Mail.Ru Group 796,33
Компания

4 мая в 16:51

Итоги третьего раунда Russian Code Cup 2017

Спортивное программирование*, Алгоритмы*, Блог компании Mail.Ru Group

29 апреля прошёл третий и заключительный квалификационный раунд Russian Code Cup 2017. По традиции чуть-чуть хвастаемся результата выкладываем разбор задач.

A. Электронные таблицы
B. Смертный бой
C. Слегка палиндромные числа
D. Дерево и многочлены
E. Красивый отчёт

В этот раз зарегистрировался 5681 человек (из них 473 новых участника) — снова бьём свои же рекорды. По результатам раунда мы взяли 202 участника в отборочный раунд. Все пять задач успешно решили пять человек.

Трое лучших:

1. На первом месте со значительным отрывом от преследователей (3 минуты) оказался Игорь Пышкин из Санкт-Петербурга.
2. Второе место занял пользователь xyz2606.
3. На третьем месте с небольшим отставанием от второго места (всего лишь 6 секунд) — Shik Chen с Тайваня.

Кроме того, в топ-10 попали:

1. Олег Меркурьев, Екатеринбург, Россия
2. Иван Смирнов, Москва, Россия
3. Александр Останин, Москва, Россия
4. Дмитрий Якутов, Санкт-Петербург, Россия
5. Maleki Mohammadreza, Тегеран, Иран
6. Kohji Liu, Токио, Япония
7. Константин Хадаев, Новосибирск, Россия

Поздравляем всех прошедших в отборочный раунд, который ждёт нас 14 мая. Ну а теперь — разбор задач.

A. Электронные таблицы

Ограничение по времени — 1 секунда
Ограничение по памяти — 256 мегабайт

Петя разрабатывает собственный редактор электронных таблиц. В его редакторе столбцы будут называться следующим образом: первые 26 столбцов — соответствующими буквами латинского алфавита: A, B, C, ..., Z. Следующие столбцы, начиная с 27, называются парами букв: AA ..., AZ, BA, BB, ..., BZ, ..., ZZ. Далее используются тройки букв: AAA, AAB, AAC, ..., затем четверки, и так далее.

Теперь Пете необходимо по номеру столбца определить его название. Помогите ему написать соответствующий фрагмент кода.

Формат входных данных

Входные данные содержат несколько тестовых примеров. Первая строка содержит целое число t — количество тестов ($1 \le t \le 1000$).

Следующие t строк содержат по одному целому числу k ($1 \le k \le 10^9$).

Формат выходных данных

Для каждого теста выведите в отдельной строке название k-го столбца.

Примеры

Входные данные
4
1
10
100
1000

Выходные данные
A
J
CV
ALL

Разбор задачи

Вычтем из k единицу, перейдя к нумерации столбцов с 0. Выясним сначала, сколько символов входит в название столбца. Для этого будем последовательно отнимать от k степени числа 26, пока очередная степень не окажется больше оставшегося числа k'.

Теперь для получения имени столбца достаточно перевести k' в 26-ричную систему счисления с цифрами A–Z и дополнить ведущими нуля (точнее, символами A) до необходимого числа символов. Это можно сделать, используя стандартные методы языка программирования (пра при этом получатся цифры 0–9, A–P, их надо будет заменить) либо самостоятельно реализовав алгоритм перевода в систему счисления с заданным основанием.

B. Смертный бой

Ограничение по времени — 1 секунда
Ограничение по памяти — 256 мегабайт

Недавно Вася начал играть в новую компьютерную игру. В конце первого уровня его ждал босс, которого нужно победить, чтобы продолжит проходить игру. У Васи есть отряд из n персонажей, i-й из которых имеет h_i единиц здоровья и a_i единиц атаки. Босс же имеет H единиц здо A единиц атаки.

Сражение происходит следующим образом:

- Если у Васи больше нет персонажей, он проиграл;
- Иначе, он выбирает одного из них и выставляет на бой против босса;
- Пусть Вася выбрал персонажа номер i. Тогда:
  - Персонаж нападает на босса и снимает ему a_i единиц здоровья;
  - Если босс еще жив, то есть его количество единиц здоровья положительно, он нападает в ответ и снимает персонажу A единиц здоровья;
  - Пока и персонаж и босс оба живы, действия повторяются.
- Если босс умер, сражение заканчивается, Вася побеждает. В противном случае все действия повторяются.
- Так как Васе еще предстоит проходить следующие уровни, он хочет потерять как можно меньше персонажей. Помогите ему — найдите минимальное возможное количество персонажей, которое он может потерять в сражении с боссом. Если же Васе не победить даже исп всех персонажей, выведите -1.

Рассмотрим тестовые примеры.

В первом примере Вася может сначала отправить сражаться персонажа 2, который три раза нанесет удар, уменьшит количество единиц здор босса на 12, и лишь затем погибнет, а затем отправить персонажа 3, который сразу же снимет у босса уже 6 единиц здоровья и убьет его. Та образом, Вася потеряет только одного персонажа.

Во втором примере даже если Вася отправит в бой по очереди всех персонажей, босс не будет убит.

Формат входных данных

Входные данные содержат несколько тестовых наборов. В первой строке задано количество тестов t ($1 \le t \le 1000$).

Каждый из следующих t тестов описывается следующим образом: в первой строке описания теста содержится три целых числа n, H, A — количество персонажей у Васи, количество единиц здоровья и атаки у босса соответственно ($1 \le n \le 10^5$, $1 \le H, A \le 10^9$).

В каждой из следующих n строк содержится два целых числа h_i, a_i — количество единиц здоровья и атаки у i-го персонажа соответственно ($\le h_i, a_i \le 10^9$).

Гарантируется, что сумма n во всех тестах одних входных данных не превосходит 10^5.

Формат выходных данных

Для каждого теста выведите ответ на него — минимальное количество персонажей, которое может потерять Вася, чтобы одержать победу на боссом, или -1, если он в любом случае проиграет сражение.

Примеры

Входные данные
```
2
4 18 4
4 5
9 4
1 6
3 3
4 27 4
4 5
9 4
1 6
3 3
```

Выходные данные
```
1
-1
```

Разбор задачи

Посчитаем количество единиц здоровья, которое i-й персонаж до своей гибели может снять боссу в одиночку: $p_i = a_i \bullet \lceil h_i / A \rceil$. После этого отсортируем персонажей по убыванию p_i и будем посылать их к боссу по одному в этом порядке, пока он не будет убит.

C. Слегка палиндромные числа

Ограничение по времени — 1 секунда
Ограничение по памяти — 256 мегабайт

Будем называть целое число слегка палиндромным, если первая цифра его десятичной записи без ведущих нулей совпадает с последней. Ва задача — посчитать количество слегка палиндромных чисел на отрезке с l по r, включительно.

В первом тестовом примере слегка палиндромные числа — 8, 9, 11 и 22.

Во втором — 1251 и 1261.

Формат входных данных

Входные данные содержат несколько тестов. В первой строке задано число t — количество тестов ($1 \le t \le 5 \cdot 10^4$).

Каждый тест задаётся одной строкой, которая содержит целые числа l и r ($1 \le l \le r \le 10^{18}$).

Формат выходных данных

Для каждого теста выведите в отдельной строке количество слегка палиндромных чисел на заданном отрезке.

Примеры

Входные данные
```
3
8 25
```

1251 1266
12 21

Выходные данные
4
2
0

Разбор задачи

Научимся считать количество слегка палиндромных чисел среди первых n натуральных чисел. Чтобы получить количество таких чисел на отрезке с l по r, нужно посчитать их количество среди первых r и $l-1$ натуральных чисел и вычесть одно из другого.

Все числа от 1 до 9 слегка палиндромные, поэтому если $n \le 9$, то ответ равен n. В противном случае прибавим к ответу 9 и посчитаем количество слегка палиндромных чисел на отрезке с 10 по n. Рассмотрим десятку чисел от $10k$ до $10k+9$, где $k \neq 0$. Среди них ровно одно слегка палиндромное, так как последняя цифра должна равняться первой цифре k. Таким образом, есть по одному слегка палиндромному на отрезках от 10 до 19, от 20 до 29, и т. д.

Пусть $n = 10k + d$, где $0 \le d \le 9$. Тогда на отрезке от 10 до n находится k слегка палиндромных чисел, если d не меньше первой цифры k (а равна первой цифре n), и $k-1$, если меньше.

D. Дерево и многочлены

Ограничение по времени — 2 секунды
Ограничение по памяти — 256 мегабайт

Ник учится на первом курсе. На курсе алгоритмов он изучает деревья. На курсе алгебры Ник изучает многочлены. А еще Ник очень любит прменять увиденное. Недавно он придумал задачу, но никак не может ее решить. Помогите ему.

Дано подвешенное дерево с n вершинами, пронумерованными от 1 до n. Исходно в каждой вершине дерева записано число 0. Для вершины обозначим как $d[v]$ глубину вершины v — количество вершин на пути от корня дерева до вершины v, в частности корень имеет глубину 1.

Дано число k. Требуется обрабатывать запросы двух видов.

1. Дана вершина v и многочлен $q(t) = q0 + q1t + q2t^2 + ... + qkt^k$. Для каждой вершины u в поддереве вершины v требуется прибавить к зна в этой вершине значение $q(d[u])$.
2. Дана вершина v и многочлен $q(t) = q0 + q1t + q2t^2 + ... + qkt^k$. Для каждой вершины u на пути от корня до вершины v, включительно, тр прибавить к значению в этой вершине значение $q(d[u])$.

Все действия выполняются по модулю $10^9 + 7$.

Помогите Нику понять, какие числа будут записаны в вершинах дерева после выполнения всех операций.

Формат входных данных

Входные данные содержат несколько тестов. В первой строке задано число t — количество тестов ($1 \le t \le 10^5$).

Далее следует описание тестов. Описание теста начинается с числа n — количества вершин в дереве, и числа k — максимальной степени многочленов ($1 \le n \le 10^5$, $1 \le k \le 20$).

Далее следует n чисел $p_1, p_2, ..., p_n$, число p_i задает номер родителя вершины i, либо равно 0, если вершина i является корнем. Гарантируетс числа p_i задают корректное подвешенное дерево.

Следующая строка содержит число q — количество запросов ($1 \le q \le 10^5$). Следующие q строк содержат запросы, описание запроса состоит числа t, равного 1 или 2 — типа запроса, после которого следует число v — номер вершины, и затем $k+1$ целое число $q_0, q_1, ..., q_k$ — коэффициенты многочлена ($0 \le q_i < 10^9 + 7$).

Гарантируется, что сумма значений n во всех тестов одних входных данных не превышает 10^5. Также сумма значений q во всех тестов одних входных данных не превышает 10^5.

Формат выходных данных

Для каждого теста выведите n чисел, для каждой вершины выведите число, которое в ней окажется после выполнения всех запросов. Еще р напомним, что все вычисления проводятся по модулю $10^9 + 7$.

Примеры

Входные данные
1
4 2

2 4 2 0
3
1 2 0 1 0
2 3 1 0 0
2 1 0 0 1

Выходные данные
12 7 4 2

Разбор задачи

Сформулируем ключевые идеи решения. Во-первых, заметим, что задачу можно решать независимо для двух типов запросов, поскольку операции не зависят от значений в вершинах.
Во-вторых, вместо хранения чисел в вершинах будем хранить сумму всех многочленов, результаты которых применялись к значениям в вершинах. Тогда в конце просто вычислим значение многочлена в каждой вершине от её глубины и получим значение в вершине.

Однако мы не можем выполнять запросы в явном виде, рассматривая все вершины, фигурирующие в запросе, и прибавляя к хранящимся в многочленах многочлен из запроса: такое решение будет работать за $O(n^2 \cdot k)$ и не уложится в ограничения по времени. Поэтому воспользу идеей отложенных операций.

В запросах первого типа нужно добавить многочлен *q(x)* в каждую вершину поддерева *v*. Для этого просто оставим пометки, что это надо сделать: будем хранить в вершине *v* многочлен *push1[v]*, при выполнении запроса первого типа прибавим многочлен *q* к многочлену *push1* Аналогично для запросов второго типа будем использовать многочлен *push2[v]*, который для каждой вершины будет хранить сумму многоч из запросов второго типа к этой вершине.

Пускай теперь *sum1[u]* — сумма всех многочленов, которые нужно было учесть в вершине *u* в запросах первого типа. Для вычисления *sum* выполним поиск в глубину от корня дерева, который будет поддерживать многочлен *cur*, прибавлять к нему значение *push1[v]* при посеще новой вершины *v* и вычитать его обратно при выходе из неё. Соответственно, находясь в вершине *u*, присваиваем *sum1[u] = cur*.

Точно так же обозначим как *sum2[u]* сумму всех многочленов, которые нужно было учесть в вершине u в запросах второго типа. Для вычисления *sum2[u]* также попользуемся обходом в глубину. Значение *sum2[u]* равно сумме значений *sum2[v]* для детей *u* и значения *push2[u]*.

Получив многочлены *sum1[u]* и *sum2[u]* в каждой вершине, вычислим их значение от величины *d[u]* и сложим, результат и будет ответом. Время работы решения — *O(nk)*.

E. Красивый отчёт

Ограничение по времени — 5 секунд
Ограничение по памяти — 256 мегабайт

Андрей занимается анализом графа подписок в одной социальной сети. Этот граф является ориентированным: если пользователь *a* подписа пользователя *b*, то пользователь *b* не обязательно подписан на пользователя *a*.

Менеджер Андрея попросил его посчитать для каждого пользователя *x*, сколько существует пользователей *y*, таких, что от пользователя *x* м добраться в графе подписок до пользователя *y*.

Печатать точное значение не имеет смысла, потому что оно смотрится некрасиво и моментально устареет, поэтому, вас интересует лишь при значение. Выполните задание менеджера и найдите эти значения с ошибкой не более чем в два раза.

Формат входных данных

В первой строке заданы целые числа n и m ($1 \le n, m \le 200\,000$) — число пользователей социальной сети и число ситуаций, когда один пользователь подписан на другого.

Далее, в m строках идёт описание графа, i-я из этих строк содержит два целых числа a_i и b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) и означает, что пользовател подписан на пользователя b_i. Каждая упорядоченная пара (a_i, b_i) встречается во входных данных не более одного раза.

Формат выходных данных

Выведите n целых чисел q_i — оценку на количество пользователей y, таких, что от пользователя i можно добраться в графе подписок до пользователя y. Если настоящее количество таких пар равно z_i, должны выполняться неравенства $q_i \le 2z_i$ и $z_i \le 2q_i$. Кроме того, допустимо д более, чем 10 пользователей вывести q_i, не удовлетворяющее указанным ограничениям.

Примечание: жюри использует для проверки ваших ответов точное количество искомых пользователей. При этом жюри не гарантирует, что число можно найти, уложившись в ограничения по времени и памяти, и рекомендует использовать возможность вывести приблизительное значение.

В примере от пользователя 1 можно добраться до всех пяти пользователей. Однако, показанный ответ 7 тоже допустим, так как отличается более, чем в два раза. Аналогично, допустимым является ответ 2 для пользователя 4.

Примеры

Входные данные
5 6
1 2
1 3
2 4
2 5
3 5
4 2

Выходные данные
7
3
2
2
1

Разбор задачи

Перед решением основной задачи решим следующую, пока слабо связанную с исходной задачу: на вход даётся последовательность элеме
($a_1, a_2, ..., a_n$), и нужно посчитать число различных элементов в последовательности. Особенность этой задачи:

1. Элементы можно считывать только последовательно и только один раз: после считывания элемента a_i произойдёт считывание $a_i + 1$, вернуться к a_i нельзя.
2. Вы не можете использовать больше O(1) дополнительной памяти.

Чтобы подойти к решению этой подзадачи, вспомним, что математическое ожидание минимума из m равномерно распределённых на отрез
[0;1] случайных величин равно $1 / (m + 1)$. Например, если вы возьмёте четыре случайных числа из отрезка [0;1], то минимальное из этих
в среднем будет равно 0,2.

Возьмём хеш-функцию $h: A \to [0;1]$, отображающую элементы a_i в числа на отрезке [0;1]. Посчитаем значение $M = min\ \{h(a_1), h(a_2), ..., h(a_
качестве ответа на задачу вернём $1 / M - 1$. Проблема этого решения: нам может «не повезти» — например, $h(a_i)$ окажется очень маленьким
существенно повлияет на ответ. Чтобы сгладить эту проблему, повторим описанный процесс несколько раз для различных хеш-функций
$h_1, h_2, ..., h_k$ для некоторого фиксированного k и усредним полученные результаты.

Вернёмся к нашей задаче поиска числа достижимых вершин. Рассмотрим сначала случай ациклического графа. Он отличается от графа с
циклами в том числе тем, что у него существует топологическая сортировка: такой порядок следования вершин, при котором рёбра графа
только от вершин, стоящих в порядке позже, к вершинам, стоящим в порядке раньше. Топологическая сортировка позволит посчитать для
графа, в котором в каждой вершине записано случайное число из отрезка [0;1], какое минимальное число достижимо из каждой вершины.
можно сделать с помощью динамического программирования: рассматривание вершин в порядке топологической сортировки позволит посч
значение для текущей вершины на основе уже посчитанных достижимых из неё. Посчитав минимальное достижимое число M_i из каждой
вершины, можно указать в качестве ответа для данной вершины $1 / M_i - 1$. Описанное требуется повторить несколько раз и усреднить
результаты.

Однако в данной задаче в графе могут быть циклы. Чтобы решить эту проблему, построим конденсацию графа. В полученном ациклическо
графе в каждой вершине находится на самом деле несколько вершин начального графа. Для всех этих вершин ответ будет один и тот же.
Отличие в подсчёте ответа будет заключаться в том, что в вершину пишется не случайное число из [0;1], а минимум из wi случайных чисе
[0;1], где w_i — число вершин начального графа, содержащихся в i-й вершине сконденсированного.

Более подробно с методом решения этой и некоторых других подобных задач любопытный читатель может ознакомиться в статье «Size-Estin
Framework with Applications to Transitive Closure and Reachability», доступной по ссылке.

Отборочный раунд совсем скоро

Напоминаем: отборочный этап состоится 14 мая с 13:00 до 15:00 по московскому времени. Чуть больше 600 участников примут участие в ра
только 50 лучших попадут в финал, который будет проведён 10 сентября.

Теперь благодаря официальному чату чемпионата вы можете задать свой вопрос напрямую организаторам. А ещё — попросить совета, подел
предложениями о решении задач, поговорить на разные темы, связанные со спортивным программированием. Вам помогут свежими идеями
добрыми словами.

rcc2017, rcc, russian code cup, разбор задач, алгоритмы

рейтинг
Mail.Ru Group 796,33
Facebook Twitter Вконтакте Instagram

ПОХОЖИЕ ПУБЛИКАЦИИ

21 сентября 2012 в 09:31
Russian Code Cup 2012: подробный разбор задач с финала в картинках, видео и примерах
+52 43,8k 244 25

25 июня 2012 в 12:00
Russian Code Cup 2012: подробный разбор задач с отборочного раунда (полуфинал)
+50 30,6k 198 20

13 июня 2012 в 19:10
Russian Code Cup 2012: Разбор задач третьего квалификационного раунда
+25 12,2k 98 14

Комментарии (0)

САМОЕ ЧИТАЕМОЕ Разра

Сейчас Сутки Неделя Месяц

Получил 1.2K звезд на GitHub с ужасной архитектурой. Как?
+22 66,2k 110 39

Реализация псевдо-3D в гоночных играх
+90 26,4k 239 16

Индексы в PostgreSQL — 1
+102 24k 430 57

Выбор MQ для высоконагруженного проекта
+30 19,4k 169 51

Алгоритм Джонкера-Волгенанта + t-SNE = супер-сила
+63 17,6k 200 2

ИНТЕРЕСНЫЕ ПУБЛИКАЦИИ

Построение графиков с двумя независимыми осями в Matlab GT
462 2 1

Valve пытается сделать работу саппорта Steam более прозрачной и эффективной GT
1,2k 1 2

Ко дню связи: история IP-телефонии
1,8k 16 11

Подключаем геймпад от PS1/PS2 к Raspberry pi GT

1,2k 3 0

Считаем до трёх: три

2,2k 10 16