АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
КОДОВЕ

Задачата е нестандартна за тази възрастова група по няколко причини.

Първо, задачата е с релативно оценяване. Тоест не е задължително да се намери оптималното решение. Всъщност, за конкретната задача няма (известен на мен) алгоритъм, който да я решава перфектно за дадените ограничения. Макар и това да е често срещано в по-горните възрастови групи (особено в А), малките състезатели по-рядко, ако изобщо, се сблъскват с такива задачи. А според мен трябва :)
Недостатък на системата, на която се провеждат българските състезания, е това, че няма как да се направи истинско релативно оценяване (тоест срещу най-доброто решение на участник, а не на автора). Което е лошо по няколко причини, които обаче няма да обсъждаме тук.

Второ, огромен брой точки (50) беше предвиден за доста малки тестове. За които съществуват оптимални решения (даже до около 20 думи, но това решение би било сложно за тази възрастова група). За да хванат 100 точки, състезателите трябваше да напишат две различни (макар и сравнително прости) решения за всяка група от тестове.

Така, сега да разгледаме и как бихме могли да решим задачата.

Общи части
Състезателите трябва да са забелязали, че едно доооста просто решение е да изпечатаме дадените ни възможни комбинации (които ще наричаме "думи") една след друга. Това решение би хванало около 15 точки (!!) с буквално 4-5 реда и почти никакво мислене. Доста щедро от моя страна.
Леко подобрение е да проверяваме преди изпечатването на всяка от дадените думи, дали предходните няколко символа (суфиксът) на предишната не са същите като първите няколко (префикса) от текущата. Ако това е така, можем да не ги печатаме. Така печелим известен брой символи и решението вече хваща 20-25 точки.
Всъщност това не е никак лошо решение, особено ако не печатаме комбинациите в реда, в който ни ги дават, ами в реда, в който се "застъпват" възможно най-много. Намирането на този ред, обаче, не е никак проста задача.

Малки тестове
За тестовете, където броят думи е малък (до 10) съществуват няколко възможни решения, които намират оптималната подредба на думите. В смисъл, ако пробваме *всички* подредби, една от тях ще е оптималната, right? А по колко начина можем да подредим до 10 елемента? 10! = 3,628,800. Което не е толкова много.
Такова брутфорс решение можем да напишем по няколко начина – или чрез рекурсия, или чрез вградената функция next_permutation(). Аз лично съм фен на рекурсията, но тъй като next_permutation() е може би по-нестандартния подход за тази възрастова група, в авторското решение е реализирано то, та можете да го погледнете. Сложността и на двата варианта е O(N! * N * K), ако реализираме решението по най-баналния начин, и O(N! * N), ако преизчислим броя на застъпващите се символи за всяка двойка думи.
Така за всяка възможна подредба на потенциалните пароли гледаме колко застъпващи се символи има между всеки две и не ги печатаме. Ако състезателите не напишат

такова решение за малките тестове, то те биха загубили около 20 точки (стига да имат хубаво решение за големите).

Големи тестове
При големите тестове вече всякакви решения, базирани на изчерпване, са обречени на неуспех. Така и намирането на оптималната подредба е ужасно трудно (всъщност не се знае дали е възможно изобщо).
Тук на помощ идва релативното оценяване – всъщност не ни трябва да намерим *оптималната* подредба, само някоя, която е що-годе добра. А кога казахме, че една подредба е добра? Когато има много застъпващи се цифри при съседните комбинации. Защо, тогава, не печатаме винаги следващата неизползвана дума, която има най-много застъпващи се цифри с предходната?
Тази алчна стратегия дава учудващо добри резултати. Даже се наложи известно мислене на гадни малки тестове за да не може само тя да хваща почти 100 точки.

Така, сега остава проблемът как да намираме следващата неизползвана дума с най-голямо застъпване. Един вариант би бил просто на всяка стъпка да проверяваме всички неизползвани и да си отбелязваме коя колко символа спестява. Това е окей, за тестовете, където има до 1000 думи, но (би трябвало да) е малко бавно за най-големите тестове.

Тук трябва да забележим, че тъй като входните думи са различни и в най-лошия случай дължината им е 6, то имаме застъпване от най-много 5 цифри. Което не е толкова зле. 5-цифрени числа са числата < 100000. Така можем да направим структура от опашки (5 * 100000 на брой), която отговаря бързо на въпроса "коя е следващата дума със застъпване от X цифри, ако предходната има суфикс числото Y (евентуално с водещи нули)". Така индексирайки опашката с индекс [X][Y] ще видим коя е следващата дума. Първо изкарваме всички думи от началото на опашката, които вече сме ползвали. След това, ако опашката не е празна, значи първата дума би ни дала валидно продължение с X застъпващи се цифри. В противен случай проверяваме опашката [X-1][Y] и така нататък, докато намерим валидно продължение. Погледнете авторското решение за конкретна реализация. Тъй като всяка дума влиза в най-много K-1 опашки, то общата сложност на вкарването и изкарването на думите от опашките е със сложност O(N * K), и тъй като всяка от думите е с K символа, то сложността на това решение е O(N * K * K).

Автор: Александър Георгиев