SPb HSE, 1 курс, осень 2021/22
Практика по алгоритмам #6

Сортировки, точки-экстремумы

8 октября

Собрано 8 октября 2021 г. в 20:19

Содержание

Сортировки и поиск

1. **Пирожки**

   Робот Иван Семеныч пробует пирожки. Содержимое пирожков делится на три типа. Всего пирожков n. Каждый пирожок можно попробовать не более одного раза. Пирожки можно менять местами. Память у робота маленькая, $\mathcal{O}(\log n)$ бит.

   Помогите Ивану Семенычу отсортировать пирожки по типу: сначала первый, потом второй, потом третий. Сортировка должна работать за линейное время.

2. **Оптимизация скалярного произведения**

   Даны два массива a и b одинаковой длины. Найти такую перестановку p, что

   a) $\sum\limits_{i=1}^{n} a_{p_i} b_i \to \max$;

   b) $\sum\limits_{i=1}^{n} a_{p_i} b_i \to \min$.

   c) Выбрать массив p из k различных чисел от 1 до n так, чтобы $\sum_{i=1}^{k} a_{p_i} b_{p_i} \to \max$.

3. **Умный бинпоиск**

   Пусть мы ищем в сортированном массиве длины n такое k: $a[k] = x$, тогда бинпоиск работает за $\mathcal{O}(\log n)$, а линейный поиск за $\mathcal{O}(k)$, что может быть меньше. Научите бинпоиск находить k за $\mathcal{O}(\log k)$!

4. **Корни многочлена**

   a) Найти корень многочлена нечетной степени за $\mathcal{O}(n \log M)$. n – степень многочлена.

   b) Зная все корни производной, найти все вещественные корни многочлена за $\mathcal{O}(n^2 \log M)$.

   c) Найти все вещественные корни многочлена за $\mathcal{O}(n^3 \log M)$.

5. **Нижняя оценка на два бинпоиска**

   Докажите, что нельзя сделать и `lower_bound`, и `upper_bound` одновременно, используя в худшем случае меньше чем $2 \log_2 n + \mathcal{O}(1)$ сравнений.

Задачи про поиск точки

Каждую из предложенных задач можно решить за время $\mathcal{O}(n)$, решения за линию от n на полилогарифм тоже приветствуются. Для разминки продифференцируйте $x^7 + 4x^3$.

1. **Точки на прямой**

   Даны n точек на прямой x_i. Найти точку x^*:

   a) $\sum\limits_i (x_i - x^*)^2 \to \min$

   b) $\sum\limits_i |x_i - x^*| \to \min$

   c) $\max\limits_i |x_i - x^*| \to \min$

   d) $\max\limits_i (x_i - x^*)^2 \to \min$

Для разминки найдите за $\mathcal{O}(n)$ площадь пересечения n прямоугольников со сторонами параллельными осям координат, поймите как выглядит на плоскости множество точек $\{\max(|x|, |y|) \leqslant d\}$, как оно выглядит после поворота на 45, опишите новую фигуру формулой.

2. **Точки на плоскости**

   Даны n точек на плоскости (x_i, y_i). Найти точку (x^*, y^*):

   a) $\max\limits_i \Big[\max(|x_i - x^*|, |y_i - y^*|)\Big] \to \min$

   b) $\max\limits_i \Big[|x_i - x^*| + |y_i - y^*|\Big] \to \min$

   c) $\sum\limits_i \Big[(x_i - x^*)^2 + (y_i - y^*)^2\Big] \to \min$

   d) $\sum\limits_i \Big[|x_i - x^*| + |y_i - y^*|\Big] \to \min$

   e) $\sum\limits_i \Big[\max(|x_i - x^*|, |y_i - y^*|)\Big] \to \min$

3. **Точки с весами**

   На прямой расположено n точек p_1, p_2, …, p_n. Каждая точка имеет вес $w_i \geqslant 0$. Требуется найти точку q: $\sum_i \Big[w_i \cdot |p_i - q|\Big] \to \min$.

   `(*)` Решить за $\mathcal{O}(n)$.

Дополнительные задачи

1. **Еще ускорение `SiftDown`**

   Модифицируйте операцию `SiftDown` для бинарной кучи так, чтобы она по-прежнему работала за $\mathcal{O}(\log n)$, но при этом делала лишь $\log_2 n + \mathcal{O}(\log^* n)$ сравнений.

2. **Convex hull**

   Доказать, что, если мы умеем строить выпуклую оболочку n точек на плоскости за время $f(n)$, мы умеем сортировать n целых чисел за $f(n)$.

3. **Hard K-best max** $(\sum_{i=1}^k a_{p_i})(\sum_{i=1}^k b_{p_i}) \to \max$ $(a_i, b_i > 0)$.

4. **Hard K-best min** $(\sum_{i=1}^k a_{p_i})(\sum_{i=1}^k b_{p_i}) \to \min$ $(a_i, b_i > 0)$.

5. **Точки на плоскости**

   f) $\max\limits_i \Big[(x_i - x^*)^2 + (y_i - y^*)^2\Big] \to \min$

   g) $\sum\limits_i \sqrt{(x_i - x^*)^2 + (y_i - y^*)^2} \to \min$

6. **Оптимизация** $\min \times \sum$

   Найти отрезок массива, на котором $\min \times \sum$ максимально. $\mathcal{O}(n)$.

   a) Все числа положительны.

   b) Числа целые, 64-битные. Решение с использованием минимума на отрезке за $\mathcal{O}(1)$.

   c) Числа целые, 64-битные. Простое решение со стеком.

7. **Коровы и стойла #2**

   То же самое, только теперь стойла стоят на окружности. Решение за $\mathcal{O}(n \log MAX)$.

Разбор задач практики

1. **Пирожки**

   Краткое условие: отсортировать массив одним проходом, используя $\mathcal{O}(1)$ доп.памяти.
   Сортировка для k типов пирожков за $\mathcal{O}(nk)$ с $k \log_2 n$ бит доп.памяти на примере k=3:

   Изначально $i_1 = i_2 = i_3 = 0$.
   Поддержим инвариант: $[0, i_1)$ – единицы, $[i_1, i_2)$ – двойки, $[i_2, i_3)$ – тройки.
   Двигаем i_3, `swap`-ами ставим $a[i_3]$ в нужное место.

```
for (int i1 = 0, i2 = 0, i3 = 0; i3 < n; ++i3)
  if (a[i3] != 3) // a[0,i1)=1 a[i1,i2)=2 a[i2,i3)=3
    swap(a[i3], a[i2]);
    if (a[i2] != 2)
      swap(a[i2], a[i1++]);
    ++i2;
```

2. **Оптимизация скалярного произведения**

   ab) **Идея:** хотим чтобы максимальный a_i стоял рядом с максимальным b_j.
   Сортируем пары $A_i = (a_i, i)$ по возрастанию.
   Если хотим max, сортируем пары $B_j = (b_j, j)$ по возрастанию, иначе по убыванию.
   Соответствие: $A_i \leftrightarrow B_i$. Ответ: $p[A_i\texttt{.second}] = B_i\texttt{.second}$.
   Доказательство (транс-неравенство): пробуем менять местами два элемента a,
   $a_1b_1 + a_2b_2 \leqslant a_2b_1 + a_1b_2 \Leftrightarrow a_1(b_1{-}b_2) + a_2(b_2{-}b_1) \leqslant 0 \Leftrightarrow (a_1{-}a_2)(b_1{-}b_2) \leqslant 0$
   $\Leftarrow (a_1 > a_2 \wedge b_1 \leqslant b_2)$. Вывод: если b отсортирован и a не отсортирован, можно поменять местами два элемента a, ответ не ухудшится, число инверсий уменьшится.

   c) $\sum_{i=1}^{k} a_{p_i} b_{p_i} \to \max$.
   Нужна max сумма из k элементов массива $c_i = a_i b_i$, берем k максимальных c_i.
   Это можно сделать за $\mathcal{O}(n \log n)$ сортировкой,
   можно за $\mathcal{O}(n + k \log n)$ построением кучи и извлечением k максимумов,
   можно за $\mathcal{O}(n)$ поиском $(n{-}k)$-й статистики и выбором всех не меньших ее элементов.

3. **Умный бинпоиск**

   `R = 1; while (a[R] < x) R *= 2;` за $\mathcal{O}(\log k)$ даст нам $R < 2k$.
   Затем фигачим бинпоиск на $[0, R]$ за $\mathcal{O}(\log k)$

4. **Корни многочлена**

   a) Пусть для определенности старший коэффициент многочлена будет положительный.
   Тогда $f(x) \to +\infty$ при $x \to +\infty$, $f(x) \to -\infty$ при $x \to -\infty$.
   Инвариант бинпоиска: $f(l) < 0$, $f(r) \geqslant 0 \Rightarrow$ в $[l, r]$ всегда есть корень.
   На каждом шаге бинпоиска вычисляем $f(m)$ за $\Theta(n)$, итого $\mathcal{O}(n \log M)$.
   Как найти начальные значения l и r?
   `r = 1; l = -1; while (f(r) < 0) r *= 2; while (f(l) >= 0) l *= 2;`

   b) Между соседними корнями производной функция монотонна, т.к. производная не меняет знак $\Rightarrow$ там не больше одного корня и его можно найти бинпоиском. Еще могут быть корни справа от max корня производной и слева от min корня (+2 отрезка на рассмотрение).
   Степень производной многочлена равна $n{-}1 \Rightarrow$ корней у неё не больше $n{-}1$.
   Время работы: $\mathcal{O}(n)$ отрезков, на каждом бинпоиск за $\mathcal{O}(n \log M)$, итого $\mathcal{O}(n^2 \log M)$.

c) Если $n = 1$, найдём корень за $\mathcal{O}(1)$ (линейное уравнение), иначе посчитаем производную, рекурсивным вызовом найдём её корни, воспользуемся решением (b).
Как считать производную? $P(x) = a_n x^n + \cdots + a_1 x + a_0 \Rightarrow P'(x) = n a_n x^{n-1} + \cdots + a_1$.
Время работы: $T(n) = T(n-1) + n^2 \log M = \mathcal{O}(n^3 \log M)$.

Замечение для самых любопытных.
На практике при наличии кратных корней этот метод не работает из-за погрешности.
Пример: $P(x) = (x-1)^{10} \Rightarrow P(1.01) = 10^{-20}$, что не отличимо от нуля даже в `long double`.
Для борьбы с кратными корнями задачу делят на две – найти кратные корни, как корни $G(x) = gcd(P(x), P'(x))$, найти оставшиеся, как корни $P(x)/G(x)$.

4.1. Задачи про поиск точки

1. **Точки на прямой**
   a) $\sum_i (x_i - x^*)^2 \to \min$.
   Минимум в нуле производной. $\sum 2(x_i - x^*) = 0 \Rightarrow x^* = \frac{\sum x_i}{n}$.
   b) $\sum_i |x_i - x^*| \to \min$.
   Идея: пусть ответ в точке x, подвигаем чуть-чуть x, улучшился ли ответ?
   Рассмотрим какой-нибудь x. Пусть слева от него l точек, справа r.
   Если сдвинем x на ε вправо, не перескочив ни через одну точку, то $\sum_i |x_i - x|$ изменится на $(l-r)\varepsilon$. Если $l < r$, движение вправо улучшает. Если $l > r$, движение влево улучшает.
   Итого брать нужно медиану, $\mathcal{O}(n)$.
   Если n нечётно, медиана $= x_{\lfloor n/2 \rfloor}$ в отсортированном порядке.
   Если n чётно, медиан две, можно взять любую точку между ними.
   c) $\max_i |x_i - x^*| \to \min$.
   Максимум либо в самой левой, либо в самой правой точке, поэтому $x^* = \frac{x_1 + x_n}{2}$.
   d) $\max_i (x_i - x^*)^2 \to \min$.
   Задача эквивалентна максимизации $|x_i - x^*| \Rightarrow x^* = \frac{x_1 + x_n}{2}$.

2. **Точки на плоскости**
   a) $\max_i \Big[\max(|x_i - x^*|, |y_i - y^*|)\Big] \to \min$.
   $\max_i \Big[\max(|x_i - x^*|, |y_i - y^*|)\Big] = \max\Big[\max_i |x_i - x^*|, \max_i |y_i - y^*|\Big]$
   $\Rightarrow$ координаты x^* и y^* не зависят друг от друга $\Rightarrow x^* = \frac{x_{\min} + x_{\max}}{2}, y^* = \frac{y_{\min} + y_{\max}}{2}$.
   b) $\max_i \Big[|x_i - x^*| + |y_i - y^*|\Big] \to \min$.
   Предварительные рассуждения.
   Бинпоиск по ответу. Внутри бинпоиска нужно уметь проверять, есть ли точка на расстоянии $\leqslant m$ от каждой (x_i, y_i). Для каждой (x_i, y_i) рассмотрим множество точек, находящихся на расстоянии $\leqslant m$ по метрике $|x - x_i| + |y - y_i|$.
   Пересекаем эти множества, проверяем, пусто ли пересечение.
   Каждое такое множество – квадрат, повернутый на 45° к осям координат.
   Пересечение множеств – прямоугольник, повернутый на 45°.

   Из разминки вы умеете пересекать квадраты со сторонами, параллельными осям координат. Повернём плоскость на 45°, найдём ответ, повернём его обратно.

Прямое преобразование: $(z_i, t_i) = (x_i - y_i, y_i + x_i)$.
Обратное преобразование: $(x^*, y^*) = (\frac{z^*+t^*}{2}, \frac{t^*-z^*}{2})$.

Откуда прямая формула: поворот точки (x, y) вокруг $(0, 0)$ на $90°$ – это $(-y, x)$.
Поворот на $45°$ – биссектриса между (x, y) и $(-y, x)$, биссектриса получается сложением векторов. Преобразование $(x_i, y_i) \to (x_i - y_i, y_i + x_i)$ – это не только поворот, но и растяжением плоскости в $\sqrt{2}$ раз. Растяжение не влияет на максимизацию.

Заметим, что после поворота и бинпоиск не нужен, получившиеся квадраты – это множества, находящиеся на фиксированном расстоянии от точки по метрике $\max(|z|, |t|)$.
Формально: $\max(|x-y|, |x+y|) = |x| + |y|$. То есть, мы свелись к предыдущей задаче:

$$\max_i\big(|x^*-x_i| + |y^*-y_i|\big) = \max_i\big(\max(|z^*-z_i|, |t^*-t_i|)\big)$$

Решение коротко.
Обозначим $z_i = x_i + y_i, t_i = x_i - y_i$.
Обозначим $z^* = x^* + y^*, t^* = x^* - y^*$.
Обозначим $dz_i = z_i - z^*,\ dt_i = t_i - t^*,\ dx_i = x_i - x^*,\ dy_i = y_i - y^*$.
Заметим, что $\max(|dz_i|, |dt_i|) = \max(|dx_i + dy_i|, |dx_i - dy_i|) = |dx_i| + |dy_i|$.
Тогда $\max_i\Big[|dx_i| + |dy_i|\Big] = \max_i\Big[\max(|dz_i|, |dt_i|)\Big]$.
Взяли решение (a) для для точек (z_i, t_i), получили $z^* = \frac{z_{\min}+z_{\max}}{2}, t^* = \frac{t_{\min}+t_{\max}}{2}$.
Перешли от (z, t) к (x, y): $x^* = \frac{z^*+t^*}{2}, y^* = \frac{z^*-t^*}{2}$.
Главная идея, запомните: $\max(|dz_i|, |dt_i|) = |dx_i| + |dy_i|$, оно же «поворот на $45°$».

Алгоритм за $\mathcal{O}(n)$.
1. $z_i = x_i + y_i, t_i = x_i - y_i$
2. $z^* = \frac{z_{\min}+z_{\max}}{2}, t^* = \frac{t_{\min}+t_{\max}}{2}$
3. $x^* = \frac{z^*+t^*}{2}, y^* = \frac{z^*-t^*}{2}$

c) $\sum_i \Big[(x_i - x^*)^2 + (y_i - y^*)^2\Big] \to \min$.

Сумма распадается на независимые суммы по каждой координате. $x^* = \frac{\sum x_i}{n}, y^* = \frac{\sum y_i}{n}$.

d) $\sum_i \Big[|x_i - x^*| + |y_i - y^*|\Big] \to \min$.

Независимые суммы по каждой координате. $x^* = x_{\text{median}}, y^* = y_{\text{median}}$.

e) $\sum_i \Big[\max(|x_i - x^*|, |y_i - y^*|)\Big] \to \min$.

1. Повернуть плоскость на $45°$, получить задачу (d).
2. Взять решение задачи (d), повернуть полученный ответ обратно.

3. **Точки с весами**

Идея: пусть ответ в точке x, подвигаем чуть-чуть x, улучшился ли ответ?

Рассмотрим какой-нибудь x, пусть суммарный вес точек слева от x равен w_l, справа w_r.
Если сдвинем x на ε вправо, не перескочив ни через одну точку, то $\sum w_i|p_i - x|$ изменится на $(w_l - w_r)\varepsilon$. Если $w_l < w_r$, движение вправо улучшает. Если $w_l > w_r$, движение влево улучшает.

$\Rightarrow$ нам подойдет самая левая точка такая, что сумма весов слева до неё (включая её саму) достигает половины от суммарного веса. Так называемая взвешенная медиана.

Найти точку можно за $\mathcal{O}(n)$ тем же алгоритмом (одноветочный Quick Sort), который ищет медиану. Внутри Qucik Sort теперь нужно смотреть не на число элементов, которые меньше разбивающего, а на их суммарный вес.

4.2. Дополнительные задачи

1. **Еще ускорение `SiftDown`**

   $\log_2 n + \mathcal{O}(\log^* n)$ сравнений.
   Пусть мин куча. Спустимся на $\log n - \log\log n$ вниз, попали в i, сравним $h[i]$ с нашим x.
   Если $h[i] > x$, то бинпоиском за $\log\log n$ побеждаем.
   Если $h[i] \leqslant x$, то мы за $(\log n - \log\log n) + 1$ сравнений свели задачу "спуститься вниз на $\log n$" к такой же, только спускаться осталось лишь на $\log\log n$, продолжим спуск.
   $(\log n - \log\log n + 1) + (\log\log n - \log\log\log n + 1) + \cdots = \log n + \mathcal{O}(\log^*)$.

2. **Convex hull**

   Пусть нужно отсортировать числа x_i. Возьмём точки (x_i, x_i^2).
   Алгоритм построения выпуклой оболочки отсортирует все точки.

3. **Hard K-best max**: $(\sum_{i=1}^{k} a_{p_i})(\sum_{i=1}^{k} b_{p_i}) \to \max$.

   Пусть $a_i + b_i = C$ (константа). Тогда какие бы k пар мы не выбрали, будет верно $\sum a_i + \sum b_i = kC$. Произведение максимально, если $\sum a_i = \frac{kC}{2}$. Т.е. если существуют такие k a-шек, мы должны их найти. Это задача о рюкзаке.

4. **Hard K-best min**: $(\sum_{i=1}^{k} a_{p_i})(\sum_{i=1}^{k} b_{p_i}) \to \min$

   ?

5. **Точки на плоскости**

   f) $\max_i \left[(x_i - x^*)^2 + (y_i - y^*)^2\right] \to \min$.
      Два вложенных тернарных поиска, по y и по x. $\mathcal{O}(n \log^2 M)$.
      Существует простой вероятностный алгоритм за $\mathcal{O}(n)$ в среднем.
      А есть даже детерминированный за $\mathcal{O}(n)$.

   g) $\sum_i \sqrt{(x_i - x^*)^2 + (y_i - y^*)^2} \to \min$.
      Два вложенных тернарных поиска, по y и по x. $\mathcal{O}(n \log^2 M)$.

6. **Оптимизация** $\min \times \sum$

   Для каждого i ищем отрезок $[L_i, R_i]$с максимальной суммой, в котором $a_i = \min$.
   Поиск L_i и R_i – независимые задачи, решаются одинаково. Разберем поиск L_i.
   Найдем за $\mathcal{O}(n)$ для каждого элемента ближайший слева меньший его l_i.

   a) Если все числа положительны, то $L_i = l_i + 1$.
   b) $L_i = \max_{j \in (l_i, i]} \big(a_i \cdot (\text{pref}_{i+1} - \text{pref}_j)\big)$. То есть, если $a_i > 0$, $\min \text{pref}_j$, иначе $\max \text{pref}_j$.
   c) $\exists$ структура данных, дающая min/max на полуинтервале $(l_i, i]$ за $\mathcal{O}(1)$.
      Можно проще: давайте по ходу вычисления l_i помнить min/max между соседними элементами в стеке. Тогда значение для $(l_i, i]$ будет на стеке, когда мы положим i.

7. **Коровы и стойла #2**

   Тот же бинпоиск, что и для обычных коров. Внутри бинпоиска нужно за $\mathcal{O}(n)$ проверять, что можно выбрать k точек на окружности на расстоянии $\geqslant m$.

   1. Отсортируем точки на окружности.
      Удвоим массив: $x[i+n] = x[i] + L$, где L – длина окружности.
   2. За $\mathcal{O}(n)$ двумя указателями $\forall i$ найдем $next[i] = \min j : x[j] \geqslant x[i] + m$.

3. Теперь для любой точки можно за $k-1$ прыжок $i \to next[i]$ проверить, можно ли взять её в ответ. Проверим точку $x[0]$.
4. a) Не зациклились за $k-1$ прыжок $\Rightarrow$ ок, расставили.
   b) Зациклились за $< k-1$ прыжков $\Rightarrow$ ни с какой начать нельзя.
   c) Зациклились ровно за $k-1$ прыжок $\Rightarrow$ окружность разбилась на $k-1$ часть, в одной из них $\leqslant \frac{2n}{k-1}$ точек. Проверим каждую из них за $\mathcal{O}(k)$.

Итого время работы $\mathcal{O}(n \log M)$.

Домашнее задание

5.1. Обязательная часть

1. **(3) Ближайшие к медиане**

   Дан массив `A[1..n]` из n различных чисел. Массив не обязательно отсортирован. Требуется найти k ближайших к медиане элементов за линейное время. Решить для двух метрик.

   a) **(1)** По позиции в отсортированном массиве.

   $$d(x, \texttt{median}) = |\texttt{pos}(x) - \texttt{pos}(\texttt{median})|,$$

   где $\texttt{pos}(x)$ — позиция элемента x в отсортированном массиве.

   b) **(2)** По значению.

   $$d(x, \texttt{median}) = |x - \texttt{median}|.$$

2. **(4) Поиск точки**

   a) **(1)** Даны n точек на прямой x_i с весами $w_i \geqslant 0$. Найти точку x^*:

   $\sum\limits_i \Big[w_i(x_i - x^*)^2\Big] \to \min$

   b) **(1)** Даны n точек на плоскости (x_i, y_i) с весами $w_i \geqslant 0$. Найти точку (x^*, y^*):

   $\sum\limits_i \Big[|x_i - x^*| + w_i|y_i - y^*|\Big] \to \min$

   c) **(2)** Даны n точек на плоскости (x_i, y_i) с весами $w_i \geqslant 0$. Найти точку (x^*, y^*):

   $\max\limits_i \Big[w_i(|x_i - x^*| + |y_i - y^*|)\Big] \to \min.$

   Дополнительный **(+1)** балл можно заработать, решив эту задачу $\mathcal{O}(\texttt{sort} + n)$.

3. **(3) Поиск двух точек**

   На прямой расположено n точек p_1, p_2, …, p_n. Каждая точка имеет вес $w_i \geqslant 0$.
   Выбрать две точки q_1, q_2: $\sum\limits_i \Big[w_i \cdot \min(|p_i - q_1|, |p_i - q_2|)\Big] \to \min.$

   Решение за $\mathcal{O}(n^2)$ даст один балл из трёх.
   Hint: перебирать можно не только координаты q_i.

4. **(3) Поиск многих статистик**

   Дан массив из n чисел и m чисел $k_1, k_2, \ldots k_m$,
   нужно за $\mathcal{O}(n \log m + m)$ для каждого i найти k_i-ую порядковую статистику.

5. **(2) Поиск статистики**

   Оцените сложность детерминированного алгоритма поиска k статистики, если в нем разбивать на группы не по 5 элементов, а по:

   a) **(1)** 7
   b) **(1)** 3

5.2. Дополнительная часть

1. **(2) Сортировка вещественных чисел**

   Даны n вещественных чисел из $[0, 1]$ в произвольном порядке. Представим, что они уже отсортированы и посмотрим на разности соседних чисел. Нужно найти максимальную такую разность за $\mathcal{O}(n)$.

2. **(3) Поиск центра**

   Даны n точек на плоскости (x_i, y_i) с весами $w_i \geqslant 0$. Найти точку (x^*, y^*):

   $\max\limits_i \Big[w_i((x_i - x^*)^2 + (y_i - y^*)^2)\Big] \to \min.$

   Будут оценены решения за линию на полилогарифм.

3. **(3) Поиск трех точек**

   На прямой расположено n точек p_1, p_2, $\ldots$, p_n. Каждая точка имеет вес $w_i \geqslant 0$.

   Выбрать три точки q_1, q_2, q_3: $\sum\limits_i \Big[w_i \cdot \min(|p_i - q_1|, |p_i - q_2|, |p_i - q_3|)\Big] \to \min.$

   Требуется решение за линию на полилогарифм.

4. **(2) Амортизированная вставка в кучу**

   Придумайте, как модифицировать обычную бинарную кучу так, чтобы `Add` работал за амортизированное $\mathcal{O}(\log \log n)$.

5. **(5) Нижняя оценка на построение кучи**

   Внимание: эту задачу можно сдавать **только до лекции**.

   Какое минимальное число сравнений нужно, чтобы построить обычную бинарную кучу? Докажите максимально точную нижнюю оценку. Для простоты вычислений предположим, что в массиве $n = 2^k - 1$ элементов.

   a) **(1)** Хотя бы $n - 1$ сравнений.
   b) **(1.5)** Хотя бы n сравнений для $n \geqslant 10$.
   c) **(3)** Оценка $1.3644 \cdot n + \mathcal{O}(1)$ без строгого доказательства.
   d) **(5)** Оценка $1.4999 \cdot n + \mathcal{O}(1)$ без строгого доказательства.