Разбор задачи «F. Бюрократия»

Будем рекурсивно обходить дерево, начиная с корня, и решать задачу для каждого поддерева. $d[v]$ – ответ для поддерева вершины v, $sz[v]$ – размер поддерева вершины v. Для каждого рёбенка u вершины v верно, что за все работы, сделанные в поддереве вершины u, человек v получит столько же денег, сколько получил человек u, плюс 1 за каждую работу. Т.е. в $d[v]$ мы должны добавить $d[u] + sz[u]$. Обработав всех детей вершины v мы учтём все работы из поддерева v кроме одной: той, которую выполнит сам человек v, получив за это единицу денег. Т.о. $d[v] = 1 + \sum\limits_{u:\ child\ v} d[u] + sz[u]$.

Время работы: $\mathcal{O}(n)$.

Note: для хранения значений $d[v]$ требуется 64-битный тип данных.

Разбор задачи «G. Дятлы»

Для каждой компоненты связности найдём число размещений вершин этой компоненты. Компоненты размера 1 оставим на потом. Компоненты размера 2 можно разместить двумя способами – первая вершина на первом дереве, вторая на втором, и наоборот. Для компонент большего размера нужно сначала проверить, можно ли вообще разместить вершины этой компоненты. Это возможно только если компонента является так называемым графом-гусеницей – простой путь, к каждой вершине которого могут быть прикреплены вершины степени 1.

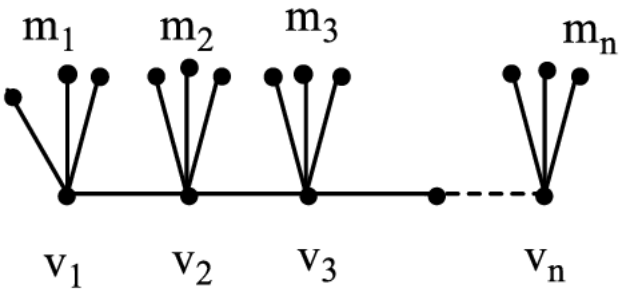

Рис. 1: Граф-гусеница

Чтобы проверить, является ли граф гусеницей, можно найти начало пути – вершину, у которой есть ровно один сосед степени не 1, и обойти весь путь, состоящий из вершин степени не 1, для каждой вершины пути проверяя, что из неё есть ровно один переход дальше по пути в вершину степени не 1.

Разместим вершину v_1 на первом дереве, m_1 её соседей степени 1 раместим на втором дереве произвольным образом упорядочив их между собой, вершину v_2 разместим сразу над верхней из m_1 вершин, m_2 её соседей степени 1 разместим на первом дереве над вершиной v_1 и т.д. Для всех вершин v_i позиция их на дереве определяется однозначно. Для вершин из групп m_i позиция определяется однозначно для всей группы в целом, но размещение вершин внутри группы может быть любым, т.е. группу m_i можно разместить $m_i!$ способами. Тогда всю текущую компоненту связности можно разместить $\prod m_i!$ способами.

Сейчас мы учли не все размещения текущей компоненты, а только те, в которых вершина v_1 размещена на первом дереве и другие вершины, размещённые на первом дереве, находятся над ней. Для каждого такого размещения существует ещё одно аналогичное размещение, в котором мы размещаем вершину v_1 не на первом дереве, а на втором, и ещё два размещения, аналогичные первым двум, в которых мы начинаем размещать вершины, начиная с v_n, а не с v_1. Т.о. итоговое количество размещений текущей компоненты равно $4 \cdot \prod m_i!$.

Найдём теперь количество размещений для всех компонент связности размера не 1. За c_i обозначим количество размещений i-й компоненты, за t обозначим количество компонент. Размещения разных компонент не могут пересекаться между собой, т.е. на дереве между вершинами одной компоненты не может быть вершин другой компоненты. Иными словами, компоненты размещаются независимо друг от друга. Если мы будем сначала размещать на деревьях компоненту 1, за ней

следом выше по деревьям размещать компоненту 2, и т.д. до компоненты t, общее количество таких размещений будет равно $\prod c_i$. Если мы будем размещать компоненты в каком-то другом порядке, количество размещений по-прежнему будет равно $\prod c_i$. Количество порядков компонент равно $t!$, и тогда количество способов разместить все компоненты размера не 1 равно $t! \cdot \prod c_i$.

Осталось разместить вершины из компонент размера 1. Обозначим количество вершин в компонентах размера не 1 за k. Пусть мы каким-то образом разместили эти k вершин, причём k_1 вершин размещены на первом дереве, и k_2 – на втором ($k_1 + k_2 = k$). Компоненту размера 1 можно разместить на первом дереве $k_1 + 1$ способами: в самом низу дерева либо непосредственно над любой из k_1 вершин. На втором дереве компоненту размера 1 можно разместить $k_2 + 1$ способами. Т. о. общее количество размещений компоненты 1 равно $k1 + 1 + k_2 + 1 = k + 2$ и не зависит от того, сколько из k вершин были размещены на первом дереве, а сколько на втором. После размещения одной компоненты размера 1, вторую можно разместить $k + 3$ способами, третью – $k + 4$ способами и т.д.

Итоговое количество размещений равно $t! \cdot \prod_{i=1}^{t} c_i \cdot \prod_{i=1}^{n-k} k + i$.

Время работы: $\mathcal{O}(n + m)$.

Разбор задачи «H. Вершинно-реберное покрытие дерева*»

Будем рекурсивно обходить дерево, начиная с вершины 1, и решать задачу для каждого поддерева. Для поддерева рассмотрим два случая: корень поддерева либо помечен, либо не помечен. $d[v][0]$ – оптимальный ответ для поддерева вершины v в случае, когда вершина v не помечена, а $d[v][1]$ – оптимальный ответ, когда вершина v помечена. Если мы не помечаем вершину v, то мы обязаны пометить каждого её ребёнка u, и тогда $d[v][0] = \sum d[u][1]$. Если мы помечаем вершину v, то мы платим её стоимость a_v, а её детей мы можем как пометить, так и не пометить, и тогда $d[v][1] = a_v + \sum \min(d[u][0], d[u][1])$.

Это решение верно всегда кроме случая, когда в дереве всего одна вершина. В этом случае мы обязаны пометить её и заплатить её стоимость.

Итоговый ответ на задачу: если $n = 1$, то ответ a_1, иначе ответ $\min(d[1][0], d[1][1])$.

Для построения списка помеченных вершин рекурсивно обойдём дерево ещё раз. Вершину v будем добавлять в список, если выполняется хотя бы одно из двух условий: предок вершины v существует и не добавлен список, или $d[v][0] > d[v][1]$.

Время работы: $\mathcal{O}(n)$.

Разбор задачи «I. Пара путей*»

В качестве корня дерева выберем вершину 1. Рекурсивными обходами дерева, начинающимися с вершины 1, будем насчитывать следующие величины:

$d_1[v]$ – количество рёбер в самом длинном пути в поддереве вершины v, начинающемся в корне поддерева. Любой путь из корня – это путь из ребёнка корня, к которому добавили корень, поэтому самый длинный путь из корня v – это самый длинный путь из какого-то ребёнка u плюс одна вершина. Т.е. $d_1[v] = \max\limits_{u:\ child\ v} 1 + d_1[u]$.

$d_2[v]$ – количество рёбер в самом длинном пути в поддереве вершины v, необязательно начинающемся в корне поддерева. Если самый длинный путь не проходит через корень поддерева, то он целиком содержится в поддереве одного из детей корня. Его длина равна $\max\limits_{u:\ child\ v} d_2[u]$. Если же самый длинный путь проходит через корень поддерева, то он состоит из двух путей от корня, продолжающих самые длинные пути в детях корня. Его длина равна $\max\limits_{u_1,u_2:\ children\ v} 2 + d_1[u_1] + d_1[u_2]$. Т.о. $d_2[v] = \max(\max\limits_{u:\ child\ v} d_2[u], \max\limits_{u_1,u_2:\ children\ v} 2 + d_1[u_1] + d_1[u_2])$.

$d_3[v]$ – количество рёбер в самом длинном пути во всём дереве, начинающемся в вершине v и проходяшего через родителя вершины v. Рассмотрим рекурсивный переход из v в её ребёнка u. $d_3[u]$ является длиной пути, начинающемся в u и идущем в v, после чего возможны два случая: путь дальше идёт либо в родителя v, либо в какого-то другого ребёнка s вершины v. В первом случае длина

пути будет равна $d_3[v]+1$. Во втором случае путь является продолжением самого длинного пути из s внутри поддерева s, и его длина равняется $d_1[s]+1$. Т.о. $d_3[u] = \max(1+d_3[v], \max\limits_{s:\ child\ v;\ s \neq u} 1+d_1[s])$.

Теперь приступим к вычислению ответа на задачу. Представим, как в дереве расположены два пути, которые максимизируют ответ. Обязательно найдётся такая вершина, что через неё проходит один из путей, и какой-то из её детей содержит все вершины второго пути и ни одной вершины первого пути. $d_4[v]$ – максимальное произведение количеств ребёр в двух не пересекающихся по вершинам путях, один из которых проходит через вершину v, а второй содержится целиком в поддереве какого-то ребёнка v.

Пусть первый путь идёт через родителя v. Тогда его можно разбить на две части: первая идёт в родителя v, и её длина равна $d_3[v]$, а вторая идёт в какого-то ребёнка u вершины v, и её длина равна $d_1[u]$, а длина всего первого пути равна $d_3[v] + d_1[u] + 1$. При этом второй путь находится в каком-то другом ребёнке s, не равном u, и его длина равна $d_2[s]$. Т.о. максимальное произведение количеств рёбер равно $\max\limits_{u:\ child\ v} (d_3[v] + d_1[u] + 1) \cdot (\max\limits_{s:\ child\ v;\ s \neq u} d_2[s])$.

Пусть первый путь не идёт через родителя v. Тогда две его части идут в детей u и t вершины v, и его длина равна $d_1[u] + d_1[t] + 2$. Второй путь всё так же находится в каком-то ребёнке s, не равном u и t, и его длина равна $d_2[s]$. Т.о. максимальное произведение количеств рёбер равно $\max\limits_{u,\ t:\ children\ v} (d_1[u] + d_1[t] + 2) \cdot (\max\limits_{s:\ child\ v;\ s \neq u;\ s \neq t} d_2[s])$.

Итоговое значение $d_4[v] =$
$= \max($

$$\max_{u:\ child\ v} (d_3[v] + d_1[u] + 1) \cdot (\max_{s:\ child\ v;\ s \neq u} d_2[s]),$$
$$\max_{u,\ t:\ children\ v} (d_1[u] + d_1[t] + 2) \cdot (\max_{s:\ child\ v;\ s \neq u;\ s \neq t} d_2[s])$$

$)$.

Ответ на задачу – максимальное значение среди всех $d_4[v]$.

Время работы: $\mathcal{O}(n)$.

Разбор задачи «J. Количество топсортов дерева*»

Оригинальный разбор этой задачи находится по этой ссылке, задача «Упорядочивания». Ниже разобрано моё собственное решение, отличающееся от авторского.

Note: все вычисления в этой задаче выполняются по модулю $10^9 + 7$.

Сначала насчитаем $\binom{n}{k}$ – число сочетаний из n по k, – по рекуррентной формуле: $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.

Будем рекурсивно обходить дерево, начиная с вершины 1, не учитывая ориентацию рёбер. Для каждой вершины v будем вычислять $d[v][i]$ – количество топсортов поддерева v, в которых сама вершина v находится на позиции i. Изначально $d[v]$ учитывает только одну вершину v, и все $d[v][i] = 0$ кроме одного: $d[v][0] = 1$ (нумерация позиций идёт с нуля). После рекурсивного вызова из ребёнка u вершины v будем вычислять новые значения $d'[v][i]$, изначально все равные 0, которые будут учитывать вершины поддерева u, и после этого присваивать $d[v] = d'[v]$.

Пусть ребро направлено **из** v **в** u. Рассмотрим два типа топсортов: топсорты уже учтённых вершин поддерева v, в котором v находится на позиции i, и топсорты вершин поддерева u, в котором u находится на позиции j. Введём обозначения:

- L_v – вершины топсорта учтённых вершин, находящиеся слева от v;
- R_v – вершины топсорта учтённых вершин, находящиеся справа от v;
- L_u, R_u – аналогично для топсорта вершин поддерева u вершины слева и справа от u;
- $sz[v]$ – количество учтённых вершин в поддереве v;
- $sz[u]$ – количество вершин в поддереве u;
- $|L_v| = i$ – количество вершин в L_v;

- $|R_v| = sz[v] - i - 1$ – количество вершин в R_v;
- $|L_u| = j$, $|R_u| = sz[u] - j - 1$ – количества вершин в L_u и R_u.

При добавлении вершин поддерева u к учтённым вершинам, из двух рассматриваемых видов топсортов получаются несколько видов топсортов, содержащих все вершины. Для каждого из них верно, что:

- все вершины из L_v находятся слева от v, т.к. они и раньше были слева от v;
- все вершины из R_v находятся справа от v, т.к. они и раньше были справа от v;
- u находится **справа** от v, т.к. ребро направлено **из** v **в** u;
- все вершины из R_u находятся **справа** от v, так как они раньше были справа от u, которая сейчас находится справа от v.

Вершины из L_u могут находится как слева от v, так и справа, поэтому позиция корня в новых топсортах может быть разной: она может быть равна i, если 0 вершин из L_u оказались слева от v, или она может быть равна $i + 1$, если 1 вершина из L_u оказалась слева от v, и т.д. до $i + |L_u|$, когда все вершины из L_u оказываются слева от корня. Т.е. получится $|L_u| + 1$ видов топсортов, в которых вершина v находится на позиции $i + k$, где $k \in [0, |L_u|]$.

Вершины из L_v и k вершин из L_u находятся слева от v, и нам не важен относительный порядок между вершинами из L_v и вершинами из L_u, т.е. k вершин из L_u можно вставить между вершинами L_v произвольным образом (но не меняя местами никакие вершины из L_v и никакие вершины из L_u). Это можно сделать $\binom{|L_v|+k}{|L_v|}$ способами. Аналогично справа от v находятся вершины из R_v, вершина u, вершины из R_u и $|L_u| - k$ вершин из L_u, и их можно разместить $\binom{|R_v|+1+|R_u|+|L_u|-k}{|R_v|}$ способами.

Т.о. из каждого топсорта уже учтённых вершин поддерева v, в котором v находится на позиции i (их количество равно $d[v][i]$), каждого топсорта вершин поддерева u, в котором u находится на позиции j (их количество равно $d[u][j]$), и каждого $k \in [0, |L_u|]$ можно получить $\binom{|L_v|+k}{|L_v|} \cdot \binom{|R_v|+1+|R_u|+|L_u|-k}{|R_v|}$ топсортов, в которых v находится на позиции $i + k$. Т.е. для всех $i \in [0, sz[v]]$, всех $j \in [0, sz[u]]$ и всех $k \in [0, j]$ нужно добавить к $d'[v][i + k]$ значение $d[v][i] \cdot d[u][j] \cdot c[|L_v| + k][|L_v|] \cdot c[|R_v| + 1 + |R_u| + |L_u| - k][|R_v|]$.

Аналогичным образом рассматривается случай, когда ребро направлено **из** u **в** v. Рассмотрим те же два вида топсортов и воспользуемся теми же обозначениями.

При добавлении вершин поддерева u к учтённым вершинам, из двух рассматриваемых видов топсортов получаются несколько видов топсортов, содержащих все вершины. Для каждого из них верно, что:

- все вершины из L_v находятся слева от v, т.к. они и раньше были слева от v;
- все вершины из R_v находятся справа от v, т.к. они и раньше были справа от v;
- u находится **слева** от v, т.к. ребро направлено **из** u **в** v;
- все вершины из L_u находятся **слева** от v, так как они раньше были слева от u, которая сейчас находится слева от v.

Вершины из R_u могут находится как слева от v, так и справа, поэтому позиция корня в новых топсортах может быть разной: она может быть равна $i + 1 + |L_u|$, если 0 вершин из R_u оказались слева от v, или она может быть равна $i + 1 + |L_u| + 1$, если 1 вершина из R_u оказалась слева от v, и т.д. до $i + 1 + |L_u| + |R_u|$, когда все вершины из R_u оказываются слева от корня. Т.е. получится $|R_u| + 1$ видов топсортов, в которых вершина v находится на позиции $i + 1 + |L_u| + k$, где $k \in [0, |R_u|]$.

Вершины из L_v, вершина u, вершины из L_u и k вершин из R_u находятся слева от v, и они «смешиваются» между собой $\binom{|L_v|+1+|L_u|+k}{|L_v|}$ способами. Справа от v находятся вершины из R_v и $|R_u| - k$ вершин из R_u, и они «смешиваются» между собой $\binom{|R_v|+|R_u|-k}{|R_v|}$ способами.

Т.о. для всех $i \in [0, sz[v]]$, всех $j \in [0, sz[u]]$ и всех $k \in [0, j]$ нужно добавить к $d'[v][i+1+|L_u|+k]$ значение $d[v][i] \cdot d[u][j] \cdot c[|L_v|+1+|L_u|+k][|L_v|] \cdot c[|R_v|+|R_u|-k][|R_v|]$.

Для ясности рассмотрим псевдокод того, что в целом происходит:

```
def dfs(v):
    d[v][0] = 1
    sz[v] = 1
    used[v] = True
    for u in out_edges[v]:
        if used[u]:
            continue
        dfs(u)
        d_prime[v] = [0] * n
        for i in range(sz[v]):
            for j in range(sz[u]):
                for k in range(j + 1):
                    d_prime[v][i + k] += _first_big_scary_formula_
        d[v] = d_prime[v]
        sz[v] += sz[u]

    for u in in_edges[v]:
        if used[u]:
            continue
        dfs(u)
        d_prime[v] = [0] * n
        for i in range(sz[v]):
            for j in range(sz[u]):
                for k in range(j + 1):
                    d_prime[v][i + 1 + j + k] += _second_big_scary_formula_
        d[v] = d_prime[v]
        sz[v] += sz[u]
```

Текущее решение не укладывается в ограничение по времени, поэтому соптимизируем его.

Рассмотрим ещё раз случай рекурсивного перехода из v в u, когда ребро направлено из v в u. Посмотрим на то, какие значения добавляются к $d'[v][i]$:

- при $j = 0$ добавляется $d[v][i] \cdot d[u][0] \cdot \binom{|L_v|}{|L_v|} \cdot \binom{|R_v|+1+|R_u|+|L_u|}{|R_v|}$
- при $j = 1$ добавляется $d[v][i] \cdot d[u][1] \cdot \binom{|L_v|}{|L_v|} \cdot \binom{|R_v|+1+|R_u|+|L_u|}{|R_v|}$
- при $j = 2$ добавляется $d[v][i] \cdot d[u][2] \cdot \binom{|L_v|}{|L_v|} \cdot \binom{|R_v|+1+|R_u|+|L_u|}{|R_v|}$
- и т.д.

У всех этих слагаемых есть большой общий множитель $d[v][i] \cdot \binom{|L_v|}{|L_v|} \cdot \binom{|R_v|+1+|R_u|+|L_u|}{|R_v|}$, и если вынести его за скобку, то за скобками останется $d[u][0] + d[u][1] + \ldots + d[u][sz[u]-1]$.

Посмотрим на то, какие значения добавляются к $d'[v][i+1]$:

- при $j = 0$ не добавляется ничего
- при $j = 1$ добавляется $d[v][i] \cdot d[u][1] \cdot \binom{|L_v|+1}{|L_v|} \cdot \binom{|R_v|+1+|R_u|+|L_u|-1}{|R_v|}$
- при $j = 2$ добавляется $d[v][i] \cdot d[u][2] \cdot \binom{|L_v|+1}{|L_v|} \cdot \binom{|R_v|+1+|R_u|+|L_u|-1}{|R_v|}$
- и т.д.

Теперь общий множитель – $d[v][i] \cdot \binom{|L_v|+1}{|L_v|} \cdot \binom{|R_v|+1+|R_u|+|L_u|-1}{|R_v|}$, а за скобками остаётся $d[u][1] + d[u][2] + \ldots d[u][sz[u]-1]$.

В общем случае к $d'[v][i+k]$ добавляется $d[v][i] \cdot \binom{|L_v|+k}{|L_v|} \cdot \binom{|R_v|+1+|R_u|+|L_u|-k}{|R_v|} \cdot \sum_{j=k}^{sz[u]-1} d[u][j]$. Последний множитель – это суффиксная сумма, которую можно получать за $\mathcal{O}(1)$.

Note: $|L_u| + |R_u| = sz[u] - 1$.

Пересчёт этого случая в псевдокоде будет выглядеть так:

```
for u in out_edges[v]:
    if used[u]:
        continue
    dfs(u)
    d_prime[v] = [0] * n
    for i in range(sz[v]):
        for k in range(sz[u]):
            d_prime[v][i + k] += _less_scary_formula_ * suf[u][k]
    d[v] = d_prime[v]
    sz[v] += sz[u]
```

Для случая рекурсивного перехода из v в u, когда ребро направлено из u в v, аналогичным образом получаем, что в общем случае к $d'[v][i+1+k]$ добавляется $d[v][i] \cdot \binom{|L_v|+1+k}{|L_v|} \cdot \binom{|R_v|+|L_u|+|R_u|-k}{|R_v|} \cdot \sum_{j=0}^{k} d[u][j]$.

Оценим время работы. От каждой вершины v мы запускаем рекурсивную функцию, самая долгая часть которой – два вложенных цикла, внутри которых идёт персчёт значений $d'[v]$. Количество итераций внешнего цикла равно количеству уже учтённых вершин, количество итераций внутреннего цикла равно количеству вершин в поддереве ребёнка. Общее количество итераций этих двух циклов равно количеству пар вершин, для которых v является `LCA`. Тогда суммарно все запуски функции отработают за время, эквивалетное количеству всех пар вершин в дереве, т.е. за $\mathcal{O}(n^2)$.