Задача A. Сложный выбор

Задача носила утешительный характер. Можно было сначала отыскать минимальное значение среди элементов $d_1, d_2, \ldots, d_n$, а затем для среди всех элементов, равных этому минимальному значению, выбрать тот, для которого окажется меньше соответствующее (по индексу) значение p. Чтобы найти последнее по номеру такой элемент, можно было либо организовать проход по массиву d, начиная с последнего элемента, и использовать для сравнения строгое неравенство, или же организовать проход, начиная с первого элемента, и использовать нестрогое неравенство, как продемонстрировано в фрагменте кода ниже.

```
int minp = 1000*1000*1000 + 10;
ans = -1;
  for (int i = 0; i < n; i++) {
    if (d[i] == mind) {
      if (p[i] <= minp) {
        minp = p[i];
        ans = i;
      }
    }
  }
```

Здесь $mind$ — минимальное значение d, найденное ранее.

Задача B. Ультиматум

В этой задаче нужно было построить непротиворечивую схему выхода газет после того, как сообщение станет известным. Это можно было сделать, например, следующим образом. Найдем среди газет, не опубликовавших сообщение, ту, которая поступила в продажу позже всех. Пусть она поступила в продажу в момент X. Далее, найдем среди газет, опубликовавших сообщение, ту, которая поступила в продажу раньше всех. Пусть она поступила в продажу в момент Y.

Можно выдвинуть предположение, что газета, поступившая в продажу в момент Y, была сверстана за 1 единицу времени. Это означает, что сообщение должно было поступить до момента $Y-1$ (ближайший слева целый момент времени был $Y-2$). Теперь осталось понять, как соотносятся моменты $Y-2$ и X. Если момент X наступил позже момента $Y-2$, предположим, что именно в момент $Y-2$ началась верстка этой газеты, и это минимально возможное время для этой газеты. Заметим, что оно будет максимальным среди всех газет: те, которые не опубликовали сообщение, могут быть сверстаны за меньшее время, а те, которые опубликовали сообщение, могут быть сверстаны за 1 единицу времени.

Если же момент X случился ранее момента $Y-2$, то для всех не опубликовавших сообщение газет также можно выдвинуть предположение, что они были сверстаны за 1 единицу времени.

Задача C. Вопрос доверия

В этой задаче нужно минимизировать максимальную фантастичность истории.

Единственная причина, почему мы не можем сказать, что все истории имели фантастичность 0 — это тот факт, что герцог иногда не верил барону, и, значит, его доверие уменьшалось. По условию же доверие никогда не могло стать отрицательной величиной.

Рассмотрим функцию $minD(fMax)$ —минимальный уровень доверия, который будет достигнут в какой-то момент времени, если фантастичность всех историй не превосходит $fMax$.

Доверие может увеличиться по формуле $d = max(d, f)$, и, следовательно, выше максимальной фантастичности оно подняться не может (если, конечно, он не было больше изначально).

Пусть для какого-то значения f значение $minD(f) < 0$. В таком случае если мы еще уменьшим верхнюю границу, то можем только ухудшить положение — ведь мы уменьшим максимально возможное значение доверия, которого барон может достичь.

В точности также, если для какого-то значения f значение $minD(f) \geqslant 0$ — то и для больших значений все будет аналогично.

Видно, что функция $minD(f)$ является монотонной, а значит по ней можно запустить двоичный поиск.

Сама функция $minD(f)$ выглядит примерно следующим образом:

– так как мы пытаемся минимизировать максимум, то, если нам не верят, считаем, что фантастичность отличалась от допустимой на минимально возможную величину, т.е. была равна $d + p + 1$.

– в случае, если нам верят, ситуация может развиваться двумя путями:

если дальше нам еще раз не поверят — значит, наш текущий уровень фантастичности не мог превосходить $fMax - p - 1$ (ведь потом нам придется прибавить $p + 1$ и все еще не выйти за границу), а также $d + p$ (иначе не поверили бы).

если же в дальнейшем нам будут только верить, можем считать, что наша фантастичность равна $min(fMax, d + p)$.

При реализации также надо проверить, что ваш уровень фантастичности нигде не стал отрицательным (такого тоже быть не может).

Заметим, что это не единственный способ решения задачи.

Задача D. Тщательная подготовка

Эта задача могла быть решена в рамках «жадного» подхода с использованием сортирующих структур данных. Сначала упорядочим пункты по времени, в которое становится возможным начать подготовку. Их можно сохранить в списке, множестве или массиве. Далее будем действовать следующим образом. Запустим « таймер» — специальную переменную, отсчитывающую дни, и заведем множество, способное хранить пункты, упорядоченные по времени, в которое их необходимо исполнить.

В момент изменения таймера будем изымать из структуры данных все пункты, для которых стала возможной подготовка, и помещать их в множество, упорядоченное по времени исполнения пунктов. После этого извлечем из множества сначала те пункты, время исполнения которых уже просрочено, а затем первый пункт, для которого время исполнения еще актуально. По завершении этой операции вновь изменим время таймера.

Если таймер стартует с нуля и каждый раз меняется на единицу, то получится решить все подзадачи, кроме последней. В последней подзадаче требуется каждый раз переходить к реально существующему моменту времени, выполняя таким образом «сжатие».

Задача E. Самый лёгкий путь

Для самой первой подзадачи можно написать любой рекурсивный перебор за $O(n^{maxK} \cdot q)$.

Для второй подзадачи можно написать динамику $dp[k][v]$ — мы закончили наш путь в вершине v, пройдя ровно k рёбер.

Получается сложность $O(k \cdot (n + m) + q)$, где m — количество ребер (по сути, это алгоритм Форда-Беллмана)

Для дальнейших оптимизаций обратим внимание, что в общем случае наш путь будет выглядеть следующим образом:

дойдем до какой-либо вершины $V0$ по кратчайшему пути $(1, V0)$, а затем будем ходить туда и обратно по ребру $(v0, v1)$, где $(v0, v1)$ — минимальное из ребер среди ребер с $v0$.

Тогда в общем случае путь будет выглядеть как

$dist[v0] + minEdge[v0] \cdot (k - distEdges[v0])$,

где $dist$ — вес какого-либо пути от вершины 1 до вершины $v0$, $distEdges$ — количество ребер в нем, $minEdge$ — вес минимального инцидентного вершине $v0$ ребра.

Но какой же путь выбрать?

В третьей подзадаче у нас дерево, поэтому до каждой вершины ровно один путь, поэтому для нее эту часть можно было выполнить за $O(n)$, просто запустив поиск в ширину или глубину.

Для четвертой и пятой подзадачи покажем следующий факт: нас интересуют только простые пути до вершины — если это не так, то внутри пути был цикл.

Если все ребра внутри цикла имеют вес больше или равный, чем $minEdge[v0]$, то мы можем удалить цикл из пути, а вместо него сделать соответствующее количество проходов по ребру $minEdge[v0]$.

В ином случае рассмотрим все ребра на пути до $v0$. Хотя бы одно из них (на цикле или нет) меньше, чем $v0$. Значит у нас есть вершина $v1$ такая, что:

$dist[v1] < dist[v0]$

$minEdge[v1] < minEdge[v0]$

Заменим кусок пути $(v1, v0)$ на проход по ребру $minEdge[v1]$. Так как это минимальное ребро из представленных на пути, то итоговая сумма может только уменьшиться. А значит выбранный нами путь $v0$ не может быть оптимальным.

Простой путь в графе из n вершин имеет путь не более $n - 1$. Поэтому мы можем для каждой длины пути от 0 до $n - 1$ включительно насчитать динамику из предыдущей подзадачи — $dp[k][n]$, причем максимальным k будет как раз $n - 1$. Эту часть мы выполнили за $O(nm)$.

Для решения подзадач 3 и 4 на полный балл для запросов $k > n - 1$ достаточно было отвечать за $O(n)$, просто проверяя все вершины в качестве конечной - итого $O(n + qn)$ и $O(nm + qn)$ соответственно.

Для решения пятой подзадачи заметим, что ответом у нас является значение прямой

$min(dp[n][v] + minEdge[v] \cdot (k - n))$,

где параметр k — это наш запрос.

Получили задачу «дано n прямых (по прямой на вершину), найдите прямую с минимальным $y(k)$».

Найдем пересечения всех прямых со всеми, получим $O(n^2)$ интересующих нас точек.

Отсортируем точки и будем обрабатывать запросы двумя указателями: перед ответом на запрос пройдемся по всем интересным точкам в промежутке и проведем обновление минимальной прямой, если это требуется.

Важно заметить, что необходимо хранить для каждой прямой, не перекрыли ли мы её ранее какой-либо другой прямой во избежание некорректных обновлений.

Таким образом, получаем решение за $O(n^2 + q + n^2 log(n^2))$.