Задача A. Тщательное планирование

Автор и разработчик задачи: Мария Жогова

Давайте представим разложение числа a_i на сумму разрядных слагаемых: $a_i = a_{i,9} \cdot 10^9 + a_{i,8} \cdot 10^8 + \ldots + a_{i,1} \cdot 10 + a_{i,0}$, где $a_{i,j}$ — значение j-й цифры числа a_i при нумерации с нуля от разряда единиц.

Наша задача — найти такую функцию f, чтобы максимизировать сумму $S = \sum\limits_{i=1}^{n} f(a_{i,9}) \cdot 10^9 + f(a_{i,8}) \cdot 10^8 + \ldots + f(a_{i,1}) \cdot 10 + f(a_{i,0})$. Иными словами, мы просто переписали сумму упомянутых выше выражений по i от 1 до n, если переназначение цифр обозначить за f.

Перепишем эту сумму, сгруппировав слагаемые по степеням десятки, а не по числу из набора. Для этого давайте для каждого числа a_i найдем

$$b_{i,k} = \sum_{j=0}^{9} \left(10^j \cdot \begin{cases} 1 & \text{если } a_{i,j} = k \\ 0 & \text{иначе} \end{cases} \right)$$

Такое выражение будет равно «вкладу» цифры k в значение i-го навыка. Ясно, что $a_i = \sum\limits_{k=0}^{9} k \cdot b_{i,k}$. Пусть $c_k = \sum\limits_{i=1}^{n} b_{i,k}$, назовем это число полным вкладом цифры k. Тогда

$$S = \sum_{i=1}^{n} \left(\sum_{j=0}^{9} f(a_{i,j}) \cdot 10^j \right) = \sum_{i=1}^{n} \left(\sum_{k=0}^{9} f(k) \cdot b_{i,k} \right) = \sum_{k=0}^{9} \left(f(k) \cdot \sum_{i=1}^{n} b_{i,k} \right) = \sum_{k=0}^{9} f(k) \cdot c_k$$

Для того, чтобы получить максимально возможное значение S, функция f должна отображать l и m, так, чтобы $f(l) > f(m) \iff c_l \geqslant c_m$. Понятно, что если f назначает цифре с не-максимальным вкладом новое значение 9, то ее можно поменять с цифрой, вклад которой максимален, и сумма увеличится. Значит нужно распределить новые значения цифр так, чтобы наибольшие новые значения были даны цифрам, имеющим максимальный вклад.

Приведенное выше распределение не учитывает то, что после переназначения в записи навыков не должно быть ведущих нулей. В каком случае это может произойти? В том случае, когда в записи всех n навыков используются 10 цифр, цифра k имеет минимальный вклад и является первой в записи хотя бы одного навыка. Чтобы этого избежать, давайте отдельно рассмотрим все цифры, с которых не начинается никакой навык. Среди всех таких цифр выберем ту, которая имеет минимальный вклад, тогда функция f должна заменять ее на 0. Теперь уберем из рассмотрения выбранную цифру, а значения f для остальных цифр выберем так же, как в общем случае. Ясно, что такое распределение дает максимальную сумму S.

Все решение выглядит так:

1. Для каждой цифры посчитаем ее вклад в общую сумму, является ли она первой в записи какого-либо навыка и общее количество различных использованных цифр в записи навыков;

2. Если в записи всех навыков были использованы все 10 цифр, найдем ту, которая имеет минимальный вклад и с которой не начинается ни один навык. Так как мы ее заменим на 0, то в итоговой сумме она никак себя не проявит. Теперь будем считать, что она имеет вклад 0 в итоговую сумму, то есть $c[q] = 0$;

3. Построим массив g, где $g[k] = (k, c[k])$. Отсортируем этот массив по не возрастанию вклада в итоговую сумму, то есть по второй компоненте каждого элемента;

4. Ответ будем считать жадным образом. Так как мы заменяем i-ю цифру в порядке не возрастания вклада на цифру $10 - i$, то $ans = \sum\limits_{k=1}^{|g|} (10 - k) \cdot g[k][0]$.

Рассмотрение цифр каждого из навыков занимает константное количество действий, а всего навыков n, значит время работы решения — $\mathcal{O}(n)$.

Задача B. В поисках Венома

Автор задачи и разработчик: Владислав Власов

В каждый момент времени мы имеем прямоугольник со сторонами x и y, и на каждом шаге из большей стороны вычитается меньшая, то есть если $x > y$, то совершается переход от пары строн оставшейся области (x, y) к паре $(x - y, y)$. Сам же процесс заканчивается, когда $x = y$ и последним действием сканируется вся оставшаяся область.

Если просто просимулировать этот процесс, как описано в условии, решение, ожидаемо, не пройдёт по времени. Однако можно заметить, что процесс очень похож на поиск НОД двух чисел, то есть *алгоритм Евклида*.

Быстрый способ провести целиком алгоритм Евклида — заменить вычитание на деление с остатком. В данном случае можно было переходить от пары (x, y) к паре $(y, x \bmod y)$, увеличивая при таком действии ответ на $\left\lfloor \frac{x}{y} \right\rfloor$, так как именно столько вычетаний заменяется одним делением.

Как и *алгоритм Евклида*, такое решение работает $\mathcal{O}(\log(a + b))$.

Задача C. Защищенная тюрьма

Автор задачи и разработчик: Даниил Орешников

По условию, ни один из разрешенных типов комнат не помещается внутри другого. Значит, если комната i-го типа имеет размеры $a_i \times b_i$, то для любой другой комнаты j-го типа ($j \neq i$) верно, что либо $a_j < a_j$, либо $b_i < b_j$, но не оба неравенства одновременно. Из этого следует, что или $a_j < a_i$ и $b_j \geqslant b_i$, или $a_i \geqslant a_j$ и $b_j < b_i$. В том и в другом случае нам достаточно расширить только один из размеров комнаты. Давайте для каждого из i типов рассмотрим оба случая отдельно и в конце выберем тот размер, за изменение которого придется заплатить меньше.

Пусть размеры комнаты i-го типа по-прежнему $a_i \times b_i$. Рассмотрим такое множество типов комнат J, для которых верно, что $a_j < a_i$, а тогда $b_j \geqslant b_i$ для любого $j \in J$. Чтобы разместить комнату типа j в комнате i необходимо как минимум увеличить размер a_j до $a = a_i$. При этом b_j можно не менять, то есть $b = b_j$. За это придется заплатить как минимум $(a + b) - (a_j + b_j) = a_i - a_j$ денег.

Чтобы найти такой тип комнат $j \in J$, за расширение которого до размеров комнаты i-го типа необходимо заплатить минимальное количество денег, нужно выбрать j с максимальным a_j. Действительно, в таком случае $a_i - a_j$ будет минимальным.

Давайте расположим все типы комнат в порядке неубывания их b_k (и возрастания их a_k, соответственно). Тогда для любого типа комнат i типы комнат $j \in J$, то есть те, у которых можно увеличить первый размер, чтобы комнату типа i разместить внутри комнаты типа j, находятся строго после комнаты i. Тогда задача поиска максимального a_j для $j \in J$ равносильна поиску максимума a_k на суффиксе отсортированного массива типов комнат. Причем этот суффикс начинается сразу после позиции на которой расположен i-й тип комнат.

Давайте отсортируем типы комнат в указанном порядке. Затем посчитаем максимумы a_k на всех суффиксах массива типов комнат (это делается за один проход с конца массива). Теперь снова пройдем по массиву (в этот раз можно с начала) и для каждого типа комнат i запомним максимум на суффиксе в массив $\texttt{ans}_a[i]$. Напоминаем, он равен $\texttt{suff}[pos(i)+1]$, где $pos(i)$ — порядковый номер комнаты в отсортированном массиве. Для комнаты, стоящей последней в отсортированном массиве, максимум на суффиксе будет не определен, будем считать, что $\texttt{ans}_a[i] = +\infty$.

Далее проделаем аналогичную последовательность действий, но для второй стороны: отсортируем по возрастанию b_k (это будет просто обратный порядок); посчитаем суффиксные максимумы; для каждого типа запомним максимум на соответствующем суффиксе в массив $\texttt{ans}_b[i]$.

После этого, можно вывести ответ. Если для i-го типа комнат ответ будет равен $min\left(a_i - \texttt{ans}_a[i], b_i - \texttt{ans}_b[i]\right)$. Для такого решения необходимо отсортировать массив типов комнат и несколько раз проитерироваться по нему. Таким образом, время работы — $\mathcal{O}(n \log n)$.

Задача D. Размещение симбиотов

Автор задачи и разработчик: Даниил Орешников

Заметим, что для размещения n пар симбиотов потребуется хотя бы $2 \cdot \lceil \frac{n}{2} \rceil$ носителей. Это достигается (например) при условии, что симбиоты из нечетной пары i размещаются в носителях из пары i, а симбиоты из четной пары j размещаются в носителях из пары $j-1$. В таком случае носители из четных рядов не потребуются и их можно будет отпустить. Очевидно, что использовать меньше носителей не получится, так как из условий размещения следует, что больше двух симбиотов в одном носителе не разместить.

Давайте решать задачу жадным образом, стремясь приблизить вид размещения к тому, которое приведено выше. Для этого поспользуемся следующим фактом: симбиотов, начиная с i-й пары, можно разместить, не затрагивая носителей из предыдущих пар. Поэтому, если при размещении i-й пары симбиотов есть возможность оставить носителя из $i-1$-й пары «свободным», следует это сделать. Действительно, если оптимальный ответ достигается заниманием этого носителя симбиотом, то где-то дальше есть свободный, до которого в каждой паре носителей размещен один симбиот с той же пары, и один — со следующей. В таком случае можно «сдвинуть» всех этих симбиотов вперед, освободив текущего, и не ухудшив ответ.

В первой паре симбиотов можно разместить любым способом, так как это ни на что не повлияет. Пусть мы уже разметили первые i пар и сейчас размещаем $i+1$-ю. Тогда

- Если в предыдущей паре в обоих носителях уже есть симбиоты, выгодно попытаться разместить там обоих симбиотов из текущей пары. Если это возможно, то носители из текущей пары останутся свободными, и их можно будет отпустить.

  Иначе стоит попробовать разместить симбиота из текущей пары с максимальным весом в носителе из предыдущей пары. Аналогично, если он не помещается, то поместить симбиота с минимальным весом. Того, который не поместился, следует разместить в любом носителе текущей пары.

- Если в предыдущей паре есть симбиот есть только в одном из носителей, выгодно аналогично предыдущему случаю попытаться также разместить симбиота с максимальным весом в нем. Иначе выгодно разместить симбиота с минимальным весом, если это, конечно, возможно.

- Если в i-й паре есть носитель, в котором нет ни одного симбиота, то выгоднее разместить симбиота из пары $i+1$ в носителе из $i+1$-й пары, тем самым добавив «свободных» носителей с предыдущего в ответ.

- Если оба носителя из предыдущей пары свободны, то, как говорилось выше, выгодно не размещать в них симбиотов вообще.

Следуя приведенным выше правилам, на каждом этапе мы будем получать размещение с максимальным числом неиспользованных носителей, а среди всех таких размещение с минимальной заполненностью носителей в последней паре. Ясно, что такое размещение в итоге приведет к размещению с минимальным количеством носителей, которого хватит для размещения всех симбиотов с соблюдением всех описанных условий.

Для решения достаточно последовательно рассмотреть каждую из n пар симбиотов, постоянно поддерживая информацию о заполненности носителей в предыдущей паре. Пар носителей также n, поэтому время работы решения — $\mathcal{O}(n)$.

Задача E. Подозрительные отчеты

Автор задачи и разработчик: Мария Жогова

Рассмотрим классическую динамику для задачи нахождения *наибольшей общей последовательности*. В данном случае мы хотим проверить, что сокращенный отчет целиком «входит» в полный отчет с указанными дополнительными ограничениями. Будем хранить $\texttt{dp}[i][j]$ — могут ли первые i столбцов гистограммы полного отчета содержать первые j столбцов противозаконного отчета, чтобы i-й соответствовал j-му.

При пересчете такой динамики достаточно перебрать k — предыдущий столбец первой гистограммы, соответствующий $j-1$-му столбцу второй. На него накладываются следующие ограничения: $s_k - t_{j-1} = s_i - t_j$ (то, что гистограмма была обрезана по горизонтальной линии, означает, что

разности высот столбцов одинаковы) и $s_x \leqslant s_i - t_j$ для всех $k < x < i$ (так как все элементы между выбранными столбцами должны быть не выше линии разреза). Таким образом,

$$\texttt{dp}[i][j] = \bigvee_{\substack{k<i \\ s_k - t_{j-1} = s_i - t_j \\ \max\limits_{k<x<i} s_x < s_i - t_j}} \texttt{dp}[k][j-1]$$

Такой пересчет в наивной реализации работает за $\mathcal{O}(nm^2)$. Для этого достаточно во вложенных циклах по i и j перебирать k по убыванию от i до 0, поддерживать максимум на отрезке от $k+1$ до $i-1$ и проверять описанные условия. Решение с такой асимптотикой проходило по времени.

Задача F. Симбиоты внутри

Автор задачи и разработчик: Константин Бац

Представим схему передачи энергии в виде ориентированного графа. Назовем органы источниками, а симбиотов — потребителями. Источники и потребители будут вершинами, а связи — ребрами. Тогда задача заключается в том, чтобы построить граф, в котором существует путь от любого источника до любого потребителя, не проходящий через другие источники (чтобы каждый из потребителей продолжил бы получать энергию при их отказе). Среди всех таких графов требуется найти граф с минимальным количеством ребер, а среди всех таких — с минимальной суммарной длиной ребер.

В таком графе должно быть хотя бы $n+m-1$ ребро. Действительно, при замене всех ориентированных ребер на неориентированные граф должен стать связным, а в связном графе не может быть меньше ребер. Заметим, что тогда в этом графе нет циклов, а значит если из a достижима b, то из b не достижима a. В таком случае все потребители должны быть достижимы из какого-то одного, с которым связаны все источники. Иначе для первого потребителя в топологической сортировке будет существовать источник, из которого он не достижим.

Поэтому минимальное число ребер достигается, когда все источники поставляют какому-то потребителю энергию, а он раздает ее другим. Схему передачи между потребителями тогда можно представить в виде подвешенного дерева. Суммарная длина связей в таком случае равна сумме евклидова расстояния от всех источников до некоторого потребителя и сумме всех ребер в дереве потребителей. Эти два слагаемых можно оптимизировать независимо, так как от выбора «первого» потребителя зависит лишь ориентация ребер в мин. остове.

Давайте минимизируем суммарную длину связей в дереве потребителей. Эту часть задачи можно решить алгоритмом поиска минимального остовного дерева в полном графе потребителей из m вершин, для чего удобно воспользоваться алгоритмом Прима с асимптотикой $\mathcal{O}(m^2)$. Найти минимальную сумму расстояний от всех источников до некоторого потребителя можно, перебрав всех потребителей и для каждого посчитав расстояние до всех источников за $\mathcal{O}(nm)$.

Итого, строим минимальный остов за $\mathcal{O}(m^2)$, перебираем всех потребителей и выбираем среди них такой, от которого расстояние до всех источников будет минимальным за $\mathcal{O}(nm)$, и выводим ответ. Общее время работы — $\mathcal{O}(m \cdot (n+m))$.

Задача G. Долгое путешествие

Автор задачи: Фитисов Артем, разработчик: Мария Жогова

Заметим, что масса любого симбиота ограничена 10^9. Поэтому масса i-го симбиота после того, как он пожертвовал собой, может дойти до симбиота с номером j, удаленного от i-го не дальше, чем на $R = \log_2 10^9 \leqslant 30$. При этом масса i-го симбиота может как-то увеличить массу j-го, и эта обновленная масса может дойти до симбиота k на расстоянии также не больше R от j. Таким образом на вес k-го симбиота могут повлиять симбиоты, удаленные от него не дальше, чем на $2 \cdot R$.

Так как у каждого симбиота есть сосед слева и справа, то масса k-го симбиота может меняться в течении первых $4 \cdot R$ лет. Далее все ближайшие симбиоты погибнут, а масса остальных симбиотов уже не будет доходить до k-го ни в каком виде.

Давайте считать, что $4 \cdot R < n - 1$. Для $1 \leqslant t \leqslant 4 \cdot R$ ответ можно предпочитать, а для больших t он очевидно будет равен ответу при $t = 4 \cdot R$.

Давайте для каждого $0 \leqslant i \leqslant n-1$ пронаблюдаем то, как его масса будет влиять на $2 \cdot R$ соседей слева и справа. Пусть $r[i][j]$ — масса i-го симбиота при условии, что j его соседей справа поделились своей массой, а $l[i][j]$ — масса i-го симбиота при условии, что j его соседей слева поделились своей массой. Ясно, что $l[(i+j) \bmod n][j] = \left\lfloor \frac{l[(i+j-1) \bmod n][j-1]}{2} \right\rfloor + a[(i+j) \bmod n]$, а $r[(i+j) \bmod n][j] = \left\lfloor \frac{r[(i+j+1) \bmod n][j-1]}{2} \right\rfloor + a[(i+j) \bmod n]$ для всех $1 \leqslant j \leqslant 2 \cdot R$.

Тогда максимальная достижимая i-м симбиотом за t лет масса $m[i][t] = \max\limits_{0 \leqslant j \leqslant t} l[i][j] + r[i][t-j] - a[i]$. Ответ на задачу равен $ans[t] = \max\limits_{i=0}^{n-1} m[i][t]$. Это верно для любых $0 \leqslant t \leqslant \min(n-1, 4R)$. Для $t > 4 \cdot R$ при условии $4 \cdot R < n-1$ ответ будет равен $ans[4R]$.

Если $n - 1 \leqslant 4 \cdot R$, то $ans[t]$ для $t < n-1$ считается так же, как в общем случае. Рассмотрим отдельно случай, когда $t = n-1$. В этом случае первый симбиот, который пожертвовал собой поделится своим весом с правым и левым соседом. Поэтому $m[i][n-1] = \max\limits_{j=0}^{n} l[i][j] + r[i][n-j] - a[i]$. Следовательно, $ans[n-1] = \max\limits_{i=0}^{n-1} m[i][n-1]$. В остальных случаях ($t > n-1$) $ans[t] = ans[n-1]$.

Общий план решения задачи.

1. Посчитаем $l[i][j]$ и $r[i][j]$ для $0 \leqslant i \leqslant n$, $0 \leqslant j \leqslant 2 \cdot R$ через рекуррентные соотношения.
2. Для всех t от 1 до $\min(4R, n-1)$ посчитаем ответ $ans[t] = \max\limits_{0 \leqslant i \leqslant n-1} \left(\max\limits_{\max(0, t-2R) \leqslant j \leqslant \min(t, 2R)} l[i][j] + r[i][t-j] - a[i] \right)$.
3. Если $n - 1 < 2 \cdot R$, то $ans[n-1] = \max\limits_{0 \leqslant i \leqslant n-1} \left(\max\limits_{\max(0, n-2R) \leqslant j \leqslant \min(n-1, 2R)} l[i][j] + r[i][n-j] - a[i] \right)$.
4. Выведем выводить ответы на запросы. Для всех t_i либо уже посчитан, либо он равен ответу при $t = min(n-1, 4R)$, который посчитан.

Общее время работы — $\mathcal{O}(R^2 \cdot n + m) = \mathcal{O}(n + m)$.

Задача H. Воссоединение с Веномом

Автор задачи: Даниил Орешников, разработчик: Константин Бац

Заметим, что любое любое изменение показателей не меняет их разницу по модулю 2. Докажем это. Возьмем два любых показателя гормонов a_i и a_j и применим к ним некоторое изменение.

- Если оба показателя увеличиваются на 1, то $(a_i - a_j) \bmod 2 = ((a_i + 1) - (a_j + 1)) \bmod 2$;
- Если один из показателей увеличивается, а второй уменьшается, то, не теряя общности, $(a_i - a_j) \bmod 2 = ((a_i + 1) - (a_j - 1)) \bmod 2 = (a_i - a_j + 2) \bmod 2$.

Поэтому уравновесить значения показателей можно тогда и только тогда, когда все три показателя изначально равны по модулю 2.

Пусть нам даны 3 характеристики $a_1 \leqslant a_2 \leqslant a_3$, которые не уравновешены, но их можно уравновесить. Ясно, что, чтобы сделать это за минимальное количество действий, стоит уменьшить a_3 на один и увеличить a_1 и a_2 на один. Это оптимально, так как любое другое действие только увеличит разницу между показателями. Таким образом, оптимальную стратегию уравновешивания показателей можно представить так:

```
while a[1] != a[2] or a[2] != a[3]:
    a.sort()
    a[1] += 1
    a[2] += 1
    a[3] -= 1
```

Понятно, что такое решение является не оптимальным. Давайте заметим, что каждая итерация в цикле с точки зрения разности между показателями равносильна $a_3 = a_3 - 2$. Тогда все итерации можно представить как уменьшение a_3 до тех пор, пока оно не станет равно a_1, и уменьшение a_2 до тех пор, пока оно тоже не станет равно a_1. В таком случае мы совершим $\frac{a_3 - a_1}{2} + \frac{a_2 - a_1}{2}$ изменений. Заметим, что обе разности поделятся нацело, так как все три показателя изначально равны по модулю 2. Это число и есть ответ на задачу.

Итак, решение задачи: проверим, что показатели гормонов равны по модулю два — если равенства нет, то ответ равен -1, иначе отсортируем по возрастанию три показателя и выведем ответ $\frac{a_3 - a_1}{2} + \frac{a_2 - a_1}{2}$. Время работы решения — $\mathcal{O}(1)$.