Алгоритмы 7

Максим Пасько

1 апреля 2018 г.

Задание 1

Предположим противное: пусть в G существует цикл. Рассмотрим этот цикл, пусть его путь такой:

$$a_1 \to a_2 \to ... \to a_k \to a_1.$$

Тогда попытаемся занумеровать эти вершины так, как сказано в условии. Тогда $a_1 > a_2 > ... > a_k > a_1$, получили противоречие $\Rightarrow$ G ациклический.

Задание 2

Пусть $n = 2$. Тогда существует простой путь длины 1. Пусть верно для k вершин. Рассмотрим турнир T с $k+1$ вершинами: зафиксируем вершину v_0 в T. Пусть простой путь в $T \setminus \{v_0\}$ такой: $v_1, v_2, ..., v_k$. Пусть $i \in \{0, 1, 2, ..., k\}$ - максимальное такое число, что $\forall j \leq i \quad \exists (v_j, v_0)$. Тогда $\forall j' > i \quad \exists (v_0, v_{j'})$. Тогда искомый простой путь будет $v_1, v_2, ..., v_i, v_0, v_{i+1}, ..., v_k$.

Алгоритм нахождения пути прямо следует из доказательства. Возьмём произвольную вершину турнира v_1, пометим её как использованную. Для каждой следующей будем искать такое максимальное i, как и в доказательстве, для которого выполняется: $\forall j \leq i \quad \exists (v_j, v_k)$, где v_k - вершина, которую мы взяли на нынешнем шаге, а v_j - использованные вершины. Таким образом, построим турнир. Асимптотика алгоритма: на k-ом шаге ищем такое $i \in \{0, 1, ..., k\}$. Итоговая асимптотика - $O(n^2)$.

Задание 3

a)

Пусть надо проверить ребро (v_1, v_2). Тогда для того, чтобы оно было прямым, необходимо, чтобы $d[v_1] < d[v_2]$, иначе предок откроется позже, чем потомок, что невозможно, и $f[v_1] > f[v_2]$, иначе предок закроется раньше, чем потомок, что опять же невозможно. Эти условия также являются и достаточными, т.к. если вершина v_2 открылась позже v_1, а закрылась раньше, то это значит, что мы, придя к вершине v_2, в какой-то момент вышли из вершины v_1.

б)

Такое ребро не должно являться ни прямым, ни обратным. Тогда добавим такое условие: $(d[v_1] > f[v_2]) \vee (f[v_1] < d[v_2])$. Таким образом, вершины v_1 и v_2 не родственные, и ребро будет перекрёстным.

Задание 4

Очевидно, что аналогичная задача - это поиск компонент сильной связности в графе. Тогда и искомым алгоритмом будет алгоритм поиска компонент сильной связности, основанный на поиске в глубину, транспонировании графа и ещё одном поиске в глубину, который работает за $O(n+m)$.

Задание 5

В комнате, в которой мы сейчас, положим две монеты. После этого, зайдём в каждую комнату, в которую мы можем попасть из той, где мы сейчас находимся, и оставим в них по одной монете. После этого, для каждой комнаты проделаем ту же процедуру, что мы и проделали для первой вершины, при этом, если мы вошли в вершину, в которой две монеты, выходим из неё обратно. Таким образом, мы обойдём все комнаты.

Этот алгоритм есть поиск в ширину, который работает за $O(|V|+|E|)$. Т.к. $|E| \leq (|V|)^2$, то его асимтотика представима как $O(|E| + \sqrt{|E|}) = O(|E|) = O(m)$.

Задание 6

Компонента сильной связности может существовать только в том случае, если добавленное ребро ведёт от вершины с большим номером к вершине с меньшим номером.

Создадим массив рёбер, отсортируем их по конечной вершине за $O(m \log m)$. Будем идти по массиву рёбер, компонента сильной связности будет определяться таким образом: если конец ребра находится между началом и концом какой-либо другой компоненты связности, а номер начала ребра больше всех номеров компоненты связности, то новая компонента связности есть объединение множеств вершин этой компоненты связности и множества вершин, которые покрывает этот отрезок.

Асимптотика алгоритма есть асимптотика сортировки - $O(m \log m)$.