4. Для каждой пары вершин в графе найти $w[a, b]$ – такой минимальный вес, что из a в b есть путь по рёбрам веса <= $w[a, b]$. O(V^2)

Решение:
Переформулируем задачу: для каждой пары точек a, b найдём путь из a в b с минимальным максимальным весом ребра.

С помощью алгоритма Прима построим минимальное остовное дерево. Затем запустим из каждой вершины дерева dfs, и будем сохранять максимальный вес ребра в процессе обхода (это и будет искомый вес).

Асимптотика: алгоритм Прима = $O(V^2)$
dfs = O(V + E) = O(V) (т.к. в дереве всегда V - 1 ребро), запускаем от V вершин
суммарно $O(V^2)$ + V O(V) = $O(V^2)$
///
Почему мы действительно найдём минимальный вес : допустим, что в минимальном остовном дереве существует ребро между вершинами c и d не минимального веса, тогда заменим его на минимальное, тогда общий вес дерева уменьшится, значит мы построили не минимальное остовное дерево, противоречие.
Далее, так как в дереве все пути определяются однозначно,
то dfs найдёт только один - наш путь с минимальным максимальным ребром, который остался после алгоритма Прима.

6. Найти в графе цикл минимального среднего веса V, E <= 2000, |w_i| <= 10**9

Решение:
Алгоритм: с помощью бинпоиска будем искать минимальный вес, при вычитании которого из весов всех рёбер существует отрицательный цикл (границы бинпоиска от -10**9 до 10**9). Проверять наличие отрицательного цикла будем с помощью алгоритма Форда-Беллмана. Найденый этим алгоритмом цикл при найденном в бинпоиске числе будет являться циклом минимального среднего веса.
Почему это работает:

Пусть w - это текущий найденный вес на данной итерации бинпоиска (mid), тогда пусть
$\sum_{i=1}^{n} (w_i - \mathrm{w})$ - это сумма всех весов рёбер в найденном алгоритмом Форда-Беллмана цикле.

Значит, $\sum_{i=1}^{n} (w_i - \mathrm{w}) < 0$. Из этого следует, что $\sum_{i=1}^{n} w_i < \sum_{i=1}^{n} \mathrm{w} = \mathrm{nw}$. Значит, $\frac{\sum_{i=1}^{n} w_i}{\mathrm{n}} < \mathrm{w}$.

А $\frac{\sum_{i=1}^{n} w_i}{\mathrm{n}}$ это и есть средний вес цикла. Поэтому нужно найти минимальное значение w, а после

сравнить со значением при w + 1 (так как знак строгий).

Асимптотика: бинпоиск = log(2 * 10**9), алгоритм Форда-Беллмона = O(VE). Проверка O(E).
Суммарно O(VE) (O(log(2 * 10**9)VE)).

8. Есть массив из нулей и единиц. В online за $\mathcal{O}(\log n)$ отвечать на запросы: поменять элемент; найти ближайший слева/справа ноль к позиции i.

Решение:
Построим дерево отрезков (построение снизу, добиваем массив единицами справа, чтобы array_size был равен степени 2-ки), в узлах будем хранить пару: минимальный и максимальный индекс нуля на подотрезке. Если на подотрезке нет нулей, то будем хранить индексы inf и -1 (inf может быть равно, например, количеству элементов массива + 1, если мы в единичой индексации).

UpdateElem: i + array_size - это индекс i-го элемента массива в дереве, следовательно, если меняем 0 на 1, то в этой node вместо [i, i] меняем значение на [inf, -1], если 1 на 0, то наоборот, иначе не меняем (два if во вставке). После чего поднимаемся наверх и обновляем значения в узлах (tree[i] = [tree[2i].min_index_of_zero, tree[2i + 1].max_index_of_sero]).
Асимптотика: O(log(n))

Запрос Max_zero_to_the_left(k):
Рекурсивно спускаемся в правого и левого сына пока tree[i].min_index_of_zero <= k и tree[i].max_index_of_zero >= k, а на выходе возвращаем максимум из того, что вернулось из левого сына и правого. Если tree[i].max_index_of_zero < k, возвращаем это значение, если tree[i].min_index_of_zero > k, возвращаем -1. Запрос Min_zero_to_the_left аналогично (вместо -1 возвращем inf соответственно).
Асимптотика: O(logn)