Годовой отчет

Автор задачи и разработчик: Прохор Ларичев

Будем воспринимать кандидатов как вершины графа, а дружбу — как его ребра. Тогда требуется выбрать среди всех максимальных клик (полных подграфов) такую, в которой `xor` значений в вершинах максимален.

Существует алгоритм Брона-Кербоша, который позволяет решить эту задачу с достаточным запасом по времени, перебрав все максимальные клики в графе за время $\mathcal{O}(3^{\frac{n}{3}})$. Поскольку этот алгоритм, скорее всего, не известен большинству из участников, ограничения в задаче были расчитаны на другое полное решение, о котором будет написано ниже.

В **первой подзадаче**, как это часто бывает, решение полным перебором проходит по времени. Если за $\mathcal{O}(2^n)$ перебрать подмножество вершин, и затем за $\mathcal{O}(n^2)$ проверить, образуют ли они клику, можно выбрать среди них максимальную по описанным критериям.

В случае $k = 0$, как во **второй подзадаче**, `xor` любого множества вершин будет равен 0, поэтому остается просто найти максимальную клику в графе. Есть множество алгоритмов, позволяющих найти произвольную максимальную клику, но здесь достаточно было воспользоваться методом «Meet in the Middle», позволяющим сократить полный перебор.

Разобьем все вершины на произвольные две равные группыЖ для каждого подмножества первой группы за $\mathcal{O}(2^{\frac{n}{2}})$ определим, является ли оно кликой, а также найдем список общих соседей. Останется только для каждой клики первой группы посмотреть на множество их общих соседей и выбрать в нем подмножество, являющееся кликой второй группы. Этот предподсчет также можно сделать за $\mathcal{O}(2^{\frac{n}{2}} \cdot n)$.

В **третьей подзадаче** выполняется $k \leqslant 3$, то есть можно перебрать возможные значение `xor` ответа, которых не более восьми, и для каждого применить решение предыдущей подзадачи, просто искать во второй группе вершин не произвольную клику среди соседей клики первой группы, а клику с конкретным значением `xor` вершин. Такие клики можно предподсчитать за $\mathcal{O}(3^{\frac{n}{2}})$, а в таком случае для MitM выгодно делить вершины на группы немного разных размеров: первая больше, вторая меньше. Это и является главным шагом на пути к полному решению.

Частный случай **пятой подзадачи** требует найти независимо максимальную клику, содержащую первую вершину, и не содержащую, и если первая не меньше второй, то выбрать в ответ ее. Для **шестой подзадачи**, аналогично, требуется сравнить максимальные клики, содержищие четное или нечетное количество вершин из первой и второй половины, соответственно. Такие клики можно найти аккуратной динамикой по подмножествам, после чего решение аналогично MitM для второй подзадачи.

Решение **четвертой подзадачи** и **полное решение** требует обобщения решения третьей. Разобьем n вершин на группы размерами m_1 и m_2, и в первой полным перебором найти все клики и `xor` значений в них, а во второй — аналогично, найти все клики, но еще для каждой маски M построить битовый бор на `xor`'ах ее максимальных подклик.

Тогда, чтобы найти максимум, переберем маску клики M_1 в первой группе, возьмем множество общих соседей ее вершин из второй группы M_2, и в битовом боре для M_2 найдем за $\log \max(a)$ значение y, максимизирующее $\left(\bigoplus\limits_{v \in M_1} a_v\right) \oplus y$.

Битовые боры при предподсчете для второй группы можно строить с использованием Small-to-Large оптимизации (взять максимальный из боров подмасок без одного элемента и персистентно добавить все недостающие подклики), но поскольку размер каждого бора для маски без однй вершины может быть меньше половины всех значений, здесь эта оптимизация не сильно поможет. Достаточно было аккуратно реализовать построение этих боров внутри перебора подмасок за $\mathcal{O}(3^{m_2})$.

Суммарное время работы такого решения равно $\mathcal{O}((2^{m_1} + 3^{m_2}) \cdot \log \max(a))$, и остается только выбрать такое разбиение n на $m_1 + m_2$, при котором эта величина минимальна. Менее эффективная реализация решения проходила только четвертую подзадачу.