Петровский Александр, M3239.
Домашняя работа 1

Задача 1

Для начала заметим, что если из вершины v существует путь в вершину u и из u — путь в v, то ответ для них будет одинаковым, так как мы можем добраться из v в u, а значит можем добраться из v и до всех вершин, что достижимы из u и наоборот: из u можем добраться до v, а также до всех вершин, что достижимы из v. Поэтому мы можем сконденсировать граф и посчитать ответы для компонент сильной связности (КСС). Для этого для каждой КСС возьмем максимальное число из вершин, принадлежащей ей. Так как граф конденсации ацикличен, то, используя динамическое программирование и dfs, получим ответ для каждой КСС.

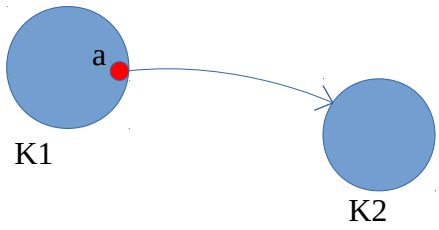
```
for v in SCC            // для каждой вершины из графа конденсации запустим dfs
    if not used[v]
        dfs(v);

int dfs(v): // dfs, считающий дп
    used[v] = true;
    mx[v] = num[v];
    for to in g[v]
        if (not used[to])
            mx[v] = max(mx[v], dfs(to))
    return mx[v]
```

Ответ для вершины — ответ для КСС, которой она принадлежит.

Задача 2

Если из вершины v можно дойти но любой вершины из множества U, то будем говорить, что вершины v покрывает множество U.
Заметим, что если некоторые вершины образуют КСС, то чтобы покрыть КСС можно взять любую одну вершину из этой КСС, поэтому сконденсируем граф. Сейчас будем работать с графом конденсаций. Задача в том, чтобы выбрать минимальное количество вершины, которые покрывают все множество в ациклическом графе. Понятно, что это множество будет содержать вершины, у которых нет входящих ребер, так как нет других вершины, которые могли бы их покрыть (потому что нет вершины, из которой в них можно попасть). Понятно, что больше вершин не требуется [Если это не так, то есть вершина, у которой есть входящее ребро, мы можем подняться к его «предку», к «предку предка» и так дойти до вершины без входящих ребер, при этом покрывая каждый раз большее множество вершин]. Также такое множество будет покрывать весь граф[так как у любой вершины графа, не принадлежащей ответу есть входящее ребро, а значит можно добраться до вершины из ответа по обратным ребрам, а значит, оно покрывается]. Итак, мы решили задачу для графа конденсации. Для ответа на исходную задачу для каждой КСС, которая вошла в ответ, выпишем любую вершину, принадлежащую ей. Понятно, что этот ответ покрывает все множество вершин и минимален по построению.

Задача 3



(а) Рассмотрим две компоненты сильной связности в графе, а именно обратим внимание на времена входа в вершины КСС K1 и времен входа в вершины КСС K2. Тут есть два случая при запуске dfs:
1) Мы зашли сначала в K1. Тогда для вершины a верно, что время входа в нее будет больше, чем времена входа любой вершины в K2.
2) Мы зашли сначала в K2. Тогда мы не попадем в K1, пока не обойдем K2 полностью, а значит время входа в любую вершину будет K2 будет меньше, чем время входа любой вершины из K1.
Заметим, что это значит, что при сортировка зависит от того, как мы запустили dfs, а это значит, что она не однозначна, а значит может получится так, что объединятся два различные компоненты сильной связности (а именно, если мы при сортировке получим, что вершина из K2 стоит раньше всех вершин из K1, то мы в КСС добавим все вершины K2 и по обратному ребру дойдем до K1 и добави все ее вершины, потому что они еще не были посещены).
Ответ: Не стоит применять такую модификацию, busted

(б) В алгоритме мы используем обратные ребра, что бы не выйти из компоненты сильной связности. После сортировки первым оказывается элемент из компоненты сильной связности, в которую не ведут ребер и из нее запускается «покраска ребер». Если мы не будет ходить по обратным ребрам, то во время покраски K1 окажемся в K2 и покрасим ее, а K1 и K2 не образуют компоненту сильной связности вместе.
Ответ: Не стоит применять такую модификацию, busted

UPDATE: Теперь изменим пункт (а): упорядочим вершины по времени выхода по возрастанию и теперь пункт (а) будет выполняться с пунктом (б)

Заметим теперь, что сортировка будет однозначна. Если мы сначала зайдем в K2, то все нормально, потому что все времена выхода из K2 будет меньше всех времен выхода из K1 и в таких случаях использование прямых ребер нам подходит, так как мы не сможем выйти за компоненту сильной связности и таким образом последовательно пройдемся по всем. Однако рассмотрим поподробнее, если наш dfs сначала зашел в K2. Тогда если мы можем дойти до красной вершинки так, что из нее будет как путь в K2, так и в некоторые оставшиеся вершины K1 (желтенький). Если мы сначала пройдемся по желтому пути, а потом перейдем в K2, то время выхода в желтеньком будет меньше, чем времена выхода в K2, а значит, что после сортировки будет так, что «желтая» вершина будет раньше всех вершин из K2, а значит, что при покраске КСС мы по прямому ребру перейдем в K2 и получится так, что K1 и K2 в одной КСС, но это не так.
Ответ: Не стоит применять такую модификацию, double busted