Конспект по вычислительной геометрии (Ковалёв)

25 декабря 2015 г.

Содержание

1 Как пользоваться этим документом

1. Писать билетики параллельно
2. Все формулы оформлять в латехе.
   - `http://www.math.uiuc.edu/~hildebr/tex/course/intro2.html`
   - `https://en.wikibooks.org/wiki/LaTeX/Mathematics`
   - `https://en.wikibooks.org/wiki/LaTeX/Advanced_Mathematics`
   - Рекомендуется смотреть конспект Сугака тут.
3. Можно пользоваться `(org-toggle-pretty-entities)` чтобы latex отображался юникодом в emacs'е.
4. **TODO** обозначают степень написанности материала. Положительное значение интерпретируется как процент. X – метка для "тут нифига не ясно и проблемы".
5. Смотреть превью формул с помощью `C-c C-x C-l`.
6. Экспортить с помощью `C-c C-e l p`. Если не работает – сначала в 'tex', а потом уже экспортировать 'pdflatex''ом. Экспорт игнорирует ошибки, поэтому сделать это руками и пофиксить их – не так плохо.
7. Если что-то не собирается, писать `@volhovm`.
8. Синтаксис теха смотреть хз где, но есть `http://detexify.kirelabs.org/classify.html` для непонятных символов.
9. Синтаксис орга – это тут:
   - `http://orgmode.org/manual/Emphasis-and-monospace.html`
   - `http://orgmode.org/manual/LaTeX-and-PDF-export.html#LaTeX-and-PDF-export`
   - `http://orgmode.org/tmp/worg/org-tutorials/org-latex-export.html`
   - Остальное (списки, хедеры, блабла) очевидно.

2 Tickets

4	1	Принадлежность точки выпуклому и невыпуклому многоугольникам
21	1	Триангуляция Делоне. Алгоритм и доказательство его корректности.
5	2	Статические выпуклые оболочки на плоскости. Джарвис, Грэм, Эндрю, Чен, QuickHull. Оболочка многоугольника, оболочка полилинии.
20	2	Триангуляция Делоне. - существование; - приводимость любой триангуляции флипами к ТД; - эквивалентность критерия Делоне для треугольников критерию для ребер.
8	3	Триангуляция многоугольника. Существование, ушная триангуляция.
17	3	Сумма Минковского (определение, вычисление)
12	4	ППЛГ и РСДС (PSLG и DCEL): определение, построение РСДС множества прямых
15	4	Трапецоидная карта.
10	5	Пересечение полуплоскостей, связь с выпуклыми оболочками
13	5	Пересечение многоугольников (PSLG overlaying)
7	6	Выпуклая оболочка в n-мерном пространстве. Quick-hull и вероятностный алгоритм.
11	6	Пересечение множества отрезков.
9	7	Триангуляция многоугольника заметающей прямой
14	7	Локализация в ППЛГ. - методом полос (персистентные деревья); - Киркпатрик.
6	8	Динамическая выпуклая оболочка (достаточно $\log^2$ на добавление/удаление)
19	8	Граф видимости и планирование движения. - построение графа видимости заметающим лучом; - сокращение графа видимости; - построение навигационного графа на трапецоидной карте; - планирование маршрута невыпуклого тела с вращением (без суммы Минковского).
3	9	Пересечение отрезков и поворот: определение, свойства, вычисление
22	9	Диаграмма Вороного. - определение и свойства; - диаграмма Вороного высших порядков, построение; - связь с подразбиением Делоне (ближайший и дальнейший); - алгоритм построения ДВ.
1	10	Skip quadtree: определение, время работы
18	10	Минимальная охватывающая окружность множества точек. Вероятностный алгоритм.
2	11	Пересечение прямоугольника с множеством прямоугольников и непересекающихся отрезков: - range tree + fractional cascading; - interval tree; - segment tree; - priority search tree; - k-d tree. van Kreveld, de Berg, Overmars, Cheong
16	11	Диаметр множества точек (вращающиеся калиперы)

3 0 1: Skip quadtree

4 0 2: Пересечение многоугольника с множеством полигонов/отрезков

5 0 3: Пересечение отрезков и поворот

Рассмотрим задачу проверить пересечение отрезков.

Вот есть у нас $S_1 = (p_{11}, p_{12}), S_2 = (p_{21}, p_{22})$.

В общем случае с Евклидовым пространством возникакуют какие-то проблемы, поэтому рассмотрим следующее определение Афинного пространства:

A – аффинное пространство, если A – такой набор точек, что:

1. В пространстве существует хотя бы одна точка.
2. $A, B, \leftrightarrow v = \overrightarrow{AB}$, причем $B = A + v$.
3. Точка + вектор = точка.
4. ... и еще 40 аксиом векторного пространства

Аффинное пространство отличается от стандартного евклидового тем, что в нем все точки равноправны, то есть ноль не зафиксирован. Типа у нас в этом пространстве есть точки, а векторы строятся из них.

Рассмотрим гиперплоскость в n-мерном аффинном пространстве. Она, очевидно, задается $n-1$ вектором, или как минимум n точками.

Рассмотрим произвольную точку A и набор векторов: $AP_1 \cdots AP_n$. Тогда если точка A принадлежит гиперплоскости, то такой набор, очевидно, линейно зависим.

Возьмем другую случайную точку B и посмотрим, как меняются координаты при переходе из системы координат, связанной с A в систему, связанную с B (очевидно, что такой набор векторов может задавать базис, если он ЛНЗ).

Theorem 1 (О повороте)**.**
Тут должно быть какое-то утверждение о повороте.

Доказательство. Рассмотрим точку X в базисах из векторов $\{\overrightarrow{AP_i}\}_i$ и $\{\overrightarrow{BP_i}\}_i$. Тут точки P_i задают гиперплоскость, то есть принадлежат ей и не линейно зависимы друг относительно друга в ней.

$$X = X_A^1\overrightarrow{AP_1} + X_A^2\overrightarrow{AP_2} + \cdots + X_A^n\overrightarrow{AP_n} = X_B^1\overrightarrow{BP_1} + X_B^2\overrightarrow{BP_2} + \cdots + X_B^n\overrightarrow{BP_n}$$

Для каждого вектора $\overrightarrow{AP_i}$ выразим его в базисе векторов $\overrightarrow{BP_i}$.

$$\begin{aligned}
&\overrightarrow{AP_1} = \alpha_1^1\overrightarrow{BP_1} + \cdots + \alpha_1^n\overrightarrow{BP_n} \\
&\cdots \\
&\overrightarrow{AP_n} = \alpha_n^1\overrightarrow{BP_1} + \cdots + \alpha_n^n\overrightarrow{BP_n}
\end{aligned}$$

Подставим выраженные AP_i в первое уравнение.

$$\begin{aligned}
X &= X_A^1\left(\sum \alpha_1^i\overrightarrow{BP_i}\right) + X_A^2\left(\sum \alpha_2^i\overrightarrow{BP_i}\right) + \cdots + X_A^n\left(\sum \alpha_n^i\overrightarrow{BP_i}\right) \\
&= \overrightarrow{BP_1}\left(\sum \alpha_i^1 X_A^i\right) + \overrightarrow{BP_2}\left(\sum \alpha_i^2 X_A^i\right) + \cdots + \overrightarrow{BP_n}\left(\sum \alpha_i^n X_A^i\right)
\end{aligned}$$

Сопоставив это с X, выраженным через $\{\overrightarrow{BP_i}\}_i$, получим следующую зависимость:

$$\left(X_B^1, X_B^2, \cdots, X_B^n\right) = \left(X_A^1, X_A^2, \cdots, X_A^n\right) \times \begin{pmatrix} \alpha_1^1 & \cdots & \alpha_1^n \\ \vdots & \ddots & \vdots \\ \alpha_n^1 & \cdots & \alpha_n^n \end{pmatrix} + \left(\overrightarrow{BA}^1, \cdots, \overrightarrow{BA}^n\right)$$

Последнее вектор перехода из точки B в A. Пусть дана точка O, которая воспринимается как ноль координат. Пусть также дана точка O', которая выражается через O. Тогда матрица A записывается следующим образом:

$$A = \begin{pmatrix} P_1 - O' \\ P_2 - O' \\ \cdots \\ P_n - O' \end{pmatrix}$$

Тут P_i и O' – это точки, координаты которых записаны отнсительно базиса $O\{e_1, \cdots, e_n\}$.

Заметим, что мы можем разбить все пространство на три класса согласно того, какой знак перехода из O в O'. A – матрица перехода от O к O',

Ориентация – свойство точки относительно базиса $O\{e_1, \cdots, e_n\}$ и гиперплоскости, заданной точками $\{P_i\}_{i=1}^n$.

Известный факт из линейной алгебры:

$$\begin{vmatrix} \overrightarrow{P_1} & 1 \\ \overrightarrow{P_2} & 1 \\ \vdots & \vdots \\ \overrightarrow{P_n} & 1 \\ \overrightarrow{A} & 1 \end{vmatrix} = \begin{vmatrix} P_1 - A \\ P_2 - A \\ \vdots \\ P_n - A \end{vmatrix}$$

Покажем, что знак детерминанта матрицы A действительно зависит от положения точки относительно гиперплоскости. Возьмем A, B, рассмотрим множество точек $\{\overrightarrow{A}t + \overrightarrow{B}(1-t)\}$.

тут какая-то магия, TODO

□

6 0 4: Локализация в многоугольнике

7 0 5: Статические выпуклые оболочки в $\mathbb{R}$

8 2 6: Динамическая выпуклая оболочка

(`CH_DYN_1`)

Начнем с подзадачи: пусть у нас есть две каких-то верхних оболочки в $\mathbb{R}^2$, разделенных по иксу. Мы хотим объединить эти верхних оболочки, проведя касательную сверху. Как такую касательную построить? (inb4 такая существует, потому что "палка сверху падает на холмики"). Как искать такую касательную за логарифм?

Очевидно, что касательная не проходит по экстремальным точкам (нарисуем большой холмик и рядом маленький).

Если мы хотим за логарифм, то че делать?

(`CH_DYN_2`)

Предположим, что есть пара точек на холмах. Будем типа пользоваться некоторым подобием бинпоиска на двух холмах сразу – четыре границы одновременно. Ну, два массивчика – это два множества точек для двух оболочек, отсортированных по иксу.

(`CH_DYN_3`) описывает классификацию всех попаданий касательной к кускам выпуклой оболочки для левой и правой кучи. Эта классификация важна, так как по ней мы будем определять текущее состояние. Как эти состояния отличать, понятно – считаем повороты. Случаи с двумя точками по одну сторону классифицируются поворотом.

(`CH_DYN_4`)

Рассмотрим случай A в `CH_DYN_2`. Рассмотрим прямую l и какую-то касательную к левой куче. Утверждается, что если мы будем поворачивать касательную вокруг точки касания, поворачивать вниз, то пересечение касательной и l как точка, будет опускаться вниз. Короче

случай A распознается так: это случай слева а), а справа г). Тогда мы можем отрезать нижние куски выпуклых оболочек.

Проверка на два случая делается за $2 \times 2 = 4$ поворота.

Рассмотрим остальные случаи, например B в `CH_DYN_2`. В этом случае мы можем откинуть нижнюю часть правой оболочки. Симметричный случай тоже очевиден.

Случай с двумя касательными тоже распознается однозначно и есть ответом.

(`CH_DYN_5`)

Пусть на правом холме у нас касательная, а на левом точка из случая а) – `CH_DYN_5` A. Тогда на левом холме мы можем откусить нижний кусок, а на правом – левый нижний от касательной. Симметрично тоже.

`CH_DYN_5` B тоже так решается, то есть можно слева откусить нижний, а справа нижний левее точки касания.

(`CH_DYN_6`)

Теперь рассмотрим самый нетривиальный случай: пусть слева б), а справа д). Рассмотрим пересечение прямых l_1 и l_2. Прямые проведем через текущие вершины и следующие выше. Проверим точку L пересечения l_2 и l_2. Тогда если прямая L лежит полностью в интервале между холмами, то можем выкинуть и у левого и у правого нижние куски. Если точка L лежит в левом холме (левее самой правой точки левого холма), то мы выкидываем весь нижний кусок только левого холма вместе с этой точкой. Аналогично с правым холмом.

Теперь мы умеем решать задачу найти касательную двух верхних полуоболочек.

Тут Славик рассказал способ найти касательную точки и многоугольника с помощью подразбиения многоугольника на подмногоугольники (каждый вложенный берет точки предыдущего через одну). Потом он типа ищет для самого вложенного треугольника касательную, а потом передвигается к более богатым многоугольникам, сдвигая касательную влево или вправо на одну вершину. Тоже алгоритм за log(n). Типа на каждом шаге есть step, мы рассматриваем текущего кандидата на касательную + step и -step. Выбираем лучшего, переходим к нему и делим шаг на два.

А как найти все четыре касательные для двух выпуклых множеств? Можно разбить на несколько и сведем к предыдущей задачи. Без этого? Нетривиальненько.

Теперь мы хотим честного итеративного построения. Можно хранить оболочки skip-листом и вместо бинпоиска просто спускаться на нижний уровень и ходить там. Вот мы идем по какому-то уровню, берем вершинку. Вдруг мы поняли, что нужно отрезать левую часть листа. Пойдем вправо. Спускаемся вниз, если нужно пойти в какую-то сторону, а та вершина уже "отрезана".

(`CH_DYN_7`)

Пусть есть оболочка, являющаяся общей частью двух оболочек. Типа дана оболочка, есть указатель на точку, по которой нужно разделиться. Причем у нас есть синяя и красная (карандашом) часть. Тогда мы можем фактически сделать две оболочки – это за 2 * logn для объединения двух скиплистов.

А как вообще все хранить, чтобы было итеративно? Будем хранить дерево, в котором листья – наши точки, а другие узлы – это верхняя оболочка сыновей. Это n*logn памяти, а хотим меньше. Причем неочевидно, как делать удаление. Как добавить? Прокинуть вершину вниз и перестроить все оболочки вверх во время просеивания. Если дерево нужно балансировать, то тоже нормально – перестроим что-нибудь.

Можно, формально, хранить немного не так: в самом верхнем узле будет храниться честная выпуклая оболочка всех точек. А в не верхнем, будем хранить только ту часть выпуклой оболочки, которая не является общей с родителем. Ну, типа, как раз синяя или красная часть. Тогда при продавливании точки вниз все проще: разбиваем текущую выпуклую оболочку (сначала корневую), объединяем за $\log n$ с чилдами. Определяем, куда кидать точку – влево или вправо. На одну часть забиваем. Так проходим вниз и добавляем вершинку. Заметим, что теперь уже не нужно хранить ничего в листах, так как два соседних листа однозначно определяются оболочкой в их паренте. Дальше строим оболочку и просеиваем вверх. Типа двух братьев берем, объединяем, отдаем паренту оболочку, себе оставляем только те части, которые

не входят в парента.

Итого мы умеем удалять и добавлять вершинки за $\log^2 n$

Антон решил пояснить за то, как нужно делать мердж skip-листов. Лист мы держим сверху за вершину самого высокого уровня. Сплит: дали нам вершинку, нашли ее в самом нижнем уровне. Удаляем, обрезаем. Идем влево, пока не можем подняться наверх, поднимаемся, делаем вершинку терминальной, и так до верхнего уровня. Аналогично идем вправо и делаем ее первой. Мердж делается так же, про асимптотику думать не нужно.

9 0 7: Трехмерные выпуклые оболочки (CHN)

Немножко модифицируем quickhull на плоскости, чтобы можно было очевидно его перенести в n-мерное пространство. Quickhull не работает хорошо с детерменированной прямой.

Давайте выберем прямую L_1L_2. Зафиксируем в надмножестве случайную точку A. Все точки, которые попали в L_1AL_2 выкидываем. Рассмотрим все точки, которые не попали внутрь. Подразобьем их лучами L_1A и L_2A. Типа будем выбирать случайные точки вверху и продолжать выпуклую оболочку.

Для каждого разбиения мы перебираем все точки и для каждой мы запоминаем грани, которые видно.

Че делать в n-мерном пространстве? Возьмем произвольный тетраэдр. На самом деле лучше брать максимально большой тетраэдр. Потом для каждой новой случайной точки мы понимаем, к какой гране он принадлежит, какие грани эта точка видит.

10 0 8: Триангуляция (опр. + уши)

Существование, ушной алгоритм.

11 0 9: Триангуляция с замет. прямой

12 0 10: Полуплоскости и выпуклые оболочки

13 0 11: Пересечение множества отрезков

14 0 12: PSLG и DCEL

15 0 13: PSLG overlaying

16 0 14: Локализация в PSLG

17 0 15: Трапецоидная карта

18 0 16: Вращающиеся калиперы

19 0 17: Сумма Минковского

20 0 18: Вероятностный алгоритм мин. охва. окружности множества точек

21 0 19: Граф видимости и планирование движения

22 0 20: Триангуляция Делоне

23 0 21: Доказательство (алгоритм + корректность)

24 2 22: Диаграмма Вороного

Статья на викиконспектах

24.1 Алгоритм и асимптотика

Антону больше нравится инкрементальный алгоритм построения диаграммы Вороного, так как он похож на Делоне.

Типа вот есть бакеты, мы там чето меняем, проводим $O(1)$ времени на каждом уровне, суммарно получается $O(n)$.

У нас есть $O(log(n))$ уровней, где есть какие-то сабсеты, для каждого мы можем построить за $O(1)$ новую диаграмму.

Как локализоваться в диаграмме Вороного, где точек $O(1)$? Тупо найти ближайшую точку, посчитав метрику.

`VOR_O` Как с помощью $n+1$ уровня найти ближайшую точку на n-м уровне? X ближайшая точка на $n+1$ уровне. A точка, которую мы хотим вернуть, то есть ближайшая к q на n-м уровне. Проведем отрезок XA и проверим все соседние грани точки X, выберем ту, которую пересекает XA. XA также может пересекать какую-то точку триангуляции. Тогда нужно перебрать все соседние прямые, исходящие из этой точки и выбрать такие две, между которыми проходит XA.

Как достроить диаграмму Вороного, если мы уже локализовались? Построим между q и A серединный перпендикуляр, пересечь его с фейсом вершины A. Будем дальше идти по соседним DCEL'ам и заворачивать, строя серединные перпендикуляры, прямые вокруг q. Таким образом, построим грань для вершины q.

Асимптотика (inb4 можно это делать, строя двойственную триангуляцию):

- Вставка: посчитаем среднюю степень, проведем регрессионный анализ, как в алгоритме Делоне.
- Локализация: пересечем $O(1)$ ребер. Это доказательство тоже копируется с Делоне. Можно сказать, что мы пройдем по количеству DCEL'ов которые не добавились на более высокий уровень. Поскольку слои диаграммы это множество Бернулли, то на каждом шаге мы добавим не больше чем сколько-то точек, а они экспоненциально убывают.

24.2 Удаление из диаграммы Вороного

`VOR_2`

Возьмем сайт, его фейс. Будем строить типа straight skeleton, двигая стороны внутрь по серединным перепендикулярам. Тогда в какой-то момент схлопнется.

24.3 Построение из триангуляции диаграмму

`VOR_1`

Как построить из триангуляции Делоне диаграмму Вороного? Возьмем диаграмму, выделим какую-то точку A. Построим серединные перпендикуляры для каждого ребра, исходяшего из A, пересечем их всех. Поймем, что получившееся пересечение сер. перпендикуляров образует ячейку Вороного.

Покажем, что такая ячейка конечна. Рассмотрим треугольник ABC. По определению, этот треугольник треугольник Делоне, поэтому точка пересечение серединных перпендикуляров лежит внутри, и расстояние от S до точек прямоугольника минимально, если точка есть пересечение серединных перпендикуляров. Более того, по свойству Делоне, в окружности не лежит никаких других точек.

Любой отрезок ячейки Вороного принадлежит ей, потому что ячейки диаграммы Вороного выпуклые. Отсюда, поскольку точки отрезка лежат в ячейке, отрезок тоже лежит. Типа сама точка A лежит ближе всего к себе. Точка пересечения сер. перпендикуляров тоже лежит в ячейке, тогда для каждых двух соседних точек прямая между ними тоже лежит, т.к. ячейка вороного выпуклый многугольник.

24.4 Построение из диаграммы триангуляции

Возьмем диаграмму Вороного и построим **разбиение** Делоне то есть могут получиться не треугольники. В этом случае можно показать, что любой такой многоугольник можно триангулировать любым образом, при этом свойство Делоне останется.

Почему в общем случае граф, в котором мы соединили сайты соседних граней, получится разбиением Делоне? Ну типа, возьмем в DCEL'е узел, который соединяет три грани. Берем их сайты, соединяем. Это получится треугольник. Прогоним обратное следствие в нужную сторону.

24.5 Высшие порядки

Диаграмма Вороного второго порядка VD^2 это: $P_1, P_2 \in V_{q_1q_2} d(p_1, q_1) = min_1, d(p_1, q_2) = min_2$.

Аналогично строим VD^k диаграмму Вороного k-го порядка.

Диаграмма Вороного $n-1$-го порядка это набор таких сайтов, что для каждого есть $n-1$ точка, и для всех точек от них есть какая-то одна самая далекая.

Как строить инеркментально? Нужно проводить алгоритм удаления точек, но не удаляя прямые, которые мы двигаем, до самого конца.

Для каждого фейса мы делаем это за: $klogk$, но

$$\sum k \log k \leq \sum k \log n = \log n \sum k = O(n \log n)$$

Сколько будет вершин в диаграмме вороного второго порядка? Столько же, сколько и ребер, вернее удвоенное количество.

ЧИТАТЬ НА ВИКИ

24.6 Диаграмма минус первого порядка

Граф Делоне двойственный диаграмме -1 порядка это верхняя крышка проекции диаграммы на параболоид.

Типа возьмем треугольник, тогда в окружности должны находиться все точки множества. Количество вершин в такой диаграмме Вороного это количество вершин выпуклой оболочки.

25 Бонусные задачи

- Из множества прямых произвольных восстановить DCEL. Можно делать инкрементально. Для трех понятно, как строить. Дальше кидаем прямую. Берем первое пересечение, локализуем точку на прямой за $O(n)$, дальше обходим соседние фейсы DCEL'а пока не найдем точку пересечения нашей прямой с какой-то другой. И так пока все не пересечем. Можно показать, что асимптотика будет норм – $O(n^2)$.

  Рассмотрим `BON-0`. l – наша прямая. Утверждается, что от пересечения нашей прямой с какой-то другой до следующего пересечения нужно пробежать не более чем $O(n)$ ребер.

  В среднем заметим, что у нас $O(n^2)$ ребер и $O(n^2)$ фейсов. Тем не менее, из этого не следует, что на каждый фейс приходится $O(1)$ ребер, может быть так, что какие-то фейсы жирные, а какие-то нет.

  Мы всегда знаем направление, в котором мы будем двигаться от точки.

  Рассмотрим все Покрасим точки с положительным наклоном относительно прямой синим цветом, а серым – с отрицательным. Будем считать только те ребра, которые лежат в DCEL'ах, которые пересекает наша прямая l.

  Посчитаем, сколько таких цветных ребер есть (кстати, синих и серых ребер будет одинаковое количество).

  Прямая l прйдет через O(n) ребер. Выкинем самую правую прямую, которая имеет синий отрезок. Тут типа индукция. Тогда есть $\exists c$, что прямая пересекает не более $c * (n - 1)$. Попробуем ее заново добавить. Пусть прямая имеет серый наклон вправо. Тогда такая прямая подразобьет не более чем $O(1)$ ребер.

26 TODO Add more