AI Masters Algorithms **Name:** *Денис Грачев*

Homework 8

6 ноября 2022 г.

Task 1

Из того что рассказывали на лекции, следует, что для этого достаточно запустить DFS из вершины s, и тогда если вершина t достижима, то мы ее посетим в процессе работы алгоритма. Для того чтобы адаптировать алгоритм для неориентированого графа достаточно добавить каждое ребро как два ребра в обе стороны. Тогда достижимость в таком ориентированном графе равносильна достижимости в неориентированном. Сложность DFS это $O(|V| + |E|)$.
Подсмотрев в следующую задачу, можно сказать что сложность просто $O(|E|)$

Task 2

Запустим поиск в глубину из стартовой комнаты. Монетками будем помечать пройденные комнаты. Таким образом если выход достижим, то мы в нем побываем. Так как в данном случае мы запускаемся из одной вершины, то количество комнат (вершин) будет не больше чем количество пройденных коридоров (ребер), поэтому сложность алгоритма будет $O(|V| + |E|) = O(m)$.

Task 3

Рассмотрим самый длинный простой путь $v_1, \dots v_{n'}$ длины $n' - 1 < n - 1$. Тогда найдется вершина v не лежащая в этом пути. Так как граф - турнир, то в нем лежит либо ребро vv_1 либо v_1v. Если есть ребро vv_1 то нашли противоречье с максимальностью пути. Так

же лежит ребро либо $vv_{n'}$ либо $v_{n'}v$. Если есть ребро $v_{n'}v$ то нашли противоречье с максимальностью пути. Следовательно оставшийся вариант v_1v и $vv_{n'}$. Так же не может быть ребра vv_2 иначе можно увеличить путь $v_1vv_2\ldots$. Следовательно есть ребро v_2v. Так же не может быть ребра $v_{n'-1}v$ иначе можно увеличить путь $v_1\ldots v_{n'-1}vv_{n'}$. Следовательно есть ребро $v_vn'-1$. Аналогично есть ребро $v_{n'-2}, v_{n'-3}, \ldots vv_2$. Противоречье так как уже есть ребро v_2v. Следовательно в какой-то момент можно было увеличить простой путь.
Решим задачу поэлементно добавляя новые вершины. Добавление новой вершины в путь стоит $O(k)$ проверок, где k длина текущего пути. Следовательно итоговая сложность $O(1) + \ldots + O(n) = O(n^2)$.

Task 4

1

Заметим что государство хочет разделить города на КСС. Так как если из города a достижим b и наоборот то это необходимое и достаточное условие чтобы они были в одной компоненте связности и в одной области. Так как компоненты связности максимальные то и количество областей будет минимальным возможным. Из рассказанного на лекции это можно сделать за $O(m+n)$ операций.

Task 5

Будем хранить добавленные m ребер как массив пар $p = [(f_1, t_1), \ldots (f_m, t_m)]$. Тогда ребра где $f_i < t_i$ не влиют на связность никак. Их мы рассматривать не будем и чтобы их удалить необходимо $O(m)$ операций. Для этого мы пройдемся по массиву и для каждого индекса посчитаем сколько таких ребер стояло перед этим индеком. После этого каждый элемент сдвинем на соотвествующее число наверх, а эти ребра просто пропустим. Таким образом удалим все неинтересующие нас ребра.

Отсортируем остаток массива в лексикографическом порядке. $O(m \log m)$ *Кажется можно использовать RadixSort за* $O(m)$.

Создадим массив k в который для каждого индекса запишем минимальный достижимый из него элемент. Для элементов из которых не выходят ребра это они же сами, поэтому их можно не создавать и не хранить. Затем пройдемся по остатку массива ребер и будем определять по формуле $k[f_i] = \min(k[f_i], k[t_i])$. Так как массив p остортирован по исходящему индексу, то сначала корректно заполняется индексы до f_i. Корректность заполнения очевидна, сложность $O(m)$. После этого отсортировав массив p мы так же получим для каждого индекса максимальный индекс из которого его можно достичь.

После этого пройдясь по полученному массиву и перепрыгивая каждый раз на следующий максимальный элемент из которого достижим текущий, мы будем итерироваться по компонентам связности данного графа и можем их посчитать. Итогвоая сложность $O(m \log m)$

Task 7

Проведем топологическую сортировку DAG. Это занимает $O(|V| + |E|)$ операций. Заметим, что наидленнейший путь выходит из вершины топологической сортировки, иначе его можно было бы продлить. Запустим DFS из вершины и будем запоминать из каких вершин мы в нее попали. Таким образом нам удастся восстановить наидленнейший путь.