Iterative compression

Итерационные алгоритмы сжатия разрабатываются для задач параметрической минимизации, где целью является нахождение небольшого набора вершин или ребер графа, удаление которых делает граф удовлетворяющим некоторому глобальному свойству. Верхней границей этого множества является параметр k

Def Процедура сжатия - это алгоритм, который по паре задача + ее решение либо вычисляет решение меньшего размера, либо доказывает, что размер текущего решения наименьший

Основана идея в том, что если мы умеем делать сжатие за FTP time, то и весь алгоритм работает за FTP time. Это позволяет использовать важную информацию о структуре решения, а не только о самой задаче, так что построить сжатие может быть проще, чем прямой FTP алгоритм.

FTP-algorithm - умеем решать за $f(k)*n^{O(1)}$, где k - параметр

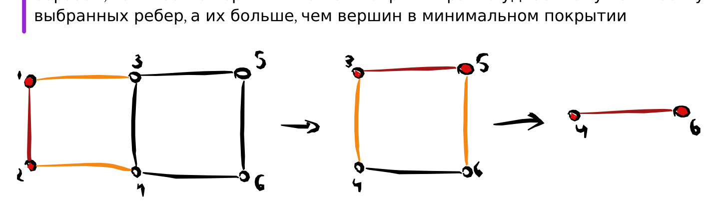
Basic technique

Основана идея в том, что если мы умеем делать сжатие за FTP time, то и весь алгоритм работает за FTP time. Это позволяет использовать важную информацию о структуре решения, а не только о самой задаче, так что построить сжатие может быть проще, чем прямой FTP алгоритм.

Def (Minimum) vertex cover - задача о нахождении минимального множества вершин в графе, такого, что любое ребро имеет как минимум один конец в этом множестве (NP-hard)

Alg (1) 2-approximation algorithm - случайным образом выбираем ребро, затем берем оба его конца в вершинное покрытие, затем удаляем из графа эти вершины и все инцедентные им ребра и повторяем процедуру.

Proof Допустим, в минимальном покрытии было бы ровно n вершин, тогда $n >=$ количества выбранных ребер, поскольку все выбранные нами ребра не смежны. Но мы брали в наше покрытие оба конца выбранного ребра, таким образом, количество вершин в нашем покрытии равно удвоенному количеству выбранных ребер, а их больше, чем вершин в минимальном покрытии



Alg (2) Предположим, нам дан Vertex cover (G, k). Используя алгоритм 2-approximation algorithm получаем множество Z. Ясно, что $|Z| <= 2k$ (иначе (G, k) не решаема), так что теперь мы можем упорядочить ветвление на Z в то время как G-Z не имеет ребер.
Шаг ветвления:

Перебираем все возможные пересечения оптимального решение X и имеющегося Z;

$X_Z = X \cap Z$

$W = X \setminus X_Z$

Ищем верш. покр. размера $\leq k$, т.ч. $X \cap Z = X_Z$.

1. В $G(W)$ есть ребро $\Rightarrow$ X - не верш. покрытие
2. Заметим, что $\forall X: X \cap Z = X_Z \Rightarrow$

$X \supset \underbrace{N_G(W)}_{\{v \in Vertex(G) \mid \exists w \in W: (v\, w) \in Edges(G)\}} \cup X_Z$

но $G-X$ не имеет ребер, так что $N_G(W) \cup X_Z$ - верш. покрытие.
Значит если найдется $X_Z \subseteq Z: |X_Z \cup N_G(W)| \leq k$ то возвращаем $N_G(W) \cup X_Z$, иначе (G, k) - не валидна.
Таким образом получаем время $\boxed{2^{|Z|} \cdot n^{O(1)} \leq 4^k \cdot n^{O(1)}}$

Полученное время все еще не является оптимальным. Причина этого заключается в том, что мы начали с большого множества Z, размер которого не превышает $2k$. Но существует простой и очень универсальный способ получения множества Z размера $k + 1$.

Alg (3) Зафиксируем произвольный порядок $(v1, v2, \ldots, vn)$ в G.
Для i из $\{1, \ldots, k\}$, обозначим через G_i подграф G, индуцированный первыми i вершинами.
Для $i = k$ мы можем принять вершинное множество G_i как вершинное покрытие X в G_i размера k.

Далее действуем итеративно:
Предположим, что для некоторого $i > k$ мы построили вершинное покрытие X_i графа G_i размером не более k, тогда в графе $G\_\{i+1\}$ множество $Z\_\{i+1\} := X\_i \cup \{v\_\{i+1\}\}$ является вершинным покрытием размером не более $k + 1$.
1. Если на самом деле $|Z\_\{i + 1\}| \leq k$, то все готово: можно просто положить $X\_\{i+1\} = Z\_\{i+1\}$ и перейти к следующей итерации.
2. Иначе $|Z\_\{i+1\}| = k + 1$, и нам необходимо сжать слишком большое решение.

Применяя этот алгоритм ветвления (2), т.е. решая $2\^|Z\_\{i+1\}| = 2\^\{k+1\}$ экземпляров Disjoint Vertex Cover, за время $2\^\{k+1\} * n\^O(1)$, мы можем либо найти вершинное покрытие покрытие $X\_\{i+1\}$ из G_i размером не более k, либо сделать вывод о том, что такого покрытия не существует.

Как этот метод может быть применен к задаче о графе?
Основная идея заключается в разработке FPT-алгоритма, который для заданного $(k + 1)$-размерного решения задачи либо сжимает его до решения размером не более k, либо доказывает что решения размера не более k не существует. Это называется шагом сжатия алгоритма.
Обычно используется метод, при котором мы начинаем с подграфа, тривиально допускающего решение размера k, и затем итеративно расширяем его.
На каждой итерации мы пытаемся найти сжатое k-мерное решение для экземпляра соответствующего текущему подграфу. Если такое решение найдено, то путем добавлением вершины или ребра мы получаем решение для следующего экземпляра, но это решение может быть больше нужного на 1. К этому решению мы применяем шаг сжатия. Мы останавливаемся, когда либо получаем решение размером не более k для всего графа, либо если некоторое промежуточное решение окажется несжимаемым.
Для того чтобы остановиться в случае, когда некоторый промежуточный экземпляр оказывается несжимаемым, задача должна обладать свойством, что оптимальный размер решения в любом промежуточном экземпляре не превышает оптимального размера решения во всем графе.

Предположим, что мы хотим решить $(\ast)$-сжатие, где подстановочный знак $(\ast)$ может быть заменен на имя проблемы, которую мы пытаемся решить.
В $(\ast)$-Compression в качестве входных данных дается экземпляр задачи $(\ast)$ с решением размера $k + 1$ и положительное целое число k. Задача состоит в том, чтобы либо найти решение размера не более k, либо сделать вывод, что решения нет.
Например, для $(\ast)$=Feedback Vertex Set дается граф G и множество вершин обратной связи множество вершин обратной связи X размера $k + 1$. Задача состоит в том, чтобы решить, есть ли в G вершинное множество с обратной связью вершин обратной связи размером не более k.

Th(1) Если существует алгоритм решения $(\ast)$-Compression за время $f(k) * n\^c$, тогда существует алгоритм решения $(\ast)$ за время $O(f(k) * n\^\{c+1\})$

Th(2) Если существует алгоритм решения Disjoint-$(\ast)$ а время $g(k) \cdot n\^O(1)$, тогда существует алгоритм решения $(\ast)$-Compression за время sum_\{i = 0\}\^\{k\}\{ C_\{k+1\}\^\{i\} * g(k-i) * n\^O(1)\} $\left[\sum_{i=0}^{k} C_{k+1}^{i} \cdot g(k-i) \cdot n^{O(1)}\right]^{(*)}$
В частности, если $g(k) = \alpha\^k$, то $(\ast)$-Compression решается за время $(1 + \alpha)\^k * n\^O(1)$

Feedback Vertex Set in Tournaments

Def Tournament - полный связный ориентированный граф

В этой задаче на вход подается турнир T и положительное целое число k, и задача состоит в том, чтобы решить, существует ли вершинное множество $X \subseteq V(T)$ размера не более k, такое, что $T - X$ является направленным ациклическим графом (эквивалентно, transitive tournament).

Def Решение X называется directed feedback vertex set.

Отметим, что Feedback Vertex Set в турнирах является частным случаем Directed Feedback Vertex Set, где входной направленный граф ограничивается турниром.

Th(3) Сначала мы рассмотрим вариант задачи компрессии, который называется Feedback Vertex Set in Tournaments Compression.
В этой задаче входные данные состоит из турнира T, направленного множества вершин обратной связи Z из T размера $k + 1$ и целого положительного числа k, а задача состоит либо в нахождении направленного вершинного множества Z из T размера k, либо сделать вывод о том, что такого множества не существует.
Lemma 4.1. (теорема 1 для турниров) Существование алгоритма для решения Feedback Vertex Set in Tournaments Compression за время $f(k) * n\^c$ влечет что Feedback Vertex Set in Tournaments решается за время $O(f(k) * n\^(c+1))$

Proof Предположим, что мы можем решить задачу сжатия вершинных множеств обратной связи в турнирах за время $f(k) * n\^O(1)$. Учитывая это, мы покажем, как можно решить исходную задачу проблему за время $f(k) * n\^O(1)$.
Возьмем произвольный порядок $(v1, v2, \ldots, vn)$ из $V(T)$ и для каждого i из $\{1, \ldots, n\}$ определяем $V\_i = \{v1, \ldots, vi\}$ и $T\_i = T[V\_i]$.
Заметим, что
1. V_k - множество вершин направленной обратной связи размера k из Tk.
2. Если X - множество вершин с направленной обратной связью из T_i то $X \cup \{v\_\{i+1\}\}$ - множество вершин с направленной обратной связью из $T\_\{i+1\}$.
3. Если T_i не допускает вершинного множества с направленной обратной связью размера не более k, то и T тоже.
Эти три факта вместе с алгоритмом для Feedback Vertex Set in Tournaments Compression, работающим за время $f(k) * n\^c$ дают алгоритм для Feedback Vertex Set in Tournaments, работающий за $f(k) * n\^ \{c+1\}$.

А теперь и сам алгоритм для Vertex Set in Tournaments Compression с требуемым временем:
В турнире T_k множество V_k является направленным вершинным множеством обратной связи размера k. Предположим, что для $i \geq k$ построено вершинное множество X_i с направленной обратной связью в турнире T_i размером не более k. Тогда в турнире $T\_\{i+1\}$ множество $Z\_\{i+1\} := X\_i \cup \{v\_\{i+1\}\}$ является вершинным множеством с направленной обратной связью размером не более $k + 1$.
1. Если на самом деле $|Z\_\{i+1\}| \leq k$, то можно перейти к следующей итерации с $X\_\{i+1\} = k$.
2. В противном случае имеем $|Zi+1| = k + 1$. Тогда за время $f(k) * n\^c$ мы можем либо построить вершинное множество $X\_\{i+1\}$ с направленной обратной связью в $T\_\{i+1\}$ размера k, либо сделать вывод о том, что (T, k) не является экземпляром множества вершин обратной связи в турнирах.

Def Теперь рассмотрим задачу, которая называется Disjoint Feedback Vertex Set in Tournaments.

Th(4) В этой задаче на вход подается турнир T вместе с множеством вершин W с направленной обратной связью, и задача состоит в том, чтобы найти множество вершин $X \subseteq V(T) \setminus W$ размера не более k, либо сделать вывод, что такого множества не существует.
Lemma 4.2. (теорема 2 для турниров) Если существует алгоритм решения Disjoint Feedback Vertex Set in Tournaments за время $g(k) * n\^O(1)$, тогда Feedback Vertex Set in Tournaments Compression решается за время sum_\{i = 0\}\^\{k\}\{ C_\{k+1\}\^\{i\} * g(k-i) * n\^O(1)\} (*). В частности, для $g(k) = \alpha\^k$ с какой то фиксированной константой α, алгоритм работает за $(\alpha + 1)\^k * n\^O(1)$

Proof Пусть Z - вершинное множество с направленной обратной связью размера $k+1$ в турнире T.
Чтобы решить, существует ли в T вершинное множество X с направленной обратной связью размера k (т.е. решить пример сжатия вершинного множества с обратной связью в турнирах (G, Z, k)), мы делаем следующее.
Предполагаем пересечение X с Z, т.е. то есть предполагаем множество $X\_Z := X \cap Z$, удаляем X_Z из T и уменьшаем параметр k на $|X\_Z|$. Для каждой гипотезы X_Z задаем $W := Z \setminus X\_Z$ и решаем Disjoint Feedback Vertex Set in Tournaments на экземпляре $(T - X\_Z, W, k - |X\_Z|)$.
1. Если для некоторого предположения X_Z найдено направленное вершинное множество обратной связи X' из T-X_Z размером не более k-$|X\_Z|$, которое расходится с W, то мы выводим $X := X' \cup X\_Z$.
2. В противном случае мы говорим что данный экземпляр неразрешим.
Но число предположений ограничено sum_\{i = 0\}\^\{k\}\{ C_\{k+1\}\^\{i\} = 2\^(k+1)-1, что доказывает лемму

Th(5) Следующим шагом мы покажем, что решение задачи Disjoint Feedback Vertex Set in Tournaments может быть выполнено за полиномиальное время. Как уже упоминалось, вместе с леммами 4.1 и 4.2 это показывает, что Feedback Vertex Set in Tournaments решается за время $2\^k * n\^O(1)$.
Lemma 4.3. Пусть T турнир, тогда
1. T имеет направленный цикл тогда и только тогда, когда T имеет направленный треугольник;
2. Если T ацикличен, тогда его можно топологически отсортировать (т.е. существует порядок вершин T, такой что для каждой дуги (u, v) $u < v$).

Alg(4) Теперь опишем алгоритм для дизъюнктивной версии задачи.
Пусть (T, W, k) - экземпляр задачи Disjoint Feedback Vertex Set in Tournaments, и пусть $A = V(T) \setminus W$. Также у нас есть условие, что $|W| = k + 1$. Очевидно, можно предположить, что и $T[W]$, и $T[A]$ индуцируют транзитивные турниры, поскольку в противном случае (T, W, k) не разрешим.
По лемме 4.3, для решения задачи Disjoint Feedback Vertex Set in Tournaments достаточно найти множество из не более чем k вершин из A, пересекающих все направленные треугольники в T.
Это дает нам следующую простую редукцию:
Reduction FVST.1. Если в T содержится направленный треугольник x, y, z с ровно одной вершиной из A, например z, то удалить z и уменьшить параметр на 1. Новым экземпляром будет $(T - \{z\}, W, k - 1)$.

Сокращение FVST.1 просто говорит, что все направленные треугольники в T, имеющие ровно одну вершину в A могут быть устранены подбором соответствующей вершины в A.
Корректность редукции FVST.1 следует из того, что мы не имеем права выбирать вершины из W.

При задании экземпляра (T, W, k) мы сначала применяем сокращение FVST.1. пока можем, поэтому с этого момента можем считать, что сокращение FVST.1 больше не применимо.
Поскольку турниры $T[W]$ и $T[A]$ ациклические, то из леммы 4.3 известно, что они оба имеют уникальные топологические порядки. Пусть топологические порядки $T[W]$ и $T[A]$ обозначим $\sigma = (w1, \ldots, wq)$ и ρ соответственно. Предположим, что X - желаемое решение, тогда $T[W \cup (A \setminus X)]$ - транзитивный турнир с единственным упорядочением, при котором, ограничивая этот порядок на W, мы получаем σ, а при ограничении на $A \setminus X$ - ρ.
Поскольку порядок σ сохраняется в порядке $T[W \cup (A\setminus X)]$, то наша модальная задача теперь состоит в том чтобы вставить в порядок $\sigma = (w1, \ldots, wq)$ подмножество максимального размера из A.

Поскольку сокращение FVST.1 больше не применимо, то для всех $v \in A$ имеем, что $T[W \cup \{v\}]$ ациклично, и позиция v в σ - это его позиция в единственном топологическом упорядочении $T[W \cup \{v\}]$.
То есть, существует целое число $p[v]$ такое, что для $i < p[v]$ существует дуга из w_i в v и для всех $i \geq p[v]$ существует дуга из v в w_i. Таким образом получаем, что

$(v, w\_i) \in E(T) <=> i \geq p[v]$ (4.1)

Заметим, что $p[v]$ определено для всех $v \in A$, оно уникально, и $p[v] \in \{1, \ldots, q+1\}$.
Теперь мы построим порядок π на A следующим образом: u находится перед v в π тогда и только тогда, когда $p[u] < p[v]$ или $p[u] = p[v]$ и u находится перед v в порядке ρ. То есть в порядок π мы итеративно включаем множества $\{v \in A : p[v] = i\}$ для $i = 1, 2, \ldots, q + 1$ и внутри каждого множества упорядочиваем вершины в соответствии с ρ (топологическим порядком на $T[A]$).

Основные наблюдения сейчас сводятся к следующему:
1. В транзитивном турнире $T - X$ топологическое упорядочение $T[A \setminus X]$ должно быть ограничением π, так как $T[W]$ остается в $T - X$, а σ - топологическое упорядочение $T[W]$.
2. С другой стороны, топологическое упорядочение $T[A \setminus X]$ должно быть ограничением ρ - топологического упорядочения $T[A]$.
Следовательно, достаточно найти самую длинную общую подпоследовательность π и ρ.

Th(6) Lemma 4.4. Пусть $B \subseteq A$. Тогда $T[W\cup B]$ ацикличен тогда и только тогда, когда вершины B из общей подпоследовательности ρ и π.

Proof Через $\rho|B$ и $\pi|B$ обозначим ограничения ρ и π на B, соответственно.
Для доказательства слева направо предположим, что $T[W\cup B]$ является ациклическим. Покажем, что $\rho|B = \pi|B$ и, следовательно, вершины B образуют общую подпоследовательность ρ и π.
Предположим, что существуют $x, y \in B$ такие, что x появляется раньше y в $\rho|B$, а y появляется раньше x в $\pi|B$. Тогда $(x, y) \in E(T)$, и $p[y] \leq p[x]$. Если бы $p[x] = p[y]$, то порядок следования x и y в π был бы определен.
и y в π определялся бы ρ, таким образом мы приходим к выводу, что $p[y] < p[x]$.
Из (4.1) следует, что $(y, w\_\{p[y]\}) \in E(T)$ и $(w\_\{p[y]\}, x) \in E(T)$. Из наличия направленных ребер (x, y), $(y, w\_\{p[y]\})$ и $(w\_\{p[y]\}, x)$, имеем, что $\{x, y, w\_\{p[y]\}\}$ индуцирует направленный треугольник в $T[W \cup B]$, что является противоречием.

Теперь докажем справа налево. Предположим, что вершины B образуют общую подпоследовательность ρ и π. В частности, это означает, что $\rho|B = \pi|B$. Чтобы показать, что $T[W \cup B]$ ациклична, по лемме 4.3 достаточно показать, что $T[W \cup B]$ не содержит направленных треугольников. Поскольку $T[W]$ и $T[A]$ ацикличны и не существует направленных треугольников с ровно двумя вершинами в W (сокращение FVST.1), то могут существовать только направленные треугольники с ровно двумя вершинами в B. Поскольку $\rho|B = \pi|B$, то для всех $x, y \in B$ с $(x, y) \in E(T)$, имеем, что $p[x] \leq p[y]$. Тогда, согласно (4.1), не существует $w\_i \in W$, для которого $(y, w\_i) \in E(T)$ и $(w\_i, x) \in E(T)$. Следовательно, не существует направленного треугольника в $T[W \cup B]$, а значит, он ациклический.

Lemma 4.5. Наибольшая общая подпоследовательность двух последовательностей из p и q элементов ищется за $O(pq)$ (Народный алгоритм)

Th(7) Lemma 4.6. Disjoint Feedback Vertex Set in Tournaments разрешимо за полиномиальное время.

Proof Пусть (T, W, k) - экземпляр задачи. Используем сокращение FVST.1 пока можем. Пусть R - множество вершин, удаленных с помощью сокращения FVST.1, а (T', W, k') - сокращенный экземпляр.
По лемме 4.4, оптимальный способ сделать T' ациклическим путем удаления вершин точно такой же, как и сделать последовательности ρ и π равными путем удаления вершин.
Таким образом, для нахождения оптимального набора удалений вершин мы можем найти самую длинную общую подпоследовательность последовательностей ρ и π, что можно сделать за полиномиальное время по лемме 4.5.
Пусть B - вершины длиннейшей общей подпоследовательности ρ и π, а $X := R \cup (V(T')) \setminus B)$.
Если $|X| > k$, то (T, W, k) - неразрешима. Иначе, X - искомое множество вершин направленной обратной связи размером не более k.

Th(8) Таким образом из лемм 4.1, 4.2, 4.6 (теоремы 3, 4, 7 в нашей нумерации) следует, что Feedback Vertex Set in Tournaments решается за время $2\^k * n\^O(1)$.

Feedback Vertex Set

Как мы знаем, X является вершинным множеством обратной связи неориентированного графа G, если $G - X$ является лесом.

Начнем с решения задачи Disjoint Feedback Vertex Set, как уже говорилось ранее, ее решение за соответствующее время даст решение Feedback Vertex Set с нуженой асимптотикой. В этой задаче в качестве входных данных дается неориентированный граф G, целое число k и множество вершин обратной связи W в G размера $k + 1$. Задача состоит в том, чтобы найти множество вершин обратной связи $X \subseteq V(G) \setminus W$ размером не более k, или доказать, что такого не существует.
Мы рассмотрим алгоритм Disjoint Feedback Vertex Set, работающий за время $4\^k * n\^O(1)$ (на самом деле существует более быстрый алгоритм со временем $\phi\^(2k) * n\^O(1)$, где $\phi = (1+\sqrt{5})/2 <$ 1,6181 - золотое сечение.

Alg(5) Пусть (G, W, k) - экземпляр вершинного множества с дизъюнктивной обратной связью, а $H = G - W$. Сначала приведем несколько правил редукции, упрощающих входной экземпляр.
Редукция FVS$\ast$.1. Удаляем из G все вершины со степенью не выше 1.
Редукция FVS$\ast$.2. Если в H существует вершина v такая, что $G[W \cup \{v\}]$ содержит цикл, то включаем v в решение, удаляем v и уменьшаем параметр, то есть новый экземпляр будет иметь вид $(G - \{v\}, W, k - 1)$.
Редукция FVS$\ast$.3. Если в G существует вершина $v \in V(H)$ степени 2 такая, что хотя бы один сосед v в G из $V(H)$, то удалим эту вершину и сделем ее соседей смежными (даже если до этого они были смежными; граф может стать мультиграфом).
Легко видеть, что редукции FVS$\ast$.1, FVS$\ast$.2 и FVS$\ast$.3 корректны и порождают эквивалентный экземпляр. Более того, все они могут быть применены за полиномиальное время.

Th(9) Lemma 4.8. Disjoint Feedback Vertex Set решается за время $4\^k * n\^O(1)$

Proof Мы будем придерживаться стратегии ветвления с нетривиальной функцией меры.
Пусть (G, W, k) - входной экземпляр. Если $G[W]$ не является лесом, то вернем, что (G, W, k) неразрешим. Таким образом, с этого момента мы предполагаем, что $G[W]$ действительно является лесом.
Алгоритм сначала применяет редукции FVS$\ast$.1, FVS$\ast$.2 и FVS$\ast$.3 пока может. Для ясности обозначим редуцированный экземпляр (тот, на котором редукции FVS$\ast$.1, FVS$\ast$.2 и FVS$\ast$.3 не применимы) как (G, W, k).
Если $k < 0$, то возвращаем, что (G, W, k) - неразрешим.
С этого момента мы предполагаем, что $k \geq 0$. Как мы знаем, H - лес, так как W - множество вершин с обратной связью. Таким образом, в H есть вершина x степени не выше 1. Более того, x имеет не менее двух соседей в W, иначе сокращение FVS$\ast$.1 или FVS$\ast$.3 было бы применено.
Поскольку сокращение FVS$\ast$.2 не может быть применено, то мы имеем что никакие две соседки x не принадлежат одной и той же связной компоненте $G[W]$.
Теперь выполним ветвление, включив x в решение в одной ветви и исключив его в другой ветви. То есть мы вызываем алгоритм на экземплярах $(G-\{x\}, W, k-1)$ и $(G, W \cup \{x\}, k)$. Если одна из этих ветвей возвращает решение, то возвращаем, что (G, W, k) - валидное состояние и разрешимо, иначе (G, W, k) - неразрешимо.
Корректность данного алгоритма следует из корректности наших сокращений и из того, что ветвление является исчерпывающим.

Для оценки времени работы алгоритма $I = (G, W, k)$,определим его меру $\mu(I) = k + \gamma(I)$, где $\gamma(I)$ - число связных компонент $G[W]$.
Заметим, что сокращения FVS$\ast$.1, FVS$\ast$.2 и FVS$\ast$.3 не увеличивают меру.
Как изменяется $\mu(I)$ при ветвлении?
Когда мы включаем в решение x, k уменьшается на 1, а $\gamma(I)$ остается прежним, и, следовательно, $\mu(I)$ уменьшается на 1.
В другой ветви k остается неизменным, а x имеет соседей не менее чем в двух связных компонентах W. А значит при включении x в W, $\gamma(I)$ уменьшается как минимум на 1.
Таким образом, мы имеем вектор ветвления $(1, 1)$, а время работы нашего алгоритма ветвления составляет $2\^\mu(I) * n\^O(1)$. Поскольку в начальный момент мы имеем $\mu(I) \leq k + |W| \leq 2k + 1$, мы получаем искомое время работы.

Th(10) Аналогично леммам 4.1 и 4.2 (теоремы 3, 4 в нашей нумерации), можно с помощью итерационного сжатия показать, что алгоритм, работающий за время $\alpha\^k * n\^O(1)$ для решения задачи Disjoint Feedback Vertex Set может быть использован для решения Feedback Vertex Set за время $(1 + \alpha)\^k * n\^O(1)$. Таким образом мы научились решать Feedback Vertex Set за $5\^k * n\^O(1)$.