функция $Penalty(A, B)$ на самом деле распадается на 2 части: зависимую только от A и зависимую только от B, с дополнительным ограничением из задачи $A \leq B$. Также отметим, что если A или B не совпадают с каким-либо из заданных значений скоростей, то, уменьшив их до ближайшей заданной скорости, ответ становится более оптимальным без изменения значения функции $Penalty$, поэтому A и B должны быть равны одной из скоростей из входных данных (или 0).

Теперь за линейное время пройдем массив данных от начала до конца и динамическим программированием посчитаем для каждой скорости V_i минимально возможное значение $Penalty1(A)$ при $A \leq V_i$, а потом за линейное время пройдем массив данных от конца в начало и динамическим программированием посчитаем для каждой скорости V_i минимально возможное значение $Penalty2(B)$ при $V_i \leq B$.

Когда части функции $Penalty(A, B)$ посчитаны для каждой возможной скорости, остается пройти массив скоростей еще раз и найти минимальное значение суммы минимально возможных $Penalty1(A)$ и $Penalty2(B)$ для каждой скорости. Найдя глобальный минимум, необходимо вывести соответствующие значения A и B, на которых достигаются минимумы $Penalty1(A)$ и $Penalty2(B)$.

K. King's Palace Garden

Очевидно, что область сада будет состоять из треугольников A_iOA_j, где O — положение дворца, а A_i и A_j — положения последовательных вершин сада. При этом, во-первых, отрезок A_iA_j виден из точки O, как идущий справа налево (т.е. переход от направления OA_i к OA_j идет против часовой стрелки), а во-вторых, внутри треугольника нет ни одного баобаба. Построим ориентированный граф G, вершинами которого будут допустимые треугольники AOB, а ребра будут идти из вершины AOB в COD тогда и только тогда, когда B и C совпадают, а переход от направления к AB к направлению CD идёт против часовой стрелки. Задача сводится к тому, чтобы найти в графе цикл, который обходит точку O ровно один раз, а сумма площадей его вершин максимальна.

Отсортируем точки A_i по направлениям OA_i. Порядок будет циклическим, т.е. во всех случаях мы считаем, что после точки A_k идут $A_{k+1}, A_{k+2}, \ldots, A_n, A_1, \ldots, A_{k-1}$.

Чтобы для точки A_k определить допустимые треугольники A_kOA_l, надо выбрать точки A_l, для которых направление A_kA_l находится против часовой стрелки от направлений $A_kA_{k+1}, ..., A_kA_{l-1}$, но по часовой стрелке от A_kO. Назовём множество вершин A_kOA_l для конкретного значения k блоком вершин G_k.

Если в графе G есть ребро из A_pOA_q в A_qOA_r, то в нем есть и все ребра из A_pOA_q в A_qOA_s для всех A_s, идущих после A_r (считая от A_q). Это нам пригодится при поиске цикла с наибольшей площадью. Второй факт, который нам понадобится — что в любом цикле найдется точка с минимальной координатой y. Поэтому нам достаточно перебрать все точки A_k, y-координата которых меньше, чем y-координата точки O, и рассмотреть циклы, которые начинаются с вершин A_kOA_l, для которых направление A_kA_l идет в направлении $(1, 0)$ или против часовой стрелки от него, и заканчиваются в одной из вершин A_mOA_k, для которых направление A_mA_k идет по часовой стрелке от $(1, 0)$. Кроме того, в цикле не должно быть вершин A_pOA_q, для которых точка A_k лежит между A_p и A_q (в порядке их нумерации).

Задача решается методом динамического программирования. Пусть A_k — предполагаемая точка цикла с минимальным y. Для каждой вершины A_pOA_q будем искать максимальную площадь вершин на путях от блока G_k, которые можно продолжить этой вершиной A_pOA_q. В начале выберем первую вершину A_kOA_l, для которой направление A_kA_l идёт против часовой стрелки от $(1, 0)$, и укажем, что для неё площадь равна $S_{kl} = 0$. Для всех остальных вершин графа установим площадь -1. Теперь для каждого блока от G_k до G_{k-1} выполним следующую процедуру. Пусть это блок G_p. Будем перебирать вершины A_pOA_q пока не найдём вершину, для которой $S_{pq} \leq 0$. Теперь для всех вершин A_pOA_r, начиная с A_pOA_q, берём максимум S площади S_{pr_1} для r_1 от p до r. В блоке G_r находим первую вершину A_rOA_s, в которую ведёт ребро из A_pOA_r. Находим площадь $S_1 = S + 2 \cdot S(A_pOA_r)$. Поскольку координаты всех вершин целые, эта площадь тоже будет целым 64-битным числом. Дальше есть три варианта. Если $s = k$ и направление A_rA_k идет по часовой стрелке от $(1, 0)$, то цикл замкнулся, мы проверяем, является ли площадь максимальной из уже найденных, и если да — запоминаем эту площадь и этот цикл. Если s идёт раньше, чем k (считая от p), присваиваем вершине A_rOA_s площадь $S_{rs} := max(S_{rs}, S_1)$. Если же s идет позже, чем k, не делаем ничего. После того, как все блоки перебраны, мы имеем максимальную площадь контура и список его вершин. Осталось их аккуратно распечатать (не забываем, что полученная нами площадь вдвое больше ответа).

Время работы алгоритма $O(N^3)$. В качестве нижней точки перебираем $O(N)$ вариантов, для каждого варианта перебираем $O(N)$ блоков, для каждого блока перебираем $O(N)$ точек (при выборе максимума из S_{pr_1} подсчитываем частичные максимумы — их пересчёт для каждой точки делаем за $O(1)$).

L. Lamps of the Mind

Вначале заметим, что каждый треугольник однозначно задается парой своих наименьших сторон (под длинами сторон будем понимать не их геометрические длины, а расстояние по дуге окружности без третей точки). В качестве первой части решения будем заполнять квадратный булев массив $a[i][j]$ означающий, что существует треугольник подходящий под ограничения у которого расстояние от первой точки до второй по часовой стрелке равно i, а до третей — j $(i < j)$. Затем, пробежавшись по этому массиву, можно найти пару минимальных сторон из i, $j - i$ и $n - j$, и посчитать количество различных таких пар (например, вновь заполняя квадратный массив).

Осталось понять как быстро реализовать первую часть. Пусть первая точка k. Рассмотрим вторую точку на расстоянии i. Посмотрим, какие точки мы можем взять в качестве третей, это все точки с положительными ограничениями кроме, может быть, совпадающих по цвету с k-ой и $k + i$-ой точками. Стало быть зная битовый массив хороших точек и точек каждого цвета, можно битовыми операциями обновить массив a[i] для k-ой точки. Заметим также, что можно брать i в качестве минимальной стороны. Получим (n/3)*n*n/32 операций.