Подзадача 1

За тази подзадача е достатъчно просто да симулираме дадените заявки. За поддържане на съдържанието на една чаша можем да използваме STL multiset или да си поддържаме сортиран масив за да можем лесно да проверим дали съдържанието на две чаши е еднакво. Авторовото решение за тази подзадача има сложност $O(NQ^2logQ)$

Подзадача 2

Единствената стъпка от предната подзадача към тази е премахване на потенциален логаритъм при сравняване на съдържанието на две чаши. Ако съдържанието се държи в multiset или сортиран масив, то това става като просто се сканират паралелно двете структури до намиране на първата разлика. Така сравняване на две съдържания има линейна сложност и общото решение става $O(NQ^2)$.

Подзадача 3

Тази подзадача иска досещане за основната стъпка в задачата – да поддържаме съдържанието на една чаша не като някакъв вид списък на напитките, а като техен **хеш**. Главното важно нещо за хеш функцията ни е, че тя трябва да не зависи от реда на елементите. Съществуват различни добри хеш функции, които изпълняват това условие – тук ще опишем само тази използвана в авторското решение:

Нека с P_i бележим i-тото просто число ($P_1 = 2, P_2 = 3 \dots$). Тогава неподреденото множество $\{a_1, a_2, \dots, a_k\}$ можем да хешираме като:

$$H(\{a_1, a_2, \dots, a_k\}) = \left(P_{a_1} \times P_{a_2} \times \dots \times P_{a_k}\right) mod\ (10^9 + 7)$$

Тъй като ако не взимахме произведението по модул то еднозначно определя елементите на множеството, то можем да сме доста сигурни, че това е добра хеш функция от гледна точка на потенциални колизии.

Използвайки тази функция, за заявки от тип 1 е достатъчно да умножим хешовете от L до R по P_k, а за заявки от тип 2 да проверим дали всички хешове в интервала са еднакви. Така една заявка има линейна сложност и решението става $O(NQ)$

Подзадача 4

Всяка чаша участва в най-много една заявка от тип 1

От допълнителното ограничение следва, че интервалите зададени в заявки от тип 1 са непресичащи се. Подзадачата има няколко възможни подходи, но най-простият от тях е да поддържаме всички интервали от тип 1 заявки в подредена структура (например set) по ред на началата им. При добавяне на нов интервал трябва да разгледаме съседните му в структурата, и потенциално да обединим интервали, чиито краища се докосват и напитки

са идентични. Заявка от тип 2 тогава става като просто проверим дали първата и последната чаша от заявката попадат в един и същи интервал в структурата ни. Сложността е $O(QlogN)$

Друг валиден подход е линейно изпълнение на заявките от тип 1 (амортизирано им сложност ще е константа) и използване на Union-Find/Disjoint-Set Union за бързо намиране на следващия различен елемент от дадена позиция.

Подзадача 5

Лазо използва само напитка 1

При това ограничение за всяка чаша е нужно да пазим само количеството от напитка 1 в нея. Тогава можем да разглеждаме двете заявки от задачата като:

1. Добави +1 на всяко число от интервал
2. Отговори дали даден интервал съдържа само еднакви числа

Тези заявки можем да поддържаме чрез сегментно дърво с lazy propagation. Ако даден интервал се състои само от едно и също число, то във върхът, който му отговаря, можем да записваме това число, а в противен случай да записваме някоя невалидна стойност (например -1). Тази информация е достатъчна за lazy propagation подход, с който да обработваме и двете заявки. Сложността е $O(QlogN)$

Подзадача 6

Пълното решение на задачата е просто обединение на хеш идеята от подзадача 3, и идеята за сегментно дърво от подзадача 5. Ако поддържаме хешовете на всяка чаша, то двете заявки от задачата се превръщат в:

1. Умножи всяко число от интервал по X (стойност зададена със заявката) по модул $10^9 + 7$
2. Отговори дали даден интервал съдържа само еднакви числа

Тези две заявки могат да се поддържат по абсолютно същия начин като в подзадача 5. Сложността е $O(QlogN)$

Потенциални колизии на хеш функцията

В тази задача колизия би довела до грешен отговор ако в даден интервал имаме различни стойности, но техните хешове са еднакви. Важно е да забележим, че ако имаме много различни стойности, и само някои от тях образуват хеш колизии, то отговорът ни все още ще е правилен, тъй като има поне две различни числа. Така всъщност шансът за грешен

отговор се вдига с линейна скорост спрямо броя заявки, а не с квадратна (т.е. не наблюдаваме тъй наречения birthday paradox). При около 10^9 валидни хеш стойности и около 10^5 заявки, шансът за такава грешка е много малък, но не е пренебрежим. Поради наличието на full feedback и възможността за избиране на произволен модул обаче, това не е реален проблем за състезателя.

Възможни, но не нужни, са по-стабилни решения, например с двоен хеш (т.е. два различни модула), или такива използващи unsigned long long като “автоматичен модул”.

Автор: Енчо Мишинев