Задача 8 – гармоничная последовательность

Пусть x и y – два первых члена последовательности
Тогда последовательность имеет вид:
$x, y, (y - x), -x, -y, (x - y), x, y, \ldots$
Длина цикла = 6

Нужно минимизировать функцию
$|x - b_1| + |y - b_2| + |(y - x) - b_3| + |-x - b_4| + |-y - b_5| + |(x - y) - b_6| + \ldots$

Немного преобразуем:
$|x - b_1| + |y - b_2| + |(y - x) - b_3| + |x - (-b_4)| + |y - (-b_5)| + |(y - x) - (-b_6)| + \ldots$

Теперь все слагаемые можно разбить на три группы:

1. Вида |x – a|
2. Вида |y – b]
3. Вида |(y – x) – c|

Создадим три отдельных массива bx, by и byx для хранения значений вычитаемых элементов в этих группах, и отсортируем их по возрастанию

Заметим, что функция является выпуклой вниз и по x, и по y, то есть имеет один минимум

Это можно увидеть, например, из таких соображений.
Зафиксируем конкретное x. Тогда сумма по первой группе превратится в константу, а вторая и третья группы соединятся в одну со слагаемыми вида |y - b|. Отсортируем их по возрастанию b:
|y – b1| + |y – b2| +|y – b2| + …, где b1 <= b2 <= b3 ….

Как изменится значение функции при увеличении y на 1? Представим числовую ось, на ней отмечены точки b1, b2,… и точка y, которая переместилась на 1 вправо. При этом к расстоянию до всех точек слева (bi <= y) добавится единица, для всех точек справа (by > y) – вычтется единица:

delta = nLeft – nRight.

При движении y слева направо сначала nLeft < nRight, но с некоторого момента станет nLeft > nRight. То есть функция сначала убывает, потом возрастает.
Для y рассуждения будут аналогичны.

Тогда вариант решения 1: используем какой-нибудь численный метод оптимизации. Например:

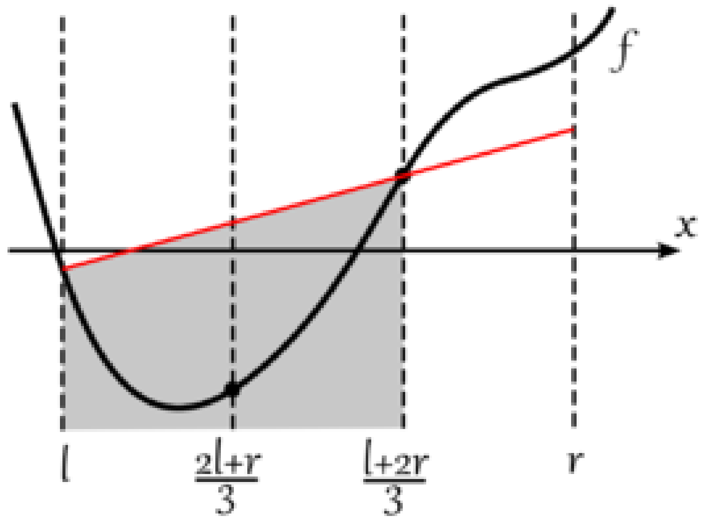
- двумерный троичный поиск
- метод градиентного спуска
- и др.

Такое решение способно набрать 100 баллов.

Разберём решение с помощью двумерного троичного поиска.

Вначале рассмотрим одномерный троичный поиск

Пусть функция f на отрезке [l, r] имеет один минимум, и нужно его найти



Посчитаем значения функции в точках a = l + (r – l) / 3, b = r – (r – l)/3
Если f(a) < f(b), то xmin Є [l, b], иначе xmin Є [a, r].
Изменяем границы поиска и дальше повторяем аналогично.

Двумерный троичный поиск - просто два вложенных одномерных. Значение x ищет внешний троичный поиск. При каждом фиксированном x задача превращается в одномерную, и значение y ищет внутренний троичный поиск.

Как быстро вычислять функцию f в точке (x, y)?

Можно воспользоваться префиксными и суффиксными суммами.
Массив префиксных сумм для некоторого массива v устроен так:
pref[i] = v[0] + v[1] + … + v[i]
Его несложно построить за линейное время.
Аналогично, массив суффиксных сумм: suf[i] = v[i] + v[i + 1] +… + v[n]

Построим следующие массивы:
prefx и sufx – для массива bx (группа слагаемых вида $|x – a_i$)
prefy и sufy – для массива by (группа слагаемых вида $|y – b_i$)
prefyx и sufyx – для массива byx (группа слагаемых вида $|(y – x) – c_i|$

Чтобы найти сумму по первой группе, ищем бинарным поиском в bx
такую позицию pos, что слева всем элементы < x, справа - >= x.
Сумма = x * pos - prefx[pos - 1] + sufx[pos] - x * (bx.size() - pos);
Частные случаи: все элементы меньше x, все больше x.
По остальным двум группам – аналогично.

Примечание: можно обойтись только префиксными суммами, так как суффиксные через них выражаются

Вопрос: какие брать начальные границы отрезка?
Достаточно от -2*10^9 до 2*10^9 (обоснование будет далее).

```
// троичный поиск по первому измерению
 long long xLeft = -2E9, xRight = 2E9;
 while (xRight > xLeft + 3) {
   long long d = (xRight - xLeft) / 3;
   long long m1 = xLeft + d;
   long long m2 = xRight - d;
   if (f(m1) >= f(m2))
     xLeft = m1;
   else
     xRight = m2;
 }
 long long fMin = f(xLeft);
 for (long long x=xLeft + 1; x<=xRight; x++)
   fMin = std::min(fMin, f(x));
```

```
// троичный поиск по второму измерению
long long f(long long x) {
  long long yLeft = -2E9, yRight = 2E9;
  while (yRight >= yLeft + 3) {
    long long d = (yRight - yLeft) / 3;
    long long m1 = yLeft + d;
    long long m2 = yRight - d;
    if (g(x, m1) >= g(x, m2))
      yLeft = m1;
    else
      yRight = m2;
  }
  long long gMin = g(x, yLeft);
  for (long long y = yLeft + 1; y <= yRight; y++)
    gMin = std::min(gMin, g(x, y));
  return gMin;
}
```

Другие методы решения

Утверждение. Существует пара (x, y), которая минимизирует сумму модулей, такая, что как минимум в двух из трех групп значения хотя бы одного модуля равны нулю.

Доказательство. Вначале докажем, что хотя бы один модуль равен 0. Пусть минимум достигается в точке (x, y), и ни один модуль не равен 0. Увеличим x на 1. Это поменяет сумму группы 1 на nLeft1 - nRight1, сумму группы 3 – на nRight3 - nLeft3. Общее изменение:

nLeft1 + nRight3 - nRight1 - nLeft3

Если же уменьшить x на 1, то получим противоположное изменение:

-(nLeft1 + nRight3 - nRight1 - nLeft3)

Отсюда следует, что мы можем уменьшить значение функции, и точка (x, y) не являлась точкой минимума.

Выполняя такие действия, на каком-то шаге мы получим нулевой модуль либо в первой, либо в третьей группе.

Теперь докажем исходное утверждение - что нулевые модули будут минимум в двух группах.

Пусть (x, y) - пара, на которой достигается минимум, и ровно один модуль обратился в ноль. Есть два варианта:
1). В ноль обратился модуль из первой группы (для второй группы рассуждения аналогичны).
2). В ноль обратился модуль из третьей группы

В первом случае просто повторяем действия с предыдущего слайда.

Во втором случае пробуем менять одновременно x и y так, чтобы $(y-x)$ не поменялось. Есть два
варианта:
$x=x+1$, $y=y+1$; или $x=x-1$, $y=y-1$;
Один из этих вариантов уменьшит значение функции, то есть точка (x, y) с одним нулевым модулей не была точкой минимума. Следовательно, должно быть нулевые модули минимум в двух группах.

Отсюда получаются такие частичные решения:

Решение за O(n^3): зафиксируем две из трех групп, переберем, какие именно модули равны нулю, получив тем самым x и y, восстановим последовательность, обновим ответ.

Решение за O(n^2log(n)): зафиксируем две из трех групп, переберем, какие именно модули равны нулю, , получив тем самым x и y, сосчитаем значение функции с помощью префиксных и суффиксных сумм, обновим ответ.

Утверждение. Пусть все числа в исходной последовательности по модулю не превосходят A. Существует правильный ответ, в котором одно из чисел x и y не превосходит по модулю A, а другое - 2A.

Доказательство.
Существует модуль вида $|x – a_i|$ или $|y – b_i|$, который равен нулю, из которого можно получить значение одной из переменных. Оно по модулю не превосходит A.

Значение второй переменной можно определить из уравнения вида $|(y – x) – c_j| = 0$ – оно не может быть по модулю больше 2A.

Подзадача 1: достаточно перебрать значения x и y в пределах от –20 до 20.

Подзадача 2: перебрать значения x и y в пределах от –200 до 200.

Подзадача 3: можно перебрать значения x и получить выражение в виде |y – y1| + |y - y1| + ..., которое необходимо минимизировать. Далее перебрать модуль, который будет равен нулю и, используя префиксные и суффиксные суммы, посчитать значение выражения.

Подзадача 4: можно перебрать, какие два модуля равны нулю и с помощью префиксных и суффиксных сумм посчитать ответ.

Решение подзадачи 5

Утверждение. Если для некоторого x найдено минимальное значение y, которое является оптимальным, то для большего x оптимальное значение y не может быть меньше найденного.

Набросок доказательства. Оптимальное значение y для фиксированного x не зависит от первой группы (её сумма - константа), а лишь от второй и третьей.
Пусть найдено оптимальное y для некоторого x. Увеличим x на 1.
Вторая группа не изменилась, то есть оптимальное y может измениться только из-за изменения третьей группы.
В третьей группе все числа увеличились на 1. Это могло вызвать как увеличение функции, так и её уменьшение.
Если функция увеличилась, то, возможно, y тоже стоит увеличить для компенсации (но это зависит ещё от второй группы).
Если функция уменьшилась, то вопрос – а почему для предыдущего x оптимальное y нельзя было ещё уменьшить? Ответ: потому что уменьшать y мешала вторая группа. Но она нам и сейчас мешает, поэтому y осталось оптимальным.

Полное решение:

Переберем две группы из трёх, в которых будут модули. равные нулю. Переберем, какой из модулей первой группы равен нулю. Будем поддерживать указатель на модуль из второй группы, при обнулении которого получается оптимальный ответ.

При переходе к следующему модулю первой группы двигаемся по модулям второй группы до тех пор, пока следующий модуль дает ответ лучше, чем текущий.

Для быстрого вычисления функции используем префиксные и суффиксные суммы.

Сложность - O(n log n).