Задача A. Дом, в котором живет кот

Посчитаем нарастающим итогом суммарное количество котов и кошек s_k, живущих на этажах, начиная с первого и заканчивая некоторым этажом k.

Когда в нашем распоряжении будут все такие суммы (их можно хранить в массиве / списке; удобно сделать также $s_0 = 0$), можно будет реализовать следующую идею.

Суммарное количество котов и кошек, живущих в доме, может быть разделено на три части: d — живущих ниже этажа, на котором живёт Бенедикт; sb — живущих на одном этаже с Бенедиктом; u — живущих выше этажа, на котором живёт Бенедикт.

Если предположить, что Бенедикт живёт на первом этаже, то $d = 0$, $sb = s_1$, $u = s - d - sb$. Запомним, какой получится разность $u - d$ (она заведомо будет положительная) и перейдём к следующему этажу

Если предположить, что Бенедикт живёт на втором этаже, то эти переменные изменятся следующим образом: $d = 0 + s_1$, $sb = s_2$, $u = s - d - sb$. Проверим, что разность $u - d$ неотрицательна (если это не так, дальнейшее рассмотрение уже не требуется) и сравним её со значением на предыдущем шаге.

Далее, предполагая, что Бенедикт живёт на третьем этаже, получим $d = 0 + s_1 + s_2$, $sb = s_3$, $u = s - d - sb$ (конечно, можно вычислять u через его предыдущее значение). Повторим проверку и сравнение для $u - d$.

Продолжая описанные действия, найдём наилучшее значение $u - d$ и соответствующее ему значение этажа.

Асимптотика подсчёта сумм и пересчёта разности $u - d$ составляет $O(n)$, где n — количество этажей.

На неполный балл задачу можно было решить, если подсчитывать u и d каждый раз «с нуля», не используя предыдущие результаты.

Задача B. Коты на заборе

Поместим данные о котах в два массива / списка (также можно воспользоваться структурой данных «очередь»): в одном будут коты, идущие по часовой стрелке, в другом — коты, идущие против часовой стрелки.

Проанализируем, встретятся ли коты из этих двух списков, стартующие в наиболее раннее (в каждом из списков) время. Если разница во времени составляет менее, чем $2 \cdot (a + b)$, то коты встретятся и не завершат обход забора.

Чтобы понять, увидит ли их разбирательство Бенедикт, нужно, чтобы разность $2 \cdot (a + b) - (t_{later} - t_{earlier})$ либо не превышала d, либо $a - d$ (в зависимости от стороны, в которую пойдёт кот). (Здесь t_{later} — время начала движения кота, который приходит к дереву позже, а $t_{earlier}$ — время начала движения кота, который приходит к дереву раньше).

Если же коты из двух списков не встретятся, то кота, который отправился в более раннее время, мы считаем завершившим обход, и выбираем из этого массива следующего кота.

Задача C. Воробьи и лужа

В этой задаче можно применить технику «двух указателей».

Поскольку требуется минимизировать время, которое Бенедикт находился на подоконнике, то можно считать, что он всегда забирается на подоконник в какую-то из минут, в которую прилетали воробьи.

Левый указатель устанавливается на начало отрезка (первая минута из проведённых котом на подоконнике, в которую прилетали воробьи), правый — на конец отрезка (последняя минута из проведённых котом на подоконнике, в которую прилетали воробьи).

Сначала оба указателя устанавливаются на первую минуту, в которую прилетали воробьи, затем мы их перемещаем. Правый указатель двигается тогда, когда сумма на отрезке ещё не превосходит искомого числа. Левый же указатель двигается тогда, когда сумма на отрезке оказывается большей искомого числа и равной ему.

Когда при некоторой позиции правого указателя получено суммарное количество воробьев, большее или равное искомому, а при позиции правого указателя на 1 меньше, получено суммарное количество воробьёв, меньшее искомого — есть шанс, что мы нашли лучший результат.

Заметим, что ответом может быть отрезок $[i..i]$. Это означает, что кот увидит только воробьев, прилетавших в минуту t_i, а находиться на подоконнике кот будет ровно 1 минуту.

Не следует забывать про то, что целые числа могут требовать 64 бита для хранения в памяти компьютера, а также про ограничение «кот увидел хотя бы одного воробья».

Задача D. Котофото

Будем говорить, что некоторая фотография x *накрывает* день $\#i$, если x непосредственно или **косвенно** лучше не выбранных фотографий, сделанных в этот день.

Поясним, что имеется в виду.

Если фотографию x выбрали в день $\#j$, то она совершенно точно накрывает дни из отрезка $[1..j]$ (поскольку выбрали её, а не какую-то другую). Будем называть в этом случае j *величиной покрытия* для фотографии x.

Допустим, что существует какая-то фотография y, такая, что

- её сделали в день $t \leqslant j$
- её выбрали в день $k > j$

В таком случае величиной покрытия y является k, а поскольку фотография x накрывает день, в который была сделана фотография y, то, следовательно, её величина покрытия тоже оказывается не менее k.

Для формирования ответа понадобятся только дни, которые накрыты менее чем n выбранными фотографиями.

Можно заметить, что в ответ войдут все дни, которые строго больше минимальной величины покрытия.

Получаем общую идею решения:

- рассматриваем выбранные фотографии, начиная с фотографии, выбранной в день n и заканчивая фотографией, выбранной в день 1.
- для фотографии, сделанной в день $\#i$, величина покрытия будет равна максимуму из i и величин покрытия всех выбранных фотографий, сделанных в дни с 1 по i.

Чтобы быстро находить максимум среди величин покрытия, будем поддерживать либо очередь с приоритетами (PriorityQueue, pq), либо стек из оптимальных рассмотренных фотографий.

В таком случае

- либо на вершине стека / pq лежит фотография, которая была сделана позже дня i — тогда просто удаляем её из стека.
- либо она сделана в дни с 1 по i — тогда берем её величину покрытия.

В конце обработки дня кладем на вершину стека текущую фотографию.

Задача E. Вкусный план

Если в некоторый момент в запасе имеется несколько упаковок корма с разными сроками годности, правильная стратегия — сначала отдать коту корм, имеющий минимальный срок годности (минимальная величина $e = d + g$).

Воспользуемся следующим приёмом: каждый день, в который можно приобрести корм со скидкой, будем приобретать новый корм, пока не наберём m упаковок, но в ответе будем учитывать только реально использованный корм.

Таким образом, при обработке дня $\#i$ (в который продаётся корм со скидкой) будем итеративно доставать из имеющихся запасов и отдавать коту корм, пока упаковки не закончатся или пока не покроем отрезок от $d[i-1]$ до $d[i]-1$ (следует положить $d[0] = 1$).

Можно представить себе, что мы достаём корм по одной упаковке в порядке возрастания срока годности, параллельно двигаясь по указанному отрезку. Если корм испортился ранее дня, который рассматривается, его следует просто игнорировать.

Практически же следует доставать из запасов сразу весь корм, приобретённый в какой-то день, и пробовать покрыть им столько дней из указанного отрезка, сколько получится. Если корм фактически использован, можно добавить его как покупку в соответствующий день.

Если запасы корма окажутся исчерпанными раньше дня $(d[i]-1)$ — значит, все оставшиеся дни будем приобретаться корм без скидки.

Если какие-то упаковки остались в запасе после того, как отрезок от $d[i-1]$ до $(d[i]-1)$ окажется покрыт, следует сравнить их срок годности со сроком годности того корма, который можно приобрести в день $d[i]$. Если у каких-либо упаковок срок годности наступает раньше, чем $d[i]+g[i]$, их также следует изъять из запасов.

После этого набираем максимально возможное количество упаковок в день $d[i]$ (чтобы суммарное количество упаковок не превысило m).

В качестве структуры данных для хранения корма, помещаемого в запас, можно использовать стек либо очередь с приоритетами (либо сортированное множество).

Заметим, что нам надо обработать все дни до $D = \max\limits_{i=1}^{n}(d[i]+g[i])$ включительно, поэтому добавим в конец списка фиктивный день $D+1$.

Задача F. Большая коробка

Лемма. Среди вершин во вводе найдётся хотя бы одна пара, образующая ребро оригинального параллелепипеда.

Доказательство леммы. Всего имеется 12 рёбер. Каждая вершина «участвует» только в 3 рёбрах. Следовательно, при изменении 3 (максимальное количества) вершин «сломалось» не более 9 рёбер.

Теорема. Среди вершин во вводе найдётся хотя бы одна тройка, образующая угол исходного параллелепипеда.

Доказательство теоремы. Возьмём пару, образующую исходное ребро. Данное ребро участвует в 4 углах граней. Поскольку было изменено не более 3 вершин, то хотя бы один угол остался исходным.

Решение задачи.

1. Переберём все возможные тройки вершин. Если тройка образует прямой угол, то предположим, что это угол исходного параллелепипеда.

2. Зная координаты 3 вершин на исходной грани, можно вычислить координату четвертой. После вычисления проверим, есть ли такая вершина во входных данных.

3. Выберем среди оставшихся вершин любую, которая образует прямой угол с одним из 4 рёбер фиксированной грани — это будет вершина с предполагаемой противоположной грани.

   Почему так всегда можно сделать? На противоположной грани 4 вершины, изменили не более 3, следовательно, хотя бы одна осталась исходной.

4. Имеем одну полную грань и одну вершину на противоположной грани — это значит, что можем вычислить «ожидаемые» координаты оставшихся 3 вершин.

5. Имея все 8 координат вершин, проверяем, что ровно $8-k$ из них присутствуют во входных данных.

Полезные советы:

- проверка, что угол является прямым, легко делается через скалярное произведение векторов: $x1 \cdot x2 + y1 \cdot y2 + z1 \cdot z2 = \cos(\pi/2) = 0$;
- вычисления «ожидаемых» координат вершин также легко делать через операции сложения / вычитания векторов.

Задача G. Очень большая коробка

Обозначим через a, b стороны дна коробки, через c — высоту коробки.

Таким образом, можем считать, что у нас использовано суммарно $a \cdot b + 2 \cdot c \cdot (a + b)$ листов, а объем равен $a \cdot b \cdot c$.

Утверждение 1. Если мы знаем a и b, то оптимальная c вычисляется однозначно из неравенства

$$a \cdot b + 2 \cdot c \cdot (a + b) \leqslant q$$

.

$$c = (q - a \cdot b)/(2 \cdot (a + b))$$

Утверждение 2. Рассмотрим объем, как функцию от сторон a и b:

$$V(a, b) = a \cdot b \cdot C(a, b) = a \cdot b \cdot (q - a \cdot b)/(2 \cdot (a + b))$$

Заметим, что a и b симметричны в данной функции.

Если бы a, b и c могли принимать любые вещественные значения, то

- либо оптимум достигается на краевых значениях ($b = 1$)
- либо оптимум достигается при $a = b$.

Рассмотрим случай $b = 1$:

$$V(a, 1) = a \cdot 1 \cdot (q - a)/(2 + 2 \cdot a)$$

Воспользуемся **приблизительной** оценкой $V(a, 1) \sim (q - a)/2$.

Максимум достигается при $a = 1$: $V(1, 1) \sim q/2$.

Теперь рассмотрим случай $b = a$:

$V(a, a) = a \cdot a \cdot (q - a \cdot a)/(4 \cdot a)$

Видно, что $V(a, a) = a \cdot (q - a \cdot a)/4$

Оптимум достигается, когда производная равна 0:

$$(q \cdot a - a \cdot a \cdot a)' = 0$$

$$q - 3 \cdot a \cdot a = 0$$

$$A = \sqrt{q/3}$$

При таком значении a значение объёма будет $\sqrt{q/3} \cdot (q - q/3)/4 = q/6 \cdot \sqrt{q/3} = q^{3/2}/(6 \cdot \sqrt{3})$

Легко заметить, что $q^{3/2}/(6 \cdot \sqrt{3}) \geqslant q/2$ при

$q^{1/2} \geqslant 3 \cdot \sqrt{3}$

$q \geqslant 27$

Таким образом, для больших q выгоднее использовать вариант $a = b$.

Однако есть важный момент: всё это время мы рассматривали вариант вещественных a, b, c. Как изменится решение для целых значений?

На самом деле достаточно перебрать все возможные значения a и b в окрестности значения $r = \sqrt{q/3}$.

Более подробно: перебираем все a в промежутке $[r - d, r + d]$, после чего внутренним циклом перебираем b в промежутке $[r - d, r + d]$ и выбираем лучший ответ.

Для d хватает 500 (проверено сравнением с ответами вплоть до $d = 2000$).

Задача H. Равномерное распределение

Поскольку порядок, в котором кот будет лежать на тех или иных своих любимых местах, не важен, отсортируем эти места по возрастанию времени, которое он готов на них провести. Также отсортируем дни по возрастанию времени, в течение которого будут отсутствовать хозяева.

Будем обрабатывать дни в порядке возрастания времени отсутствия хозяев.

В день i кот можем выбрать для отдыха только место j, для которого $t[j] \leqslant d[i]$. Назовём такие места «активными».

Перед выбором места «активируем» все места, для которых $d[i-1] < t[j] \leqslant d[i]$ (заметим, что однажды активированное место будет активным до конца).

Из всех активных мест лучшим выбором всегда будет место, на котором кот лежали меньше всего раз (жадная стратегия).

Доказательство. Пусть на данный момент максимум по количеству лежаний среди активных мест равен x, а минимум равен y.

Если мы выберем место, на котором кот лежал w раз, такое, что $w > y$, то мы

- возможно увеличим максимум (если $w = x$);
- точно не увеличим минимум.

Отсюда следует, что любой выбор места, на котором кот лежал $w > y$ раз, может лишь увеличить разность $x - y$, но никак не может её уменьшить.

Для эффективного выбора места с наименьшим количеством лежаний можно воспользоваться очередью с приоритетами (или структурой данных с аналогичным функционалом, например, сортированным множеством).

Задача I. Мячики

Чтобы прийти к решению, обсудим, как можно проверить, удовлетворяет ли некоторый набор предпочтений заданной последовательности.

Например, рассмотрим набор предпочтений «красный», «жёлтый», «зелёный» (RYG).

В таком случае в последовательности могут присутствовать следующие пары переходов:

- RY (были доступны все мячики, кот играл с R и закатил его под шкаф, стал играть с Y)
- YR (кот поиграл с Y, возможно, закатил его под шкаф, но из-под шкафа достали R и Y, если он там тоже оказался).
- YG (R закатился под шкаф некоторое время назад, Y закатился только что, у кота остался только G)
- GR (кот поиграл с G, возможно, закатил его под шкаф, из-под шкафа достали R и Y, а также G, если он там оказался)

Заметим, что в последовательности никогда не могут встретиться переходы RG и GY.

Аналогичным образом можно вычислить «запрещённые» переходы для всех других возможных перестановок цветов.

Итоговое решение выглядит следующим образом.

- переберём все возможные перестановки цветов мячиков
- проверим, что последовательность из входных данных **НЕ содержит** «запрещённые» переходы для рассматриваемой перестановки.
- если не содержит запрещённых переходов — добавим перестановку в список кандидатов на ответ.

В итоге получаем три варианта наполнения списка кандидатов на ответ.

- список кандидатов пуст — выводим FFF
- в списке оказалось два или более кандидатов — выводим NNN
- в списке единственный кандидат — выводим этот набор предпочтений в качестве ответа.