**Разбор задач окружного этапа Всероссийской олимпиады школьников**

***I тур***

### *Задача A. Газета*

Здесь требовалось максимизировать время, затраченное мэром на срывание газет, соблюдая при этом условие на разницу между количеством газет, сорванных с первой, второй и третьей попыток. Анализ вариантов приводит к следующим наилучшим случаям.

Обозначим $c = n/3$ (деление выполняется нацело). Тогда:

- для $n \bmod 3 = 0$ количество газет, сорванных с первой попытки, следует выбрать равным $c-1$, количество газет, сорванных со второй попытки, – равным $c$, а количество газет, сорванных с третьей попытки, – равным $c+1$;
- для $n \bmod 3 = 1$ количество газет, сорванных с первой попытки, следует выбрать равным $c$, количество газет, сорванных со второй попытки, – также равным $c$, а количество газет, сорванных с третьей попытки, – равным $c+1$;
- для $n \bmod 3 = 2$ количество газет, сорванных с первой попытки, следует выбрать равным $c$, количество газет, сорванных со второй попытки, – равным $c+1$, и количество газет, сорванных с третьей попытки, – также равным $c+1$.

После этого останется лишь посчитать время, которое затратил мэр на срывание газет, и сопоставить его с временем, которое затратил на срывание газет журналист.

### *Задача B. Реклама*

Ограничения позволяют решать задачу перебором. Посчитаем для каждого перекрестка $(i, j)$ расстояния от него до каждого из магазинов и выберем среди этих расстояний максимальное $d_{i,j}^{max}$. Далее останется найти минимальное из всех $d_{i,j}^{max}$.

### *Задача C. Парк*

Опишем один из возможных путей решения. Найдём сначала точку пересечения линии, соединяющей дубы, и линии, соединяющей центры площадок. Затем определим расстояния от центров площадок до линии, заданной дубами. После этого рассмотрим допустимые случаи взаимного расположения дубов и площадок. Поскольку гарантируется, что канал можно построить, то принципиально различных случаев два, и оба они проиллюстрированы рисунками в условии: когда дубы уменьшают ширину канала и когда они не влияют на неё.

Когда линия дубов расположена “между” площадками (как на рисунке, иллюстрирующем первый пример), нужно будет выбрать в качестве ширины канала большее из расстояний от линии, заданной дубами, до площадок (с учетом радиусов площадок и “берегов” единичной ширины и р).

Когда же расположение линии дубов не уменьшает ширину канала (как на рисунке, иллюстрирующем второй пример), то в формулу войдут оба расстояния от линии, заданной дубами, до площадок (и, конечно, оба радиуса).

### *Задача D. Надёжное дело*

Отсортируем проекты по неубыванию процентов от суммы необходимого финансирования. Заведём массив из 100000 элементов (по одному элементу на каждую тысячную долю процента) и выполним суммирование вложений в проекты нарастающим итогом (посчитаем частичную сумму). После этого предпросчёта ответ на запрос о привлечённом финансировании для некоторой величины процента выполняется обращением к соответствующему элементу массива.

### *Задача E. Сертифицируй это*

Пусть Потап заранее определил время $k$ (количество дней), через которое он собирается получить лицензию. Тогда Потапу нет смысла посещать города, которые Брюквин посещает за первые $k-1$ дней. Из оставшихся городов Потапу выгодно выбрать $k$ самых "дорогих", а итоговая стоимость лицензии будет равна числу $k$, помноженному на вес минимального взятого города.

Простейшая реализация этого подхода имеет сложность $O(n^2 \cdot \log n)$ и не укладывается в ограничения по времени. Заметим, что при увеличении $k$ на единицу множество городов, посещаемое Потапом, меняется не более, чем на два элемента – один элемент из-за посещения дополнительного города Брюквиным и один элемент из-за посещения "дополнительного" города Потапом. Будем поддерживать (используя, например, set в C++ или TreeSet в Java) множества выбранных и невыбранных городов. Тогда на каждой итерации мы должны удалить из соответствующего множества город, посещенный Брюквиным, после чего, если этот город был в множестве выбранных, взять два самых "дорогих" города из множества невыбранных в множество выбранных, в противном случае – один.

## *II тур*

### *Задача A. Красивый номер*

Из двух цифр можно составить 8 различных номеров автомобилей. После этого останется определить, какие из них допустимы (т.е. превосходят по числовому значению номер покровителя Митрофана), и выбрать из них три наименьших. Если выбрать три наименьших не получится, вместо недостающих придётся вывести –1.

### *Задача B. Три шанса*

В задаче требовалось аккуратно подсчитывать, в какой срок будет сдано то или иное задание. Можно было, например, завести массив из 100000 элементов (по количеству потенциально существующих курсов), и для каждого элемента выставлять одну из пометок: "не приступал к изучению", "выполнил задания до первого срока", "выполнил задания до второго срока", "выполнил задания после второго срока". Каждая из этих пометок является "поглощающей" по отношению к предыдущей, а вот "обратной силы" пометки не имеют: если какое-либо задание по курсу было сдано после второго срока, то выполненное позже в срок задание по этому же курсу уже не изменит пометку в массиве.

По завершении обработки данных следует посчитать количество пометок “выполнил задания после второго срока” и вывести полученное число и номера соответствующих элементов.

### *Задача C. Короче...*

Сформируем сначала “словарь”, в котором будем хранить ключевые слова. Для этого можно как воспользоваться структурой set в C++ или TreeSet в Java; можно хранить слова и в массиве (их количество невелико и позволяет выполнить линейный поиск).
Что же касается текста, его обработаем следующим образом. При прочтении очередного слова проанализируем, является ли оно ключевым. Если слово – ключевое, его придётся пропустить, если же слово не является ключевым, сохраним его длину. После того, как весь текст будет проанализирован, упорядочим массив длин по невозрастанию.
Теперь можно применить “жадную стратегию” и пошагово уменьшать длину текста на величину наиболее длинного слова, которое в текущий момент можно удалить. Разумеется, не следует забывать о пробелах.
Если в процессе удаления слов в какой-то момент длина текста всё же примет значение, меньшее или равное $n$, следует вывести количество удалённых слов. Если же массив длин слов исчерпан, а длина текста всё ещё остаётся слишком большой, придётся вывести –1.
Заметим, что достаточно простой способ учета пробелов состоит в том, чтобы добавить “виртуальный” пробел в конец текста, после чего учитывать каждое слово вместе с последующим пробелом. В этом случае числа $m$ и $n$ “виртуально” увеличиваются на 1, все длины слов также увеличиваются на 1, и после этого “виртуальная” длина текста при удалении слова уменьшается на его “виртуальную” длину.

### *Задача D. Игра*

Заведём $n$ стеков – по одному на каждый стакан, а также массив, в $j$-ом элементе которого будет содержаться номер стека, в котором стакан #$j$ находится в текущий момент. В начальный момент времени в каждом из стеков находится стакан с соответствующим номером (а в элементе массива #$j$ – число $j$).
При перемещении очередного стакана нужно удалить его с вершины стека, где он находился, поместить его на вершину нового стека и обновить значение в массиве.
При запросе о местоположении стакана следует обратиться к массиву и определить, в каком из стеков он находится. После чего вывести в качестве ответа на запрос номер стакана, находящегося на вершине соответствующего стека.

Можно обойтись и вовсе без стеков. Рассмотрим детально операцию перемещения. Обратим внимание, что при этой операции у многих стаканов (всех в стопке) меняется самый верхний над ними, но при этом только у перемещаемого стакана меняется самый нижний под ним. Аналогично, только у перемещаемого стакана меняется стакан, который находится непосредственно под ним. Кроме того, только для двух лежащих в самом низу стопки меняется верхний стакан стопки.
Будем хранить три массива:
в $a$ – для каждого стакана номер самого нижнего стакана в его стопке;
в $b$ – для каждого стакана номер стакана, находящегося непосредственно под ним в стопке;
в $c$ – для каждого находящегося внизу стопки стакана стакан, находящийся вверху стопки, в которой он лежит.

Согласно наблюдениям выше, значения в этих трёх массивах меняются один или два раза при запросе типа 1; при этом, используя массивы *a* и *c*, можно вывести ответ на запрос типа 2.

### *Задача E. Утомлённые не солнцем*

Пусть *k* (= 5) – максимальная сложность. Тогда утомлённость от одной задачи имеет *k* возможностей.

Будем решать задачу методом динамического программирования. В качестве состояния будем использовать утомлённость от предыдущей задачи *x* и оставшийся набор задач *S* (как неупорядоченный набор).

Мы начинаем с значения $x=0$ утомлённости от предыдущей задачи и набора *S*, данного в условии. На каждом этапе мы перебираем, задачу какой сложности $y=(1..k)$ из текущего набора *S* мы разбираем следующей, тогда новый набор $S'=S\setminus\{y\}$ (предыдущий без задачи), а новое значение утомлённости $x'=(y+x)/2$ .

Если *S* пуст, то ответ $w(S,x)=0$ , в противном случае ответ получается оптимальным выбором *y*: $w(S,x)=min(w(S',x')+x')$ .

Оценим количество состояний в этой динамике. Количество возможных наборов *S*, достижимых из начального, есть произведение $(c(i)+1)$ количеств чисел $i$ в начальном наборе. Это значение максимально, если все $c(i)$ равны, итого для $n>k$ получаем $k\cdot((N+k)/k)^k$ . Итоговая сложность нашего решения есть $O(k^2\cdot((N+k)/k)^k)$ времени и $O(k\cdot((N+k)/k)^k)$ памяти, в ограничениях задачи это $25\cdot21^5$ примерно 100 млн (достаточно быстрых) операций и примерно 80МБ, что с запасом укладывается в ограничения задачи.