Задача A. Киоск с напитками

В задаче необходимо было подсчитывать количество различных видов напитков, которые мог бы приготовить робот учитывая наличие различное количество ингредиентов.

Для подсчёта этого количества необходимо хранить количество оставшихся ингредиентов на данный момент. Для этого можно завести специальную структуру данных, позволяющую по названию ингредиента быстро получать и изменять его значение. Подходящая структура — map в C++ или HashMap в Java. Также в похожей структуре данных необходимо хранить рецепты напитков для быстрого получения доступа к ним. При использовании структур данных для хранения и обновления значений достаточно промоделировать все запросы создания напитков.

Задача B. Магический дождь

Будем решать задачу методом динамического программирования. Заметим, что r в задаче не превосходит 5. Пусть тогда $dp[i][j][mask]$ — максимальная длина орошенных участков, для первых i деревень, призвав дождь j раз таким образом, что призывы туч на последних $2r$ точках совпадают с маской $mask$. Тогда из состояния $dp[i][j][mask]$ нужно обновить состояния $dp[i+1][j][mask*2]$ и $dp[i+1][j+1][mask*2+1]$, причем во втором случае: $dp[i+1][j+1][mask*2+1] = max(dp[i+1][j+1][mask*2+1], dp[i][j][mask] + add[i][mask])$. Здесь $add[i][mask]$ - изменение длины орошенных участков, если в i-ой точке и точках, соответствующих $mask$, были вызваны тучи. Итоговая сложность $O(n*k*2^{2*r})$.

Задача C. Поросячий бизнес

Найдем наименьшую цену стога c_i. Тогда для постройки дома необходимо $W \cdot c_i$ монет. Чтобы максимизировать прибыль, нужно продать дома всем, кто готов купить его больше, чем за $W \cdot c_i$ монет.

Задача D. Филворды

Так как задача требует найти все варианты, а слова могут быть нетривиально расположены, придётся перебрать все возможные варианты.

Для этого следует запустить поиск в глубину из каждой клетки филворда. Для составления последовательности без самопересечений можно заполнять пройденные во время поиска в глубину клетки произвольным символом (естественно, не строчной латинской буквой). Тогда за константное время можно проверить, взята ли буква.

Даже с ограничением слов в 10 символов (и, следовательно, глубины поиска) выходит слишком много вариантов для переборa, поэтому нужна эвристика, чтобы не обходить варианты, которые точно не могут помочь найти слово из словаря.

Для этого следует проверять, может ли найденная последовательность быть префиксом слова из словаря.

Для эффективности стоит хранить словарь в виде бора. Тогда можно будет быстро ответить, существует ли слово в словаре такое, что выделенная последовательность букв в поле филворда является его префиксом. Также небольшой оптимизацией будет добавить счётчик оставшихся слов с данным префиксом, чтобы не проверять те префиксы, все слова с которыми уже найдены.

Найденные слова можно помечать в самом бору или добавлять в контейнер, озаботившись защитой от повторений (например, используя std::set), а после просто вывести.

Задача E. Про Дашу

Научимся определять, в скольких раундах Даше необходимо поучаствовать, чтобы определить, сколько очков нужно вложить в характеристику, чтобы выиграть в сравнении. Для этого можно воспользоваться бинарным поиском поучаствовав в $log(20) = 5$. Выполнять поиск для каждой характеристики не получится, т.к. для этого потребуется 30 раундов. Заметим, что вложив какое-то количество очков в некоторую характеристику, оставшиеся очки можно вложить в прочие характеристики, причем вкладывать очки стоит так, чтобы вложенные очки были максимально приближены к очередной итерации бинарного поиска. Для полного балла, данного решения достаточно.

Так же в первом раунде можно вложить во все характеристики поровну, достаточно сильно уточнив границы поиска, поскольку гарантируется, что в сражении можно выиграть.