Алгоритмы и структуры данных

Лекция 22
Геометрия. Выпуклые оболочки.

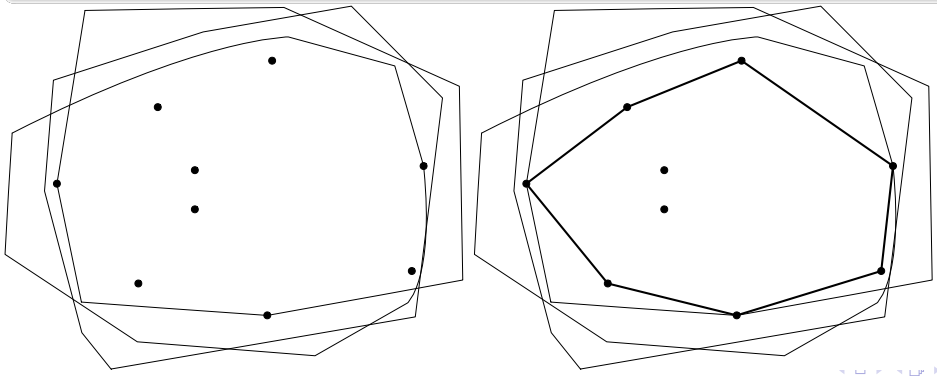
Сергей Леонидович Бабичев

Definition (Выпуклое множество)

Выпуклым множеством точек S называется такое множество, в котором для любых двух точек p и q отрезок (p, q) полностью принадлежит множеству.

Definition (Выпуклая оболочка множества)

Пусть S — множество точек в $\mathbb{R}^2$. Тогда **выпуклой оболочкой множества** S называется пересечение всех таких выпуклых множеств F таких, что $S \subset F$.



Выпуклая оболочка

Definition (Альтернативное определение)

Другое определение: выпуклая оболочка (ВО) $\mathrm{conv}(S)$ — минимальное по включению выпуклое множество, содержащее S.

Lemma

Выпуклая оболочка — всегда многоугольник.

Построение выпуклых оболочек

Задача 1. На плоскости задано множество точек S. Построить выпуклую оболочку $\mathrm{conv}(S)$.

В дальнейшем под термином *выпуклая оболочка* или $\mathrm{conv}(S)$ мы будем понимать границу выпуклой оболочки как области или фрагмент такой границы. Будем обозначать H.

Простой алгоритм нахождения выпуклой оболочки

1. Для каждой пары точек $P_1, P_2 \in S, P_1 \neq P_2$ построим прямую, проходящую через них.
2. Если остальные точки лежат в одной полуплоскости, образованной этой прямой или на отрезке P_1P_2, то отрезок P_1P_2 — одно из рёбер $\mathrm{conv}(S)$.

Обоснование: выпуклая оболочка — пересечение таких полуплоскостей.

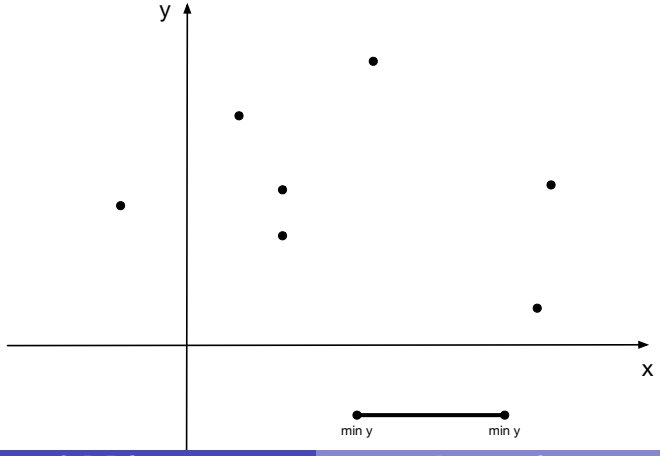
Сложность алгоритма:

- $\frac{N \cdot (N-1)}{2}$ пар точек для проверки.
- Сложность каждой проверки $O(N)$.
- Общая сложность

$$T = O(N^3).$$

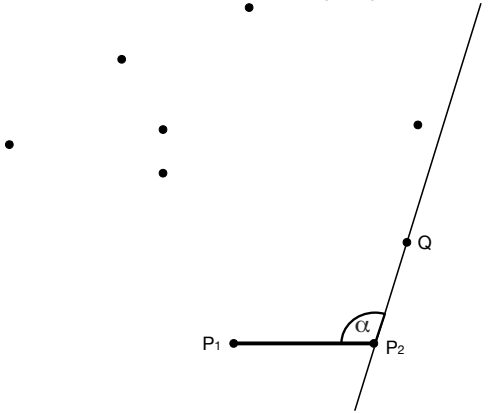
Алгоритм Джарвиса

- Известен как заворачивание подарка.
- Рассмотрим самую нижнюю точку P_2 множества S. Если таких точек несколько — возьмём самую правую из них.
- Эта точка принадлежит границе $\mathrm{conv}(S)$.
- Все остальные точки лежат либо выше её, либо левее.



Алгоритм Джарвиса

- Предположим, мы нашли одно ребро, лежащее на границе BO и имеется ещё одна точка P_1 на этом ребре. Какая следующая точка?

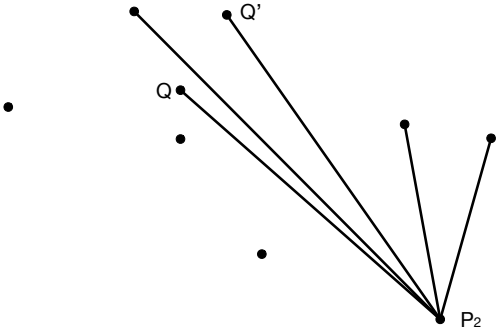


- Идея: это точка Q, дающая наибольший угол α из всех возможных между векторами $\overrightarrow{(P_2, P_1)}$ и $\overrightarrow{(P_2, Q)}$.

Алгоритм Джарвиса

- Перебираем все точки Q'. Для каждой такой точки находим угол $\angle P_1 P_2 Q'$.
- Очередной точкой в H становится такая, которая даёт наибольший угол.
- Как обойтись без тригонометрии?
- Введём понятие упорядоченности точек: Q' предпочтительнее Q, если

$$\overrightarrow{(P_2, Q)} \times \overrightarrow{(P_2, Q')} < 0. \tag{1}$$



- Обязательное условие: $P_2 \in H$.
- Будем искать самую предпочтительную точку Q для всех $Q \notin H$.

Алгоритм Джарвиса

- Для каждой очередной вычисленной точки контура P_2 ищем Q — аналог максимума.
- По всем $Q' \in S\ H$

  если $\overrightarrow{P_2, Q} \times \overrightarrow{P_2, Q'} < 0$

  то $Q = Q'$

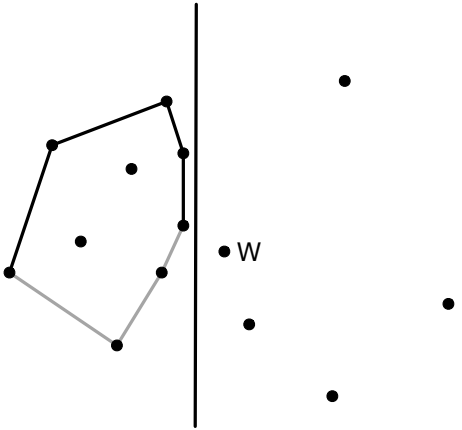
  Если $\overrightarrow{P_2, Q} \times \overrightarrow{P_2, Q'} = 0$, то выбираем точку более удалённую от P_2.
- Сложность: за $O(N)$ находим новую вершину из H.

$$T = O(N \cdot |H|).$$

Алгоритм Эндрю

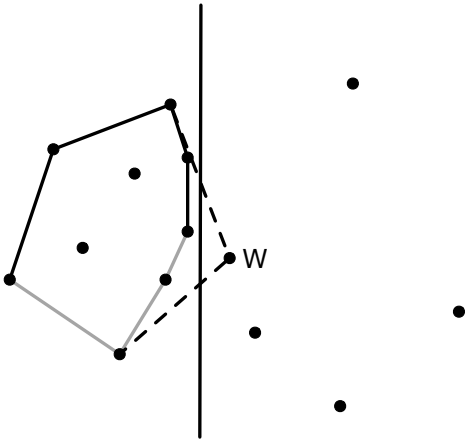
- Ещё известен как алгоритм «сортировка по координатам».
- Все точки сортируются: вначале по координате x, при равных x — по координате y.
- Алгоритм строит две части оболочки — верхнюю и нижнюю.
- Самая левая точка принадлежит H.
- Самая правая точка принадлежит H.
- Верхняя и нижняя части — движение от самой левой точки до самой правой.
- Будем говорить «верхняя оболочка» и «нижняя оболочка».
- Храним отдельно оболочки в разных векторах/стеках.

- Алгоритм по-очереди рассматривает новые точки в порядке сортировки и поддерживает инвариант: для всех рассмотренных точек имеется выпуклая оболочка.
- Это — вариация на тему «заметающая прямая».

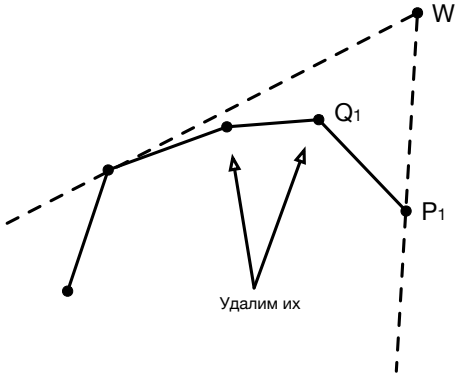


- Пусть уже часть построена и добавляется новая точка W.
- Каждая добавленная точка всегда принадлежит выпуклой оболочке, построенной на подмножестве.

- Идея: построим касательные к уже готовой оболочке.
- Удалим все точки, попавшие внутрь угла, образованного касательными.



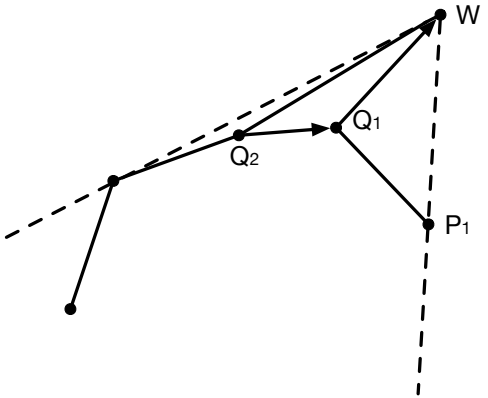
- Если удалять их в правильном направлении, операция займёт O(1).



- Правильно: двигаемся по направлению от P_1 с конца до точки первого касания.
- Удаляем с вершины либо вектора, либо стека, все, вошедшие в угол.
- Точка касания — предельная точка со следующим свойством: все точки после неё удаляются.
- Заметим: P_1 нужно удалить, если $\overrightarrow{(P_1, Q_1)} \times \overrightarrow{(P_1, W)}$ отрицательный для верхней оболочки и положительный для нижней.

Критерий удаления:

$$\overrightarrow{(P_1, Q_1)} \times \overrightarrow{(P_1, W)} < 0 \tag{2}$$



Асимптотика

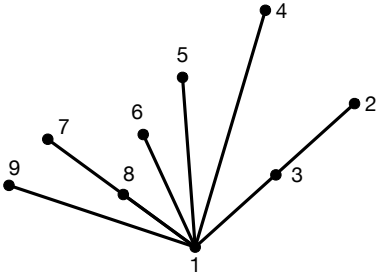
- Часть 1. Сортировка $O(N \log N)$.
- Часть 2. N раз добавление точки. При этом каждая точка удаляется не более одного раза. Амортизированная сложность части $O(N)$.

$$T = O(N \log N)$$

Алгоритм Грэхема

- Другое название — «сортировка по углам».

1. Найдём самую нижнюю точку P по координате y, при нескольких таких — и самую правую по координате x — как в алгоритме Джарвиса. Инвариант: точка принадлежит H. Положим её в стек.
2. Отсортируем точки по радиус-векторам от точки P до других.



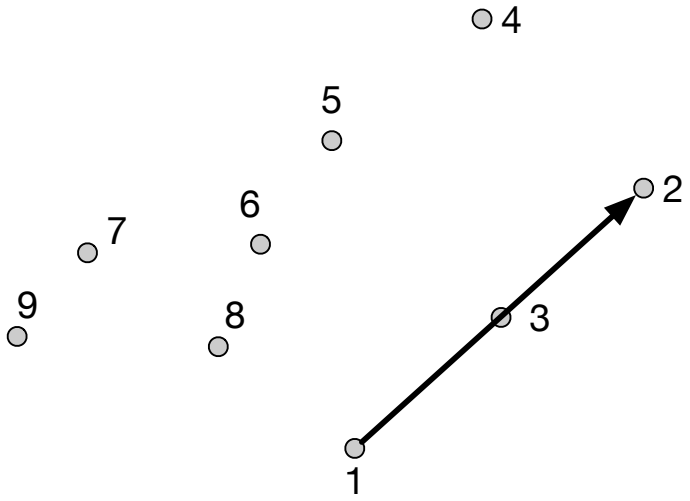
3. Если направления векторов совпадают — самым приоритетным будет самый длинный.
4. Выберем самую приоритетную точку из оставшегося множества W и исполняем операцию корректировки стека.
5. Повторяем пункт 4 пока имеются точки для обработки.

Алгоритм Грэхема: операция корректировки стека

a. Если в стеке менее двух элементов — кладём W в стек и заканчиваем корректировку.
b. Рассмотрим два верхних элемента стека — самый верхний P_1, далее P_2 а также новую точку W.
c. Если $\overrightarrow{(P_2, P_1)} \times \overrightarrow{(P_1, W)} > 0$, то кладём в стек W и заканчиваем корректировку.
d. Если $\overrightarrow{(P_2, P_1)} \times \overrightarrow{(P_1, W)} = 0$, то в из вершин P_1 и P_2 остаётся та, которая имеет наибольшее расстояние до W.
e. Если $\overrightarrow{(P_2, P_1)} \times \overrightarrow{(P_1, W)} < 0$, то извлекаем из стека P_1 и повторяем операцию.

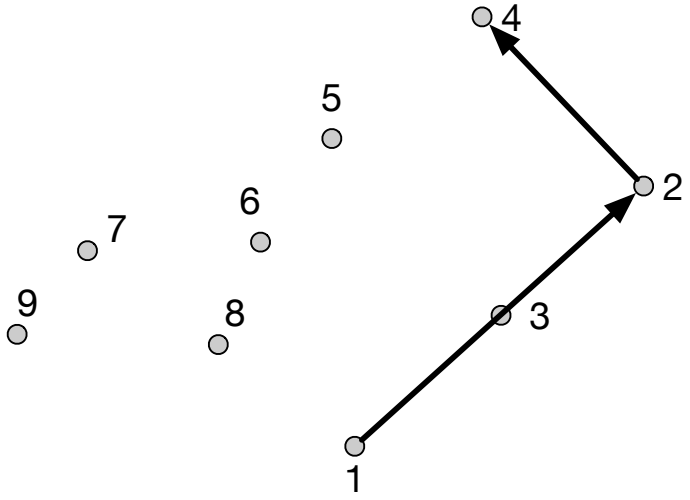
Алгоритм Грэхема

В стеке 1 и 2. Появление 3 к изменению ситуации не привело — случай d операции корректировки.



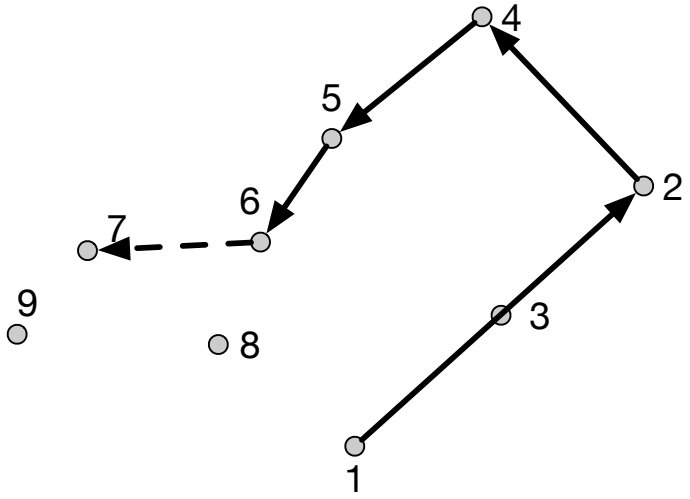
Алгоритм Грэхема

Вершина 4 просто добавляется в стек.



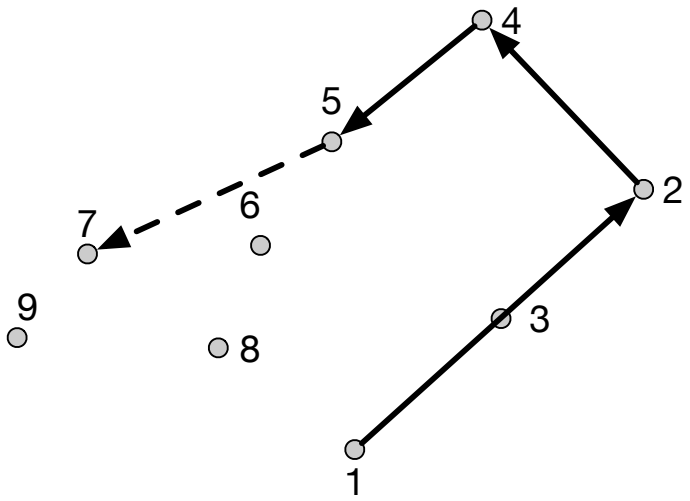
Алгоритм Грэхема

Вершины 5 и 6 были добавлены в стек, пришла вершина 7. Срабатывает случай e — из стека удаляем вершину 6, не кладя туда пока ничего. Держим вершину 7 в руке.



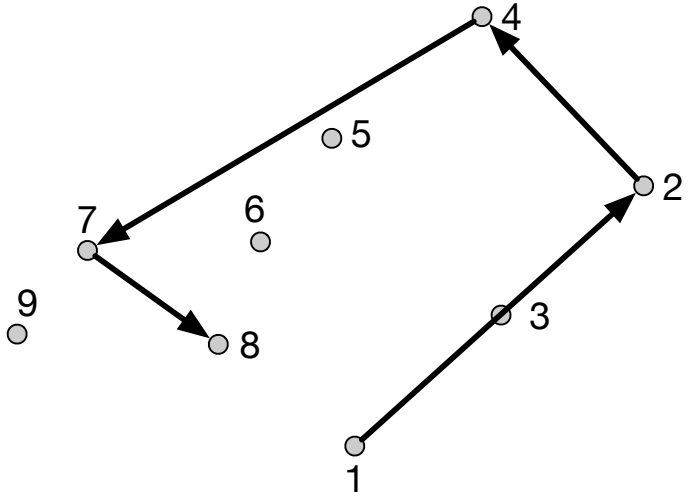
Алгоритм Грэхема

Повторение корректировки сейчас удалит вершину 5, после чего вершину 7 можно помещать в стек.



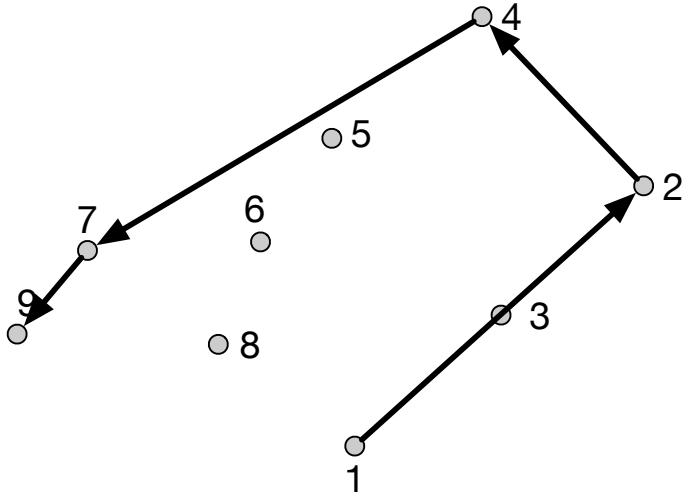
Алгоритм Грэхема

Добавлена вершина 8.



Алгоритм Грэхема

После появления вершины 9 вершина 8 убрана из стека. Алгоритм закончен.



Алгоритм Грэхема: замечания по реализации

- Если задача состоит только в построении выпуклой оболочки — короче всего код в алгоритме Грэхема.
- При сравнении расстояний корни не извлекаем.
- Алгоритм Эндрю, хотя и более длинный, позволяет строить оболочку динамически.

Динамическая выпуклая оболочка (ДВО)

Задача 2. На вход алгоритма подаются запросы двух типов: добавить точку и вывести границу выпуклой оболочки множества. S расширяется и каждый раз нужно перестраивать оболочку.

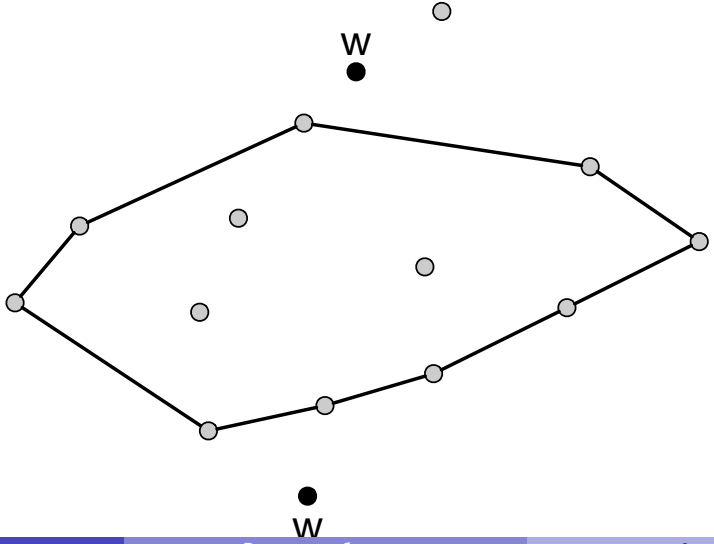
Решение: С точки зрения двух подоболочек, верхней и нижней, используемых в алгоритме Эндрю, новая точка W может прийти в 5 разных мест:

1. слева — $W_x < P_{minx}$ — чисто алгоритм Эндрю;
2. справа — $W_x > P_{maxx}$ — развёрнутый алгоритм Эндрю;
3. над верхней подоболочкой;
4. под нижней пдоболочкой;
5. между подоболочками или на границу.

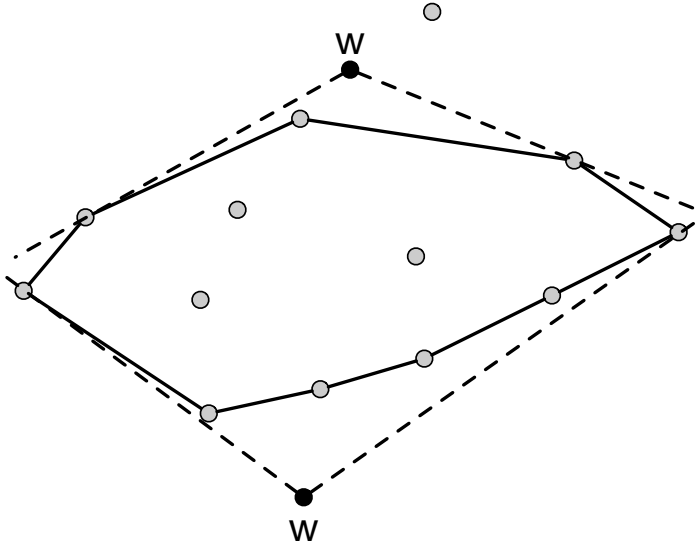
- Как понять, куда попала точка, если она не слева и не справа?

ДВО

- Уже имеется построенная ВО и *упорядоченные* множества `upper` и `lower`.
- Бинарным поиском находим ребро, над которым или под которым находится W.



- Проведём касательные от W к $upper$.
- Зная касательные, можем удалить все правые из левого куска и все левые из правого.



ДВО: технические вопросы

От структуры данных, хранящей вершины, требуется следующее:

- добавить вершину;
- удалить вершину;
- переместиться на следующую;
- переместиться на предыдущую.

Первые две операции для массива или вектора $O(N)$. Последние две — быстрые, $O(1)$. Итого для вставки N вершин $O(N^2)$.

Для сбалансированного дерева все операции за $O(\log N)$. N операций амортизированно даст $N \log N$. Хорошо хранить вершины в $std :: set$. Имеются итераторы $++$ и $--$, $erase$, $insert$.

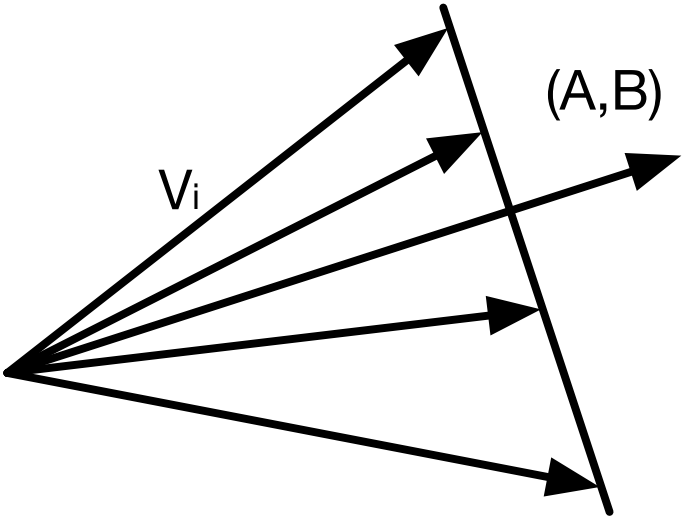
Convex Hull Trick.

Задача 3. Имеется массив v пар (x, y) и заданные (a, b).
Нужно для каждой пары (a, b) давать ответ: какая из пар массива даёт максимальную сумму $ax_i + by_i$.

Можно каждый запрос исполнять за $O(N)$. Но запросов много.

Convex Hull Trick

Из ЛА: ГМТ таких точек, что $(x, y) \cdot (a, b) = const$ есть прямая, ортогональная (a, b).

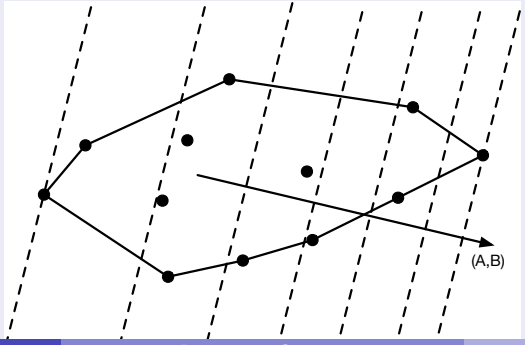


Lemma

Максимум скалярного произведения вектора $\overrightarrow{v} = \overrightarrow{(A, B)}$ на радиус-векторы, концы которых образуют множество S, достигается на границе выпуклой оболочки H, если $H = \mathrm{conv}(S)$.

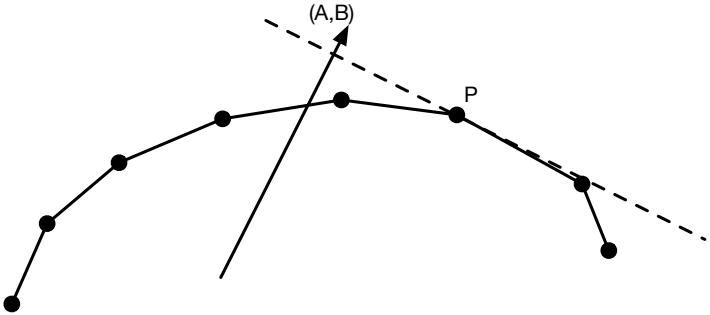
Доказательство.

Пусть имеется $\overrightarrow{(a, b)}$. Возьмём прямую и будем двигать её вдоль направления (a, b), чтобы найти экстремальную точку. Самая крайняя точка — одна из вершин ВО.



Алгоритм Convex Hull Trick

- Пусть $b > 0$. вектор смотрит вверх. Рассмотрим $upper-$ часть S.



- Рисуем все ортогональные (a, b) прямые и смотрим на их проекцию на прямую, параллельную вектору. Это — унимодальная кусочно-непрерывная функция с максимумом в точке P.
- Для $b > 0$ и $upper$ она сначала возрастает, потом убывает.

Алгоритм Convex Hull Trick

- Максимум находится бинарным поиском по производной (лучше даже можно тернарным по значениям).
- Если для точек m и $m+1$ скалярное произведение возрастает, то левую границу переносим вправо.
- Если убывает — правую границу переносим влево.
- Равны — нашли нужное ребро.
- Нижняя область — аналогично, только ищем минимум.

```
while (r - l > 1) {
    m = (r+l)/2;
    x = dot(pm, ab);
    y = dot(pm+1,ab);
    if (x < y) l = m;
    if (x > y) r = m;
    if (x == y) // ans
 }
```

Сложность алгоритма Convex Hull Trick

- Построение выпуклой оболочки $H = \mathrm{conv}(S)$ есть $O(N \log N)$ (предподсчёт).
- Поиск требуемой вершины в H есть $O(\log |H|)$ на одну вершину.

Динамический Convex Hull Trick

.

Задача 4. Повторяем условие предыдущей задачи и включаем операцию добавления пары.

Решение:

- От структуры данных, хранящей вершины, теперь требуется следующее:
  - найти вершину по номеру (для двоичного поиска по вершинам);
  - добавить вершину с заданным номером;
  - удалить вершину с заданным номером.
- `std::set` теперь не подходит.
- Используем декартово или splay дерево.
- Стоимость каждой операции $O(\log(N))$.
- Для запроса вставки потребуется $O(\log(N))$ таких операций.
- Для запроса поиска потребуется $O(\log(N))$ таких операций.

$$T = O(\log^2 N)$$