Преамбула. Думаю, не стоит приводить базовые определения из теории графов, потому что мы прошли курс АЛКТГ -_-

1. На вход задачи подаётся граф G и его верины s и t. Постройте алгоритм, который за время $O(|V|+|E|)$ проверяет, что вершина t достижима из вершины s. Решите задачу как в случае, когда G неориентированный граф, так и в случае, когда G ориентированный граф.

Для решения этой задачи воспользуемся опцией "звонок другу"и позовем DFS (алгоритм поиска в глубину). Как известно, алгоритм поиска в глубину работает за $O(|V|+|E|)$, так что по времени он нам подходит.

Теперь слегка модифицируем его докажем корректность:

Модификация алгоритма: на каждом шаге проверяем, является ли вершина, в которой мы находимся на текущий момент, вершиной t.

Доказательство корректности: как известно, DFS обходит все вершины графа за указанное выше время. Значит, рано или поздно он либо обойдет весь граф (это будет означать, что данная вершина не достижима), либо зайдет в нужную нам вершину (это, очевидно, будет значит, что вершина достижима). Корректность тривиальна.

Время: поскольку модификация нашего алгоритма на каждом шаге добавляет $O(1)$ операций, то сложность будет такая же, что и у DFS. Значит, этот алгоритм справится за $O(|V| + |E|)$ операций.

P.S. для ориентированного графа тоже используем алгоритм DFS и радуемся жизни (нет, т.к. скоро мидтерм).

2. Докажите, что каждый турнир на n вершинах содержит (простой) путь длины $n-1$. Постройте алгоритм, который получив на вход турнир, находит в нём такой путь, и оцените асимптотику его времени работы.

Докажем с помощью метода математической индукции.

База: турнир из двух вершин. Очевидно, путь длины $2 - 1 = 1$ существует.

Пусть верно для n вершин. Докажем для $n + 1$ (то есть найдем путь длины n):

Берем турнир из n вершин и добавляем туда вершину, которая инцедентна каждой вершине этого турнира.

Если эта вершина "проиграла"всем, то просто добавляем ее в конец найденного ранее (для n вершин) пути. Аналогично, если вершина "выиграла"всех, добавляем ее в начало пути.

Если эта вершина находится между, то в нее можно прийти из t вершин и из нее можно уйти в $n-t$ вершин. Очевидно, найденный ранее путь соединяет одну из t вершин (v_1) с одной из $n - t$ вершин (v_2).

Тогда вместо ребра (v_1, v_2) добавляем нашу вершину и соединяем ее с v_1 и v_2. Мы получили путь длины n. Ура.

Теперь алгоритм:

1) По алгоритму, который был разобран на семинаре, ищем компоненты сильной связности графа.

2) Проходим все вершины компоненты сильной связности.

3) Переходим к следующей КСС и делаем то же самое.

4) Алгоритм завершается, когда мы обошли последнюю КСС.

Корректность тривиальна: (мы обошли все вершины, поэтому длина пути $n-1$).

Время работы алгоритма равно времени работы DFS, то есть $O(|V|+|E|)$, что в случае с полным графом равно $O(n^2)$.

3. В графе G был проведён поиск в глубину. Время открытия и закрытия вершин сохранено в массивах d и f. Постройте алгоритм, который используя только данные из массивов d и f (и описание графа) проверяет, является ли ребро e графа G прямым ребром; перекрёстным ребром. *См. определения в Кормене (глава про поиск в глубину).*

1) Берем две вершины a и b и ребро p, соединяющее их.

2) Проверяем, является ли одно из этих ребер потомком другого (используя алгоритм с семинара).

3) Проверяем, являются ли a и b последовательными вершинами. (Если нет – то ребро p прямое, в противном случае – перекрестное).

Сложность алгоритма зависит только от степеней вершин a и b, то есть она равна $O(max(deg(a), deg(b)))$.

4. В государстве между n городами есть m одностронних дорог. Было решено разделить города государства на наименьшее количество областей так, чтобы внутри каждой области все города были достижимы друг из друга.

1. Предложите эффективный алгоритм, который осуществляет такое разделение, докажите его корректность и оцените асимптотику.

Используем алгоритм разбиения графа на КСС (разбирали на семинаре). Каждая КСС – одна из искомых областей. Корректность тривиальна. Сложность алгоритма – $O(|V|+|E|) = O(m+n)$.

2*. Государство решило добиться того, чтобы из каждого города можно было добраться до каждого. В силу бюджетных ограничений, было решено построить минимальное число односторонних дорог (не важно какой длины), необходимое для достижения этой цели. Предложите алгоритм, решающий задачу.

*Решение предполагает, что государство уже разделило город (граф) на области (КСС).

1) Строим конденсат графа – сложность $O(n)$, т.к. проходимся в худшем случае по каждой вершине.

2) Строим матрицу инцидентности для конденсата.

3) Чтобы из каждого города можно было добраться до каждого, необходимо добавить цикл в наш граф.

4) Делим вершины на две части: (1) в которые никто не входит и (2) из которых нельзя выйти.

5) На каждом ходу берем вершину из 1-ой части и вершину из второй части. Проверяем, что это не одна и та же вершина (такое может случиться с независимой КСС), и соединяем их взаимовыгодным ребром.

Сложность алгоритма – $O(n)$.

5. Вам нужно выбраться из лабиринта. Вы не знаете, сколько в нем комнат, и какая у него карта. По всем коридорам можно свободно перемещаться в обе стороны, все комнаты и коридоры выглядят одинаково (комнаты могут отличаться только количеством коридоров). Пусть m - количество коридоров между комнатами. Предложите алгоритм, который находит выход из лабиринта (или доказывает, что его нет) за $O(m)$ переходов между комнатами. В вашем расположении имеется неограниченное количество монет, которые вы можете оставлять в комнатах, причем вы знаете, что кроме ваших монет, никаких других в лабиринте нет, и вы находитесь в нем одни.

Используем алгоритм DFS, но вместо закрашивания вершин в серый и черный цвета кладем в комнату одну и две монеты соответственно. Также на каждом шаге проверяем, является ли комната выходом из лабиринта.

Таким образом, мы либо обойдем весь лабиринт, либо найдем выход. Время работы алгоритма – $O(m)$.

6. Дан орграф на n вершинах ($V = \{1, \dots, n\}$), который получен из графа-пути (рёбра, которого ведут из вершины i в $i+1$) добавлением ещё каких-то m данных ребер. Найдите количество сильно связных компонент за $O(m \log m)$.

7. На вход задачи поступает описание двудольного графа $G(L, R, E)$, степень каждой вершины которого равна двум. Необходимо найти максимальное паросочетание в G (которое содержит максимальное количество рёбер). Предложите алгоритм, решающий задачу за $O(|V| + |E|)$.

Поскольку наш граф двудольный и степень каждой вершины равна двум, то это граф-цикл! Проверяем четность нашего графа, чтобы понимать, будет ли у нас одинокая вершина. Далее построить максимальное паросочетание в таком графе проще простого! (Просто берем каждое второе ребро, за исключением, быть может, последнего) Поскольку мы проходим по всем ребрам, то наш алгоритм работает за $O(|E|)$.

8. Все степени вершин в неориентированном графе равны $2k$. Все его ребра покрашены в несколько цветов. Предложите $O(V + E)$ алгоритм, который находит в этом графе эйлеров цикл, в котором цвета всех соседних ребер разные (либо выводит, что такого цикла нет).