Задача A. Водонагреватель

Сначала кратко опишем общее решение, а затем рассмотрим подробно решения по группам тестов.

1. Общие упрощения Для начала рассмотрим подходы, упрощающих общей процесс решения.

1. Ответ можно представить в виде формулы $ans = w \cdot (T_p \cdot p + T_q \cdot q)$, где
   - ans — ответ на задачу;
   - T_p — суммарное количество минут работы на тарифе p;
   - T_q — суммарное количество минут работы на тарифе q.

   Так как w, p, q даны во входных данных, а T_q можно вычислить как $k - T_p$, то задача сводится к вычислению времени работы нагревателя на тарифе p.

   Псевдокод итогового вычисления ответа:

   ```
   <Вычисление Tp>

   Tq := k - Tp

   ans := w * (Tp * p + Tq * q)

   Вывести(ans)
   ```

2. В данной задаче в том или ином виде осуществляются две операции со временем:
   - сравнение двух моментов времени;
   - прибавление к времени определенного промежутка времени.

   Для облегчения взаимодействия со временем (прибавления и сравнения) можно перейти от пары «часы, минуты с начала часа» к значению «минуты с начала суток».

   Пусть (h, m) — время в парном формате, а t — время в минутном формате.

   Тогда $t = h \cdot 60 + m$, причем $0 \leqslant t < D$, где $D = 24 \cdot 60$ - количество минут в сутках.

   Теперь обе операции сводятся к аналогичным операциям над целыми числами.

   Псевдокод перехода к минутному формату времени:

   ```
   <Ввод данных>

   t1 := h1 * 60 + m1 # время начала тарифа p
   t2 := h2 * 60 + m2 # время конца тарифа p
   ts := s * 60 + u # время начала работы нагревателя
   D := 24 * 60

   <Подготовка данных>
   ```

3. В задаче действуют два тарифа p и q, причем p действует только с $t_1 = (h_1, m_1)$ до $t_2 = (h_2, m_2)$.

   Для решения задачи в том или ином виде надо уметь отвечать на вопрос «какой тариф действует в момент времени t», что аналогично вопросу «принадлежит ли t полуинтервалу $[t_1; t_2)$».
   - если $t_1 < t_2$ (тариф действует в рамках одних суток), то проверка сводится к двойному неравенству $t_1 \leqslant t < t_2$.
   - чтобы не обрабатывать отдельно случай $t_1 > t_2$ (действие тарифа переходит через границу суток 23:59 - 00:00), можно поменять местами тарифы. В таком случае уже тариф q действует в рамках одних суток на полуинтервале $[t_2; t_1)$.

Псевдокод подготовки данных к решению:

```
<Переход к минутному формату времени>

Если (t1 > t2) {
    Обменять(p, q)
    Обменять(t1, t2)
}

<Вычисление Tp>
```

2. Решение эмуляцией Самым простым для понимания и реализации решением является поминутная эмуляция работы нагревателя.

Переберем все минуты работы от 0 до $k-1$ и для каждой отдельно проверим, к какому тарифу она принадлежит.

Остается определиться с переходом от номера минуты i к абсолютному времени в сутках.

Легко заметить, что если обозначить время начала работы за $t_s = (s, u)$, то i-я минута работы будет равна в абсолютном времени $(t_s + i)\ mod\ D$, где mod - операция взятия остатка от деления. В самом деле, если время будет равно D, то это аналогично моменту 00:00 уже в следующих сутках.

Итоговая асимптотика решения получается равной $O(k)$, что являлось достаточным для прохождения всех подгрупп по времени на основных доступных языках.

Псевдокод вычисления Tp эмуляцией:

```
<Подготовка данных к решению>

Tp := 0
Цикл (i от 0 до k - 1) {
    t := (ts + i) mod D
    Если (t1 <= t) И (t < t2) {
        Tp := Tp + 1
    }
}

<Вычисление и вывод ответа>
```

3. Аналитическое решение Ответ может быть вычислен аналитически (с помощью формул) без необходимости явной эмуляции.

Можно выделить два случая относительно всего периода работы нагревателя:

3a. начало и конец расположены в рамках одних суток;

3b. начало и конец расположены в разных сутках.

Для упрощения обработки случая 3b можно использовать следующее разбиение:

- работа с начала работы до конца первых суток (3a);
- работа на протяжении нескольких полных суток (возможно 0);
- работа с начала последних суток до конца работы (3a).

Рассмотрим вычисление T_p для случая 3a: если период работы нагревателя в текущих сутках задан полуинтервалом $[b; e)$, то T_p равно количеству минут в пересечении полуинтервалов $[b; e)$ и $[t_1; t_2)$.

Задача нахождения пересечения двух интервалов на числовой прямой имеет очень простое и красивое решение в общем виде: если $L = max(b, t_1)$ — максимум из начал интервалов, а $R = min(e, t_2)$ — минимум из концов, то количество минут в пересечении $M = max(0, R - L)$.

Псевдокод подпрограммы вычисления количества минут в пересечении:

```
Пересечение(b, e, t1, t2) {
    L := max(b, t1)
    R := min(e, t2)
    M := max(0, R - L)
    Вернуть(M)
}
```

В случае работы на протяжении полных суток $T_p = t_2 - t_1$ (по сути это случай 3a при $b = 0$, $e = D$).

Псевдокод вычисления Tp аналитически:

```
<Подготовка данных к решению>

Tp := 0

tf := ts + k
Если (tf <= D) {
    # случай 3a
    Tp := Пересечение(ts, tf, t1, t2)
} иначе {
    # случай 3b

    need := k # оставшееся до конца работы время

    # первые сутки
    Tp := Пересечение(ts, D, t1, t2)
    need := need - (D - ts)

    # полные сутки в середине
    Full := need div D # div - операция целочисленного деления
    Tp := Tp + (t2 - t1) * Full
    need := need mod D

    # последние сутки
    Tp := Tp + Пересечение(0, need, t1, t2)
}

<Вычисление и вывод ответа>
```

4. Подгруппы Рассмотрим особенности каждой отдельной подгруппы.

Подгруппа 1 - Минутный формат необязателен.
- Достаточно случая 3a для аналитического способа.

Подгруппа 2 - Минутный формат необязателен.

- Необходимы оба случая 3a и 3b для аналитического способа.

Подгруппа 3
- Минутный формат рекомендуется.
- Достаточно случая 3a для аналитического способа.

Подгруппа 4
- Минутный формат рекомендуется.
- Необходимы оба случая 3a и 3b для аналитического способа.

Подгруппа 5 отличается от подгруппы 4 необходимостью использования 64-битных типов данных для промежуточных вычислений и окончательного ответа, так как максимальное значение ответа равно $10^6 \cdot 10^6 \cdot 10^6 = 10^{18}$.

Задача B. Примеры

Сначала опишем решение на полный балл, а затем прокомментируем отдельные группы.

Определим момент времени от начала решения примеров, в который ребята пойдут гулять.

Как можно предположить, это случится, когда Кеша и Мелентий одновременно закончат решать очередные примеры — разумеется, при условии, что каждый из них уже решил достаточное количество примеров.

Чтобы найти такой момент времени, нам необходимо определить наименьшее общее кратное чисел tk и tm. Это несложно сделать, используя алгоритм Евклида для отыскания наибольшего общего делителя, а затем соотношение $a \cdot b = lcm(a, b) \cdot gcd(a, b)$, где gcd — это Great Common Divisor, наибольший общий делитель, а lcm — Least Common Multiple, наименьшее общее кратное.

Наименьшее общее кратное должно быть делителем того момента времени, в который Кеша и Мелентий закончат решать примеры одновременно. Так что можно выяснить, кто из ребят закончит решать желаемое количество примеров позже и найти ближайшее к этому моменту окончания решения такое число. Для этого достаточно выяснить, делится ли этот момент окончания на lcm нацело, и в случае отрицательного ответа добавить «недостающий» остаток.

Если бы количество примеров было бесконечным, то это было бы решением задачи. Однако примеров всего n штук, так что их может просто не хватить. Поэтому придётся рассмотреть ещё некоторые ситуации, а именно, когда один из ребят закончит решать все примеры.

Например, предположим, что Кеша решает пример за меньшее время, чем Мелентий, так что на решение всех примеров Кеше потребуется меньше времени. Так что, решив все примеры, Кеша будет звать Мелентия гулять. Можно считать, что он будет делать это каждую минуту. Мелентий же откликнется на приглашение не раньше, чем решит необходимое с его точки зрения количество примеров.

Следовательно, в этой ситуации возможны два варианта. Первый вариант — Кеша решит все примеры до момента, когда Мелентий решит необходимое количество примеров. Это значит, что ребята пойдут гулять, как только Мелентий дорешает свои примеры.

Второй вариант — Кеша закончит решать примеры уже после того, как Мелентий решит необходимое количество примеров. В этом случае ребята пойдут гулять, когда Кеша дорешает все примеры, а Мелентий закончит решать очередной пример. Чтобы отыскать этот момент времени, нужно найти число, ближайшее к моменту окончания решения примеров Кешей и делящееся нацело на tm.

Аналогично можно рассмотреть и «зеркальную ситуацию», когда Мелентий тратит на решение одного примера меньше времени, чем Кеша.

Далее потребуется выбрать, какой из моментов времени случится раньше: когда ребята закончат решать свои очередные примеры одновременно или же когда кто-то из них решит все имеющиеся примеры.

Определившись с моментом, в который ребята отправятся гулять, несложно вычислить, сколько примеров решит каждый из них к этому моменту: достаточно разделить это время на tk для Кеши и tm для Мелентия. Если полученные значения превышают n, их можно просто заменить на n.

Задача C. Роллер

Сначала опишем полное решение, а затем прокомментируем отдельные группы тестов.

Представляется достаточно очевидным, что нужно «бороться» с наиболее длинными асфальтовыми участками. Пройдём по строке, задающей покрытие участков, и выделим из неё последовательности участков, на которых уложен асфальт. Хранить такие последовательности участков можно в виде пар (*номер первого участка*, *количество участков*) (можно хранить вместо количества участков номер последнего участка, это уже вопрос реализации).

Отсортируем пары по убыванию количества участков в них и выберем первые две последовательности, состоящие из наибольшего количества участков.

Поскольку мы можем поменять асфальт ровно на двух участках, нам потребуется рассмотреть два разных варианта замены. Первый вариант — заменить участки, находящиеся посередине каждой из двух последовательностей. Второй вариант — использовать обе замены на более длинной последовательности (разбив её на три фрагмента, отличающихся друг от друга по длине не более, чем на один участок).

Разумеется, можно провести анализ, показывающий, при каком соотношении количеств участков в первых двух последовательностях стоит выбирать первый, и при каком — второй вариант. Но если пары, описывающие последовательности, сохранены в сортирующей структуре данных, достаточно изъять из неё эти последовательности и поместить в неё фрагменты, полученные после замен, после чего сделать запрос наибольшего элемента.

Конечно, следовало аккуратно обработать ситуацию, когда существует только одна последовательность асфальтовых участков.

В первых двух группах тестов можно было просто перебрать все возможные варианты замен двух участков. Третья группа допускала использование каких-либо неэффективных вариантов сортировки (или подобных алгоритмов).

В четвёртой и пятой группах рассматривался частный случай, когда имеется одна последовательность асфальтовых участков (в пятой группе требовалось определить её границы). Шестая группа предполагала, что существует ровно одна последовательность участков из плитки, и это давало гарантии, что последовательностей асфальтовых участков не более двух. Седьмая группа оказывалась фактически «интегрирующей» предыдущие группы.

Наконец, восьмая группа требовала полного решения задачи.

Задача D. Экскаватор

Сначала опишем полное решение, а затем приведём соображения, которые могли позволить решить задачу частично.

В этой задаче имеется два вида событий — поступление заявки с некоторого участка и прибытие экскаватора на некоторый участок. Заявки хронологически упорядочены во входных данных, выполняются также хронологически, поэтому никакой сортировки не требуется. Однако на «событийный ряд» заявок требуется «наложить» движение экскаватора. Таким образом, это достаточно типичная задача на применение техники «двух указателей».

Можно действовать следующим образом. Поместим все события-заявки в какую-либо линейную структуру данных (массив или список). Именно по этой структуре будет двигаться один из указателей. Будем фиксировать, на каком участке находится экскаватор, а также день, до которого включительно экскаватор будет там находиться (или же первый день, в который экскаватор окажется на следующем участке — это уже детали реализации). Фактически, пара, состоящая из номера участка и дня, и является вторым указателем.

Как понятно, в день поступления первой заявки экскаватор отправится на первый участок и будет там работать в течение k дней. Зная последний день его работы на первом участке, будем перемещаться по списку заявок до тех пор, пока день поступления заявки будет меньшим или равным дню, до которого работает экскаватор. Отыскав номер последнего такого участка, узнаем, во-первых, количество участков, которые придётся обходить Кеше (как разность между номерами участков), и, во-вторых, количество дней, в течение которых ему придётся это делать (как разность между днём, в который экскаватор отправится на другой участок, и днём, в который поступит заявка с рассматриваемого участка).

Далее следует увеличить номер участка, на котором будет находиться экскаватор, на единицу, и вычислить день, до которого он будет там находиться, добавив к «текущему» дню k. После этого

нужно повторить описанные выше шаги, направленные на поиск последнего участка, с которого поступит заявка до дня окончания работы экскаватора.

После того, как список заявок будет пройден полностью, ответ будет сформирован: после поступления последней заявки количество участков, которые придётся обходить Кеше, возрастать уже не будет.

Важный момент заключается в том, что необходимо поддерживать переменную, хранящую максимальное количество участков, которые Кеше приходилось обходить в течение одного дня, и переменную, хранящую суммарное количество таких дней. В случае, если обнаруживается новый максимум, сумма должна быть сброшена в ноль.

Первая группа тестов предполагала, что заявки поступили с двух участков, притом в разные дни. Для получения баллов по ней было достаточно написать условный оператор, определяющий, поступит ли заявка со второго участка до завершения работ на первом. В случае положительного ответа максимальное количество участков, которые нужно было обходить, становилось равным двум, а количество дней вычислялось как количество дней от поступления второй заявки до окончания работ экскаватора на первом участке.

Группы тестов со второй по четвёртую допускали решение с помощью «ежедневной» симуляции процесса. Пятая группа тестов могла быть пройдена решением с квадратичной асимптотикой — полным просмотром последовательности заявок (от начала до конца) для вычисления количества участков, которые нужно обходить в данный момент. Однако при полном просмотре требовалось уже выполнять проверки для дней,в которые происходили события — поступление заявки или же перемещение экскаватора.

Шестая и седьмая группа тестов предполагали наличие у участника решения в частном случае, когда экскватор работал на каждом участке в течение одного или двух дней.

Для прохождения последней, восьмой группы тестов требовалось полное решение задачи.

Задача E. Котики

Сначала опишем полное решение, затем прокомментируем отдельные группы.

Фактически в этой задаче нужно было выполнить симуляцию процесса, каждый раз предполагая, что Кеша гладит кота, которого уже гладил раньше — конечно, если это в принципе было возможно.

Чтобы выполнить эту симуляцию, нужно использовать структуру данных, позволяющую хранить пару (*масть котика*, *момент последнего поглаживания*), а также быстро отвечать на запрос, в какой момент погладили котика такой масти. Такую структуру можно было написать вручную — массив или список, упорядоченный, например, по мастям в лексикографическом порядке, на котором можно выполнить бинарный поиск. Но также можно было воспользоваться структурой «словарь» (dict в Python, Map в Java, map в C++).

Если дополнить пару, хранящуюся в структуре данных, дополнительным полем для хранения количества котиков каждой масти, которых уже погладили, по завершении обработки входных данных будет несложно подсчитать, сколько же всего было котиков, и сколько было котиков какой масти. При желании можно хранить эту информацию отдельно, в аналогичной структуре данных.

В первых трёх подзадачах масть котика задавалась единственным символом. Таким образом, мастей котиков было не более 26, что заметно упрощало их обработку.

В четвёртой и пятой подзадачах предлагалось разобрать частные случаи, когда количество мастей котиков равно двум и трём соответственно.

В шестой подзадаче дополнительных ограничений не вводилось, требовалось полное решение задачи.