В докладе будут использоваться более-менее стандартные графские обозначения, например, $H-v-X$ для обозначения графа H, из которого удалили вершину $v \in V(H)$ и подмножество вершин $X \subset V(H)$. Расстояние между вершинами x и y в графе H будем обозначать как $d_H(x, y)$. Очевидно, $d_H(x, y) = d_H(y, x)$ для неориентированного графа H.

Задача о самой важной вершине – дан связный взвешенный (веса всёх рёбер положительные) неориентированный граф G и две вершины в нём: s и t. Хотим удалить вершину G, отличную от s и t так, чтобы максмимизировать расстояние между s и t в оставшемся графе (если пути между s и t в оставшемся графе нет, то положим расстояние равным $+\infty$).

Наивный алгоритм нахождения такой вершины работает за $\Theta(nm \log n)$ в графе с n вершинами и m рёбрами: перебираем удаляемую вершину v ($n-2$ варианта) и ищем кратчайший путь от s до t в графе $G-v$ (алгоритм Дейкстры, требует $\Theta(m \log n)$ времени в самой простой реализации).

Наш алгоритм будет иметь время работы $O(m \log n)$ или даже $O(m + n \log n)$, если использовать фиббоначиевые кучи. Более того, он для каждой вершины графа v графа посчитает длину кратчайшего пути между s и t в графе $G - v$, что даёт намного больше информации, чем просто номер самой важной вершины.

Алгоритм основан на нескольких несложных наблюдениях, но, как говорил, Сергей Витальевич, “Две пропущенные подряд тривиальности образуют непреодолимое препятствие”.

Def. Деревом кратчайших путей связного взвешенного графа H с центром в вершине $r \in V(H)$ называется такое любое такое корневое дерево T на множестве вершин $V(H)$, что

- r — корень T.
- Все рёбра T являются рёбрами исходного графа H.
- Для каждой вершины $v \in V(G)$ верно, что $d_G(r, v) = d_T(r, v)$.

Lemma. *Пусть H — связный взвешенный граф, а $v \in V(H)$ — его вершина. Тогда у H есть дерево кратчайших путей с корнем в v.*

Lemma. *Пусть T — какое-то дерево кратчайших путей для H. Тогда $d_T(u, v) = d_H(u, v)$, если u является предком v в дереве T.*

Рассмотрим у графа G какое-нибудь дерево кратчайших путей T с корнем в s, а также нисходящий путь $P = (v_0, v_1, \ldots, v_k)$ от s к t в этом дереве, где $v_0 = s$ и $v_k = t$. По определению дерева кратчайших путей путь P кратчайший в G. Поэтому $d_{G-u}(s, t) = d_G(s, t)$, если $u \notin P$.

Поняли, что стоит удалять только вершины из P. Пусть $i \in [0, k-1]$. Введём такие обозначения:

- U_i — пустое множество вершин при $i = 0$, иначе компонента связности графа $T - v_i$, содержащая вершину v_{i-1}, то есть вершины "сверху" от v_i.
- D_i — поддерево вершины v_{i+1} в дереве T, то есть вершины "снизу" от v_i ("низ" — направление к t).
- O_i — все вершины поддерева вершины v_i в дереве T, за исключением самой вершины v_i и всех вершин из множества D_i. В некотором смысле, вершины находящиеся "сбоку".

Lemma. *$d_G(s, u) = d_{G-v_i}(s, u)$ для каждой вершины $u \in U_i$*

Несложно видеть, что любой путь из s в t, не проходящий через v_i, где $i \in [1, k-1]$ выглядит так: сперва он как-то ходит по $U_i \cup O_i$, а потом впервые заходит в D_i и уже из какой-то вершины D_i приходит в t. Оказывается, верна такая лемма:

Lemma. *$d_G(u, t) = d_{G-v_i}(u, t)$ для каждой вершины $u \in D_i$. То есть в G существует кратчайший путь между u и t, не проходящий через v_i.*

То есть каждый интересующий нас путь из s в t в графе $G - v_i$ выглядит так: сперва он как-то ходит по $U_i \cup O_i$, приходит в некоторую вершину $x \in U_i \cup O_i$, потом проходит по какому-то ребру $(x, y) \in E(G)$ в вершину $y \in D_i$, а потом проходит из y в t по пути длины $d_G(y, t)$.

Таким образом, $d_{G-v_i}(s, t) = \min d_{G-v_i-D_i}(s, x) + w(x, y) + d_G(y, t)$, где минимум берётся по всем таким рёбрам $(x, y) \in E(G)$, что $x \in U_i \cup O_i, y \in D_i$.

Величину $d_{G-v_i-D_i}(s, x)$ можно посчитать так: это $d_G(s, x)$, если $x \in U_i$, а для $x \in O_i$ её можно посчитать с помощью алгоритма Дейкстры: можно понять, что кратчайшему пути из s в $x \in O_i$ в графе $G - v_i - D_i$ нет смысла снова заходить в U_i после того, как он уже зашёл в O_i. Поэтому нужно просто сперва положить $d_{G-v_i-D_i}(s, x) = \min d_{G-v_i-d_i}(s, z) + w(z, x) = \min d_G(s, z) + w(z, x)$, где минимум берётся по всем таким рёбрам $(z, x) \in E(G)$, что $z \in U_i$. После этого нужно применить обычный алгоритм Дейкстры на вершинах O_i и рёбрах, оба конца которых принадлежат O_i. Эта стадия алгоритма работает за $O(m \log n)$ суммарно и позволяет насчитать $O(n)$ информации, позволяющей за $O(1)$ отвечать на запрос "чему равно $d_{G-v_i-D_i}(s, x)$".

Наконец, нужно как-то посчитать эти минимумы для каждого i, вообще говоря каждый минимум может браться по $O(m)$ рёбрам, поэтому наивным образом этот минимум считать нельзя. Однако можно заметить, что при переходе от v_i к v_{i+1} множество рёбер, по которым берётся минимум, изменилось не сильно:

- Удалились рёбра с одним концом в множестве $U_i \cup O_i$, а другим в множестве $D_i \setminus D_{i+1} = O_{i+1} \cup \{v_{i+1}\}$.
- Добавились рёбра с одним из концов в множестве $\{v_i\} \cup O_{i+1} = (U_{i+1} \cup O_{i+1}) \setminus (U_i \cup O_i)$ и другим в множестве D_{i+1}.

- Наконец, рёбра с одним концом в множестве O_i, а другим в множестве D_{i+1} стали вносить другой вклад в ответ: раньше они вносили $d_{G-v_i-D_i}(s,x) + w(x,y) + d_G(y,t)$, а теперь $d_{G-v_{i+1}-D_{i+1}}(s,x)+w(x,y)+d_G(y,t)$. Разница в том, что $d_{G-v_i-D_i}(s,x)$ для $x \in O_i$ мы считали отдельно с помощью Дейкстры, а $d_{G-v_{i+1}-D_{i+1}}(s,x) = d_G(s,x)$.

Мы хотим иметь множество рёбер, в котором мы можем удалять часть имеющихся рёбер, менять части имеющихся ребёр "вклад" и добавлять немного новых рёбер и, наконец, спрашивать минимальный "вклад" ребра из множества (чтобы узнать $\min d_{G-v_i-D_i}(s,x) + w(x,y) + d_G(y,t)$, то есть $d_{G-v_i}(s,t)$). Любая куча способна поддерживать такие операции. Если аккуратно посчитать, то всего операций, когда i пробегает значения от 1 до $k-1$, будет $O(m)$, поэтому эта часть алгоритма работает за $O(m \log n)$.