Гигантский дракон

Автор задачи и разработчик: Константин Бац

Для начала постарамся переформулировать все данные в условии критерии в терминах скалярного и векторного произведений векторов, чтобы не приходилось в явном виде вычислять углы во избежание неточностей. Для примера возмьмем условие на то, что угол между векторами $\overrightarrow{u}$ и $\overrightarrow{v}$ имеет величину строго между 0 и 45° и направлен по часовой стрелке. Равносильное утверждение будет выглядеть как $0 < -(\overrightarrow{u} \times \overrightarrow{v}) < \overrightarrow{u} \cdot \overrightarrow{v}$, так как для таких углов $\sin\alpha < 0$, $\cos\alpha > 0$ и $|\cos\alpha| > |\sin\alpha|$. Здесь под $\overrightarrow{u} \cdot \overrightarrow{v}$ подразумевается скалярное произведение $|v||u|\cos\alpha = u_x v_x + u_y v_y$, а под $\overrightarrow{u} \times \overrightarrow{v}$ — векторное произведение $|v||u|\sin\alpha = u_x v_y - u_y v_x$. Для остальных условий можно построить аналогичные неравенства, тем самым избавившись от необходимости явно вычислять углы.

В **первой подзадаче** $n \leqslant 5$. Заметим, что в таком случае мы можем построить не более одного сегмента дракона. Давайте переберем все тройки вершин, которые могут стать сегментом, проверим наличие ребер в каждую из S и соблюдение всех условий. Из всех подходящих троек выберем ту, которая имеет максимальную мощность.

Во **второй подзадаче** можно было развить предыдующую идею и реализовать перебор на основе обхода в глубину (но без отметок о посещенности). Запустим dfs из вершины S, пытаясь каждый раз пойти по основанию нового сегмента. Соответственно, переходы делаем только когда выполняются все необходимые условия на ребро как на основание сегмента. Иными словами, перед переходом надо проверить, что новое ребро находится под углами не больше 45° относительно последнего ребра в dfs и отрезка из головы до текущей вершины, и что среди детей текущей вершины есть два других ребра, которые подходят на роль лап.

Ответом будет один из наиболее глубоких путей из S. Если в dfs поддерживать мощность дракона на пути от S до текущей вершины, достаточно просто будет взять максимум из всех полученных в какой-либо момент значений.

Для решения **остальных подзадач** следует проанализировать структуру дракона. Можно доказать, что дракон — дерево. Действительно, комбинация условий на углы $\measuredangle(\overrightarrow{A_{i-1}B_{i-1}}, \overrightarrow{A_iB_i})$ и $\measuredangle(\overrightarrow{A_1A_i}, \overrightarrow{A_iB_i})$ не позволяет цепочке оснований «закручиваться». Формально это можно описать следующими двумя утверждениями:

1. С каждым сегментом вершины отдаляются от вершины S. Иными словами, для каждого сегмента верно, что его конец находится дальше (в смысле декартового расстояния) от S, чем его начало. Действительно, в противном случае угол между $\overrightarrow{SA_i}$ и $\overrightarrow{A_iB_i}$ был бы не только больше 45°, но и больше 90°.

2. Ограничение на длину лап и угол между лапами и основаниями не позволяет использовать внешнюю вершину лапы в качестве начала или конца другого сегмента или другой лапы в последующих сегментах. Если посмотреть на геометрическое место вершин, которые могут подходят на роль L_i или R_i, окажется, что это сектор круга радиуса $|A_iB_i|$ с центром в точке A_i, тогда как вся последующая структура дракона будет располагаться в противоположной от этого круга полуплоскости, отделенной от круга касательной к нему в точке B_i.

Из первого утверждения можно сделать вывод, что достаточно рассматривать вершины в порядке их отдаления от S и находить для каждой максимального дракона, заканчивающегося в ней, методом динамического программирования. Из второго — что можно выбирать лапы произвольным подходящим образом, потому что это не повлияет на возможность построить следующие сегменты.

В **третьей подзадаче** нам изначально давалось дерево. Подвесим дерево за вершину S. Рассмотрим произвольную не стартовую вершину v. Если она входит в какого-то дракона, то ребро от v до ее предка в дереве p является основанием какого-то сегмента, так как основания сегментов образуют пусть от S до v. Если v входит в еще какой-то сегмент, то не более чем в один, и основание этого сегмента — ребро от v до какого-то ребенка v. Лапами этого сегмента должны являться два других ребра от v до каких-то детей.

Давайте отсортируем вершины по удалению (в ребрах) от S и, рассматривая их от самой ближней до самой дальней, для каждой посчитаем максимального заканчивающегося в ней дракона. Для этого необходимо уметь для ребра, инцидентного вершине, быстро подбирать правую и левую лапу. Давайте отсортируем все ребра, инцидентные вершине v и идущие к детям, по углу между ребром и прямой от v до S. Теперь будем рассматривать ребра по увеличению угла и иметь две очереди с поддержкой минимума. В одной очереди будем хранить ребра с углом между ним и текущей вершиной от $-45°$ до $0°$ не включительно, а в другой — ребра с углом от $0°$ до $45°$. В качестве значения минимума будем использовать длину ребра. По мере обработки ребер мы будем перекладывать ребра из одной очереди в другую. Также здесь следует аккуратно обрабатывать случай, когда рассматриваемое ребро параллельно (с частичным наложением) еще нескольким. В таком случае необходимо, чтобы все параллельные текущему ребра не лежали ни в одной очереди. Для их временного хранения можно использовать третью очередь или просто массив.

Таким образом, для каждого ребра от v к ребенку мы можем понять, если ли подходящие для него лапы. Если существует лапа, подходщая под условия, то она будет лежать в нужной очереди и минимальное значение в этой очереди будет не больше, чем длина рассматриваемого ребра.

Итого, мы умеем за линейное время для каждого потенциального основания $v \to u$ находить подходящие лапы. Поскольку мы рассматриваем вершины в порядке увеличения растояния, то мы уже знаем длину максимального дракона до вершины v, а значит можем прорелаксировать динамику вперед $dp_u \leftarrow dp_v + |uv|^2$.

В **четвертой подзадаче** координата x была равна 0 или 1. Если рассмотреть всех возможных драконов в такой ограниченной полосе, то окажется, что y координаты вершин, входящих в основание, будут строго мнотонно убывать или возрастать, а x координтаты будут чередоваться (не может быть вертикального основания сегмента, так как для него не найдется лапы).

Лапы будут тоже вести себя особым образом. Одна из лап будут параллельна оси OX (иначе она будет длиннее основания), а вторая будет так же, как и основание, менять координату x на противоположную. Это позволяет решать задачу с помощью динамического программирования.

Давайте отдельно рассмотрим случай, когда y кордината будет возрастать и убывать. Не теряя общности, пусть она убывает. Для точки, например, $(0, y)$ выберем ближайшую снизу точку $(0, y_l)$ для левой лапы (если мы заранее отсортируем точки по координате, это можно сделать за $\mathcal{O}(1)$), затем переберем все ребра на «другую» сторону в порядке увеличения их угла относительно оси OY, то есть в точки вида $(1, ?)$, и заметим, что для всех ребер, имеющих длину не меньше $y - y_l$, и не последних в порядке увеличения угла, есть левая лапа (в точку $(0, y_l)$) и правая — следующее по углу ребро. Таким образом, динамику можно посчитать за линейный проход по ребрам «вниз» из каждой вершины.

Вспомним, что искомый дракон — дерево и про каждое потенциальное ребро в исходном графе можно сказать, как оно будет направлено в драконе. Поэтому для каждой вершины можно найти все возможные входящие и исходящие ребра, которые могут являться основаниями сегментов. Далее, так же как в третьей подзадаче, будем рассматривать вершины в порядке отдаления от вершины S. Поскольку у нас произвольный граф, а не дерево, в качестве растояния стоит брать декартово растояние от вершины S.

Ограничения **пятой подзадачи** позволяли перебрать комбинации сегментов, которые будут соеденяться в вершине v за время $\mathcal{O}(deg_{in}(v) \cdot deg_{out}(v))$. Для каждого начинающегося в v сегмента так же за линейное время можно проверить существование правой и левой лапы. Это позволяло не использовать очередь, в отличие от группы три. Время работы такого решения можно грубо оценить, как $\mathcal{O}(nm)$.

Ограничения **последней подзадачи** требовали совместить все ранее озвученные идеи вместе: отсортировать вершины по удалению от стартовой вершины, в каждой вершине отсортировать все входящие и исходящие ребра по углу, и быстро находить для каждого исходящего ребра оптимальное входящее основание сегмента. Поскольку подходящие по углу основания образуют непрерывный отрезок в массиве отсортированных по углу ребер, это можно делать с помощью очереди с максимумом (скользящее окно) или с помощью дерева отрезков.

Немного подробнее: сначала за линейное время найдем для всех исходящих ребер, можно ли

к ним присоединить подходящие лапы с помощью очереди, использованной в одной из прошлых подгрупп. Далее переберем исходящие из вершины ребра, для которых существуют подходящие лапы. Для каждого такого ребра нужно понять, какие входящие в вершину основания сегментов нам подходят. Если они тоже отсортированы по возрастанию угла, можно найти первое и последнее подходящее бинпоиском и сделать запрос в построенном на них дереве отрезков.

Либо, что более оптимально — если исходящие ребра перебирать в порядке увеличения угла, то подходящие входящие ребра будут изменяться по принципу скользящего окна, так что максимальное значение мощности дракона через эти ребра можно находить с помощью очереди с поддержкой максимума. Итого, при обработке вершин мы один раз обработаем каждое входящее и исходящее ребро. Также нам придется отсортировать ребра, поэтому общее время работы решения $\mathcal{O}(m \log m)$.