1.

Для нахождения самих путей необходимо хранить двумерный массив предков. То есть, при расчете D^k_{ij}(матрицу вводили на лекции) надо еще рассчитать π^k_{ij}, где π^k_{ij} – предок вершины j на пути из i через промежуточные вершины $\{1, ..., k\}$.

$$\pi^0_{ij} = \begin{cases} null, \text{ если } i = j \text{ или } w_{ij} = \infty \\ i, \text{ иначе} \end{cases}$$

$$\pi^k_{ij} = \begin{cases} \pi^{k-1}_{ij}, \text{ если } D^{k-1}_{ij} \leq D^{k-1}_{ik} + D^{k-1}_{kj} \\ \pi^{k-1}_{kj}, \text{иначе} \end{cases}$$

Сложность алгоритма не изменится по сравнению с классической версией: $O(n^3)$ (вершины пронумерованы от 1 до n).

Корректность следует из того, что мы запоминаем всех предков, так что однозначно можем восстановить путь.

2.

Заметим, что при наличии цикла отрицательного веса в матрице D(вводили на лекции) появятся на диагонали отрицательные числа. Действительно, релаксация расстояния между вершинами с $i = j$ произойдет только в случае когда вершина k такая, что $D_{ik} + D_{kj} < 0$, что и означает наличие цикла отрицательного веса.

Таким образом, для проверки наличия цикла отрицательного веса необходимо после работы алгоритма Флойда-Уоршелла проверить диагональные элементы (D_{ii}) на отрицательность. Если есть хотя бы один отрицательный элемент, то и есть цикл отрицательного веса.

Сложность алгоритма такая же, как у классической версии, проход диагонали не ухудшает ее.

Корректность следует из доказанного утверждения.

3.

1. Алгоритмом Дейкстры найдем кратчайшее расстояние между этими двумя вершинами

2. Также найдем кратчайший путь от вершины a до вершины u и кратчайший путь от v до вершины b, так что итоговый путь от a до b: $w_{au} + w_{uv} + w_{vb}$, где w_{au} – вес пути $a \to u$, w_{uv}– вес ребра $u \to v$, w_{vb} – вес пути $v \to b$.

Алгоритм должен вернуть минимум из этих двух значений.

Корректность:

Алгоритм Дейкстры работает корректно только для графов с неотрицательными ребрами, так что 2 пункт этого алгоритма позволяет отдельно проверить случай пути, проходящего по ребру отрицательного веса, так как мы заранее знаем вершины ребра с отрицательным весом, можно построить путь конкретно через это ребро. Ведь может так получиться, что путь больший длины обладает меньшим весом за счет ребра отрицательного веса. Таким образом, будут рассмотрены всевозможные пути и выбран минимальный.

Сложность:

В данном алгоритме 3 раза применяется алгоритм Дейкстры: один раз в первом пункте ($a \to b$) и два раза во втором пункте ($a \to u$ и $v \to b$). Таким образом, итоговая сложность: $O(|V|^2)$.

4.

1. Как сказано в указании, булева и тропическая алгебра не являются кольцами, а являются полукольцами, так как, нет противоположного элемента относительно сложения. А алгоритм Штрассена применим только к кольцам, так как требует наличия этого противоложного элемента относительно операции сложения в ходе быстрого умножения матриц.

2. Так как алгоритм Штрассена не работает с булевыми матрицами, о чем было сказано в первом пункте, необходимо их записать как числовые матрицы и перемножать в кольце вычетов по модулю $n + 1$, таким образом, обспечив наличие противоположного элемента относительно сложения. То есть, вместо дизъюнкций конъюнкций (было оговорено на лекции) будет вычисляться сумма произведений по модулю $n + 1$. Поэтому, чтобы узнать значение дизъюнкции конъюнкий, надо проверить вычисленную сумму произведений на равенство нулю.

Корректность:

Корректность алгоритма вытекает из определения кольца вычетов по модулю n.

Сложность:

Сложность данного алгоритма будет улучшена за счет применения алгоритма Штрассена, дополнительные операции не ухудшают сложность, так что итоговая сложность: $O(n^{\log_2 7} \log n)$

5.

Выберем любую вершину v_1 в дереве и сделаем поиск в ширину от нее, обозначим самую дальнюю как v_2. А затем от v_2 сделаем еще раз поиск в ширину и получим самую удаленную вершину от нее: v_3. Расстояние между этими вершинами является диаметром, а центр, значит, находится посередине, если число вершин четное, а если нечетное (обозначим вершины на диаметре от 1 до $2n + 1$), то n и $n + 1$.

Корректность:

Дерево – это связный ациклический граф. Если все ветви дерева имеют одинаковую высоту, то центр будет находится в корне, иначе он будет находится посередине пути от листа самой длинной ветви до листа второй по длине ветви, описанное расстояние является диаметром, то есть, максимальная длина кратчайшего пути между всеми вершинами графа.

Сложность:

Обход графа, как было показано ранее на лекциях, стоит $O(|V| + |E|)$. В данном алгоритме делается два таких обхода, так что итоговая сложность: $O(|V| + |E|)$.