Краткий разбор задач первого тура

Задача A. Последовательность

Автор задачи — В. Гурович

Если $k = 1, 2$, то ответ равен самому k. В противном случае первый раз число k встретится в тройке $(k-2), (k-1), k$, это будет $(k-2)$-я тройка в последовательности. Следовательно, искомая позиция равна $3k-6$.

Задача B. Офис

Автор задачи — В. Матюхин

Сумма всех чисел во входном файле равна суммарному количеству посещений офиса всеми его работниками в течение месяца. А это в точности 27 умножить на количество работников, поскольку каждый из них посетил офис ровно 27 раз.

Задача C. Книга

Автор задачи — Е. Андреева

Благодаря небольшим ограничениям можно было найти ответ перебором по количеству страниц. Достаточно запустить цикл от 4, считать количество цифр в номере текущей страницы и прибавлять к уже посчитанным. В момент, когда это количество станет равным N, текущее число страниц станет искомым.

Задача D. Карточки в метро

Автор задачи — В. Матюхин

Обозначим через T_i $(0 \le i \le 70)$ тарифную стоимость i поездок за месяц (эта таблица приведена в условии, $T_0 = 0$). Приведём эту таблицу к разумному виду T'_i $(0 \le i \le 70)$: $T'_{70} = T_{70}$, $T'_i = \min(T'_{i+1}, T_i)$ при $0 \le i \le 69$. Массивы T и T' будут различаться из-за немонотонности T.

Тогда за n поездок за месяц по одной карточке с неё спишется

$$cost(n) = T'_{\min(70,n)} + 10.0 \cdot (n > 0) + 15.71 \cdot \max(n - 70, 0)$$

Выражение $(n > 0)$ по определению равно 1, если $n > 0$, и 0 иначе.

Чтобы найти ответ, остаётся только перебрать количество использований k Петиной карточки за их совместные поездки (оно может изменяться от 0 до $2C$). Для каждого такого случая суммарные затраты братьев равны $cost(A + k) + cost(B + 2C - k)$. Искомый ответ равен минимуму этого выражения по всем $0 \le k \le 2C$.

Задача E. Стройка-2

Алексей Гусаков

Пусть x — некоторый угол. Проделаем следующие построения. Пусть A — точка на большей окружности, соответствующая вершине угла x.

Рассмотрим наименьший по величине угол с вершиной в точке A, содержащий две меньшие окружности. Обозначим точки пересечения границ этого угла с большей окружностью как B и C. Очевидно, что если в нашей задачи существует решение, то для некоторого угла x построенный таким образом треугольник ABC будет искомым. Пусть d — ориентированное расстояние от прямой BC до объединения двух меньших окружностей. В частности, если обе окружности лежат целиком по нужную сторону от прямой BC, то d положительно, а если по другую, то отрицательно. То есть, неформально говоря, d характеризует то, насколько нам подходит наш угол x. Введем функцию $f(x) := d$. Теперь решение задачи сводится к нахождению такой точки x, в которой функция примет неотрицательное значение. Для этого можно было использовать какой-нибудь численный метод для нахождения максимума функции на отрезке. Есть стандартный алгоритм для поиска максимума выпуклой вверх функции — троичный поиск. Идея заключается в делении отрезка на три равные части и отбрасывания левой или правой части в зависимости от значений в точках деления. В данном случае функция выпуклой, конечно, не является. Но неплохим способом оказывается разбиение отрезка на достаточно большое количество частей и надежда на то, что на каждой из них функция выпукла. Также есть метод Ньютона, который позволяет, зная начальное приближение для корня функции, хорошо его уточнить. В нашей задаче можно было с помощью метода Ньютона найти корни производной функции f.

Задача F. ЕГЭ *Автор задачи — В. Гуровиц*

Вообще говоря, эта задача не решается с помощью хитрой конструкции. Однако при данных ограничениях стандартная применявшаяся участниками схема давала неверный ответ лишь на нескольких тестах: 5 5, 9 9, иногда 9 11 (и так далее). Благодаря отсутствию штрафа за посылки большим их количеством можно было вычислить проблемные тесты, написать перебор и запустить для них на ночь. После чего вбить полученные результаты в программу и получить 100 баллов.

Однако существует и точное решение, которое быстрее перебора с возвратом. Этот подход называется динамическим программированием по профилю (см. http://informatics.mccme.ru/moodle/mod/book/view.php?id=290). Было решено не требовать реализации изломанного профиля (что несколько сложнее, но и работает быстрее). Основная трудность этой задачи заключается в том, что профилей получается слишком много, чтобы действовать влоб. По моим представлениям, в этой задаче есть два способа определить профиль. Далее будем считать, что $n \leq m$, где n — высота, а m — длина таблицы.

Первый способ по моему мнению более муторный, чем второй. Профилем назовем вертикальную полосу из n ячеек, в каждой из которых либо пусто, либо проведена одна из диагоналей, при этом никакие две не имеют общих концов. Тривиальный подход даёт оценку 3^n на количество профилей. Несложный подсчёт по рекуррентной формуле даёт результат 8 119 при $n = 10$. Количество переходов получается около 2 750 000. Из-за таких объёмов сами переходы желательно хранить в быстрой для добавления структуре, например в списке.

Для эффективной работы программы принято перенумеровывать профили (в данном случае числами от 0 до 8 118), ведь 3^{10} гораздо больше 8 119. Однако при должной сноровке можно обойтись и без этого. Появятся лишние проблемы с хранением переходов, зато можно избавить себя от комбинаторных изысканий, которые в данной задача оказываются не совсем тривиальными.

Пример реализации первого способа с нумерацией профилей есть в архиве решений жюри (http://olympiads.ru/zaoch/2009/problems/1-sols.rar) под названием `f_bv_ok2.cpp`.

Второй способ оказывается более коротким, особенно когда нужно найти только максимальное количество проведенных диагоналей без схемы. Рассмотрим столбец таблицы, в котором некоторая корректная конфигурация диагоналей. Некоторые узлы таблицы справа от данног столбца оказываются занятыми (в них заканчивается какая-то диагональ), а некоторые — свободными. Теперь чтобы провести диагонали в стобце справа корректно, нужно лишь, чтобы они не конфликтовали в самом столбце и чтобы они не начинались в занятых вершинах. Итак, из всей информации о предыдущем стобце достаточно знать лишь конфигурацию занятых и свободных вершин. Это и будет профилем. Формально — это число от 0 до $2^{n+1} - 1$, каждый бит которго содержит информацию о том, занят ли соответствующий узел таблицы, или свободен.

Практика показывает, что вычислять переходы для каждого профиля быстрее всего бэктрекингом (а не динамикой, об этом позже). В результате получится матрица p_{ij}, где $0 \leq i, j < 2^{n+1}$. Число p_{ij} равно максимальному количеству диагоналей, которое можно провести в квадратиках столбца так, чтобы слева они не пересекались с конфигурацией узлов i, а справа бы получалась в точности конфигурация узлов j. Осталось обсудить, как найти эту максимальную конфигурацию (это потребуется для вывода ответа).

Прежде всего, для всех m столбцов можно запустить полный перебор расстановок диагоналей. Получится $O(3^n m)$, что нас устраивает. Однако эта задача решается за $O(nm)$. Через i_k обозначим

k-й бит профиля i; $\bar{i}_k := 1 - i_k$. Пусть нам надо провести максимальное количество диагоналей в столбце высоты k, чтобы при этом они не пересекались с конфигурацией узлов i слева, и образовывали в точности конфигурацию j справа (здесь учитываются только первые $k+1$ бит этого числа). Через a_k обозначим ответ на этот вопрос при дополнительном условии: последняя (k-я) ячейка пуста. В b_k будет то же самое, только в последней ячейке стоит $/$, для c_k — $\backslash$. Тогда $a_0 = b_0 = c_0 = 0$, и верны следующие рекуррентные соотношения:

$$
\begin{aligned}
a_{k+1} &= \max(a_k\bar{j}_k, b_k\bar{j}_k, c_k),\\
b_{k+1} &= \max(a_k+1, b_k+1)\bar{i}_{k+1}j_k,\\
c_{k+1} &= \max(a_k\bar{j}_k+1, c[k]+1)\bar{i}_k j_{k+1}.
\end{aligned}
$$

Остаётся восстановить по этой динамике ответ. Реализация этого подхода имеется в `f_bv.cpp`.

Задача G. Эвакуация *Автор задачи — К. Абакумов*

Создадим ещё одну вершину, соединим её со всеми выходами. Для ответа на вопрос задачи остаётся обойти граф в ширину из созданной вершины, пометив вначале её расстоянием -1. При реализации можно не создавать вершину, а сразу добавить все выходы в очередь, пометить их расстоянием 0 и действовать дальше как при поиске в ширину.

Задача H. Последовательность-2 *Автор задачи — Д. Васильев*

Разложим данные числа на простые множители:

$$A = p_1^{\alpha_1}p_2^{\alpha_2}\dots p_k^{\alpha_k}, \qquad N = q_1^{\beta_1}q_2^{\beta_2}\dots p_m^{\beta_m}$$

Если в наборе $q_1, \dots q_m$ найдется число, которого нет среди $p_1, \dots, p_k$, то ни один элемент последовательности никогда не разделится на N. В противном случае вычислим минимальную степень r, в которую надо возвести A, чтобы результат делился на N. Несложно убедиться, что

$$r = \max_{p_i = q_j}\left\{\left\lceil\frac{\beta_j}{\alpha_i}\right\rceil\right\}$$

(для каждого q_j находится p_i, такое что $p_i = q_j$, $\lceil\cdot\rceil$ — округление вверх). Первоначальная задача сводится теперь к такой: какой элемент последовательности $x[k]$ первым станет больше либо равен r. Тогда $x[k+1] = A^{x[k]}$ будет первым элементом, делящимся на N.

Оказываетя, ограничения позволяют вычислять k простым возведением в степень (последовательным вычислением элементов):

```
index := 1;
c := 1.0; // c вещественного типа
while ( c + 1e-9 < r ) do begin
    c := A^c; // возведение в степень
    index := index + 1;
end;
```

Дело в том, что $r < 30$. Поэтому возведение в степень на втором шаге (когда $c = A$) возможно лишь в случае $A < 30$. Аналогичное будет происходить на любом шаге: операция возведения в степерь будет только с A и c меньшими 30. Поскольку 30^{30} вмещается в `double`, то переполнения происходить не будет.

После завершения цикла в `index` будет записан ответ на первоначальную задачу.

Задача I. Самолёт
Автор задачи — В. Матюхин

Ошибиться при реализации в этой задаче было значительно проще, чем придумать правильное решение. Пусть w_1 — время, прошеднее в первом городе с отлёта до прилёта самотётика; w_2 — время от прилёта до отлёта во втором городе. Тогда $(w_2 - w_1)$ равно удвоенному времени полёта.

В такую реализацию могут закрасться три ошибки. Во-первых, $(w_2 - w_1)$ может быть отрицательным. Нужно следить за этим, и при необходимости прибавлять 24 часа. Во-вторых, нельзя забывать округлять результат $c = (w_2 - w_1)/2$ вверх, если он получился не целый. В третьих, c может оказаться равным нулю. В таком случае нужно выводить не `00:00`, а `12:00`.

Кроме такого решения допустимо ещё было перебрать все возможные времена полёта (от 0.5 минут до 11 часов и 59.5 минут), для каждого из них разницу во времени (от -11 часов и -59.5 минут до 11 часов и 59.5 минут), и проверить входные данные моделированием. Это избавляет от разбора случаев, однако догадаться о том, что c и разница во времени могут быть только целыми либо полуцелыми, всё равно пришлось бы.

Задача J. Авангардная архитектура
Автор задачи — М. Густокашин

Сначала мы опишем тривиальную динамику за $O(W^4HN)$, после улучшим её до $O(W^3HN)$, а потом соптимизируем её до $O(W^2HN)$. Через a_{tc} обозначим приклекательность c-й квартиры на t-м этаже.

Пусть A^t_{ijm} равно максимальной привлекательности дома, занявшего первые t этажей и состоящего из m кубиков, причём на самом верхнем (t-м) под него пользованы кубики с i-го по $(j-1)$-й включительно. Тогда

$$A^t_{ijm} = \sum_{c=i}^{j-1} a_{tc} + \max_{l,r}\{A^{t-1}_{l,r,m-j+i}\}$$

Максимум берётся по всем $l < r$, таким что отрезки $[l, r-1]$ и $[i, j-1]$ пересекаются, это условие берётся из “законов физики” задачи.

Для удобства пересчитаем a так, чтобы a_{tc} было равно сумме привлекательностей квартир t-го этажа с самой левой до $(c-1)$-й включительно. Тогда вместо $\sum\limits_{c=i}^{j-1} a_{tc}$ можно писать $a_{tj} - a_{ti}$.

Пусть теперь B^t_{ijm} — максимальная привлекательность дома из m кубиков, занявшего первые t этажей, причём на самом верхнем этаже кубики с i-го по $(j-1)$-й точно использованы (также использованные кубики могут быть непосредственно слева от i и справа от $(j-1)$). Чтобы написать рекуррентное соотношение, рассмотрим несколько случаев. В первом слева от i используется ещё хотя бы один кубик, во втором — справа от $j-1$, в третьем на t-м этаже заняты только кубики с i-го по $(j-1)$-й. Тогда

$$B^t_{ijm} = \max\Big\{B^t_{i-1,j,m}; B^t_{i,j+1,m}; a_{tj} - a_{ti} + \max_{i \le p < j} B^{t-1}_{p,p,m-j+i}\Big\}$$

Чтобы ускорить посчёт этой формуле, можно перед вычислением t-го этажа предпосчитать все максимумы $C_{ijm} = \max\limits_{i \le p < j} B^{t-1}_{p,p,m}$. Это действие не повлияет на асимптотику решения, зато уменьшит основные затраты в W раз.

Реализацию этого решения можно посмотреть в `j_bv.cpp`.

Задача K. Электронное табло
Автор задачи — А. Шестимеров

Обозначим строки через s_1 и s_2, а их длину — через n. В этой задаче для каждого сдвига длины $0 \le a < n$ нужно определить, равен ли префикс строки s_1 длины $n-a$ суффиксу строки s_2 (той же длины), и для всех случаев равенства выбрать минимальное a. Другими словами, нужно найти максимальный префикс s_1, который равен суффиксу s_2. Рассмотрим строку $s = s_1 s_2$ и построим для неё префикс-функцию π. Её значение $\pi(2n)$ и будет ответом на вопрос задачи.

Задача L. Заливка

Автор задачи — О. Самойленко

Вместо таблицы из 0 и 1 будем работать с графом, построенном на областях одного цвета как на вершинах. Он будет планарным и двудольным (хотя в нашем решении это роли не играет). За один ход можно выбрать вершину и перекрасить её в противоположный цвет, после чего слить со всеми соседями в одну. Через $\rho(u, v)$ будет обозначать кратчайшее расстояние между u и v.

Центральной вершиной графа назовем такую вершину, что максимальное расстояние от неё до вершин графа минимально, т.е. $\left(\max\limits_v \rho(u,v)\right) \to \min$. Центром графа называется множество центральных вершин. Выберем из них одну, её обозначим через u, максимальное расстояние от неё до вершин через k ($k = \rho(u) = \max\limits_v \rho(u,v)$). Существует путь p длины не менее $2k$, содержащий u, и являющийся кратчайшим. Несложными рассуждениями (и небольшим разбором случаев) можно показать, что за один ход длину этого пути можно уменьшить самое большое на 2 (важно, что p кратчайший). Таким образом, для перекраски всего графа потребуется не менее k ходов. Однако, если кликать всё время на u, то за k ходов весь граф заведомо перекрасится в один цвет. Обосновано, таким образом, что ответ на вопрос задачи равен k.

Итак, в этой задаче нужно вычислить $k = \min\limits_u \rho(u)$. Следовательно, надо в некотором порядке просматривать вершины графа, вычисляя для каждой значение ρ, а когда истечёт время — вывести минимальный найденный результат. Время истечёт почти наверняка, ведь на одну вершину тратится $O(NM)$ действий, и асимтотика всего алгоритма равна $O(N^2M^2)$.

Если порядок перечисления вершин прямой или рандомный, то скорее всего такое решение не пройдет пары-тройки тестов. В решении `l_sr.cpp` вершины отсортированы по удалённости от центра таблицы, что позволяет за выделенное время проходить все тесты.

Ещё в этой задаче можно было использоваться отсечения другого характера. Пусть $\rho(u) = \rho(u, v_0)$. Рассмотрим ещё одну вершину v_1. По неравенству треугольника (из того, что пути кратчайшие) $\rho(v_0, v_1) \geq \rho(u, v_1) - \rho(u, v_0)$. Поэтому если $\rho(u, v_1) - \rho(u) \geq k$, где k — текущий ответ, то $\rho(v_1) \geq k$, поэтому просматривать её не имеет смысла.

С такой эвристикой программа работает около 0.1 секунды. Вот соответствующий ключевой фрагмент кода Александра Желубенкова, который первым получил `OK` по этой задаче.

```
for i:=1 to kcol do
  if fl1[i]=0 then
  begin
    get_ans(i);
    for j:=i+1 to kcol do
      if anst-d[j]>=ans then fl1[j]:=1;
  end;
```

Краткие разборы писали:
Алексей Гусаков (по задаче «E»)
Василевский Борис <vasilevskiy.boris@gmail.com>