1 Обязательная часть

1. ***Счастливые билеты***

   Решается динамикой. Заметим, что кол-во способов получить счастливый билет P_{2n}^k, где k —сумма цифр в половине, равняется квадрату кол-ва способов получить число k из n слагаемых P_n^k, так как две половины билета не зависят друг от друга. Таким образом

   $$ans = \sum_{k=0}^{9n} (P_n^k)^2$$

   Что такое P_n^k? Это сумма кол-ва способов разбить число, меньшее k на что-то от 0 до 9, на $n-1$ слагаемое. То есть:

   $$P_n^k = \sum_{i=0}^{9} P_{n-1}^{k-i}$$

   Начальное значение очевидно: $P_0^0 = 1$. Считаем, что $P_n^{m<0} = 0$. Работает за $\mathcal{O}(9n \cdot n)$.

2. ***Без двух одинаковых подряд.***

   Надо набрать n чисел. Первое число может быть любым из трех. Второе - любое, кроме первого. Третье - любое, кроме второго. И тд. Получаем, что для всех чисел после первого два варианта подстановки. Итог: $3 \cdot 2^(n-1)$.

3. ***Без двух нулей подряд.***

   Нельзя два нуля подряд. То есть если стоит ноль, то следующее либо единица, либо двойка. Если же стоит единица или двойка, то следующим может быть любое из трех чисел. То есть кол-во последовательностей длины n - сумма умноженного на тройку кол-ва не нулей в предыдущей последовательности и умноженного на двойку количества нулей. D_n - нулей возможно на n шагу. L_n - всего возможно вариантов последовательностей на n шагу.

   $$D_n = L_{n-1} - D_{n-1} \text{ - так как 0 рождается только из 1 или 2.}$$
   $$L_n = 2D_{n-1} + 3 \cdot (L_{n-1} - D_{n-1}) = 3L_{n-1} - D_{n-1}$$

   Начальные значения: $D_1 = 1$, $L_1 = 3$, так как длины 1 есть только последовательности 0; 1; 2. Работает за линию, так как мы просто заполняем одним проходом два массива размера n.

4. ***Кубы.***

   Предподсчет делаем за кубический корень: считаем все кубы от 1 до $n^{\frac{1}{3}}$. Затем динамика. Бежим слева направо по числам от 1 до n. $f[n] = min(f[n - c[i]])$, где $c[i]$ - все кубы из подсчитанного массива. Запоминаем в map уже найденные значения. Работает за $n^{\frac{4}{3}}$. Бежим по всем числам - отсюда n, на каждом шаге делаем куб из n сравнений, отсюда $n^{\frac{4}{3}}$.

5. ***Zero***

   Для удобства посчитаем все возможные произведения трех соседних в массив. То есть теперь в массиве d на месте i будет храниться $a[i-1] \cdot a[i] \cdot a[i+1]$. Теперь бежим динамичкой слева направо по этом массиву.

   $$f[i] = max(d[i] + f[i-2], f[i-1])$$

   То есть мы либо берем произведение d[i], тем самым зануляя iй элемент в a, теперь есть смысл брать только число через одно, соседние произведения обнуляются, либо не берем его, тогда можно брать и соседнее число и тд. Начальное состояние: $f[0] = d[0], f[1] = max(d[0], d[1])$. Работает за линию: подсчет произведений+динамика.

6. ***Взвешенный бинпоиск.***

   Динамичка. Есть массив весов c. Смотрим на его подотрезки $[i][j]$. Хотим узнать, за какую минимальную сумму весов сможем на этом подотрезке найти элемент в худшем случае. То есть:

   $f[i][j] = min(max(f[i][k-1], f[i][k+1]) + c[k])$, для всех k на промежутке [i..j].

   Работает за куб, так как квадрат состояний, в каджом ответ за линию ищем. Начальные состояние: $f[i][i] = c[i]$.

2 Дополнительная часть

1. ***Кубы.***

   Делаем то же самое, что и в обязательной про кубы, но вторая часть - ленивая динамика. То есть запоминаем в map уже найденные значения. Работает не сильно быстрее предыдущего: за кол-во достижимых состояний.

2. ***Кактус.***

   Умеем считать за линию паросочитания для цикла, дерева, смеси деревьев и цикла. Задача: посчитать паросочитания для циклов, торчащих из одной вершины (возможно, из этой же вершины торчат еще какие-то деревья). Что делаем. Убираем все ребра, выходящие из вершины-сцепки. Получаем много отдельных деревьев/циклов/циклов с деревьями. Считаем отдельно паросочитания для них.

   Теперь надо вершину убранную вернуть. Бежим по ребрам, выходящим из этой вершины, пытаемся отметить их (максимум одно ребро отмечено). Пусть отмечено ребро, ведущее в струтуру i (обзовем структурой связный кусок, остающийся после убирания вершины-сцепки). Все возможные структуры отметим множеством g. Тогда пытаемся обновить ответ такой величиной: $d[i][true] + \sum d[g \backslash i][false] + 1$. Таким образом выберем максимальное паросочетание, если в вершине-сцепке обязательно есть отмеченное ребро. И осталось учесть вариант, если ни одно не отмечено: $\sum d[g][false]$. Работает по идее за линию:).