \varnothing$ (нужно исп. непр. опр.)

① Возьмем миним. охв. куб (мы можем построить выпуклую оболочку и куб если точек $\ge d+1$)

Возьмем куб $\parallel OX$ — наимен-ий, пересекли грань, в ней есть как минимум 1 точка из A — k мерн ($k \le d$, если $k = d$, то мы нашли грань)

$A_F = A \cap F$

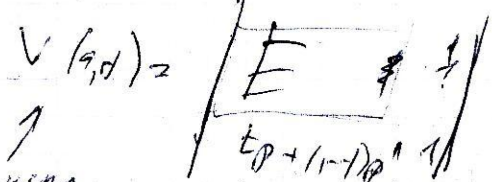
$\# A_F \ge 1,\ 0 \le \dim(A_F) \le d-k$.



$tp + (1-t)p$

$\min_{a \in A \setminus F} (\cdot V(a)) = f(d)$

$d-1$ мин. точек на огр. гранях куба



непр.

Как только мы найдем точку, то $f = 0$.

Короче мы взяли $p' \to p$. и $d-1$ точек из F, нашли точку из A, добавили ее, уменьшили кол-во точек в

☆ Тут были причитывания алг. на hull ☆

26.03.2015

Сегодня: 1) Динам. hull

2) Выпукл. оболочка за $O(n \log n)$ Chan

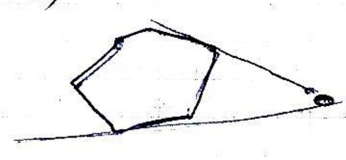
3) Пересечение полупл-ей

4) CH $\mathbb{R}^3$

1. Инкр. оболочка

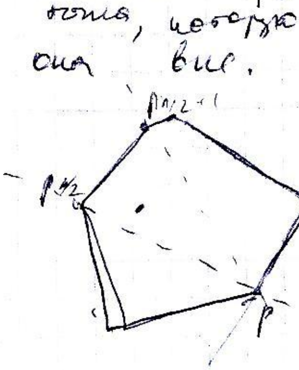
Проверка внутри или снаружи, найти тангенсы (касательные)

Кас:



Как понять, внутри мы или нет?

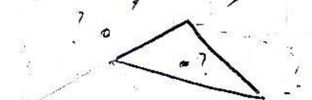
Возьмем P_0 — мин. элемент CH. $\forall P_{n-1}$, если точка, которую мы проверяем, лежит справа, тогда она вне.



Найдем сектор, в котором лежит точка.

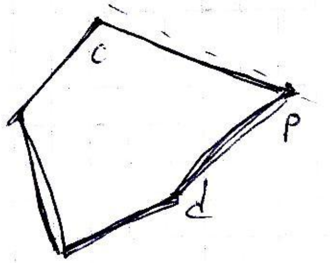
Возьмем $P_{n/2}$, проведем $PP_{n/2}$, поиск перебором и двоичным поиском.

Когда определили сектор, смотрим, вне или внутри сектора



Реализация: lower_bound (p+1)

Как искать касательные?

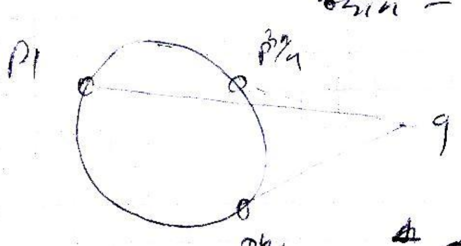


Проверим поворот qpc

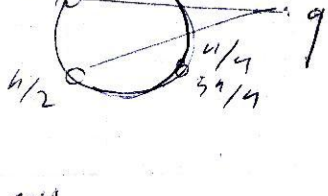
Точки бывают 3х типов:
1) опорные
2) по час ↺ вып.
3) по час ↻ вып.

проверим, что p c и d с одной стороны qp.

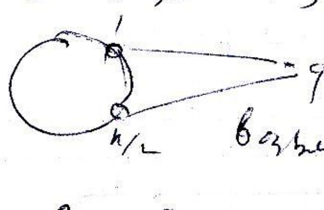
Если кас. — то −− или ++.
вып — +− вогн. —+



Если разные выпуклости, то все ок, делим пополам

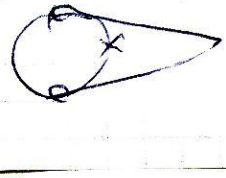


→ возьмем 3n/4,



возьмем n/4

Когда найдем



— хранить значения лучше в декартовом, а не в массиве

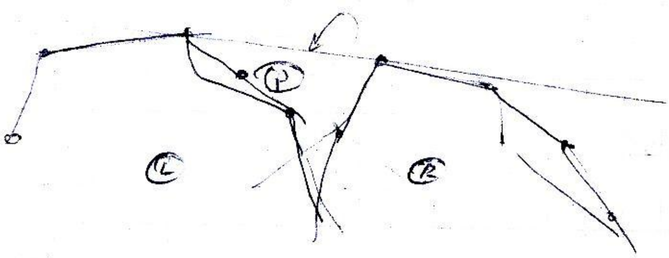
• Удаление точки (декремент.)

Алгоритм: del/add за $O(\log^2 n)$

Будем уметь поддерживать верхнюю цепь. можно за $O(\log n)$ (не рассказываем)

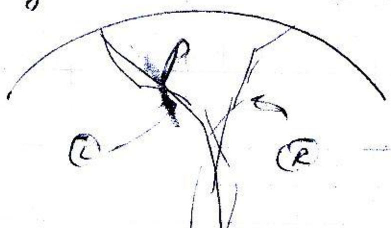
Будем держать дерево (сбаланс. двоич.) упоряд. по X точек. Когда для каждой вершины уметь как-то восстанавливать СН сверху. Точка отмечается

в т. хранится ~~КНН~~ опт. верхняя для под. поддерева + мост между СН детей



Хотим в P хранить что-то, что может сделать СН из СН сыновей

Сделаем так

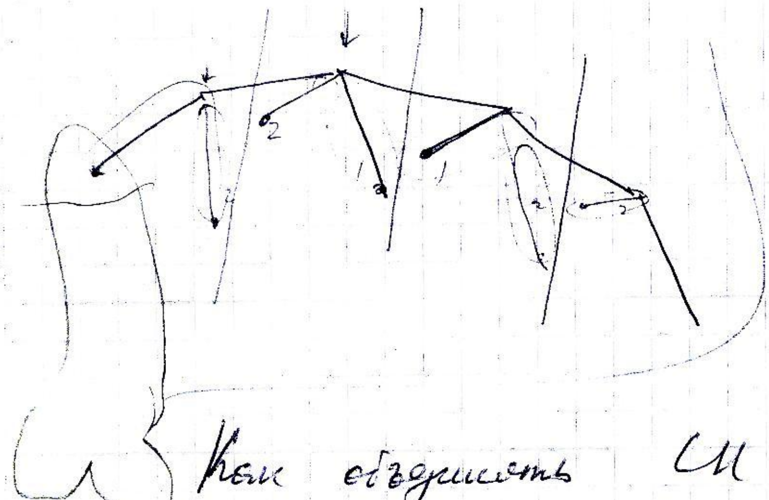


Точка в P хранит всю СН, ~~а в L и R хранится~~ и еще какие-то доп. вершины, которые позволяют разбить СН на 2

в узле храним кусок цепи и указатель откуда клеить подцепи

add/del:

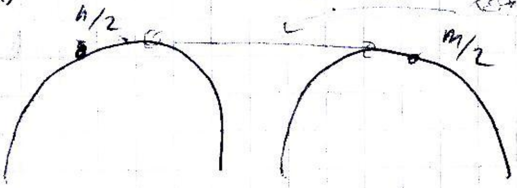
Прошли вниз, разрезанием вверх с восстановлением построением новой СН

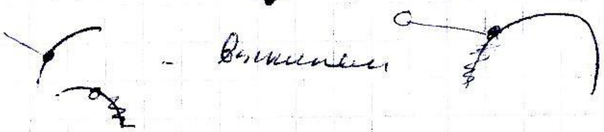


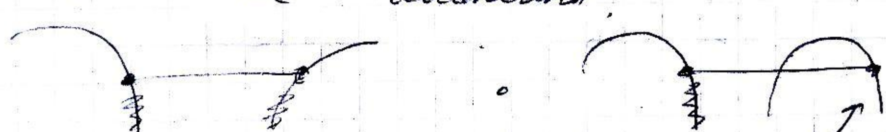
Так, идея.

Как объединить СИ при s[illegible]?

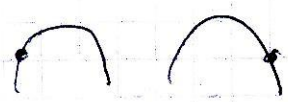
Делаем биткоин. [illegible] 2 опорные вершины.



- оп — оп - V.
- оп -  - вершина
- оп —
- ☆ - оп аналогично.

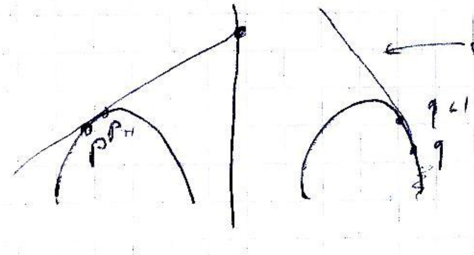


нельзя здесь.



- вершина не освещена

Делаем так: построим нас:



это выше, это вершина, [illegible]

Алгоритм Чана (Chan)

Построение CH за быстрее чем $O(n \lg n)$

Можно быстрее асимптотически, если в оболочке вершин << чем всех.

(Quickhull за $O(n \lg k)$ нечасто)

Пусть мы знаем число k. k - число точек в CH

Разобьём все точки на $\lceil \frac{n}{k} \rceil$ групп, для каждой такой группы строим CH за $k \lg k$

$$\lceil \tfrac{n}{k} \rceil k \lg k = n \lg k$$

для каждой группы построим опорную прямую за $\lg k$. Касательные за $\lg$ частей, суммарно

$\lceil \frac{n}{k} \rceil \lg k$, на шаге Jarvis'ом обходим.

$k \lceil \frac{n}{k} \rceil \lg k = n \lg k$. Как найти k?

Будем искать в виде 2^{2^t}

Запустим при $k = 4$, не получилось - запустим

—//— $k = 16$, ...

Вывод $\sum_{t}^{\lg\lg k} n \lg 2^{2^t} = n \sum_{1}^{\lg\lg k} 2^t = n \lg k$

Хотя он оптимален, ~~но~~ quickhull быстрее

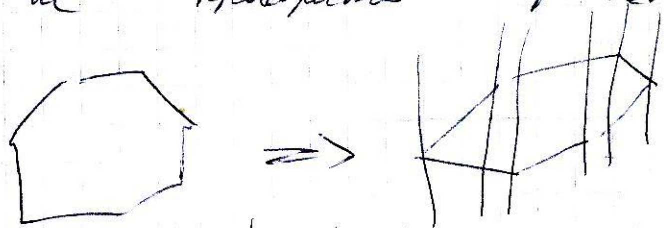
2.04.2015

Построение многомерных оболочек (3d, 4d-1)

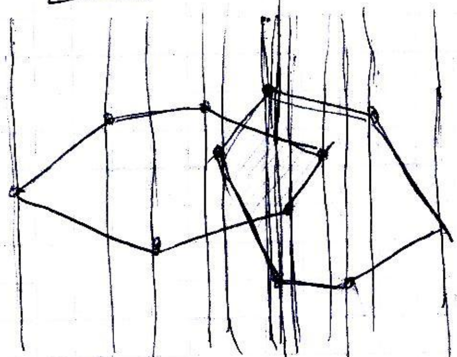
1) Найдём за $O(n)$ $d+1$ из точек.

9 ап. 2015

1) Как проверить пересечение многоуг.?



разбиение на трапеции



Пересечение многоугр. — это пересечение верт. трапеций

Локализация — проверка точки на принадлежность фигуре.

Мы как-то за $n \log n$ делали бинпоиском.

2) Невыпукл. многоугольники.

Внешняя граница, определит внутренность.

Она делит плоскость на 2 части, внутр. границы должны пересекаться в н-ть

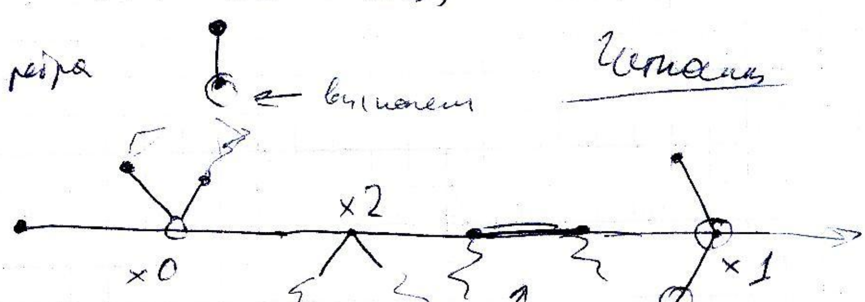
border: [point]
inner_border [[point]]

3) Определим принадлежность точки многоугр.
(пусть inner_border нет)

Возьмем все ребра

Частности

Луч



Бросить луч можно горизонтально (го вправо).
Если в одну сторону, чем больше отрезков наверх.

Для внутр. границы так же

4) Локализация за $O(1)$ асимпт.

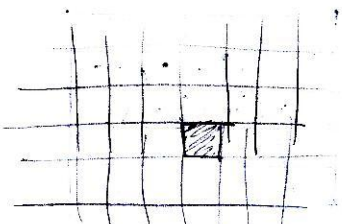
Зафиксируем масштаб многоугр.

Квантизация

Разбиение плоскости на клетки (hash)

$$floor\left(\frac{p}{size}\right)$$ хранение

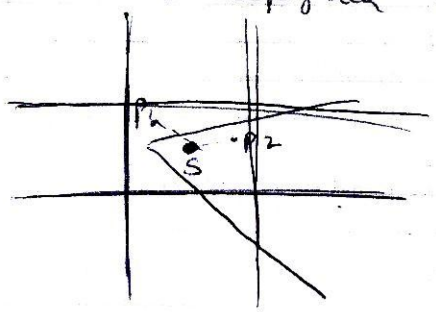
где клетки



Нас интересует квантизация точки относит. клетки,
Кол-во отрезков, пересекающих клетку.

ℓ - размер клетки,
$2\ell \sim \ell$
$\ell \geq$ длина ребра (средняя) окружн.

Предположим, что для каждой клетки мы знаем, находится ли ее вершина внутри многоуг.

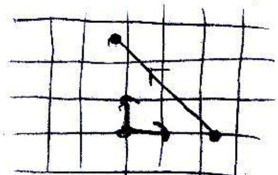


Если ∂ p_i и S ~~посчит.~~ кол-во пересеч., где их квантиз. совпадает

Мы уже знаем внешний контур против часовой стрелки

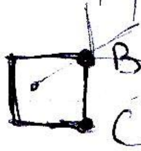
Но лучше бежать по манхэттенскому отрезку

б) Как определить, какие клетки пересекает отрезок?
Растеризация

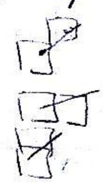


$[x, x+1) \times [y, y+1)$

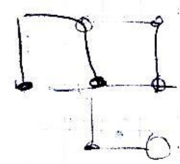
Проверяем отрезки в генератор пор-ке (слева направо)
Пусть отрезок идет вверх. Проверим поворот с точкой B



поворот $0 \to x+1, y+1$
$1 \to x+1, y$
$-1 \to x, y+1$

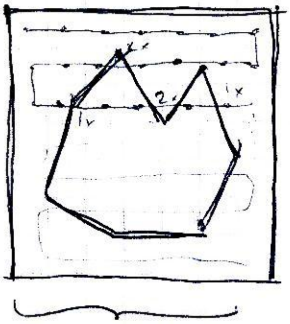


Вниз проверим поворот с C
Если 0, то вправо.



В итоге, мы можем сохранить лишь так отрезки. Итерирование по клеткам.

в) Пусть у нас есть bounding box - прямоуг. по сетке. Для внешних точек клеток квантиз. не нужно. (внутри горизонт $X_{max}-X_{min}$, $Y_{max}-Y_{min}$)
Для внутр. точек смотрим заметание



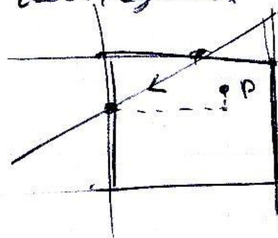
$O(n)$

кол-во ребр. (или ?) по условию!

Но мы хотим запомнить $O(n)$ ячеек, потому что

Точки в среднем выпуклые.

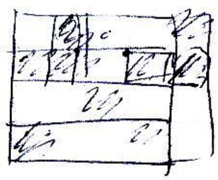
Или для ячеек в HashMap относительно пересеч. отрезков необходимо:



интерполяция определение направл. ребра, если не пересекли.

Таким образом, мы проинициализировали все ячейки в HashMap

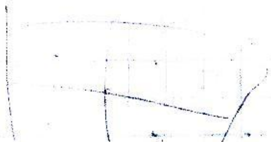
Что-то...?

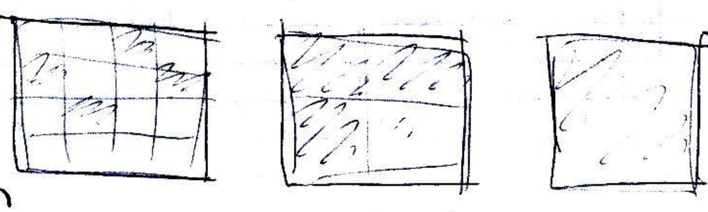


что-то скрыли, ищем соседние...? Что?

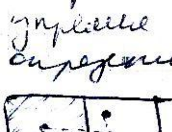
7) Тут что-то про квадродеревья, но Антон думает, что мы знаем, что это.

Закрасим квадрат, если его подквадраты закрашены, иначе их





Спускаемся по квадродереву, пока мы не упремся в незакраш. ячейку, потом как-то определим пограничную через соседние



+ Есть проблема с памятью — $O(n)$ закраш. ячеек. И квадро-дерево хранит.

$O(n \log)$ памяти (дерев: кол-во ячеек)

xy → [segment_id]

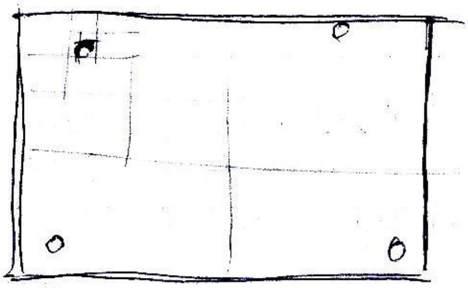
Можно вместе с Map[Cell → Data] ~~...~~ Map[~~Seg~~ Cell]]

Можно без квадродерева, хранить для ячейки hashmap вместо дерева

16.04.2015

Что делать, если есть несколько зон, но они несвязны и тогда глубина квадродерева большое.

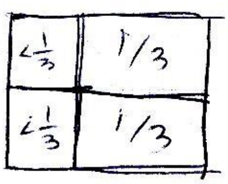
Хотим <u>множество</u> многоугольников.



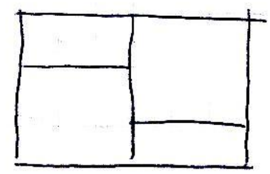
1) Делать сетку, если в зоне только 1 точка, в лист

—: может быть диагональное распределение точек, тогда это неэффективно.

2) Делим по $\frac{1}{3}$



3) K-D дерево

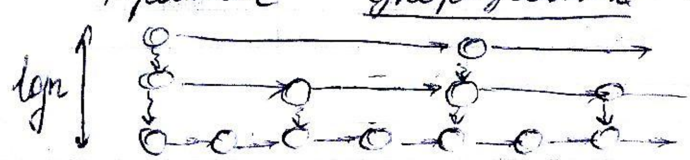


—: в таких деревьях нельзя сразу можно добавить элемент, теряем инкрементальность.

Это можно исправить перестраиваемым деревом — google Overmars. Можно поддерживать сбалансированно за $O(\log^2 n)$

Skip-list:

Хранит <u>упорядоченное</u> множество:



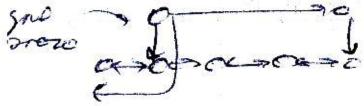
Это назыв. Бернуллиевский процесс

Сделаем примерно $\log$ таких уровней, тогда за $O(\log n)$ можно искать, а по памяти $2\cdot O(n)$.

+ хотим добавлять/удалять.

При добавлении с вероятностью p добавляем на уровень вверх, не добавляем с $(1-p)$,

И ещё список должен быть двусвязным



Удаление — удаляем столбец.

Mem — $O(n)$

Высота столбца $E(M) = 1 + \sum k p^k (1-p)$ — матожидание высоты столбца

$M \approx \sum k p^{k-1} = \left(\sum p^k\right)'_k = O(1)$

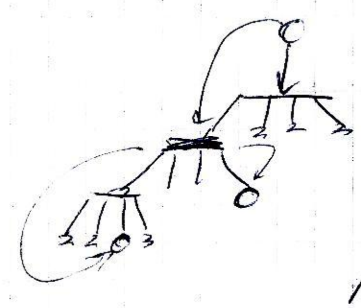
+ Поверим (там сложно) высота — $\log n$

Skip-quad-tree

Потребуем от квадродерева сжатие путей — мы добавляем в ноде только если ≥ 2 пути

Использование shortcut'ов

В ноде храним четыре указателя, но если...

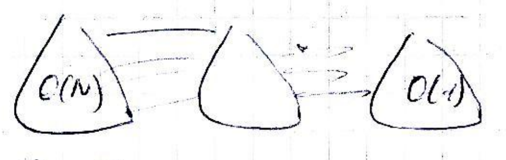


Вершина существует, если в ней есть хотя бы 2 поддерева.

Уровни у детей фиксированы, могут меньше только указатели.

При добавлении можем добавить 1 лишнюю вершину, при удалении — удалить лишнюю.

Добавим Skip-структуру, добавление вероятностно.



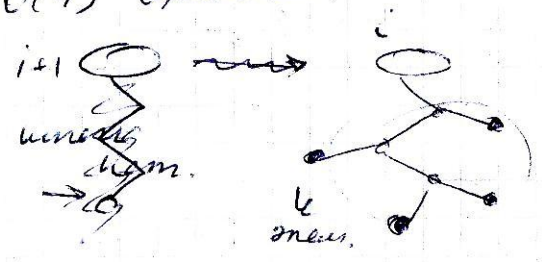
s-level $i \to i+1$
$x \geq 1 \quad x \geq 2$

Работаем только с точками, не с узлами.

Base tree, s-level = 0

Тогда локализовавшись в верху (самое глубокое в skip-дереве), можем перейти ниже по skip-дереву.

Почему мы на каждом skip-level проводим $O(1)$ времени?



Заходим в структуру, но уткнулись в $\varnothing$ — идем дальше на i уровень.

$$\sum k(1-p)^k = O(1)$$

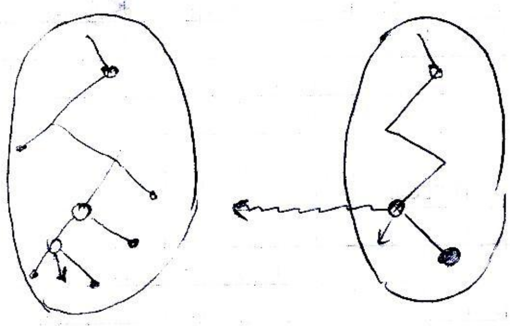
— глубина поддеревьев, которых нет в $i+1$, но есть в i — $O(1)$

Короче говоря, маловероятно, что поддеревья будут очень длинными при переходе к базе skip-структуры.

Как это использовать вообще?

Мы хотим локализоваться — найти вершину в skip-quadro-дереве, внутри которой есть базовая точка

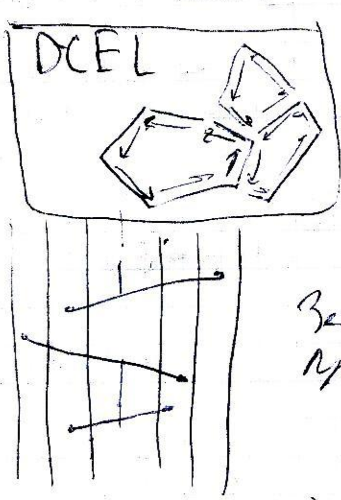
Нам нужно знать, для каждого выкрашенного куска, что в его области есть вершины. вершины



Всегда спускаемся на базовый уровень skip-tree, а там уже разберемся.

Мы хотим динамически добавлять отрезки. Будем думать, что узел наполненный, если он содержит k отрезков. При добавлении k+1 отрезка чем нам сетка не нравилась? Не динамично!

Как еще можно локализоваться?

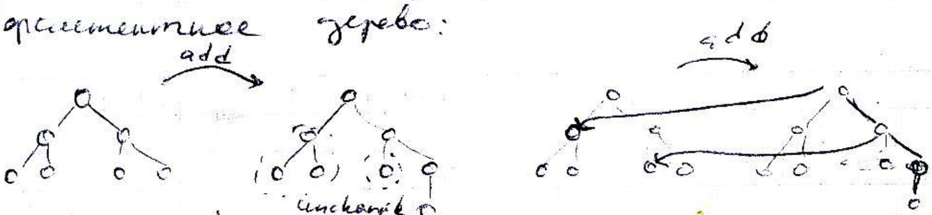


Разделим пл-ть на полосы, которые содержат концы отрезков. Всего $O(n)$ полосы. В каждой полосе $O(n)$ точек.

За $O(\log n)$ можно локализоваться. Проблема — сильно хранимость

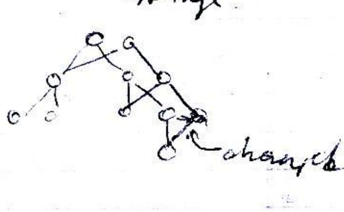
Хотим $< O(n^2)$ памяти.

Персистентное дерево:



Simple tree

change:



delete: the same.

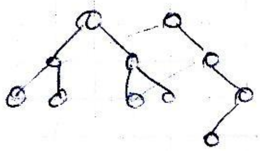
$O(\log N)$ памяти

$O(\log N)$ времени. можно за $O(N)$

23.04.2015

Сбалансированные персистентные деревья, хранить параллельно.

Не обязательно почти на дереве при write — персистентное дерево само делает все хорошо



делаем CAS

CAS(newroot, oldroot)

происходит редко (тогда если редко пишем)

Если не совпало, убьем newroot и сделаем добавление еще раз.

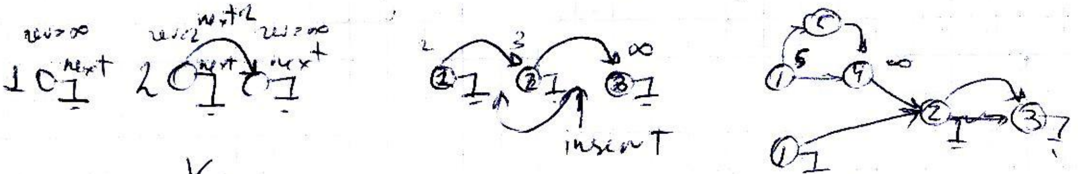
Мы делали поиск, за log найти поиск, храним n log n памяти для n отрезков.

- На листах персистентность:

На каждую операцию $O(n)$ 

Поэтому время храним в вершине 2 указателя, тогда в вершине есть revision — половинке

если идем по списку в $2 \le$ revision, идем по next2, иначе по next 2



Копируем узел, если надо.

На этом: амортизационно $O(1)$ времени, памяти — тоже

list

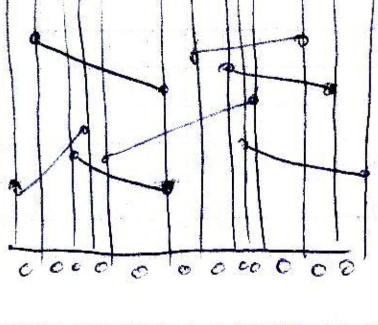
[root]

[revision]

- А с деревом?

next, bool, revision id   если rev ≤ rev id

left or right subtree

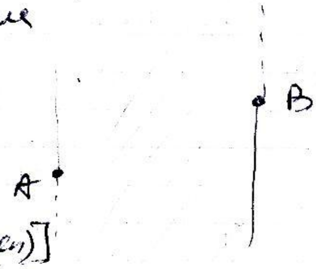
- 

Rotate ~~Graph~~ на (X, Y) на ... x заливаем

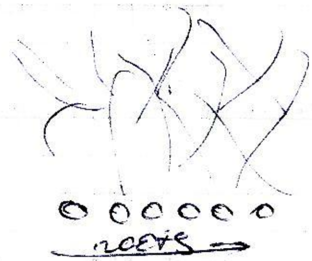
skew $(x, y) \longmapsto (x + \varepsilon y; y)$

Событие — добавл. отрезка

$$A \leq C < B$$



У нас полоска → [(root; qn)]

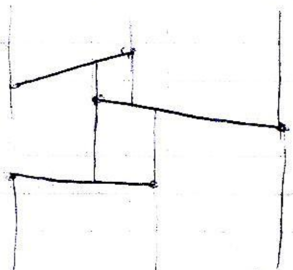


инкремент добавление отрезков в структуру — offline задача.

Tarjan 1986 article about

30.04.2015

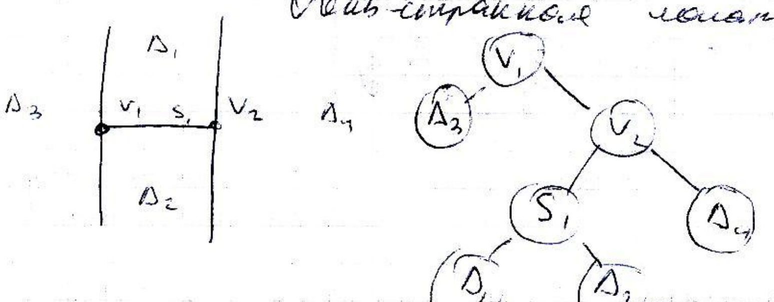
Давайте вместо сетки делать трапеции.



$O(n)$ трапеций по ф. Эйлера

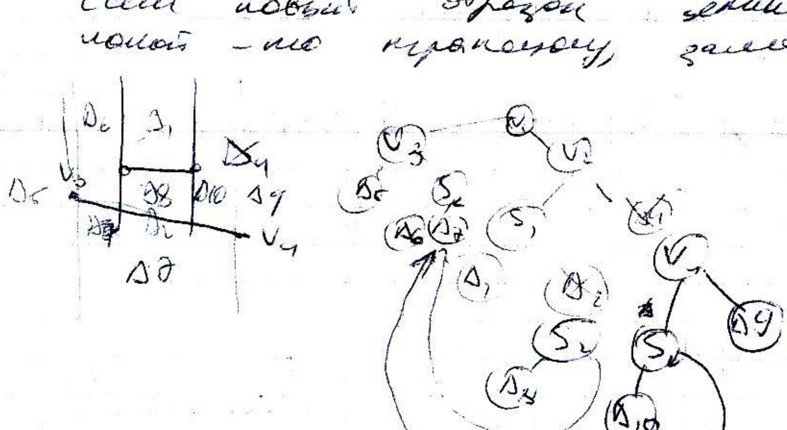
Δ, s-node, v-node

Очень странная локализ. структура



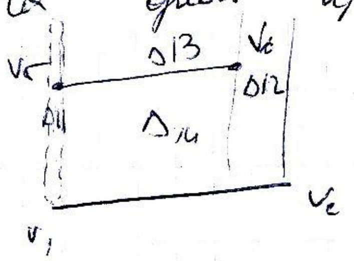
Добавим сюда новый отрезок

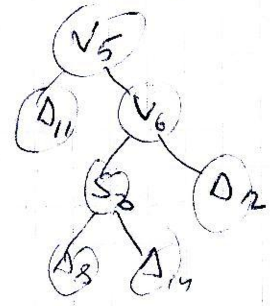
Если новый отрезок целиком попадает в новой-то трапецию, заменим его.



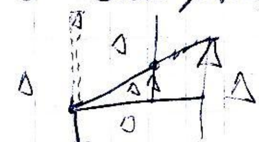
struct trapezoid {
trapezoid* l_top,
l_bot,
r_top,
r_bot,
vertex* left, *right;
}

Из двух правых (верш)



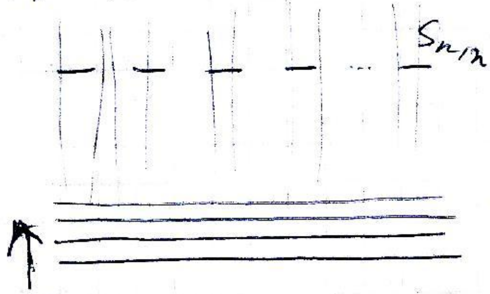


в ситуации



можно так же

Проблема: много одинаковых длинных отрезков в дереве:



Важен порядок добавления ребер.
Давайте рандомно!

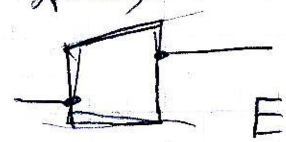
удлиняет очень

q - глуб., i - кол-во отр.

$i \to i+1$

q на сколько изменилось

Будем думать, какой отрезок на i+1 шаге нужно убрать, чтобы увел. глубину q.



- 4 таких отрезка

$$E = \sum \frac{4}{i+1} \cdot 3 \le 12 \log N$$

вер-ть взять плохой отрезок

худший вариант увелич. глубины

С памятью:

$i \to i+1$

+ 2 вершины, + кол-во пересеч. трапецоидов.
хуже — $O(n)$

Рассмотрим, сколько Δ затронет отрезок, если мы его вынем.

Каждый трапецоид касается 4 отрезков, всего $\Delta = O(n)$

$$f(s_i, \Delta_j) = \begin{cases} 1 & s_i \text{ def } \Delta_j \\ 0, & \text{otherwise} \end{cases} \qquad N' \text{ - кол-во } \Delta$$

$$\sum_j^{N'} \sum_i^{N} f(s_i, \Delta_j) \le 4N' \qquad \Big| \Rightarrow E = O(1)$$

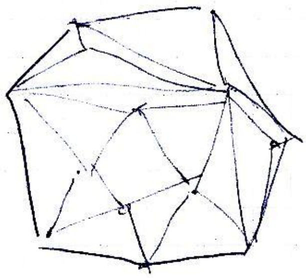
$$\| \sum^{N} \sum^{N'} f(s_i, \Delta_j) = N \cdot E$$

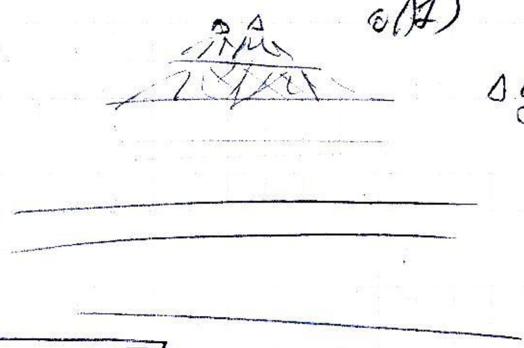
именно Δ нач.

K. Mulmuley 94

Имплементационно — буферизирование, модификация, много.

Kirkpatrick algorithm





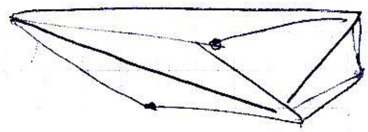
o(Δ)

Δ lg N

Δ_{i+1}

Δ_i

Δ_1



Пока у нас связь на уровне "лежит" не один Δ в другом, если вставляем вершину и перетриангулируем.

$\Delta_i \to \Delta_{i+1}$: удаляем набор точек, что-то перетриангулируем.

Δn;

У каждого Δ не больше $O(1)$ потомков и не выбираем слишком больших.

Будем ограничивать вершины степенью и выбрасывать, если степень $\geq d$.

$\alpha \leq \frac{1}{d}$ Средняя степень вер. в триангул. — 6.

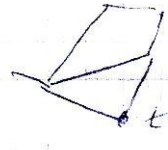
$d \leq 12$ $\alpha \geq \frac{1}{12} \cdot \frac{1}{2}$

2: или триангулировать, локализация в Δ — 3х изящных поворота

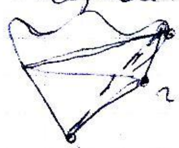
7.05.2015

Внешние триангуляции. 3х-а внешние ~~треугольники~~, которые в двойственном графе — нет.

Если мы найдем внешн. в графе вершин и найдем их отражение, то это захватит соседние узлы



t — 3-о



2

Просто проверяем соседей

затриангулировать за $O(n^2)$

Хранить вершины

1) List — too much malloc's
2) List on vector — Vector<(T, size_t prev, size_t next)>

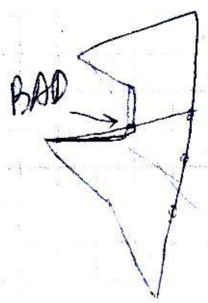
Можно за $O(n \log n)$

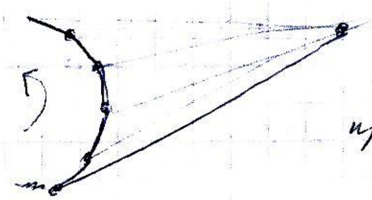
Монотонный многоугольник (относительно прямой l) если любая прямая $\perp l$ пересекает границу многоуг. не больше 2х раз

Многоугольники с дырками не монотонны.
Монотонный многоугольник можно триангулировать за линию

Диагональ монот. многоуг. бьет его на 2 монот.

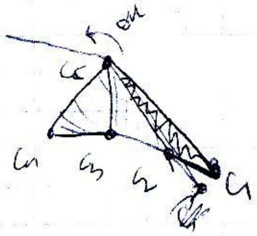
Если бы все повороты были левые, мы бы просто соед. точки по заметающей прямой.





Что делать с вершинами, имеющими правый поворот

Идем снизу вверх по часовой, если встретили правый поворот, отрезаем пред. вершины пока не станет левым

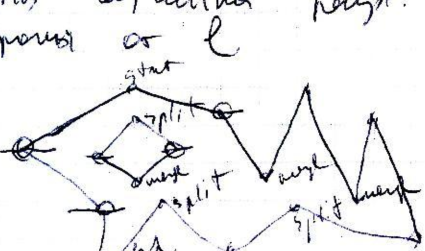


на c_5 отрезаем $c_{4,3}$

Итого за $O(n)$ работает.

Как искать монот. многоугол.?

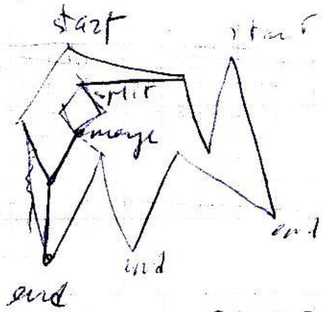
Пусть вершина регул. если ее соседи по разные стороны от l



Если соседи вверху — это end-вершина (поворот I)
снизу — start-вершина

Если в □ есть хотя бы одна split/merge-вершина, он не монотонен.

Перебьём □ на полосы по вершинам

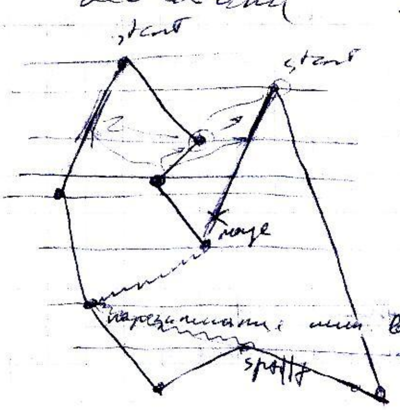


⌇\ — границы опр. ребрами

Смотрим на левые ребра

Сохраним с помеченного самую низкую вершину которая выше заметающей прямой

Добавим вершины в очередь с приоритетом.



Снизу заметающ. прямая — упор. мн-во отрезков
Храним вместе с полосой низшую вершину выше прямой

Что с этим сделать? DCEL, помечаем faces, находим face триангулируем отдельно.

14.05.2018 Пересечение многоугольников

1. Пересечение ∀ отрезков

Как найти т. пересечения отрезков? В рациональных BigInt всё ок, а если float, то точка неточно будет хорош.

$\varepsilon = 2^k$, разобьём площадь на квадраты, будем выдавать представителем.

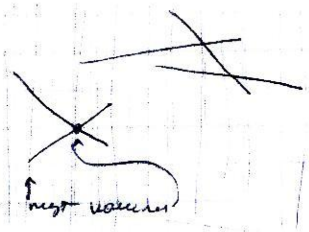
point intersect (segment_t a, segment_t b, int prob)
Приближаем float-отрезки — рациональными.
$floor(p \cdot 2^k) \rightarrow \frac{1}{2^k}$

Нужно считать в интервальном.

За $O(n^2)$ пересечение всех отрезков!

Можно построить Q-tree, т.е. хз.

Можно заметать прямой:



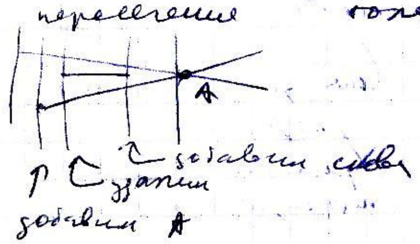
Храним события (→ точки) в лексикогр. порядке, вытаскиваем.

Чтобы обработать пересечение, нужно обработать соседей

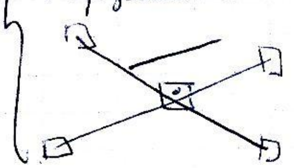
На каждой операции изменяется $O(1)$ отрезков, поэтому проверок $2n + k$ суммарно.

$O((n+k)\log n)$ $\quad k \le n^2$

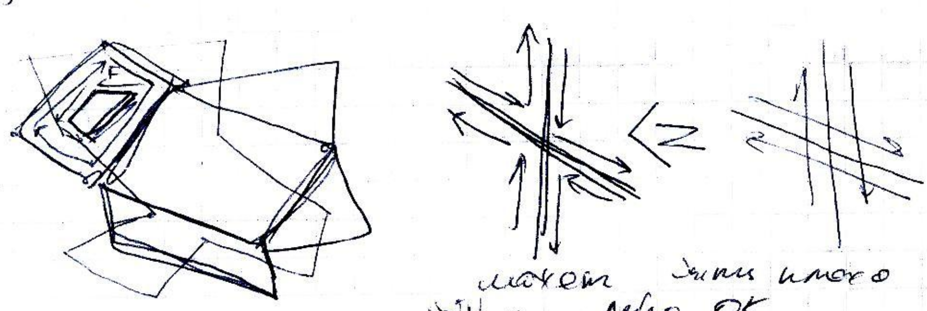
$O(n^2)$ памяти в очереди — дебаггинг: храним пересечение только для пары отрезков:



Хотим возвращать представителей в □. Если все точки переведем в представителей может поменяться топология.



2. Мы получили пер. ∀ отрезки. Хотим пересечение двух DCEL.

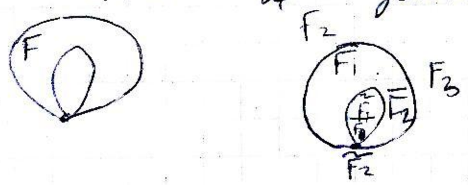


может очень много с контуров ребра OK, как потом починить части?

Просто пробежимся по ребрам, если граф связен. К внутренним изменения неочевидно.

Перед перестройкой графа храним старые фейсы, тогда сможем хранить старые версии.

Но этого не достаточно:



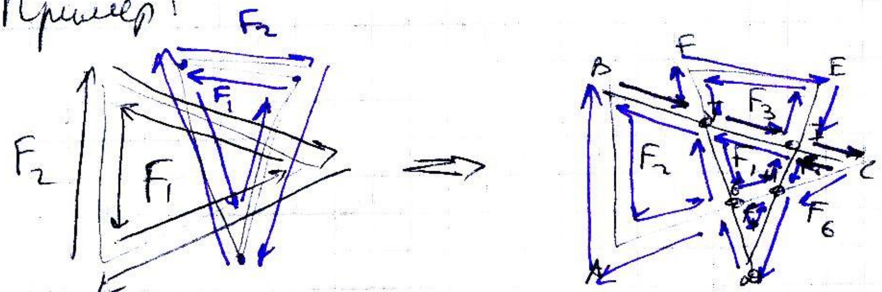
хотим $\bar{F_1} = (\tilde{F_1} \cap F_1)$

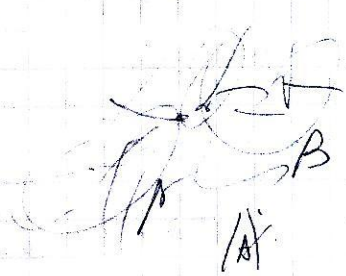
$\bar{F_2} = \tilde{F_2} \cap F_1$

$\bar{F_3} = F_2 \cap \tilde{F_2}$

А если не пересекаются.

Пример:





DCEL

28.05.2015

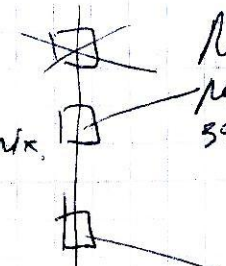
1) найти $\forall$ hotpixels (начало/конец отрезка и точки пересеч.)

2) Пересечение отрезков и hotpixels

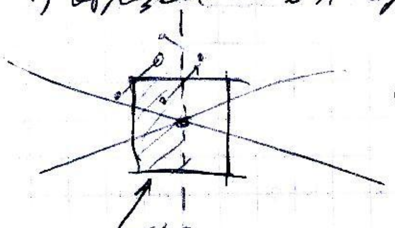
События

1) нач/конец отрезка
2) $\cap$ пары отрезков
3) Отрезок входит в hotpixel
4) Отрезок выходит из hotpixel.

} init hotpix.

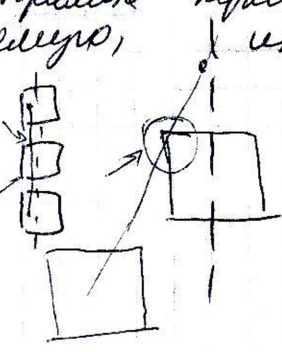


Порядок размера hotpix зависит от задачи



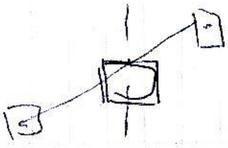
no events here before creation

а) Все прямые пересекающие замет. прямую, или б)



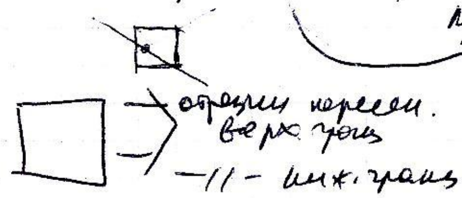
Сделаем когда отрезок не пересекает 3. прямую, тогда в верт. прямой есть hotpixel'и с началом

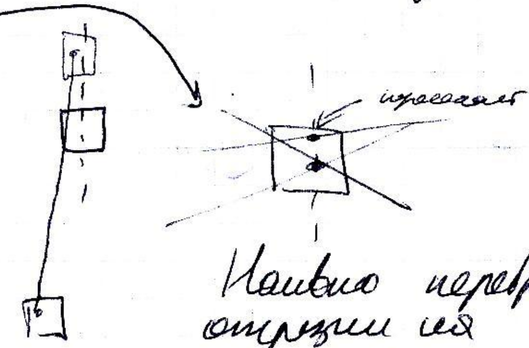
или



4) проверяем боковые грани

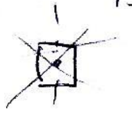
б) Создавали hotpixel до здесь.
прелест





a)

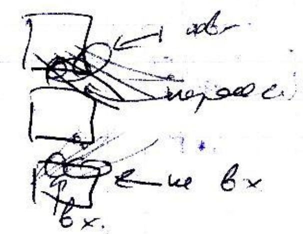
1) Отрезок пересекает З. П. внутри hotpixel, точно добавим



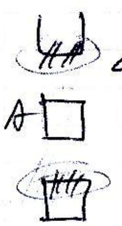
Наивно перебрать все отрезки на З.П-оси) ТИПА ДОЛГО(((((

2) Отрезок пересекает З.П вне hotpix

проверим все



исход.



← рассматриваем, они не пересек. между собой все потому не попадем не равно, не пересек A нижние аналогично.

Инициализация:

1) Добавление h.p.
2) Добавление отрезков, сингулирующих h.p.
3) Добавл. ∀ отрезки в статус что пересекают З.П. внутри h.p.
4) ∀ hp сверху и снизу, предрасч исход, зап. вход.

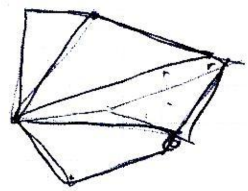
5) обновить кол-во вход и исход. отр.
6) В кол-во отрезков добавим в. и нижн. (добавим в ~~даже после З.П~~ очередь н/в. границы или отрезки)

4.06.2015

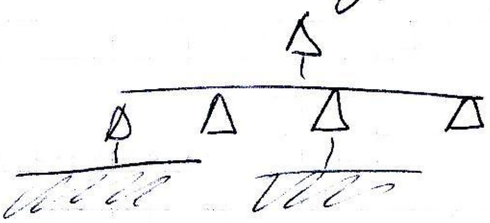
Как триангулировать мн-во точек?

Просто создаем ребра, чтобы не нарушить планарность по ходу дела.

Можно построить CH, построим ее триангуляцию наивно,



Построим дерево на гранях



Из и локализуем точки.

Spacial indeces

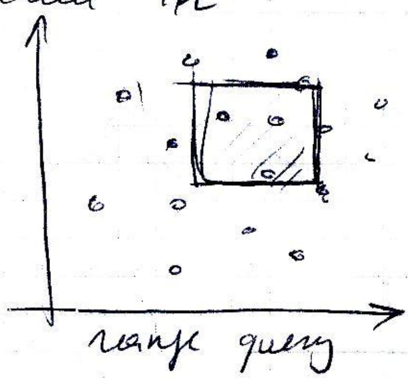
хотим вернуть точки в (a, b)

Можно lowerbound

$\log n + k$

Можно пользоваться skip-list'ом или деревом если нужно +− в дерево.

Если $\mathbb{R}^2$



range query

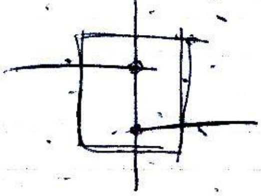
Можно дерево отрезков интервалов, если не хотим модиф.

Skip-quad tree не оч, потому что глубина большая.

Можно k-D tree (лучше, глубина константно меньше)

+ Можно поддержать в узле $\frac{maxdepth}{mindepth} < \alpha$ и перебалансировать

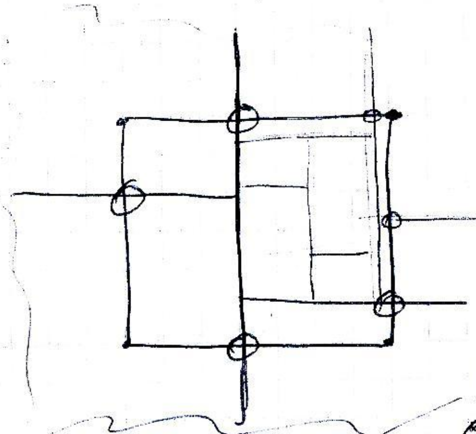
Как квадродерево отвечать



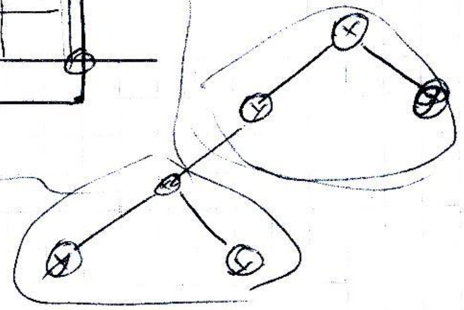
Для поддеревьев в range полностью вкладываем ответ, иначе идем в детей.

Такое, грубо — $O(k \log n)$

Хотим точнее



K+nodes - кол-во nodes которые мы посетим.



на каждом уровне берем 2

С 4 границами мы пересекаемся единожды (когда зашли, уже не все границы

$$K(N) = 2K\left(\frac{N}{4}\right) + C$$

K(X) - кол-во nodes пот. мы

подходит $\surd$ $K(N) = 3K\left(\frac{N}{4}\right) + C$

+O(n) памяти

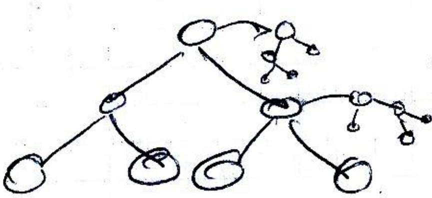
Master's theorem

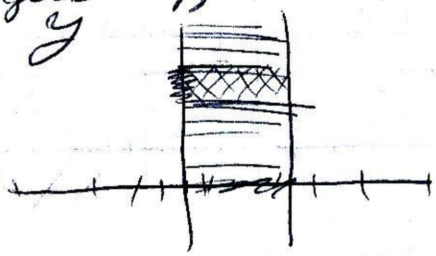
почему-то не достаточно

Кормен.

$K + \sqrt{n}$ на одну вершину.

Построим сбаланс. дерево по x, где каждого узла ещё дерево по y



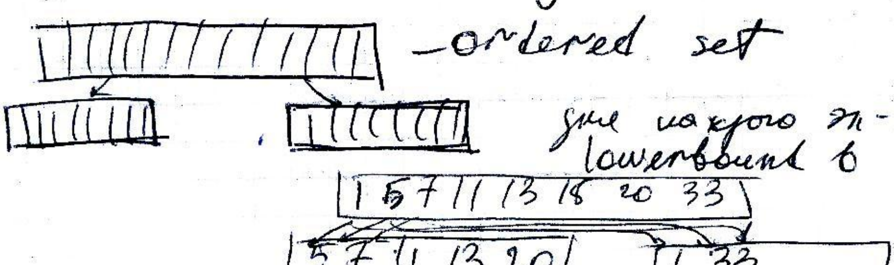


$O(\log^{d-1} n)$ время

$O(\log^{d} n)$ памяти

Можно ещё лучше:

fractional cascading

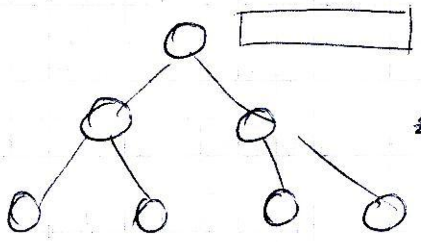


— ordered set

где каждого эл-та храним lowerbound в нижнем подмассиве

помощ.

Поиск диапазона:



← найти lowerbound

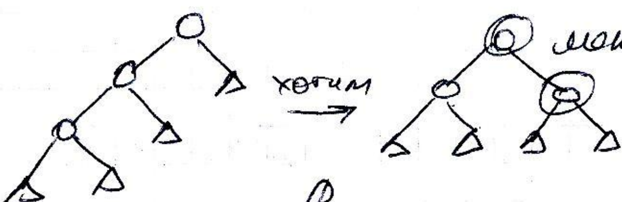
← за $O(1)$ поддерживаем

← —//—

Можно уменьшить время до $O(\log^{d-1} n)$. Если d-мерное, то только на последнем уровне

Как динамически строить структуру для range query без fractional cascading? ← только статич.

Если без балансировки за $\log^2 n$ добавление. А с балансировкой есть проблема:

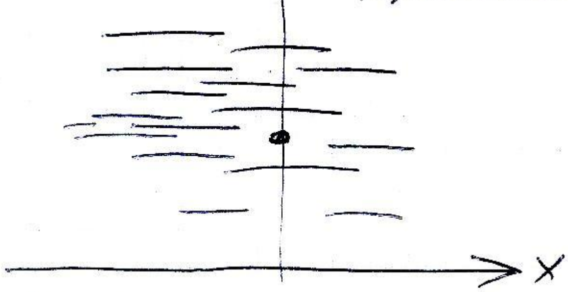


← меняется структура вершин. пересчитаются диапазоны по y (надо долго)

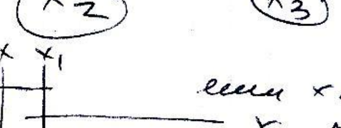
Решить можно: перебалансировать только в случае нарушения инварианта

Хотим по точке выдать все многоугольн., которые ее содержат.

Можно попроще задачу: хотим знать, какие отрезки пересекают луч:



Можно строить дерево отрезков. А мы построим дерево интервалов



лево ← x_1 ← все отрезки пересек. x_1

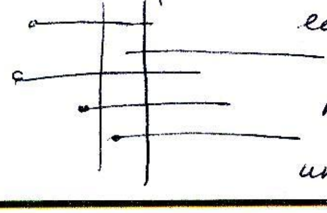
отрезки, пол. не пересек. x_1 и левее

—//— и право

⟨left [ ] incr sort.; right [ ] decr.⟩

С в каждой вершине.

$O(k + \log n)$ } $O(n)$ памяти

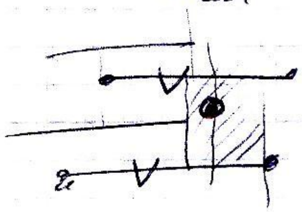


если x левее x_1, пойдем по left, иначе right →

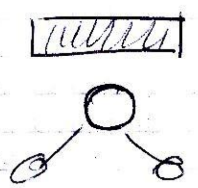
Построение: возьмем $\frac{1}{2n}(\sum start + \sum end)$ — медиана
потом выбрали слева, посередине, справа и т.д.

Динамически сложно — какая-то сложная балансировка.

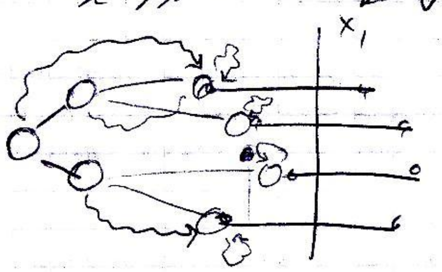
Offtop: применение дерева интервалов для локализации.
Ищем пару отрезков, ограничивающих луч.



Давайте в вершине хранить все отрезки отсортир. по y



+ храним отрезок с min x уже поддерева
тная глоб., не хорошо ли
Храним вершина левее
все дерево в каждой вершине (дерево отрезков)



Алгоритм Тришович, Ковалёв сам но очень плохо.

тут — сам. дерев. влево.

$O(\log n)$

Проверяем поворот и выходим если не левее самого левого

Пытаемся понизить верхнюю границу и повысить нижнюю, держим 2 указ.

1 сентября 2015

d - dim

P - мн-во прям.

Q(q) - прямоуг. запрос

хотим $\{p \in P,\ p \cap q \neq \emptyset\}$

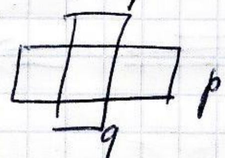
$S =$

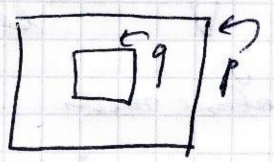
$S = A \cup B \cup C$, где $A = \{p \in P,\ \exists i \in \text{corners}(p)\ p_i \in q\}$

$B = \{p \in P : \exists i : s_i \in \text{sides}(p),\ s_i \cap q\}$

$C = \{p \in P : \exists i,\ q_i \in \text{corners}(p),\ q_i \in p\}$

A:


B:


C:


A и B было на 1 паре, я встречал папу в аэропорту. C сейчас, это нихера не понятно :(

Интервальное дерево:

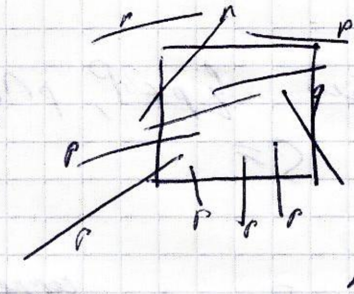
паралл. отрезки, верт. прямая, какие пересекает?

1. Берем среднее от всех концов отрезков — медиану
2. Все что пересек. верт. прямая медианная (проверка за $O(n)$), пишем в корень дерева
3. Повторяем для детей

B решается интервальными деревьями.

1. Строим два дерева интервалов для верт. и горизонтальных отрезков по $p \in P$.
2. Для сторон q проверяем (4 проверки, по 2 на дерево)

Задача: мн-во отрезков, прямоугольник,

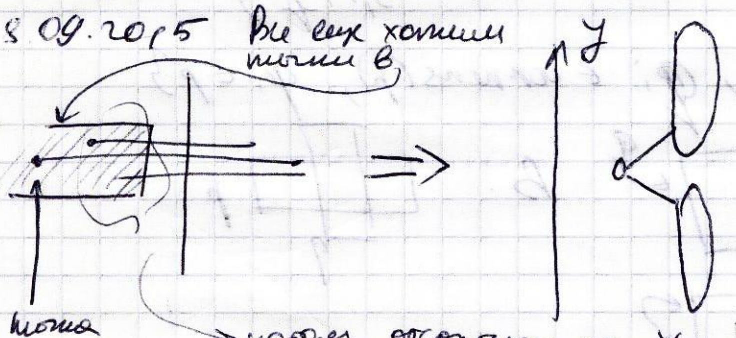


отрезки непересекающиеся

Можно построить на p ∈ P трапецоидную карту, локализовать угловые точки q, потом посмотреть по карте, пока не обойдем q по периметру

Можно построить верт. линии через концы, потом построить персист. дерево.

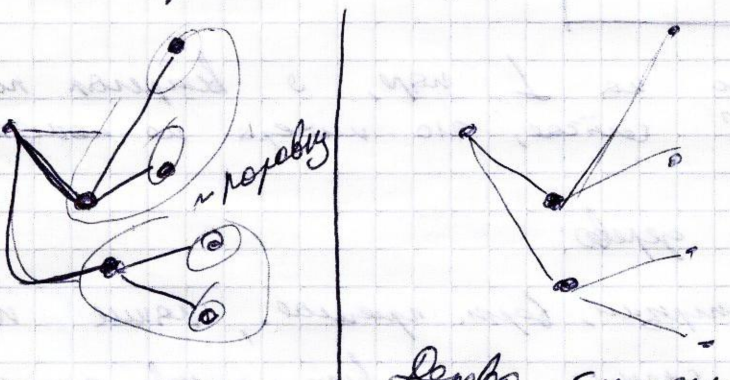
8.09.2015 Все еще хотим точки в



точка с min x

потом отсортим по y, разрежем на 2 равных части

Строим heap. (не бин. дерево поиска) соединяем ребенка с самой левой верш. поддерева.



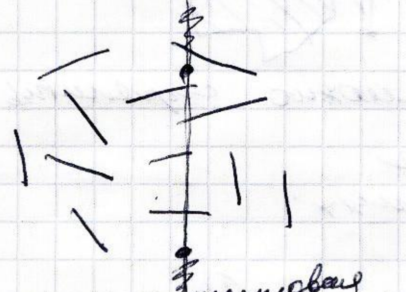
Идем вниз пока не окажемся правее вертикальной границ.

Дерево строим заранее

```
Node:
    Node *left,
    Node *right;

    range_r y;
    point_r p;

Node *root;

trav (x, range_r y, Node * root)

    if (root -> p.x > x) return
    if (y.contains(root -> p.y)) yield root
    if (root -> left.y.intersects(y)) trav(x, y, root -> left)
    if (root -> right.y.intersects(y)) trav(x, y, root -> right)
```

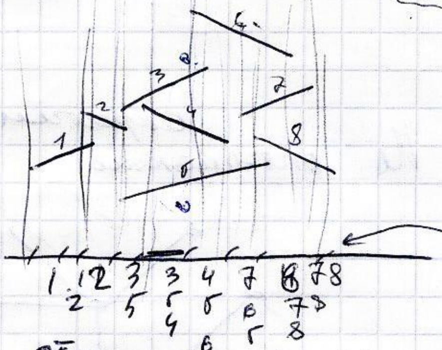
Задача: ор. отрезки, кот. пересекаются ~~прямой~~ отрезком
+ хотим инкрементальность!

← без этого решено в IV семе

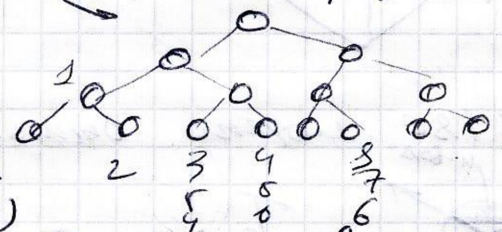


тестовая прямая

Метод полос:



Сделаем из этого дерево отрезков на эл-тах



и поднимем выше наверх, избавляясь от $O(n^2)$ памяти.

В каждом узле отрезки отсортир. по y

Время — $\log^2 n + k$ $n\log n$ памяти.

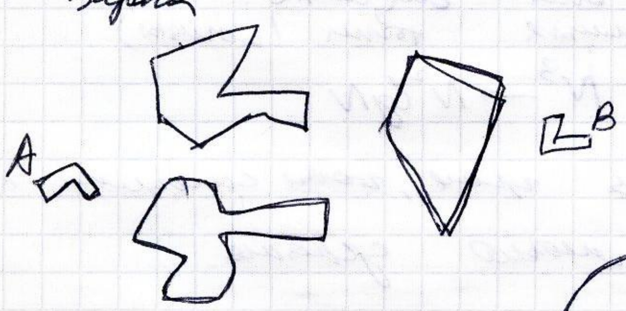
Из этого можно сделать динамическую структуру.

Алгоритм: локализация — $\log \cdot \log$ ← бинпоиск в вершине.

C по дереву вниз

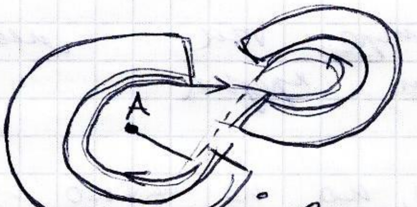
Добавим моторику.

Задача



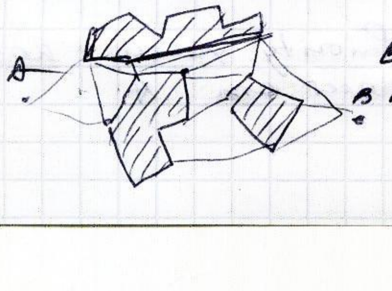
Перевести (не)точечный объект через полигональные препятствия ← можно, конеч.

• Известность карты — ?



Обходить по часов. стрелке ног.

Обходим все вершины, не пересекающие.

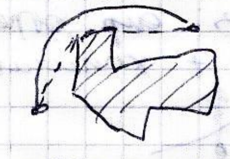


← Можно достроить этот граф видимости.

n^2 памяти, n^2 времени.

1. Путь - ломаная. Если нет и оптимально, то

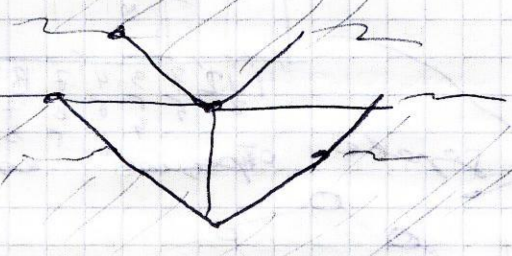
Между невыпуклых объектов



его можно спрямить

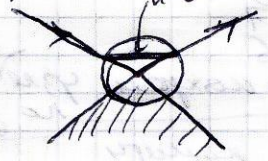
Сократим кол-во ребер

2. Не оптимально в касательных:



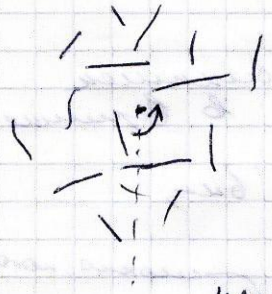
Такое не может быть оптимальным.

Угол < 180 может быть спрямлён н-м △



То есть такие ребра можно выкинуть.

3. Еще оптимизация



Находим за $O(n)$ отрезки, которые пересекают луч вниз.

Используем (нечто похожее на) Бентли-Оттмана, по повороту (луча) заметаем.

Проходим лучом и собираем только первые отрезки — они видимые.

Так определим видимые точки (концы)

$$N^3 \to N^2 \lg N$$

4. Динамически строить граф, тогда сэкономим память

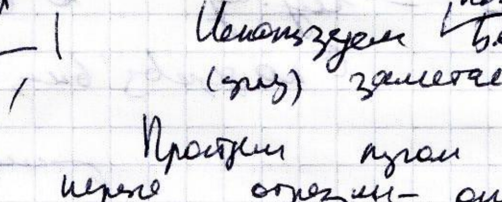
В дейкстре так можно сделать.

Сохраняем время

5*. Можно, но не нужно - сложнее [$n \log n$ памяти, ~~[illegible]~~ запрос + локализ. в план. графе

см. Митчелл

Мысль —



Priority queue по расст. от A

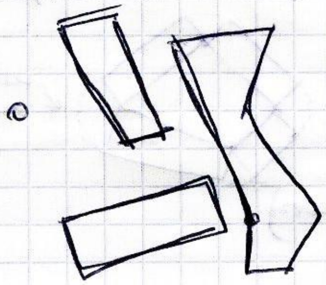
15.09.2015

Я опоздал на лекцию на 1 час :С

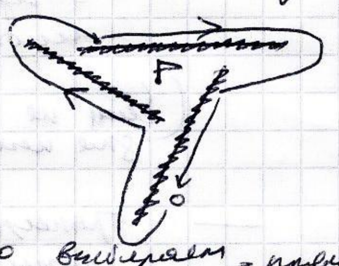
Что-то про шейдеры, однородные координаты.

Приближенное решение задачи о прохождении [illegible] объектов.

Задача:



Можно держаться рукой, BUG0 но это [illegible]:

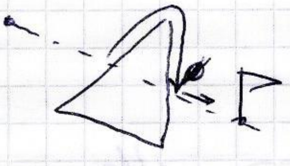


стороны случайно выбираем ← правую пусть

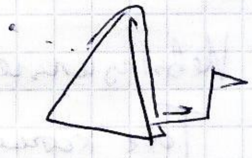
BUG1: обходим препятствие, пока не пересечем

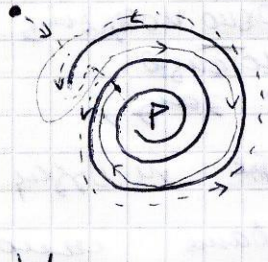
BUG0: обходим преп. пока не ближе всего к F

BUG0



BUG1

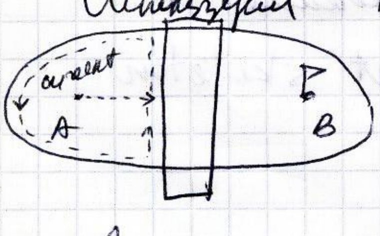




— BUG1 значительно лучше BUG0

Есть алгоритм CBUG:

Используем BUG0/BUG1 с ограничением:



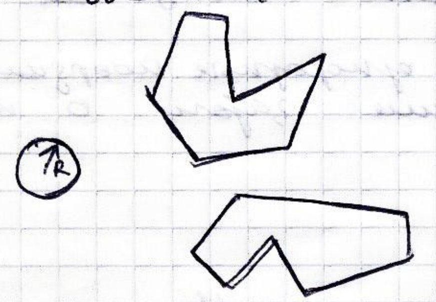
Строим эллипс на AB, упираемся в него. Если что — увеличиваем эллипс в 2 раза. Эллипс — еще одно препятствие

Со зрением — то же самое, потому что мы зрением "осматриваем" объекты так же, как бы мы их обходили

Если объект не точечный, то без зрения — почти так же, со зрением сложно.

Провести объект (по точкам) через поле

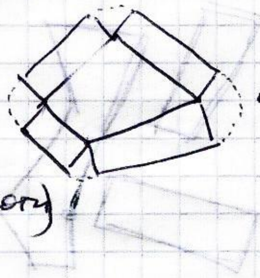
1. Круг:



а) Полигоны выпуклые

расширим полигон

решение примерное



потеря обхождения многоугольнику

конечное кол-во точек.

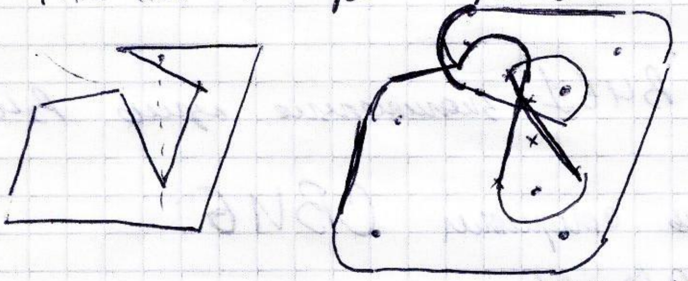
(Если не граф видимости) тогда его не нужно

— трапецоидной картой — искать путь по середине трапецоидов

— триангуляция — по центрам △

б) Невыпуклые

Тоже хотим расширить

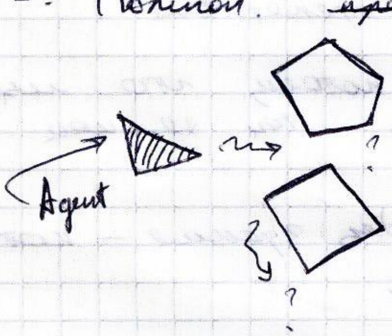


Можно построить нечестную буф. зону, просто не будем учитывать само-пересечения

честная буферная зона — сложно

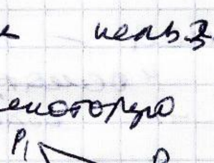
straight skeleton

2. Полигон. объект ~~произвольный~~

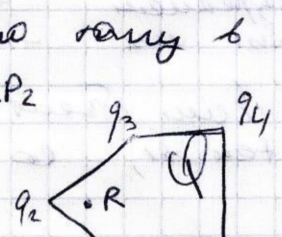


а) Пусть нельзя поворачивать

Выберем некоторую точку в полигоне



$p + p_1$
p_2
p_3



Объекты пересекаются,

если $\exists R \in \{P_i\}$, $R \in \{q_i\}$
в полигоне

for t in Agent
  for t' in Q
    if (p+t) == t' intersect $\to$ p = t' − t
    ret

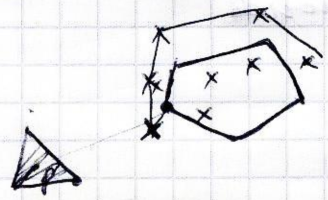
~~tta~~

outside $\uparrow$ p лежит в невып. многоугол.

$Q' = \{q - p : q \in Q \quad p \in Agent\}$

Сумма Минковского P и Q:

$$P \oplus Q = \{p+q,\ p \in P,\ q \in Q\}$$



22.09.2016

Проверим, что $A, B \in CH \Rightarrow A \oplus B \in CH$.

$\triangle$ По опр.

$c_1 = \underset{\in A}{a_1} + \underset{\in B}{b_1}$

$c_2 = a_2 + b_2$

$c = t c_1 + (1-t) c_2 =$

$= t c_1 + c_2 - t c_2 =$

$= t a_1 + (1-t) a_2 +$

$+ t b_1 + (1-t) b_2$

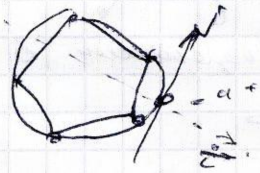
∇

Правда, что если A, B — полигоны, то $A \oplus B$ тоже

$CH(\{a_i + b_j\}) \subseteq A \oplus B$ — очевидно

докажем, что они совпадают, то есть других точек нет

разложим по $\vec{i}, \vec{j}$



$a + b$ — строго за полигоном, по макс. проекции

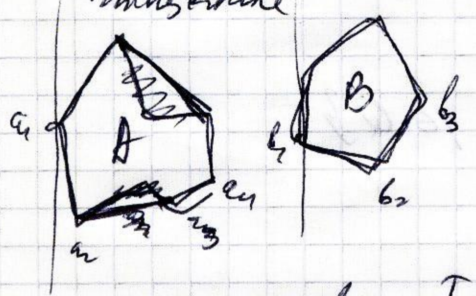
$\underbrace{(a^i + b^i)\vec{i}}_{\text{важно}} + (a^j + b^j)\vec{j}$

Найдем макс. проекцию на i среди A и B

Ещё раз: тела нашли мин. точку в $A \oplus B$, где і минимальна, но тоже во всех в жив.
Обнаружим, что такое $c = a + b$, но a и b — точки на границе.

Построение $\oplus$ за n^2 очевидно. Быстрее?
$\log n$ ← выпукл. оболочка

Выпуклые



Возьмём самую левую точку у полигонов

$a_1 + b_1$ — I точка

II — ? $a_1 + b_2$, $a_2 + b_1$, $a_2 + b_2$?

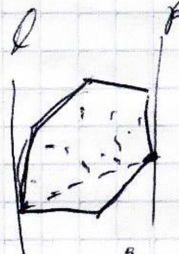
$\measuredangle$ $\overrightarrow{b_2 - b_1}$, $\overrightarrow{a_2 - a_1}$, приложим их к a_1

a_1, $a_1 + (b_2 - b_1)$, a_2 ← сравним повороты

$\Downarrow\Downarrow$ ← катаем макс. "левое" направление.

Получим $O(m+n)$ точек вместо $O(mn)$

Rotating calipers



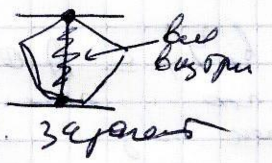
Найти диаметр множества?

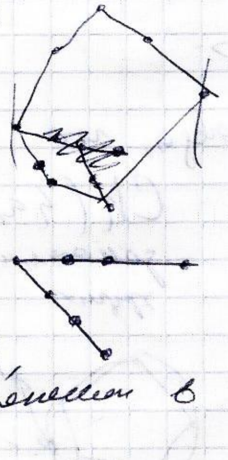
Диаметр — на выпукл. оболочке

$\Leftarrow$

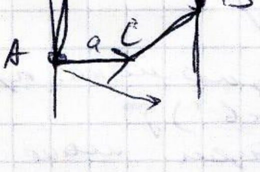
Выберем левую и пр. точку

Мы должны перебрать все такие пары, что





Аналогично с пред. задачей



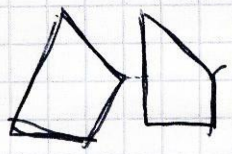
B инвертируем вектор b, перенесём в A

Затем сравним

$A, C, A + \overrightarrow{(a-b)}$

Можно смотреть min — ширина мн-ва

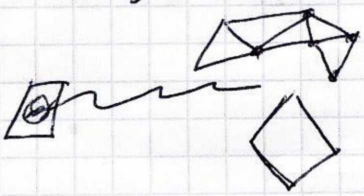
Мин. расстояние между двумя непересек. полигонами.



перебираем пары точка – отрезок

Д/З)))0)0

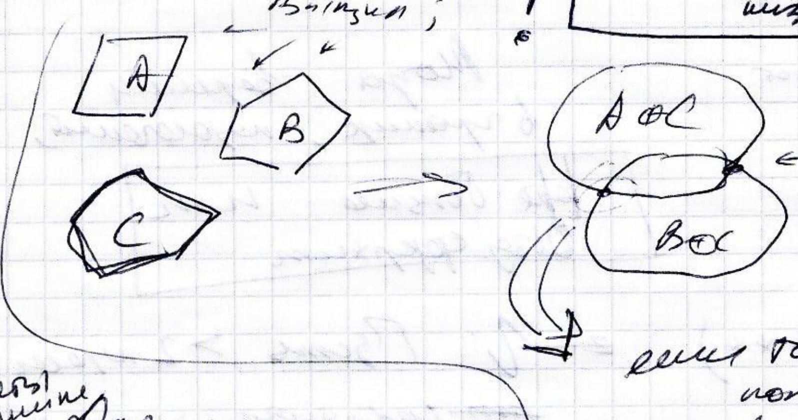
Невыпуклые как, ну з



Можно невыпуклые полигоны триангулировать, потом сделать одну минковского по треугольникам, потом слияние.

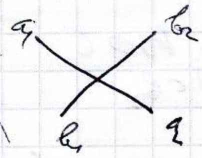
верхняя оценка – $(mn)^2$ – при обыч. выпукл.

! Можем более и невыпукл. низкую оценку.



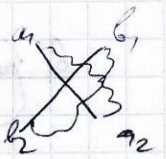
Д-во позже (***)

если только 2 пересеч. полигонов, то



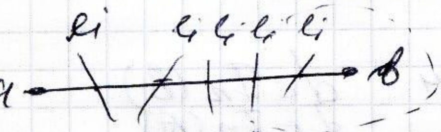
тогда $a_i, b_i \in A$, $a_j, b_j \in B$

иначе точек пересечения 4



2 монеты в вершине

Для многих отрезков 1 мн-в



С отрезки дел на 2 вида

мы платим за каждую забытую точку, которая является точкой полигона, содерж. в.

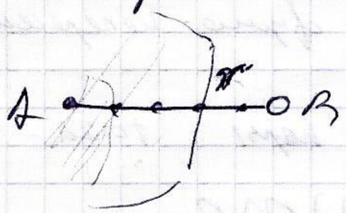
Крайние точки платят за соседние

за эти платят сами отрезки

Мы не можем распилить

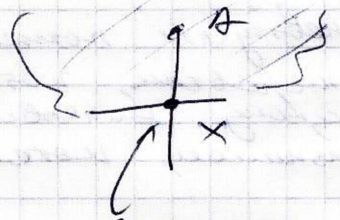
Лёша

∢ ребро расширенного Δ.

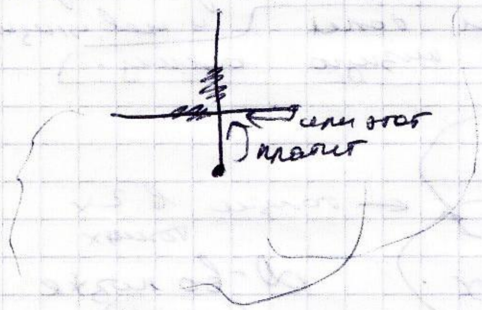


↑ самое дальнее от A пересечение AB с ребром, которое

Дадим 1 монету с A на Г. Вторую оставим.



граничная точка нового объединения расшир. Δ



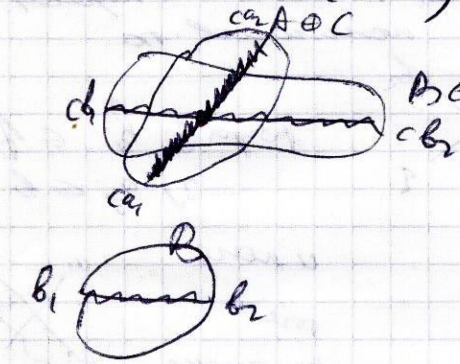
Тогда вершины в границе, пересечений,

$\boxed{\text{не больше } 4m. \text{ Объед. содержит}}$

Д-во (***) ⇐ Пусть > 2 пересеч.

содержится целиком в $A \oplus C$

— " — в $B \oplus C$

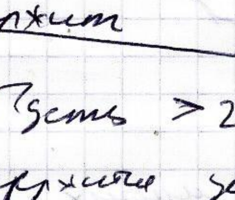


Хотим найти $d \in Int(B)$, $d \in Int(A)$

29.09.2015

* Тут было пересечение выпукл. и невыпукл. $\geq O(mn)$

Невыпукл + невыпукл.

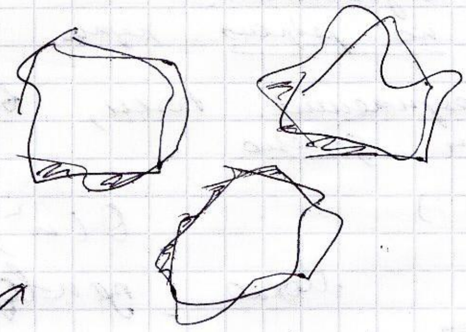
m^2n^2 — минимум. Пример —  m^2n^2

Вернемся к первонач. задаче.

Можно сделать неточное решение (точное — $O(n^4)$ по памяти)

$\exists$ поворот на малый угол — ок

Построим много карт для разных углов

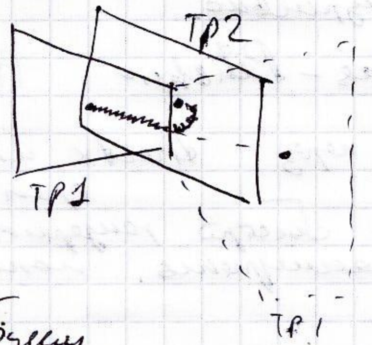


В узких местах можно и двигать $\parallel$ и поворачивать.

В заграждённых только $\parallel$ перенос

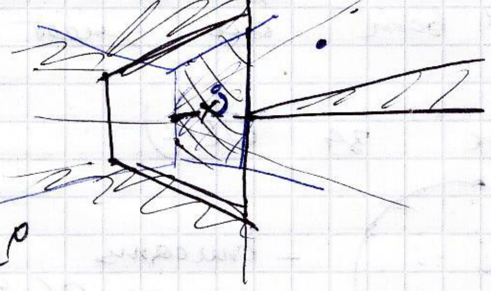
две карты для n° и $(n+\varepsilon)^\circ$

Можно попробовать построить трапец. карту

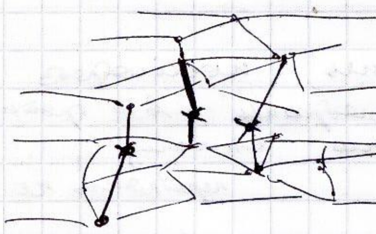


— добавл. вершины, соединяющие слои.

Лучше добавлять также в центр пересечения.



Будем линковать



n°

$(n+\varepsilon)^\circ$

$\ast$ — центры пересеч. трапезоидов

памяти $(n+m)$ если все выпуклое и (n^2m^2) если все невыпуклые.

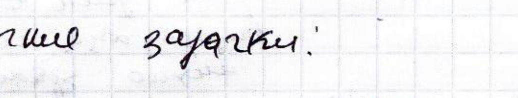
Итого памяти $\ast$ слой $\ast$ кол-во дискретных угла

n^2 углов примерно

Path planning
Motion planning
Piano moving

Легкие задачки:

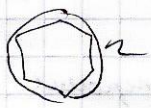
Рисуем карту, дороги. Проблема — 10^9 точек в дорогах. Есть полилиния и масштаб.



хотим из — сделать - - -

— сгладить

Можно расширить прямую Минковским многоугольником — правильным.

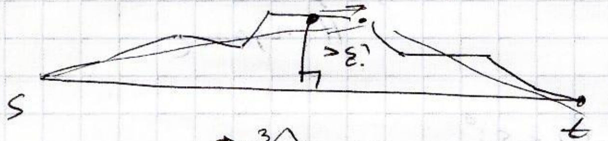


Потом с помощью сделаем какой-нибудь граф видимости.
Построим кратчайший по рёбрам путь.

Вариант — построить соединение точек, выкинуть рёбра, которые слишком далеко.

Я это придумал :)

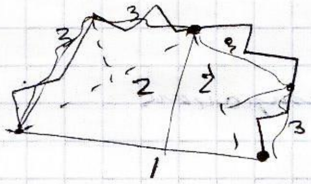
$O(n^2)$



Можно за $n\log$: (в среднем)

st — подходит?

Смотрим макс. расстояние, строим прямую если > ε. И так же рекурсивно.



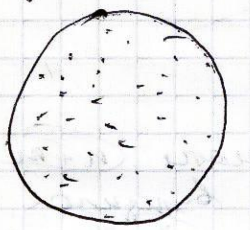
Это алгоритм Douglas–Peucker Пекер

С видеокартами общаются через directx или openGL.

С помощью openGL можно быстро рендерить.
И вот оно умеет расширять полилинии.

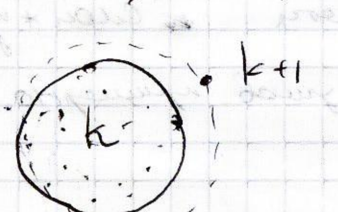
~

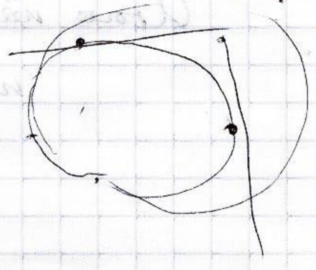
Радиус за $O(n)$



— описать окружность, посмотрев
за $O(n^4)$ перебрать все тройки
за $O(n^3)$ построить оболочку и перебирать по 3

Утверждение: если есть окр. для k и эту k+1, тогда (·) k+1 на границе.





Δ

Если удалим из этой выборки, диск не изменится

Если диск ∈ всей выборки, тогда её можно удалить

▷

min disc(P)

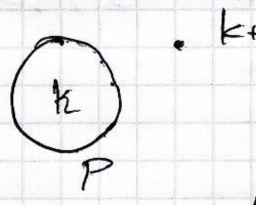
min disc2(x, P); ← точка, не ∈ P, лежит на границе.

min disc3(x, y, P)

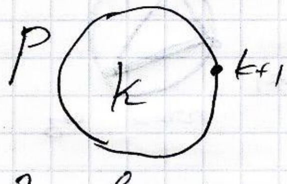
Вызываем mindisc

В первом приближ. $O(n^3)$

Но на деле ... $O(n)$! Куб — худшее невозможно.
Пошаффлим за $O(n)$



С какой вер-ю $k+1 \notin P$?
Непонятно.



С какой вер-ю $k+1$ на границе?
не меньше $3/k$

В mindisk2 вер-ь вызвать mindisk3 — $2/k$

Тогда mindisk2 — за линию. И mindisk — тоже.

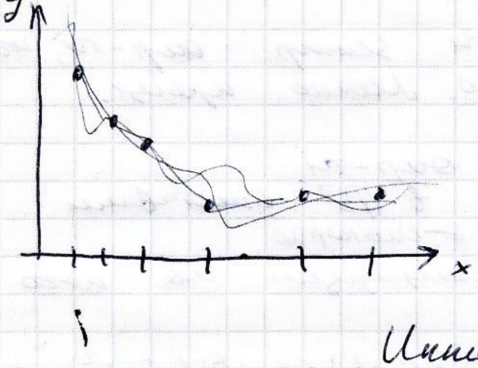
Реализацию можно в пару строк? А то!

$\boxed{\text{Как?}}$ в 1 цикл из 1 проверки.

6.10.2015 Триангуляция

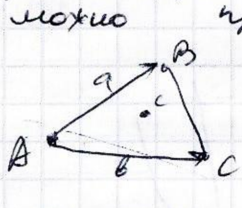
Можем разбить СН на Δ, будет работать.
Можно интерполировать:

∃ есть поле; ф-я $y = f(x)$. Пусть мы знаем значение ф-ии в конечн. мн-ве точек.
Что мы можем сказать о хар-ре точек?



Это ML: выбираем ф-ю с параметрами, пытаемся минимизировать — это наз. аппроксимацию.

Интерполяция — наоборот: пусть точки заданы очень точно, давайте получим только при старших членах ряда Фурье или Тейлора.
А можно приблизить отрезками



$c = \alpha a + \beta b$. $A = A + \alpha(B-A) + \beta(C-A)$

$f(c) = \alpha(V_B - V_A) + \beta(V_C - V_A)$

билинейная интерполяция

Билин. гарантирует непрерывность, но не гладкость.

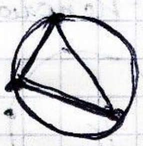
Оптимальная триангуляция — Делоне, если еще триангул. min периметра.

Критерий оптимальности — не зная ничего о хар-ре ф-ии избежать больших перепадов по наклонам.

Чтобы, $\sum\limits_{\Delta} \operatorname{grad} f$ была бы ближе к 0.

Как раз пригод. Делоне минимизирует эту величину.

$\forall\ \Delta, \in$ триангул.



В описанной окр-ти (внутри) не должно быть точек из набора на границе — ОК.

Лемма: эквив.:
около люб. Δ можно опис. окр., не содерж. точек $\Longleftrightarrow$ около ребра можно —//—

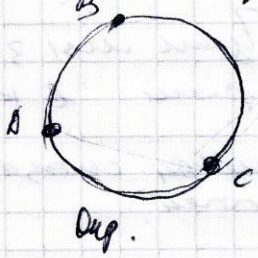
Δ

$\Rightarrow$ очев.

$\Leftarrow$ Д/З !!!

∇

Решим задачу:



$C \in$ окр? на границе?

Можно найти центр окр-ти, но это сложно. Можно проще.

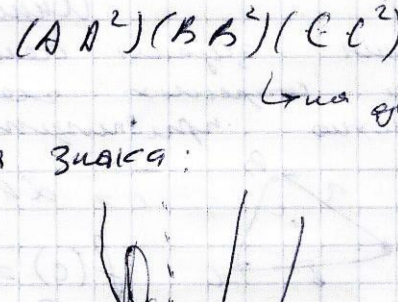
$\exists$ 4 точки на окр-ти.
каждой точке в $\mathbb{R}^2$ сопоставим (p, p^2) $\leftarrow$ скаляр
Получим параболоид. Спроецируем на него точки

Если $ABCD \in \overset{\text{гран.}}{\text{окр}} \Rightarrow (A\ A^2)(B\ B^2)(C\ C^2)(D\ D^2)$ $\leftarrow$ на одн. пл-ти

если D ниже, то

Тогда решение — проверка знака:

$$\begin{vmatrix} A_x & A_y & A^2 & 1 \\ B_x & B_y & B^2 & 1 \\ C_x & C_y & C^2 & 1 \\ D_x & D_y & D^2 & 1 \end{vmatrix} \gtrless 0$$



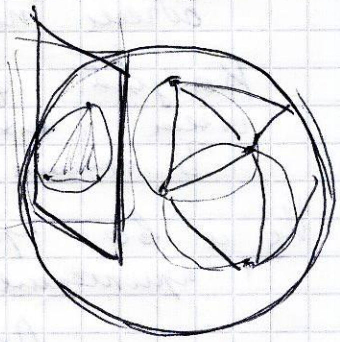
Тогда параболоид — СН (нижн. часть). Можно добавить фиктивную точку 0, 0, 1, 0 в однородных координатах.

Тогда СН сфер ⇒ триангул. Делоне сущ.

~

Пусть есть точки на сфере. Хотим триангуляцию на сфере, чтобы она была Делоне

Ответ — СН точек на сфере

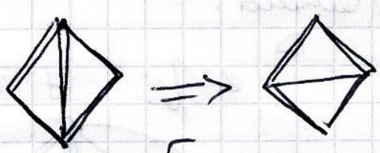


У окр. пересеч. точек, если все остальные по другую сторону секущей пл-ти.

Тогда очевидно СН работает.

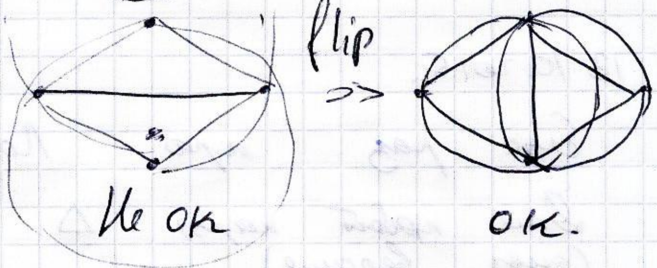
~

Пусть flip — операция



Можно ли из абстрактной триангуляции сделать Делоне за конечное число flipов

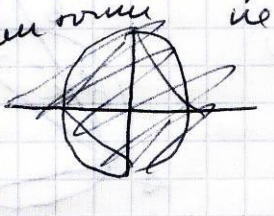
Для 4х точек: все работает.



Формально перенесение в 2 опр. Делоне через ребра, где ребра — диагонали (локальная версия)

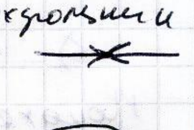
Только 1 диагональ — ок, если точки не лежат на окр.

$\Leftrightarrow$: обе диагон. окей



$\alpha+\beta<180$
$\gamma+\delta<180$, тогда четырехугольник плох.

$\Leftrightarrow$: обе диаг. плохи

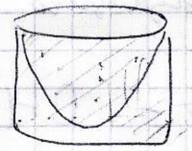


Аналогично

Отсюда давайте флипать только плохие ребра.

Посмотрим объем фигуры, поднятой на параболоид, нижнюю часть.

тогда флип изменяет объем подграфика V -



Каждый флип уменьшает объем нижн. части (строго)

V огр.
Кол-во триангл < ∞

∃ триангл. делоне мин. объем.

Из и все, прямо следует относительно критерия „хороших ребер"

Локальная оптимальность.

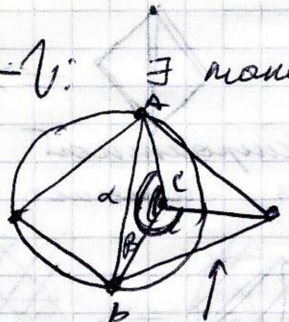
$\Rightarrow$

глобальн. оптим.

Лемма: ↗

Δ

$\Leftarrow$ ¬: ∃ лока и окр, но всё плохо:



возьмем такое AB, что $\angle ACB$ − max.

покажу $\beta > \alpha$ − больше на дуг.

соседний Δ

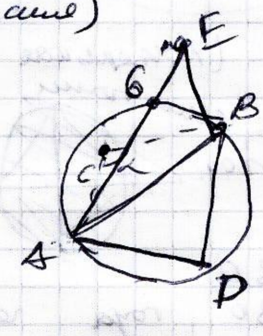
13.10.2016.

Еще раз про лок. опт. $\Rightarrow$ глоб. опт.

Для любых пары Δ усл. Делоне верно (диаг. верное)

Δ

$\Leftarrow$ ¬:



C-ка смежна с Δ

Тогда найдем среди всех C такое, что α − макс.

Δ по стороне AB смежн. Δ − AEB.

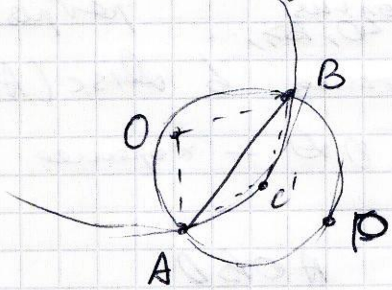
E не в окр. по локальн. Делоне

Покажем, что $\angle ACE > ACB$ (добавл. факт, что дуга GE непустая)

Также $c \in \mathcal{D}(ABD)$, $\angle ACB < \angle ACE$
диск

Покажем $c \in \mathcal{D}(AEB)$:

Хорда AB общая для $\mathcal{D}(ABD)$ и $\mathcal{D}(AFB)$



$O \in Disc(ABD)$, потому что $\angle AOB + \angle ADB > 180$

Аналогично

$\angle AOB + \angle AC'B > 180$,

потому что $\angle AC'B > \angle ADB$
след. из опир. на хорды.

Отсюда $\angle ACB$ — не max.
($\angle ACE$ больше). —×←

▽

* *

Триангул. Делоне — мин. остовное дерево в евкл. простр-ве. (норм-во)

(вес ребра — $\sqrt{\Sigma(\,)^2}$)

Δ Пусть ≠:

ребро в ост. дереве, но не из Делоне

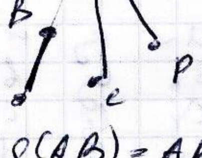
Тогда по опр. Делоне
$\forall$ окр на AB есть точки в ней.

AC и CB не лежат в MST.

Выкинем AB, тогда C в одной компон. св-сти с A или с B. Тогда восстановим нужное ребро (AC или CB), тогда вес MST — не миним.)

▽

Парижская метрика



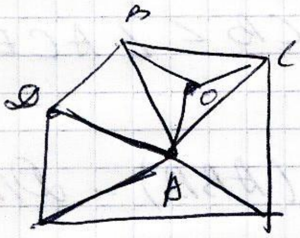
$\rho(A,B) = AB$

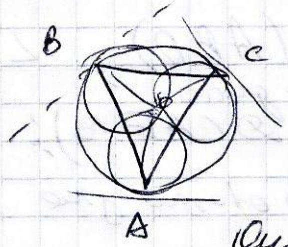
$\rho(C,D) = CO + CP$

Флипы дают возможность строить Делоне инкрем.

Возьмем Делоне, добавим точку

Утв: мы не флипаем уже готовые ребра — покажем, что инцидентны ← AO BO CO



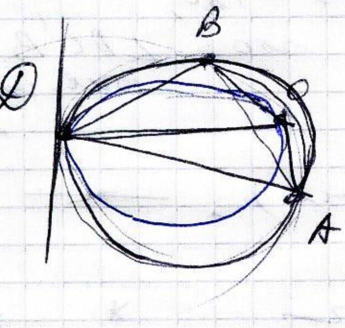


Построим 3 окр-ти, доказывающих, что
OB OC OA — норм. рёбра.
Окр-ти должны касаться O, B, ...

O — единств. точка в $Disc(ABC)$

Тогда флипнем AB, ребро OD — хорошее

Покажем, что нужно флипнуть AOBD
И никогда больше не надо DO



DBA содержит O
м.б.

Будем флипать только
противолеж. O.

Сдуваем окружности $Disc(DBA)$
до $Disc(DO)$, которые всегда <

Итого: флипаем только рёбра, противолежащие
O в неделон. треугольнике.

Храним исход. из вершины рёбра.

Берём next, проверяем
не нужно flip => след. ребро
нужно flip => остаёмся, т.к.
добавилось ещё 1 ребро

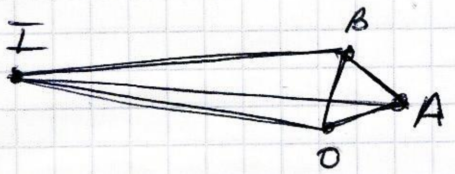
$$\begin{vmatrix} a_x & a_y & a^2 & 1 \\ b_x & b_y & b^2 & 1 \\ c_x & c_y & c^2 & 1 \\ d_x & d_y & d^2 & 1 \end{vmatrix} < 0$$ — тогда нужно flip'ать,
где: ...a, b, c ∈ Δ, d — провер. ∈ Δ

На параболу спроецировать вып. оболочку
+ I — беск. удалённая точка (0, 0, 1, 0)
эта вершина соед. со всеми
=> Бесконечные Δ выпукл. face'ами

Находим треугольник для точки — смотрим
Δ с двумя конечными и 1 бесконеч.
точками:

$$\begin{vmatrix} a_x & a_y & a^2 & 1 \\ b_x & b_y & b^2 & 1 \\ 0 & 0 & 1 & 0 \\ d_x & d_y & d^2 & 1 \end{vmatrix} = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ d_x & d_y & 1 \end{vmatrix}$$

Нужно для
локализации
вне CH

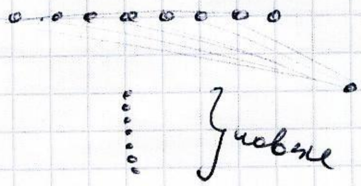


Надо ли флипать IBD?

Асимптотика

локализация точки + число флипов

Пример, когда все плохо.



А если пошаффлить?

Пусть мы добавили i точек и добавим $i+1$-ю.

~~$i+1$ 'ą т~~

Новая точка мож. быть последней

⟹ средняя степень вершины – 3

⟹ $O(3)$

Локализация – Киркпатрик (д-во в Де Берге)
Deviller

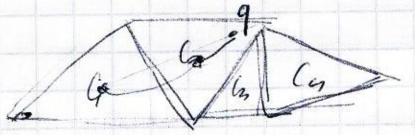
В нее все упирается т.к. с кол-вом флипов ничего не сделаешь.

Скип-лист в $\mathbb{R}^2$ (смотри обычный skip-list)

Пусть есть мн-во точек, построим $\log n$ Бернулл. мн-в.

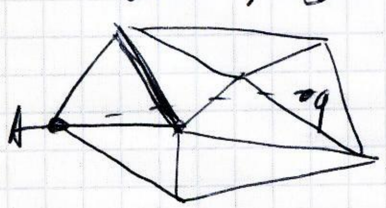
Для подмножеств – триангуляцию Делоне

По всем вершинам верхнего уровня идем пока можем (отрезок $\ell_i q$ пересекает ребро), потом спуск. вниз.



20.10.2015

Еще раз локализацию в триангуляции



Выберем вершину A, поймем в какой Δ ведет мысленная прямая Aq.

За кол-во ребер можно локализоваться.

Алг. Deviller'a

Этот алгоритм лучше, потому что он более устойчив к отсутствию шаффла. В $O(n^2)$ флипов сложно упереться, а локализация может быть $O(n)$ глубины локализ. структуры

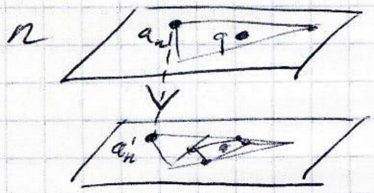
Итак, Devillers

Имеем триангуляцию (стек таких)

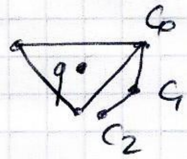
Как локализовать q? Хотим найти ее в DT_{redn}. Потом поднимать, и бросать монеты, если хотим пробрасывать треугольники вверх. (Бернулли). Локализация вне CH с помощью (0,0,1,0)

← На 1м уровне перебрать за $O(n)$

Найдем ближ. вершину до q в уровне n



Пот то есть Δ, содержащий точку q.



Найдем c_0 — ближ. до q точка среди вершин Δ.

А дальше по соседним так, что

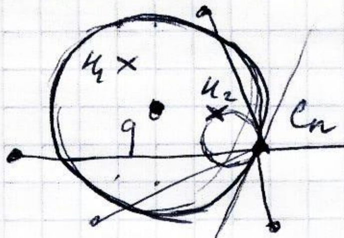
$$\rho(c_{i+1}, q) < \rho(c_i, q).$$

C_n — искомая точка.

Тогда C_n — глобально оптимальна, то есть $\forall x \in D_i \;\; \rho(q, x) > \rho(q, C_n)$.

Δ (такой алгоритм обеспечивает глоб. оптимальность)

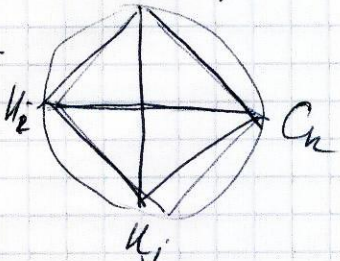
← 1: пусть мы в C_n сейчас, из нее нельзя улучшить расстояние, но не глобально — не min



Проведем касательную к окр. в т. C_n

Найдем мин. окр-ть, опирающуюся на C_n и H_i, где все $H_i \in$ Окр, но не связаны с C_n.

Окружность O' (вот эта новая) не содержит внутри ни одну точку, тогда $C_n H_i$ — ребро триангуляции.

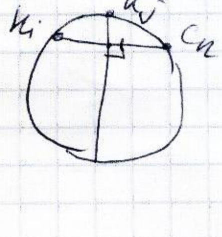


* { Почему его нет? H_j, вот такое — тогда если на O' две точки, и обе диагонали одинаково хороши. но ребра $C_n H_j$ тоже нет

*: Антон сказал что это ~~бред~~ D/3

Просто ребро Делоне и его нет.

Непорядок! Триангуляция недостроена.



→←

∇

Докажем, что переход $i \to i+1$ (локализация + жадный алгоритм) за $O(1)$. Если с шаффлом.

Формально:
1. $deg(q_i) = O(1)$ · Степень q_i на ур-не $i+1$
2. $|\{e_n : e_n \cap q_i q\}| = O(1)$ · Кол-во пересеч. ребер
3. $|\{p_n : p_n \in B(q, |qq_i|)\}| = O(1)$

Δ

1. Вопрос в том, что q_i — не случайно выбранная.
Давайте рассматривать добавление новой точки — регрессионный анализ.

Рассм. P_i, P_{i+1} — мн-ва точек на i и $i+1$ ур-х

Рассм. $P_i \cup q$, построим граф: $V = \{x,\ x \in P_i \cup q\}$
$E = \{qb,\ closest(q) = b$, $closest(b) = q\}$

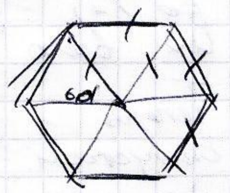
Это nearest neighbour graph.

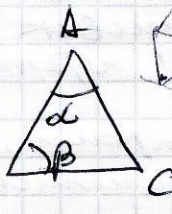
↑ если несколько, то выберем худшую

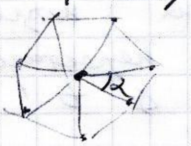
Докажем, что степень в таком графе ≤ 6.

Δ

от противного: $\exists$ есть > 6 вершин. Пусть есть 7 вершин







Тогда есть такой $\angle\alpha$, что $\angle\alpha < 60°$ (по принц. Дирихле), тогда в этом Δ есть $\angle\beta$, что $\angle\beta > 60°$, тогда по теореме синусов, тогда $BC < AC$, отсюда AC — не минимум для C.

∇

Покажем что на i-м ур-не $deg_i(q_i) = O(1)$ в среднем.

Рассм. $i+1$ ур-нь $P_{i+1} \cup \{q\}$. Берем из P_{i+1} одну вершину, рассмотрим какой она была ближайшей на i-м ур-не.
Какая была степень вершины, суммир. ее степень и усредняем.

$\forall p \in P_{i+1} \cup q$ $\quad \dfrac{\sum_{P_{i+1}} deg(closest_{P_i}(p))}{|P_{i+1}|} =: \Omega$ ← степень в триангл. Делоне на $i+1$ ур-не

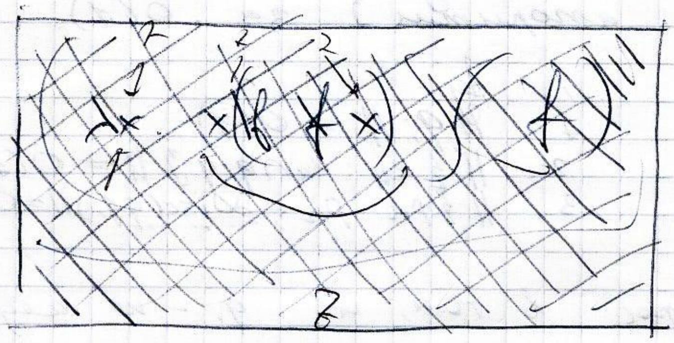
$\boxed{\text{NN graph строим на } P_i}$

Можно рассмотреть $\forall p \in P_{i+1} \cup q \setminus P_i$

Но мы знаем, что $\sum_{r \in P_{i+1} \cup q \setminus P_i} deg_{P_{i+1}}(closest_{P_i}(r)) \le$

Это грубая мажоранта → $\le 6 \cdot \sum_{r \in P_{i+1}} deg_{P_{i+1}}(r)$, Тогда $\Omega = 36$

Пояснение Радуга ⇓



Примечание: строим инкрементальную триангуляцию

1 слой [| t |] n   После добавления t порядок добавления не важен.

только добавлены

n слоёв [| m | | t |] n

С для позиции m известно, до какого уровня вверх мы пробросим вершину.

Аналогично после добавл. t точек порядок не важен. Запомн. позиции, которые дошли до i+1 ур-ня. Порядок в запомн. позициях опустим. Их всего k штук.

Теперь посмотрим на позицию t, тоже опустим. ∡ (k+1)! перестановок

Это нужно, чтобы не ∡ t! перестановок.

[| | | t] — тут перестроим, ∡ запрашиваем, — те, что пробросились на i-й уровень.

все на i+1

Тогда тут запрашиваем. элементы могут быть выбраны не более 6 раз. Для каждой из возм. перестановок

△

3. $\|\{p_n : p_n \in B(q, qq_i)\}\| = O(1)$

c в P_{i+1}

Посчитаем немного другое:

Сколько окружностей, для $\forall r \in P_{i+1}$, имеющих центр в r, радиусом $\rho_i(r, closest(r))$, содержат в себе случ. точку уровня $i+1$?

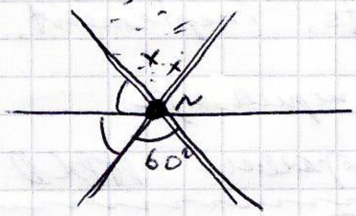
Точки из P_i в P_{i+1} не интересны, шары будут с $r = 0$.

Возьмем из $P_{i+1} \setminus P_i$

Теперь считаем для каждой $r \in P_i$

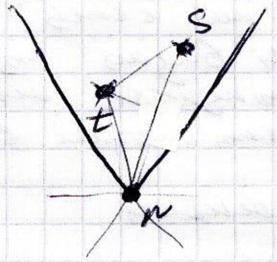
$$\left[\sum_{r \in P_i} \sum_{r' \in P_{i+1}} \delta\left(r \in B(r', \rho(r', closest_{P_i}(r')))\right)\right] / |P_{i+1}|$$

Это аналогично постановке вопроса.
Разобьем на 6 частей все пр-во



Возьмем точки r', которые лежат в секторе. Посортим их по расст. до r. Возьмем 2.

Δ × – точки из i   через красные
• – из i+1



Мы знаем, что $\rho(rs) > \rho(rt)$

$\xrightarrow{?}$ $\rho(st) < \rho(rs)$

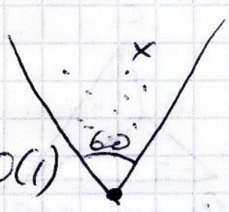
Δ

rs или rt – наиб. сторона, через углы доказывается за 7 класс

Все красные точки дальше ближ. черной (×) ∇.
точки не могут содерж. окр-ти, содерж. r.

Аналогично не очевидно, что все точки красных ближе ближайшей черной ×

Пример



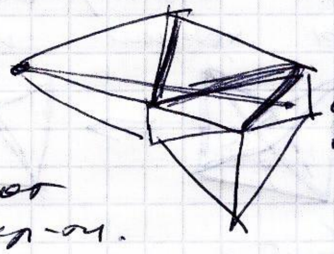
Матожидание того, что первая точка – черная = кол-во красных точек.

это черная $\sum (i-1)p^{i-1} q = O(1)$

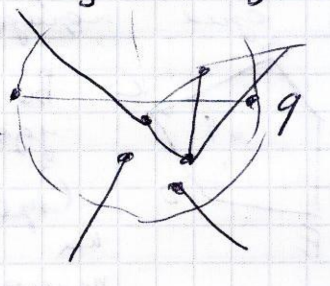
С вер-ю переходим на сл. уровень.

2. Рассмотрим ребро $q_i q$. В среднем сделать $O(1)$ флипов. Как мы вставили флипов в Делоне?

Мы флипаем ребра плохие.



Построим окр-ть на диаметре $q_i q$.
Возьмем ребра, кот. имеют хотя бы 1 конец в окр-ти.
Ребер вне окр-ти
Точек в окр-ти $O(1)$ из 3 пункта.
Тогда ребер тоже.

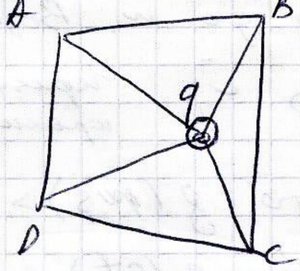


Ребер, которые пересекают $q_i q$, но не имеют концов в шаре, тоже $O(1)$.
Когда будем флипать, все такие ребра сломаем тоже.

27.10.2015

Сегодня будет: remove, constraints, implement, datasets

1. Remove: удаляем точку из триангуляции.



Удалим q, перестроим $ABCD$ — звездн.

О $ABCD$ — звёздный, то есть $\exists q$, что $\forall a_i$ a_i видна из q.

Ядро звёздного — выпуклое мн-во, из которого все ост. точки видны.

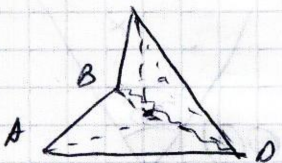
1) Найти ядро — пересечь полуплоскости

2) Триангулировать звёздный

нельзя отрезать ABC — не ухо

- Имея ядро, соединим его со всеми, потом флипаем пока q нельзя будет убрать.

Будем выбирать "ухо" — $\angle < 180^\circ$, такое что q не входит в это ухо.



без уха BAD тоже звёздный

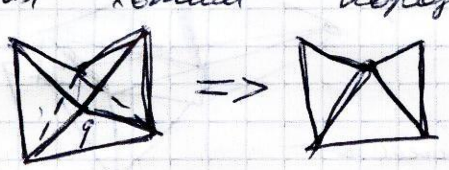
- Без ядра: найти ядро за $O(n)$ с помощью пересеч. полупл.

Без поиска ядра:
У звёздного многоугольника можно безболезненно отрезать ухо. (хорошее)

Нужно за $O(n)$ или $O(n \log n)$

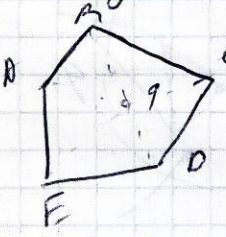
$\boxed{Д/З}$ Как-то...

Как перестроить $ABCD$?
Мы хотим порядок флипов, такой



← это верно.

Такой порядок есть: это обратный порядок добавления вершины q. В обоих случаях сущ. не один порядок.



Все ребра $ABCDE \in$ триант. Добавим при удалении $qA \dots qE$.

Спроецируем многоугольник на параболоид, предположим, что триангуляция есть. Возьмем точку и будем поднимать ее из $(x_0, y_0, -\infty)$ в $(x_0, y_0, +\infty)$

Видимость = снаружи параболоида

У нас будут появляться (исчезать) грани в порядке удаления / приближения точки.

Существует точка на l, из которой видно ровно BE, BC, CE. Это ровно пересечение l с параболоидом. $= Q \equiv (q, q^2)$

P – мн-во точек

$DT(P)$ – триангуляция

q – удаляемая точка

$DT(P)$ и $DT(P \setminus q)$ отличаются только в области смежности q.

Возьмем очередь с приоритетом; запихнем уши с $\angle < 180°$, приоритет – видимость уха с прямой l (мин.)

Первое ухо которое мы перестаем видеть (при подъеме по l) — первое.

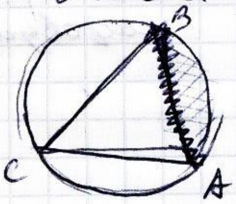
Добавляя уши в этом порядке мы получим уши Делоне. за $n \log n$

Почему? — $\boxed{\text{Д/З}}$

2. Constraints

Хотим Делоне когда ребра зафиксированы.

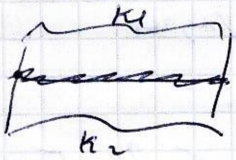
Условие Делоне:



В сегменте AB (#) могут быть точки. В остальных зонах окруж. – нельзя.

Хотим в constraint Делоне добавить constraint ребро. (Constraint'ы не пересек.)

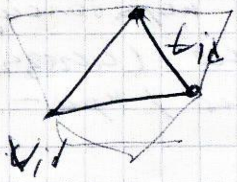
Разбиваем прямой новой на 2 монот. многоугольника.



Все удалим нафиг и заново потриангул.

3. Implement. details

Пользоваться DCEL – удобно, но избыточно.



Храним <tid, Vid>   point = <x, y>

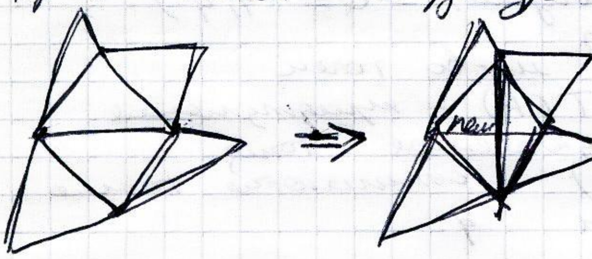
Vertex = <point, triangle_id>

triangle = <vertex_id[3], triangle_id[3]>

Смежные

Это меньше чем DCEL раза в два.

Flip в такой структуре:



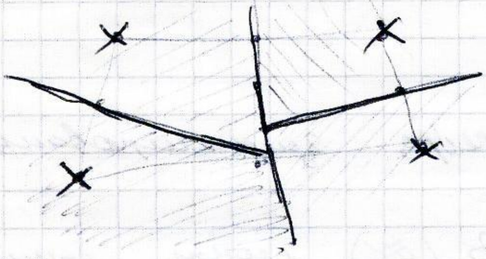
добавить 2 Δ, переставить triangle.triangle_id

Диаграмма Вороного

$f: \mathbb{R}^2 \to P$, P - конечное множество точек, упорядочено.

$f: \mathbb{R}^2 \to P\{0 \dots n-1\}$, $P = \{p_0 \dots p_{n-1}\}$

Графики f: (соотв. диаграмма)



Але метрика не важна, например с манхэттенской будет под 45°?.. Можно для Парижа!

И даже метрику графовую, с препятствиями.

Данные на выкладках:

Λ Ячейка Вороного - пересечение полуплоскостей

Λ Топология: P - мн-во из n сайтов.

Если все лежат на одной прямой — Vor(P) — n-1 паралл. линия. Иначе Vor(P) связна и все ребра — отрезки/лучи

Δ wiki ▽

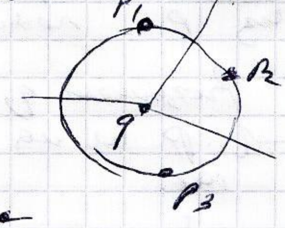
Λ Размер Vor(P): ≤ 2n-5 вершин
≤ 3n-6 ребер

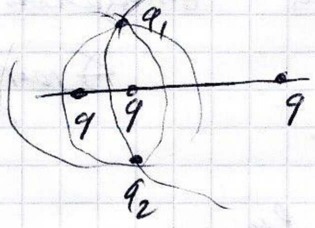
Δ $v - e + f = 2$ — по Ф. Эйлера ▽

О Наиб. пуст. окр. q по отнош. к P ($C_P(q)$) — наиб. окр. с центром в q, не содержащая ни одного сайта P

Л q — вершина $Vor(P) \Leftrightarrow C_P(q)$ содержит $\geq$ сайтов

Л Серед. перпендикуляр на $q_1 q_2$ — ребро $Vor(P) \Leftrightarrow$ на нем есть q, что $C_P(q)$ содержит только q_1, q_2

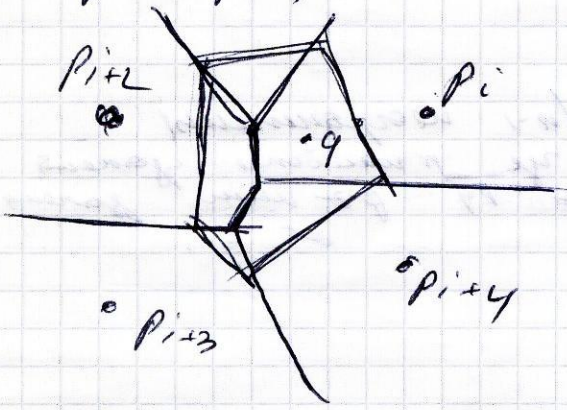




Т Если соединить все сайты, соотв. смежным ячейкам Вороного, то получим триангуляцию Делоне.

Построение:

- Наивно: $O(n^2 \log n)$ — пересеч. полуплоскостей
- Инкрементально: храним DCEL, добавим в диаграмму q. Локализуем q в ячейке сайта p_i. Дальше найдем соседа p_{i+1}, проведем серед. перпендикуляр qp_i, он пересечет границу p_i с p_{i+1}, построим qp_{i+1}, и т.д.



Каждый шаг за $O(i)$, значит суммарно за $O(n^2)$

О Диаграмма k-го порядка. Ячейка Вороного k-го порядка — $V(p_1 \dots p_k)$ — множество $\{x \mid \forall j \in \{1..k\}\ \forall i \notin \{1..k\}\ \rho(x, p_i) \geq \rho(x, p_j)\}$.

- Инкрементально за счет $k-1$ уровня. Возьмем S — мн-во сайтов для ячейки на $k-1$ уровне. Пересечем с диаграммой 1 уровня для $P \setminus S$.

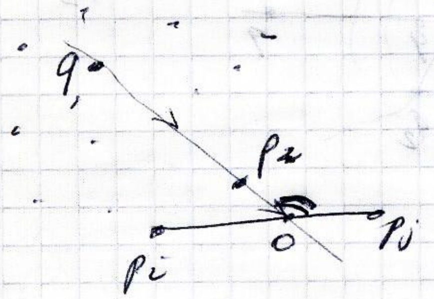
Пересечение $V_{k-1}(S) \cap V_1(p_i) \Rightarrow V_k(S \cup \{p_i\})$

Работает за $O(k \cdot n^3)$

Опр farthest-point — диаграмма — $n-1$ уровень.
В каждой ячейке все точки находятся наиболее далеко удалены от какого-то сайта.

Лемма $\forall q$ самый удаленный от неё сайт из P лежит на выпуклой оболочке P

$\triangleright$ Возьмем q, проведем луч qp.
т.к. p не на выпуклой оболочке, тогда внутри неё.



$\exists$ ребро, кот. пересекает qp; тогда либо p_i либо p_j дальше q.
из свойств большего угла (из $\angle qpp_j$, $\angle qpp_i$)

$\triangleleft$

Лемма Сайт внутри CH не может иметь ячейку в V_{n-1}

Лемма Каждый сайт q, $q \in CH(P)$ имеет ячейку в V_{n-1}

$\triangleright$ по индукции $\triangleleft$

Лемма Все ячейки в V_{n-1} неограниченны:
$\triangleright$ луч из q в p, где p наиболее удаленная от q. тогда $\forall z \in pq$ pz наиб. далеко от z.

Алгоритм построения V_{n-1}

Построим $CH(P) = \{p_1 \dots p_n\}$. Запомним порядок обхода и случайно переставим точки.

Далее строим инкрементально, начиная с трех. 3 трех- пересекает полуплоскостей.

Для p_i ячейка встанет между ячейками $cw(p_i)$ (clockwise) и $ccw(p_i)$

Они были соседями, поэтому достроим серед. перпендикуляр к $p_i\,ccw(p_i)$. обойдем ячейку $ccw(p_i)$, найдем с какой ячейкой он пересек. и так далее.

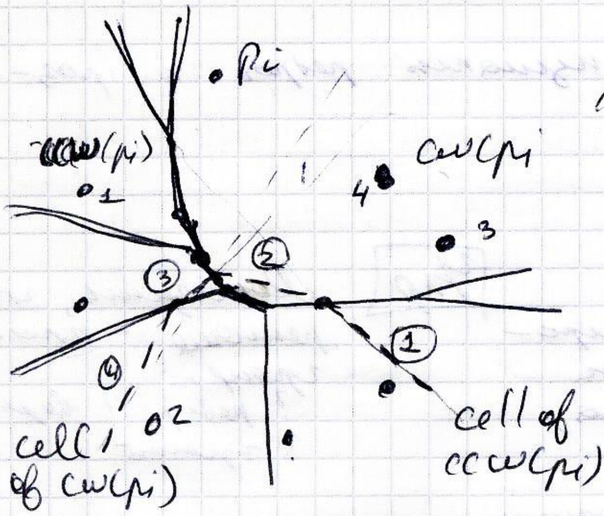


Иллюстрация к V_{i-1}

24.11.2015

Straight skeleton

Motivation: есть картинка, на ней есть объекты, нужно посчитать кол-во. Но там есть шум.

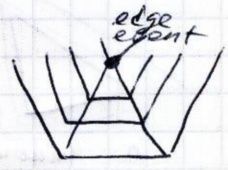
Эрозия: ▪ → ▫. Расширение: обратное. Шум уйдет.

Аналогично — карта и подсчет островов. Тут расширение испортит топологию острова. А хотим оставить.
Хотим расширить все острова: расширить полигон (диагр. Вороного для отрезков) — сложно и нетривиально. Straight-skeleton — это упрощ. без арок. Для выпуклых легко.

Применение: рисовать в игре крыши для зданий можно straight-skeleton'ом! Можно еще строить границы для рек, гор, проверять их топологию на карте.

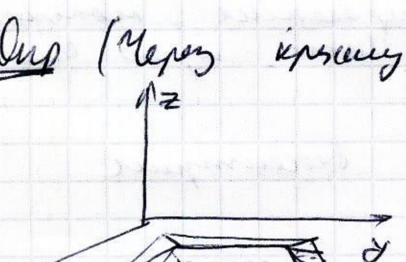
Опр (Процед. опред.) Берем полигон, кажд. сторона дв. внутрь с одинак. скоростью.

Концы ребер соединены. Пересекаем биссектрисы углов соседних.



По SS можно быстро построить оффсет время t

Опр (Через крышу)



Мерджим внутрь ребра и приподнимаем на ΔX. Потом опускаем вниз на пол-ть. Это есть пересечение полупространств (для выпуклых)

Есть обобщение — можно изменять рёбра с разной скоростью.

Построение $O(n\log n)$

Находить биссектрисы, первое пересечение, сортировать по расст. от т. пересечения до ребра, потом их соединить.

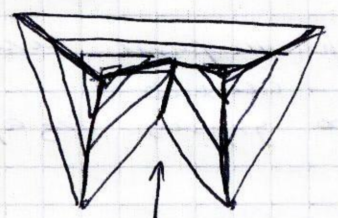
$\boxed{\text{Упр}}$ Доказать, что решение — планар. граф.
? рёбер ? вершин
? граней

Потом расширяем полигон.
Храним двусв. список полигона, в узле ещё 2 указ. на лев/прав ребро.
При добавл. биссектр. добавл. её в список и новые добавл. в список.
Удаление — $O(1)$ могу.
Сортировка — $n\log$, итого $O(n\log n)$

~ ~ ~

Для невыпуклого: $O(n^2\log n)$

Felkel's algo:



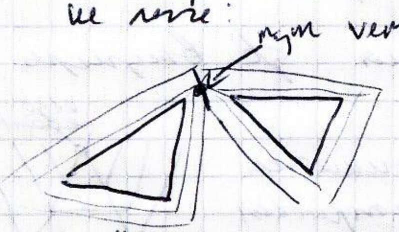
тут перевал.

что если было только edge event'ы, а сейчас есть ещё split-event'ы.
Есть edge и valley.

За n^2 можно перебрать все пары вам. верш. — отрезок и проверить, породит ли split-event

$a_1x+b_1y+c_1=t$ — ур-е прямой в мом. врем. t.
$a_1^2+b_1^2=1$

Можно расширять наружу.
Не легче: тут vertex event!



с какого-то момента острова объединяются.

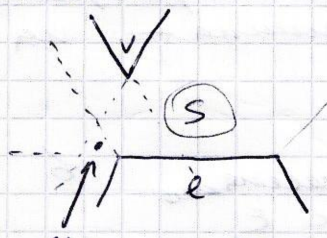
Тогда решение split-случая — это решение

$$\begin{cases} a_1x+b_1y+c_1=t \\ \vdots \\ a_3x+b_3y+c_3=t \end{cases}$$

Тут всё работает в рац. арифметике, можно если всё уже отнормировано. Иначе, нужно считать корень.

Проблема, нужно для сравнения биссектрис считать $r = S/P$ и корни.

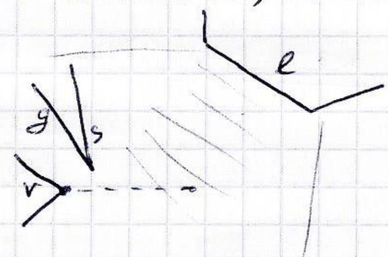
В такой интерпретации split-event'ы — это решения ур-е для ребра x2 и ~~другого двух~~.
Edge event — для соседних трех.



← предикат

Если пересечение бисс. не в области S, то не добавляем == не split event.

Может быть, что у нас пересеч. раньше:



Вершина s может "перебить" столкновение v с e
Это решается пересечением v с g — входит в шаг 2

Для каждого ребра соседние — $O(n)$
Перебираем <вогн. верш. + ребро> — $O(n^2)$

Для каждого события $O(n) * O(n)$ еще добавим и еще очередь с приор. — $\log n$

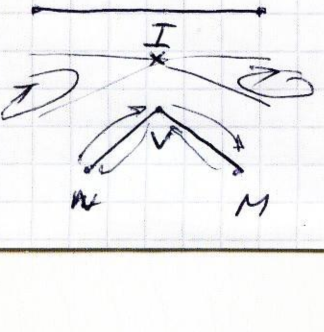
Итого вот добавили, посортили.

Достаем события. Если edge event — как раньше, просто заменим 2 узла в списке на 1. $O(1)$ событий добавилось.
Если split-event, то могли создать еще вогнутую вершину.

---

split-event'ы могут организовать новые.

В нашем план. графе не более n граней, потому что каждая грань растёт единожды. Тогда не более n вершин, n ребер. Отсюда для каждой верш. не более n опер.
Отсюда n^2



— распались на 2 списка. Просто вместо I сделаем v_1 и v_2, разорвем список на 2.