Побег с горной базы

В первой подзадаче можно перебрать все варианты расстановки вертолетов, и выбрать оптимальный. Решение работает за $O(2^n \cdot n)$.

Заметим, что вертолеты всегда выгодно ставить в листы дерева. И если количество листов меньше, чем k, то ответом является n. Решим вторую подзадачу методом динамического программирования. Будем перебирать листы в порядке, в котором они встречаются в каком-то эйлеровом обходе дерева. Состоянием будет пара чисел (i, v) — количество вертолетов, которые уже поставлены, и номер последнего листа, в который был поставлен вертолет. Допустим, мы хотим поставить вертолет в лист u, тогда нужно обновить значение $dp[i+1, u]$ значением $(dp[i, v] + depth(u) - depth(lca(u, v)))$, где $depth$ — глубина вершины, а lca — наименьший общий предок. Получившееся решение работает за $O(k \cdot n^2)$ или $O(k \cdot n^2 \cdot \log(n))$ в зависимости от реализации нахождения наименьшего общего предка.

Для решения третьей подзадачи, нужно оптимизировать состояние динамического программирования из предыдущего абзаца для графов с маленькой высотой. Вместо того, чтобы хранить номер последнего взятого листа, будем хранить $depth(lca(v, u))$, где v — последний взятый лист, а u — текущий лист. Теперь, пусть мы переходим от листа номер u к следующему листу номер q. Несложно доказать, что $depth(lca(v, q)) = \min(depth(lca(v, u)), depth(lca(u, q)))$. Поэтому, при переходе от листа к следующему, нужно обрезать значение глубины lca с последней взятой вершины до глубины lca текущего и следующего листа. В остальном, переходы аналогичны переходам из предыдущего абзаца. Получилось решение, работающее за $O(k \cdot D \cdot n \cdot \log(n))$, где D — максимальная глубина вершины. Заметим, что оно укладывается в ограничения как третьей, так и второй подзадач.

Заметим, что самую глубокую вершину всегда выгодно взять. Допустим, мы ее не возьмем. Тогда начнем подниматься из нее до корня, пока не встретим вершину, из которой достижима взятая. Заменим эту взятую вершину на ту, из которой поднимались, и ответ не ухудшится. Поэтому, возьмем самую глубокую вершину в ответ, и удалим из дерева все вершины на пути от взятой вершины до корня. При этом, дерево может распасться на несколько. Теперь, аналогично можно доказать, что снова выгодно взять в ответ самую глубокую вершину (теперь глубина вершины считается от корня того дерева, в котором она лежит). Снова возьмем ее и удалим из дерева путь от нее до корня. Будем делать там k раз. Каждый раз мы находим самую глубокую вершину и удаляем путь за $O(n)$. Поэтому, получилось решение, работающее за $O(n \cdot k)$.

Отсортируем все вершины по глубине. Будем рассматривать их в таком порядке. Когда мы рассматриваем вершину v, будем подниматься из нее до корня, пока не встретим помеченную вершину. Обозначим за c_v количество непомеченных вершин, по которым мы успели пройти. После чего пометим все вершины, по которым прошли. Оказывается, что ответом является сумма k максимальных чисел среди c_v. Нужно доказать, что для вершин, которые войдут в ответ, c_v будет равно глубине в момент удаления в алгоритме из предыдущего абзаца. Доказательство оставим читателю в качестве упражнения :). Получили решение, работающее за $O(n)$.