Конспект по вычислительной геометрии (Ковалёв)

21 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/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: $AP_1....AP_n$. 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.
Короче типа хех мемосы вот шмяк шмяк фигакс

Доказательство.

$$\begin{aligned} X &= X_A^1 AP_1 + X_A^2 AP_2 + ... + X_A^n AP_n \\ &= X_B^1 BP_1 + X_B^2 BP_2 + ... + X_B^n BP_n \end{aligned}$$

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

Подставим выраженные AP_i в первые уравнение.

$\mathrm{X} = \mathrm{X}_A^1 \left(\sum \alpha_1^i \vec{BP_i}\right) + X_A^2 \left(\sum \alpha_2^i \vec{BP_i}\right) + ... + X_A^n \left(\sum \alpha_n^i \vec{BP_i}\right) = \vec{BP_1} \left(\sum \alpha_i^1 X_A^i\right) + \vec{BP_2} \left(\sum \alpha_i^2 X_A^i\right) + ... + \vec{BP_0} \left(\sum \alpha_i^n X_A^i\right)$

$$\left|\sum_{i=1}^{n} a_i b_i\right| \le \left(\sum_{i=1}^{n} a_i^2\right)^{1/2} \left(\sum_{i=1}^{n} b_i^2\right)^{1/2}$$

□

Theorem 2 (о саси)**.**
кокойтотекст))

Theorem 3 (о саси2)**.**
кокойтотекст2))

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)

10 0 8: Триангуляция (опр. + уши)

Существование, ушной алгоритм.

11 0 9: Триангуляция с замет. прямой

12 0 10: Полуплоскости и выпуклые оболочки

13 TODO Add more