A Нужно выбрать более короткий из путей Киев - Кременчуг и Киев - Харьков - Кременчуг. Их длины a и $b+c$. Ответ $min(a, b+c)$.
Сложность $O(1)$.

B Пусть $max(e) = E$. Тогда для всех $r_i \leq E$ можно взять $e_i = r_i$ и тогда $|r_i - e_i| \cdot c_i = 0$, а для $r_i > E$ выгодно взять $e_i = E$ и тогда $|r_i - e_i| \cdot c_i = (r_i - E) \cdot c_i$.
Отсортируем пары (r_i, c_i) по r_i. Если i это наименьший индекс такой, что $r_i > E$, то наименьший ответ будет $(r_ic_i + r_{i+1}c_{i+1} + ... + r_nc_n) - (c_i + c_{i+1} + ... + c_n) \cdot E + k \cdot E$.
Тогда предпосчитаем суммы $(r_ic_i + ... + r_nc_n)$ и $(c_i + ... + c_n)$ на суффиксах. Будем менять E от 0 до $max(r)$, поддерживать наименьшее i, что $r_i > E$ и находить для каждого E наименьший ответ за $O(1)$.
Сложность $O(nlogn + max(r))$.
(Можно брать только E равные r_i-м и тогда сложность будет $O(nlogn)$)

C Пусть $|s| = n$. Поиск подпоследовательности в строке сводится к последовательному рассмотрению символов в последовательности и нахождению в строке этого же символа, правее чем предыдущий найденный, но как можно левее.
Переформулируем задачу. Изначально мы находимся в позиции перед первым символом. Каждый раз мы выбираем букву и прыгаем в ближайшую справа такую же букву в строке. Мы хотим, чтобы момент, когда мы не можем прыгнуть (за неимением такой буквы справа) настал как можно раньше.
Предпосчитаем для каждой пары (позиция, буква) позицию в которую мы прыгнем за $O(n \cdot alphabetSize)$.
Если мы будем жадно выбирать букву которая перебросит нас как можно правее, то это даст нам минимальную длину строки и 50 баллов.
Давайте посчитаем минимальное требуемое количество шагов, чтобы мы не могли прыгнуть, и минимальную лексикографически букву, используя которую этого можно достичь. Будем считать эти значения с конца. За счет предпосчитанных прыжков мы можем считать эти значения в позиции за $O(alphabetSize)$ через уже посчитанные. Затем, так как мы сохраняли лексикографически минимальные буквы, позволяющие получить минимальную длину, мы можем восстановить ответ, прыгая по ним, начиная из позиции перед первым символом строки.
Сложность $O(n \cdot alphabetSize)$.

D Пусть $max(a) = A$. Будем отвечать на запросы в офлайне. Считаем все запросы и сгруппируем их по правой границе.
Будем обрабатывать элементы слева направо. Для каждого числа k будем хранить последнюю позицию с числом кратным k ($lst[k]$). Для этого можно рассмотреть все делители числа a_i (их количество можно оценить как $O(a_i^{\frac{1}{3}})$. Чтобы быстро их перебирать предпосчитаем делители для чисел от 1 до A за $O(AlogA)$) и обновить значения lst. Перед этим также хотим найти ответ для всех запросов с правой границей в i. Для этого хотим найти наибольшее k такое, что последние два числа кратных k были правее левой границы запроса. Для этого давайте поддерживать set в котором хранятся пары из позиции левого из последних двух чисел кратных k и число k. Причем будем удалять из этого set'а пары (pos, k) для которых существует пара лучше их (числа не левее и k не меньше). Тогда, чтобы найти ответ на запрос с левой границей l просто найдем в set'е пару (pos, k) с наименьшим $pos >= l$, тогда k из этой пары это и будет ответ.
В итоге, последовательность действий такая:
1. Для каждого $d|a_i$ добавляем в set пару $(lst[d], d)$.
2. Находим ответы на все запросы с правой границей в i.
3. Обновим значения $lst[d]$ для всех $d|a_i$.
Сложность $O(A^{\frac{1}{3}}nlogn + AlogA)$.

E Давайте создадим таблицу 16×50. Одну половину 8×50 зальем белым цветом, а другую - черным. Теперь на белую половину добавим $b-1$ черных клеток не нарушая связности белой половины. Аналогично, добавим $a-1$ белых клеток на черную половину.
Сложность $O(a+b)$ (на самом деле $O(nm)$, но нам достаточно такого размера таблицы, чтобы мы могли закрасить в каждой половине $max(a, b) - 1$ клеток).