Аннотация

Данная работа посвящена изучению деревьев диаметра 8 с минимальным количеством независимых множеств. В ходе работы был сужен класс всех деревьев диаметра 8 до деревьев с определенной структурой прикорневых поддеревьев и с некоторыми ограничениями на их размер. В результате работы для этого класса деревьев удалось сформулировать полиномиальный алгоритм поиска структуры дерева с минимальным количеством независимых множеств для заданного размера дерева.

Содержание

1 Введение

1.1 Определения

Диаметр дерева - это количество ребер в наибольшем простом пути между двумя вершинами дерева. Независимое множество (н.м.) - это множество вершин, каждая из которых не соединена с любой другой из этого множества. Пусть количество вершин в дереве равно n, диаметр дерева равен d. Такое дерево называется (n, d)-деревом. Тогда (n, d)-минимальным(максимальным) деревом называется дерево из n вершин диаметра d с минимальным(максимальным) количеством независимых множеств. Центральная вершина дерева диаметра d - это вершина, расстояние от которой до любой другой не превосходит $\lceil \frac{d}{2} \rceil$. Известно, что у деревьев чётного диаметра центральная вершина единственная. Прикорневая вершина - вершина соединенная с корнем. Прикорневое поддерево - максимальное по включению поддерево с корнем в прикорневой вершине, не включающее корень. Лес прикорневых вершин (поддеревьев) - лес, полученный из исходного дерева удалением корня дерева.

$i(v)$ - количество независимых множеств в поддереве вершины v.

$i_-(v)$ - количество независимых множеств в поддереве вершины v, не содержащих вершину v.

$i_+(v)$ - количество независимых множеств в поддереве вершины v, содержащих вершину v.

$i(F)$ - количество независимых множеств в дереве (лесу).

$i_-(F)$ - количество независимых множеств в лесу, не содержащих корневые вершины деревьев леса F.

$i_+(F)$ - количество независимых множеств в лесу F, содержащих все корневые вершины деревьев леса F.

$size(F)$ - количество вершин в дереве/лесу F.

$size_-(F)$ - количество вершин в дереве(лесу) F, не считая корень(корни).

Замечание. $\frac{i_-(*)}{i(*)} \leq 1$

Замечание. Количество н.м. (включая, не включая, произвольно) от пустого дерева/леса полагаем равным 1.

1.2 Предыдущие результаты

Известно, что при всех $n \geq 2$ граф-звезда S_n содержит максимально возможное количество н.м. в классе всех n-вершинных деревьев [1]. В работе [2] показано, что при любых значениях $n, d \geq 3$ единственным (n, d)-максимальным деревом является граф, полученный присоединением пути P_{d-1} к центральной вершине звезды S_{n-d+1}. Интересно, что при всех значениях параметров n и d n-вершинное дерево диаметра d, содержащее минимально возможное количество максимальных по включению н.м., также является единственным и имеет похожу структуру. Этот результат был доказан в работе [3]. С другой стороны, (n, d)-минимальные деревья устроены намного сложнее, и задача их описания для произвольного значения диаметра d до сих пор остается открытой. В работе [4] описаны (n, d)-минимальные деревья для $d \leq 4$ и любых n, а также $(n, 5)$-минимальные деревья при достаточно больших значениях n. В 2009г. Дайняк в работе [5] получил некоторые оценки на количество н.м. в (n, d)-минимальных деревьях при различных значениях $d \geq 6$. В 2021 году Талецкий в работе [6] получил точную структуру $(n, 6)$-минимального дерева при $n > 160$.

2 Деревья диаметра 8

2.1 Основная технология

В работе Талецкого [6] для минимизации количества н.м. в дереве диаметра 6 используется подход с заменой поддеревьев на другие, не меняющие количество вершин в дереве, но уменьшающих количество н.м. Опишем его.

Пусть дано дерево (n, d)-дерево T с корнем r. Разобьем лес прикорневых поддеревьев $T_1...T_k$ на два множества(леса): F_0 и F_1. Пусть есть лес F_2 такой, что $size(F_2) = size(F_1)$. *Корректной заменой* g в дереве T называется замена леса F_1 на F_2, не меняющая диаметр дерева. Как видно из определения, дерево T' полученное в результате применения к дереву T замены g по-прежнему (n, d) - дерево.

Посчитаем разницу в количестве н.м. до и после замены g

$$i(T) = i_-(T) + i_+(T) = i(F_1)i(F_0) + i_-(F_1)i_-(F_0)$$

Аналогично

$$i(T') = i(F_2)i(F_0) + i_-(F_2)i_-(F_0)$$

Посмотрим на разницу

$$i(T') - i(T) = i(F_0)(i(F_2) - i(F_1)) + i_-(F_0)(i_-(F_2) - i_-(F_1))$$

Назовём замену g *уменьшающей*, если

$$i(T') - i(T) < 0 \tag{1}$$

Видно, что если g - уменьшающая, то T - не (n,d)-минимальное.

Распишем 1 более подробно. Положим

$$\begin{aligned} F_{12} &:= i(F_1) - i(F_2), \\ F_{21\_excl} &:= i_-(F_2) - i_-(F_1) \end{aligned} \tag{2}$$

Тогда 1 переписывается следующим образом:

$$i(F_0)(-F_{12}) + i_-(F_0)(F_{21\_excl}) < 0$$

или

$$\frac{i_-(F_0)}{i(F_0)} F_{21\_excl} < F_{12} \tag{3}$$

Замечание. В дальнейшем мы рассматриваем только те замены в которых, как и в работе [6] Талецкого, $F_{12} > 0$, если не оговорено иного.

В предположении этого замечания мы можем сформулировать *достаточное условие* того, что замена g - уменьшающая:

1. $F_{21\_excl} \leq 0$

2. $F_{21\_excl} > 0$. Тогда 3 эквивалентно

$$\frac{i_-(F_0)}{i(F_0)} < \frac{F_{12}}{F_{21\_excl}}. \tag{4}$$

Замечание. В работе Талецкого [6] все используемые замены удовлетворяют условию *достаточности*.

Лемма 1. *Если условие достаточности выполнено для замены g дерева T, то условие достаточности выполнено для замены g дерева $T \cup F'$, где F' - произвольный лес, подвешенный к корню дерева T. Т.е. замена g - уменьшающая для $T \cup F'$.*

Доказательство. Заметим, что F_{21_excl} и F_{12} те же для $T \cup F'$. Из определения *достаточности*

1. Если $F_{21\_excl} \leq 0$ для замены g дерева T, то $F_{21\_excl} \leq 0$ для g дерева $T \cup F'$.

2. Если $F_{21\_excl} > 0$ и $\frac{i_-(F_0)}{i(F_0)} < \frac{F_{12}}{F_{21\_excl}}$ для замены g для терева T, то $F_{21\_excl} > 0$ для g дерева $T \cup F'$.

   Обозначим FF_0 - лес, не участвующий в замене для дерева $T \cup F'$. Тогда

$$\frac{i_-(FF_0)}{i(FF_0)} = \frac{i_-(F')}{i(F')}\frac{i_-(F_0)}{i(F_0)} \leq \frac{i_-(F_0)}{i(F_0)} \leq \frac{F_{12}}{F_{21\_excl}} \tag{5}$$

□

2.2 Результаты для прикорневых поддеревьев диаметра 6

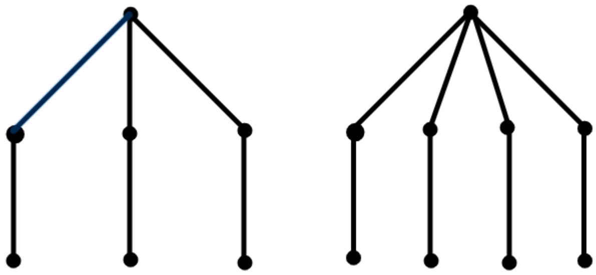

Рис. 1: $\mathbf{M_7}$ слева, $\mathbf{M_9}$ справа

Введём обозначения: M_7 - дерево из 7 вершин вида 3 пути P_2 присоединенных к корню. M_9 - дерево из 9 вершин вида 4 пути P_2 присоединенных к корню. См. рисунок 1.

Теорема 1. *Пусть дано $(n, 8)$-дерево T с корнем в r. Пусть из корня - центральной вершины дерева - ребра ведут в вершины $v_1, v_2, v_3, ..., v_h$, которые в свою очередь являются корнями соответствующих прикорневых поддеревьев $T_1, T_2, ..., T_h$. Пусть каждое из прикорневых поддеревьев T_i является деревом диаметра 6 с центральной вершиной в v_i. Пусть $\forall T_i size(T_i) > 160$. Пусть при этом дерево T содержит наименьшее количество н.м. при заданных условиях. Рассмотрим любое прикорневое поддерево T_i с корнем в v_i как отдельное дерево. Рассмотрим его лес прикорневых поддеревьев F_i. $size(F_i) = size_-(T_i) = 7k_i + mod_i$, где $k_i = \lfloor \frac{size(F_i)}{7} \rfloor$, $size(F_i) \equiv mod_i \pmod 7$.* ***Тогда верно следующее***:

$$F_i = \begin{cases} k_i M_7, & mod_i = 0 \\ (k_i - 5)M_7 \cup 4M_9, & mod_i = 1 \\ (k_i - 1)M_7 \cup M_9, & mod_i = 2 \\ (k_i - 6)M_7 \cup 5M_9, & mod_i = 3 \\ (k_i - 2)M_7 \cup 2M_9, & mod_i = 4 \\ (k_i - 7)M_7 \cup 6M_9, & mod_i = 5 \\ (k_i - 7)M_7 \cup 3M_9, & mod_i = 6 \end{cases} \tag{6}$$

Доказательство. По *Теореме 1* в работе Талецкого [6] заключение теоремы выполнено, если мы рассматриваем T_i как отдельное $(n, 6)$-дерево, и заключение совпадает с нашим. В работе Талецкого к такой структуре $(n, 6)$-минимального дерева приходят, используя **только** замены, для которых выполнено *достаточное условие*. По лемме 1 *достаточное условие* для замены будет по-прежнему выполнено, если к корню дерева присоединить произвольный лес. В нашем случае этот лес - это дерево $T \setminus T_i$. Поэтому все уменьшающие замены в работе Талецкого будут уменьшающими и для дерева $T = (T \setminus T_i) \cup T_i$ с корнем в v_i. Отсюда заключаем, что если T_i имеет какую-то отличную структуру от 6, то обязательно будет существовать последовательность уменьшающих замен $\{g_k\}$ строго внутри T_i приводящих T_i к структуре 6, для которых выполнено *достаточное условие*. *Противоречие.* □

Для простоты введем следующее обозначение:

Определение. (N, D, d) называется дерево диаметра D, такое что все его поддеревья диаметра d, и центральная вершина этих поддеревьев соединена с корнем дерева. (N, D, d)-*минимальным* называется (N, D, d) дерево, содержащее наименьшее кол-во н.м.

Замечание. В дальнейшем считаем, что размер $n + 1$ каждого прикорневого поддерева (N, D, d) дерева больше 161. Тогда $k = \lfloor \frac{n}{7} \rfloor \geq 23$. Обозначим $k_{min} = 23$.

2.3 Близость размеров прикорневых поддеревьев

В данном разделе мы покажем, что в $(N, 8, 6)$-минимальном дереве размеры поддеревьев достаточно близки.

Лемма 2. *Пусть прикорневое поддерево* T $(N, 8, 6)$*-минимального дерева с корнем в* r*. Пусть его размер* $n + 1$*,* $k = \lfloor \frac{n}{7} \rfloor$*,* $n \equiv m \pmod 7$*. Пусть* F *- лес прикорневых поддеревьев* T*. **Тогда*** $\frac{i(F)}{i_-(F)}$ *возрастает с ростом* k *(*$\frac{i_-(F)}{i(F)}$ *возрастает с убыванием* k*).*

Доказательство. Из теоремы 1 выразим $i_-(F), i_+(F), i(F)$:

$m = 0$:

$$i_-(F) = 27^k$$

$$i_+(F) = 8^k$$

$$i(F) = 35^k$$

$m = 1$:

$$i_-(F) = 27^k \cdot 3$$

$$i_+(F) = 8^k \cdot 2$$

$$i(F) = 35^{k-5} \cdot 97^4$$

$m = 2$:

$$i_-(F) = 27^k \cdot 3$$

$$i_+(F) = 8^k \cdot 2$$

$$i(F) = 35^{k-1} \cdot 97$$

$m = 3$:

$$i_-(F) = 27^k \cdot 9$$

$$i_+(F) = 8^k \cdot 4$$

$$i(F) = 35^{k-6} \cdot 97^5$$

$m = 4$:

$$i_-(F) = 27^k \cdot 9$$

$$i_+(F) = 8^k \cdot 4$$

$$i(F) = 35^{k-2} \cdot 97^2$$

$m = 5$:

$$i_-(F) = 27^k \cdot 27$$

$$i_+(F) = 8^k \cdot 8$$

$$i(F) = 35^{k-7} \cdot 97^6$$

$m = 6$:

$$i_-(F) = 27^k \cdot 27$$

$$i_+(F) = 8^k \cdot 8$$

$$i(F) = 35^{k-3} \cdot 97^3$$

Отсюда видно, что заключение леммы выполнено для любых m. $\square$

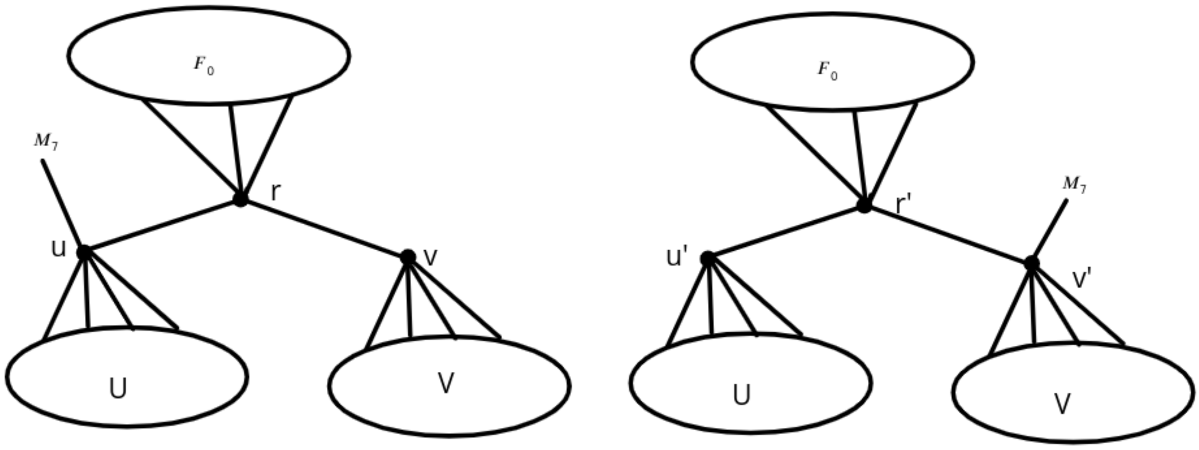

Рис. 2

Лемма 3. *Рассмотрим в $(N, 8, 6)$ дереве T два $(n, 6)$-поддерева с корнями в u и v. Лес $UV_{old} = u \cup v$ (u и v рассматриваются как поддеревья). Пусть лес прикорневых деревьев u равен $U \cup M_7$, лес прикорневых деревьев v равен V. Пусть $size(U) + 7 \geq size(V)$. Рассмотрим лес UV_{new} который состоит из двух деревьев с корнями в u' и v', чьи прикорневые леса равны U и $V \cup M_7$ соответственно.* ***Тогда*** *замена UV_{old} на UV_{new} уменьшающая тогда и только тогда, когда $\frac{i_-(V)}{i(V)} \cdot \frac{i(U)}{i_-(U)} > 1$ (см. рисунок 2).*

Доказательство. Посчитаем количество н.м. до замены:

$$i(T) = i_-(r) + i_+(r)$$

Посчитаем отдельно $i_-(r)$ и $i_+(r)$:

$$i_-(r) = i(F_0) \cdot i(u) \cdot i(v) = i(F_0) \cdot (i_+(u) + i_-(u)) \cdot (i_+(v) \cdot i_-(v)) =$$
$$= i(F_0) \cdot (i_-(M_7) \cdot i_-(U) + i(M_7) \cdot i(U)) \cdot (i_-(V) + i(V)) =$$
$$= i(F_0) \cdot (i_-(M_7) \cdot (i_-(U) \cdot i_-(V) + i_-(U) \cdot i(V)) + i(M_7) \cdot (i(U) \cdot i_-(V) + i(U) \cdot i(V)))$$

$$i_+(r) = i_-(F_0) \cdot i_-(u) \cdot i_-(v) = i_-(F_0) \cdot i(U) \cdot i(M_7) \cdot i(V)$$

Аналогично после замены:

$$i_-(r') = i(F_0) \cdot (i_-(M_7) \cdot (i_-(V) \cdot i_-(U) + i_-(V) \cdot i(U)) + i(M_7) \cdot (i(V) \cdot i_-(U) + i(V) \cdot i(U)))$$
$$i_+(r') = i_-(F_0) \cdot i_-(u) \cdot i_-(v) = i_-(F_0) \cdot i(U) \cdot i(M_7) \cdot i(V)$$

Тогда разница:

$$i(T') - i(T) =$$
$$i(F_0) \cdot (i_-(M_7) \cdot (i(U) \cdot i_-(V) - i(V) \cdot i_-(U)) + i(M_7) \cdot (i(V) \cdot i_-(U) - i(U) \cdot i_-(U))) =$$
$$= i(F_0)(i(V) \cdot i_-(U) - i(U) \cdot i_-(V)) \cdot (i(M_7) - i_-(M_7))$$

$i(F_0) > 0$ и $i(M_7) - i_-(M_7) > 0$, поэтому:

$$i(T') - i(T) < 0 \Leftrightarrow i(V) \cdot i_-(U) - i(U) \cdot i_-(V) < 0 \Leftrightarrow$$
$$\Leftrightarrow \frac{i(V)}{i_-(V)} < \frac{i(U)}{i_-(U)} \Leftrightarrow \frac{i_-(V)}{i(V)} \cdot \frac{i(U)}{i_-(U)} > 1$$

□

Замечание. В лемме можно заменить M_7 на любое непустое дерево P, т.к. нам лишь необходимо $i(P) - i_-(P) > 0$.

В дальнейшем мы будем пользоваться вычислениями на компьютере. Все программы написаны на языке Python 3.0 и не используют сторонних библиотек.[1]

Замечание. Как можно заметить все выражения ($i_-,\ i_+,\ i$) для количества н.м. для прикорневого дерева (леса, вершины) имеют вид:

$$\sum_{j=1}^{r} c_j \cdot a_j^k, \quad a_j \in \mathbb{Q}, c_j \in \mathbb{N} \tag{7}$$

Сумма и произведение выражений вида 7 - это выражение вида 7. Выражение такого вида можно представить как два вектора(списка, массива): $factors$ ($\{a_j\}$) и $power\_bases$ ($\{c_j\}$). [2]

Замечание. Если имеется лес F, то можно добавить вершину, назначить ее корнем и соединить со всеми деревьями леса. Такую вершину будем обозначать r_F[3]. Очевидно, что $i_-(v_F) = i(F)$, $i_+(v_F) = i_-(F)$. Вершина r_F может совпадать с уже имеющейся вершиной v дерева T, если мы рассматривали лес F как подмножество дерева T. Когда мы считаем количество н.м. для r_F, мы считаем количество независимых множеств именно для дерева с корнем r_F и поддеревьями из леса F, а не количество н.м. для вершины v в дереве T.

Теорема 2. *Пусть T_{main} - $(N, 8, 6)$-минимальное дерево.* ***Тогда***

$$\exists K \geq k_{min}\ \forall T \text{ - прикорневое дерево } T_{main} \quad \lfloor \frac{size_-(T)}{7} \rfloor \in [K, K+2]$$

Доказательство. В дереве T есть хотя бы две прикорневые вершины u и v. Из всех возможных пар вершин u и v рассмотрим с максимальной разницей $size_-(u) - size_-(v)$. Если $\lfloor \frac{size_-(u)}{7} \rfloor - \lfloor \frac{size_-(v)}{7} \rfloor \geq 2$, то теорема выполнена.

Покажем, что иначе быть не может. Зафиксируем $q \geq k_{min}$. Мы будем перебирать всевозможные пары остатков по модулю 7 (m_u, m_v). Будем рассматривать $delta > 0$. Имеем дерево

[1] Весь проект смотрите на github

[2] В коде программы класс для выражений вида 7 называется *IndSetCntExpr* (см. приложение A).

[3] Смотрите класс $PreRootVertex$ в приложении B, соответствующий прикорневым вершинам, т.е. центральным вершинам каких-то $(n, 6)$-минимальных деревьев.

T из леммы 3. Пусть $size\_(u) = 7q + m_u$, $size\_(v) = 7(q - delta) + m_v$ (предполагаем, что $q - delta \geq k_{min}$, но это не очень важно, т.к. критерий 3 не будет зависеть от q, а только лишь от разности размеров поддеревьев). Если замена, описанная в лемме 3 - уменьшающая, то по лемме 2, подобная замена для $size\_(u) = 7q + m_u$ и $size\_(v) = 7(q - delta - 1) + m_v$ тоже уменьшающая. Поэтому достаточно найти минимальное $delta$ для каждой пары остатков (m_u, m_v). Выполним перебор с помощью компьютера (см. приложение D)[4]. Результаты[5]:

```
mods: 0 -> 0; delta: 1; criterion: 1
mods: 0 -> 1; delta: 1; criterion: 157565625/88529281
mods: 0 -> 2; delta: 1; criterion: 105/97
mods: 0 -> 3; delta: 1; criterion: 16544390625/8587340257
mods: 0 -> 4; delta: 1; criterion: 11025/9409
mods: 0 -> 5; delta: 1; criterion: 1737161015625/832972004929
mods: 0 -> 6; delta: 1; criterion: 1157625/912673
mods: 1 -> 0; delta: 4; criterion: 88529281/72335025
mods: 1 -> 1; delta: 1; criterion: 1
mods: 1 -> 2; delta: 3; criterion: 912673/893025
mods: 1 -> 3; delta: 1; criterion: 105/97
mods: 1 -> 4; delta: 3; criterion: 9409/8505
mods: 1 -> 5; delta: 1; criterion: 11025/9409
mods: 1 -> 6; delta: 3; criterion: 97/81
mods: 2 -> 0; delta: 2; criterion: 97/81
mods: 2 -> 1; delta: 1; criterion: 1500625/912673
mods: 2 -> 2; delta: 1; criterion: 1
mods: 2 -> 3; delta: 1; criterion: 157565625/88529281
mods: 2 -> 4; delta: 1; criterion: 105/97
mods: 2 -> 5; delta: 1; criterion: 16544390625/8587340257
mods: 2 -> 6; delta: 1; criterion: 11025/9409
mods: 3 -> 0; delta: 4; criterion: 8587340257/7595177625
mods: 3 -> 1; delta: 2; criterion: 97/81
mods: 3 -> 2; delta: 4; criterion: 88529281/72335025
mods: 3 -> 3; delta: 1; criterion: 1
mods: 3 -> 4; delta: 3; criterion: 912673/893025
mods: 3 -> 5; delta: 1; criterion: 105/97
mods: 3 -> 6; delta: 3; criterion: 9409/8505
mods: 4 -> 0; delta: 2; criterion: 9409/8505
mods: 4 -> 1; delta: 1; criterion: 42875/28227
```

[4]Технические функции, используемые при замене леса, смотрите в C

[5]criterion - это значение критерия из леммы 3

```
mods: 4 -> 2; delta: 2; criterion: 97/81
mods: 4 -> 3; delta: 1; criterion: 1500625/912673
mods: 4 -> 4; delta: 1; criterion: 1
mods: 4 -> 5; delta: 1; criterion: 157565625/88529281
mods: 4 -> 6; delta: 1; criterion: 105/97
mods: 5 -> 0; delta: 4; criterion: 832972004929/797493650625
mods: 5 -> 1; delta: 2; criterion: 9409/8505
mods: 5 -> 2; delta: 4; criterion: 8587340257/7595177625
mods: 5 -> 3; delta: 2; criterion: 97/81
mods: 5 -> 4; delta: 4; criterion: 88529281/72335025
mods: 5 -> 5; delta: 1; criterion: 1
mods: 5 -> 6; delta: 3; criterion: 912673/893025
mods: 6 -> 0; delta: 2; criterion: 912673/893025
mods: 6 -> 1; delta: 1; criterion: 1225/873
mods: 6 -> 2; delta: 2; criterion: 9409/8505
mods: 6 -> 3; delta: 1; criterion: 42875/28227
mods: 6 -> 4; delta: 2; criterion: 97/81
mods: 6 -> 5; delta: 1; criterion: 1500625/912673
mods: 6 -> 6; delta: 1; criterion: 1
```

Максимальное значение $delta = 4$. Заметим, что после замены из леммы 3,

$$size_-(u') = 7(q-1) + m_u, \quad size_-(v') = 7(q - delta + 1) + m_v$$

т.е. $\lfloor \frac{size_-(u')}{7} \rfloor - \lfloor \frac{size_-(v')}{7} \rfloor = delta - 2 \leq 2$. □

Определение. Назовем k - *базой* $(N, 8, 6)$-дерева, если

$$k = \min \lfloor \frac{size_-(T)}{7} \rfloor \mid T \text{ - прикорневое поддерево}$$

Замечание. В качестве K из теоремы 2 можно взять *базу* $(N, 8, 6)$-минимального дерева.

2.4 Полиномиальный алгоритм

Рассмотрим *достаточное условие* уменьшающей замены g. Условие 4

$$\frac{i_-(F_0)}{i(F_0)} < \frac{F_{12}}{F_{21\_excl}}$$

Т.к. $\frac{i_-(F_0)}{i(F_0)} \leq 1$, то условие можно усилить до

$$1 < \frac{F_{12}}{F_{21\_excl}} \tag{8}$$

Определение. Полученное условие назовем *сильным достаточным*[6].

Из *сильной достаточности* следует *достаточность*, поэтому лемма 1 так же выполнена. Более того, в условии *сильной достаточности* вообще неважно, какой лес F_0.

Замечание. В дальнейшем мы будем пользоваться только *сильно достаточными заменами*.

Важное техническое замечание:

Замечание. Можно с помощью компьютера за небольшое время определить выполняется ли *сильное достаточное условие*, начиная с какого-то $k \geq M$ (k - это *база*). Т.к. F_{12}, F_{21_excl}, их сумма, разность и произведение - это выражения вида 7, то начиная с какого-то k, в 7 элемент с наибольшим основанием степени a_j будет больше чем сумма модулей других, потому что мы имеем экспоненциальный рост. В программе мы будем перебирать это k, начиная с единицы. Смотрите функцию *sign_on_limit* в A.

По предыдущей теореме 2 мы знаем, что в $(N, 8, 6)$-минимальном дереве размеры поддеревьев достаточно близки. Зафиксируем k - *базу*. Это позволяет нам организовать перебор для поиска структуры $(N, 8, 6)$-минимального дерева, при фиксированном числе прикорневых вершин $q \geq 2$. Для начала посмотрим, что получается при $q = 2$.
Будем перебирать два упорядоченных числа (a, b), $a < 7$[7], $b < 21$. Для такой пары будем перебирать упорядоченную пару (c, d) такую, что $c < 21$, $d < 21$, $c + d = a + b$. Понимаем это следующим образом: есть два прикорневых поддерева A, B,

$$size\_(A) \equiv a \pmod 7$$
$$\lfloor \frac{size\_(A)}{7} \rfloor = k$$
$$size\_(B) \equiv b \pmod 7$$
$$\lfloor \frac{size\_(B)}{7} \rfloor = k + \lfloor \frac{b}{7} \rfloor$$

Аналогично для C, D

$$size\_(C) \equiv c \pmod 7$$
$$\lfloor \frac{size\_(C)}{7} \rfloor = k + \lfloor \frac{c}{7} \rfloor$$
$$size\_(D) \equiv d \pmod 7$$
$$\lfloor \frac{size\_(D)}{7} \rfloor = k + \lfloor \frac{d}{7} \rfloor$$

[6] Смотрите функцию *is_swap_edge* в приложении C

[7] Меньше 7, т.к. иначе мы неверно зафиксировали K и можем его увеличить

Мы будем пробовать заменять лес прикорневых деревьев, состоящий из A, B, на лес, состоящий из C, D, используя *условие сильной достаточности*. Среди всех таких замен мы будем брать замену, для которой выполняется *условие сильной достаточности* с наименьшим k. Если *условие сильной достаточности* выполнено, то проводим ребро из вершины (A, B) в (C, D). *Оптимальными* будем называть те вершины(пары), из которых нет ребер. Выполним перебор с помощью компьютера (см. приложение E):

```
max k value is: 23
allowed pairs count of pre-root subtrees: 17
(0, 0)
(0, 1)
(0, 2)
(0, 7)
(0, 9)
(0, 11)
(0, 13)
(0, 15)
(1, 2)
(1, 4)
(2, 2)
(2, 4)
(2, 15)
(2, 17)
(4, 4)
(4, 6)
(6, 6)
```

Определение. Если из корежа x есть ребро в кортеж y, то будем писать $x \to y$.

Замечание. Ребра обладают свойством транзитивности.

Мы можем применить подобную технологию для количества прикорневых поддеревьев 3 и более. Только мы можем пользоваться результатами с предыдущих итераций. А именно, зафиксируем q - количество прикорневых поддеревьев. Если кортеж $x = (x_1, x_2, ..., x_q)$-*оптимален*, то так же *оптимален* его любой "подкортеж" $x' = (x_{i_1}, x_{i_2}, ..., x_{i_r})$. Иначе существует какая-то уменьшающая замена для x', удовлетворяющая условию *сильной достаточности*, значит не зависящая от F_0, и тогда мы можем применить эту замену к x, значит x - не *оптимальный* кортеж длины q.

Таким образом, мы можем перебирать кортежи x длины q такие, что любой их "подкортеж" длины $q - 1$ *оптимален*. Более того, перебираем кортежи y длины q по такому же принципу, в которые пытаемся провести ребро из x.

Ниже очень важное замечание.

Замечание. Мы не рассматриваем замены, которые уменьшают *базу*, т.к. база имеет ограничение снизу. К примеру, ребро $(0,0,1) \to (6,8,9)$ никогда не будет рассмотрено. [8]

Определение. (N,D,d,q) - это (N,D,d) дерево с q прикорневыми поддеревьями.

Приведем здесь результаты программы для всех q от 3 до 8 (код программы смотрите в F).

$q = 3$

```
(0, 0, 0),
(0, 0, 1),
(0, 0, 2),
(0, 0, 7),
(0, 0, 9),
(0, 0, 11),
(0, 0, 13),
(0, 0, 15),
(0, 1, 2),
(0, 2, 2),
(0, 2, 15),
(0, 7, 7),
(0, 7, 9),
(0, 9, 9),
(0, 9, 11),
(0, 11, 11),
(0, 11, 13),
(0, 13, 13),
(1, 2, 2),
(2, 2, 2),
(2, 2, 4),
(2, 2, 15),
(2, 4, 4),
(4, 4, 4)
```

$q = 4$

```
(0, 0, 0, 0),
(0, 0, 0, 1),
```

[8] $(0,0,1) =_{BASE-1} (7,7,8) \to (6,8,9)$

(0, 0, 0, 2),
(0, 0, 0, 7),
(0, 0, 0, 9),
(0, 0, 0, 11),
(0, 0, 0, 13),
(0, 0, 0, 15),
(0, 0, 1, 2),
(0, 0, 2, 2),
(0, 0, 2, 15),
(0, 0, 7, 7),
(0, 0, 7, 9),
(0, 0, 9, 9),
(0, 0, 9, 11),
(0, 0, 11, 11),
(0, 0, 11, 13),
(0, 0, 13, 13),
(0, 1, 2, 2),
(0, 2, 2, 2),
(0, 2, 2, 15),
(0, 7, 7, 7),
(0, 7, 7, 9),
(0, 7, 9, 9),
(0, 9, 9, 9),
(0, 9, 9, 11),
(0, 9, 11, 11),
(0, 11, 11, 11),
(2, 2, 2, 2),
(2, 2, 2, 4),
(2, 2, 4, 4)

$q = 5$

(0, 0, 0, 0, 0),
(0, 0, 0, 0, 1),
(0, 0, 0, 0, 2),
(0, 0, 0, 0, 7),
(0, 0, 0, 0, 9),
(0, 0, 0, 0, 11),
(0, 0, 0, 0, 13),
(0, 0, 0, 0, 15),
(0, 0, 0, 1, 2),
(0, 0, 0, 2, 2),

(0, 0, 0, 2, 15),
(0, 0, 0, 7, 7),
(0, 0, 0, 7, 9),
(0, 0, 0, 9, 9),
(0, 0, 0, 9, 11),
(0, 0, 0, 11, 11),
(0, 0, 0, 11, 13),
(0, 0, 0, 13, 13),
(0, 0, 1, 2, 2),
(0, 0, 2, 2, 2),
(0, 0, 2, 2, 15),
(0, 0, 7, 7, 7),
(0, 0, 7, 7, 9),
(0, 0, 7, 9, 9),
(0, 0, 9, 9, 9),
(0, 0, 9, 9, 11),
(0, 0, 9, 11, 11),
(0, 0, 11, 11, 11),
(0, 2, 2, 2, 2),
(0, 7, 7, 7, 7),
(0, 7, 7, 7, 9),
(0, 7, 7, 9, 9),
(0, 7, 9, 9, 9),
(0, 9, 9, 9, 9),
(0, 9, 9, 9, 11),
(0, 9, 9, 11, 11),
(2, 2, 2, 2, 2),
(2, 2, 2, 2, 4)

$q = 6$

(0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 1),
(0, 0, 0, 0, 0, 2),
(0, 0, 0, 0, 0, 7),
(0, 0, 0, 0, 0, 9),
(0, 0, 0, 0, 0, 11),
(0, 0, 0, 0, 0, 13),
(0, 0, 0, 0, 0, 15),
(0, 0, 0, 0, 1, 2),
(0, 0, 0, 0, 2, 2),
(0, 0, 0, 0, 2, 15),

(0, 0, 0, 0, 7, 7),
(0, 0, 0, 0, 7, 9),
(0, 0, 0, 0, 9, 9),
(0, 0, 0, 0, 9, 11),
(0, 0, 0, 0, 11, 11),
(0, 0, 0, 0, 11, 13),
(0, 0, 0, 0, 13, 13),
(0, 0, 0, 1, 2, 2),
(0, 0, 0, 2, 2, 2),
(0, 0, 0, 2, 2, 15),
(0, 0, 0, 7, 7, 7),
(0, 0, 0, 7, 7, 9),
(0, 0, 0, 7, 9, 9),
(0, 0, 0, 9, 9, 9),
(0, 0, 0, 9, 9, 11),
(0, 0, 0, 9, 11, 11),
(0, 0, 0, 11, 11, 11),
(0, 0, 2, 2, 2, 2),
(0, 0, 7, 7, 7, 7),
(0, 0, 7, 7, 7, 9),
(0, 0, 7, 7, 9, 9),
(0, 0, 7, 9, 9, 9),
(0, 0, 9, 9, 9, 9),
(0, 0, 9, 9, 9, 11),
(0, 0, 9, 9, 11, 11),
(0, 2, 2, 2, 2, 2),
(0, 7, 7, 7, 7, 7),
(0, 7, 7, 7, 7, 9),
(0, 7, 7, 7, 9, 9),
(0, 7, 7, 9, 9, 9),
(0, 7, 9, 9, 9, 9),
(0, 9, 9, 9, 9, 9),
(0, 9, 9, 9, 9, 11),
(2, 2, 2, 2, 2, 2)

$q = 7$

(0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 1),
(0, 0, 0, 0, 0, 0, 2),
(0, 0, 0, 0, 0, 0, 7),
(0, 0, 0, 0, 0, 0, 9),

(0, 0, 0, 0, 0, 0, 11),
(0, 0, 0, 0, 0, 0, 13),
(0, 0, 0, 0, 0, 0, 15),
(0, 0, 0, 0, 0, 1, 2),
(0, 0, 0, 0, 0, 2, 2),
(0, 0, 0, 0, 0, 2, 15),
(0, 0, 0, 0, 0, 7, 7),
(0, 0, 0, 0, 0, 7, 9),
(0, 0, 0, 0, 0, 9, 9),
(0, 0, 0, 0, 0, 9, 11),
(0, 0, 0, 0, 0, 11, 11),
(0, 0, 0, 0, 0, 11, 13),
(0, 0, 0, 0, 0, 13, 13),
(0, 0, 0, 0, 1, 2, 2),
(0, 0, 0, 0, 2, 2, 2),
(0, 0, 0, 0, 2, 2, 15),
(0, 0, 0, 0, 7, 7, 7),
(0, 0, 0, 0, 7, 7, 9),
(0, 0, 0, 0, 7, 9, 9),
(0, 0, 0, 0, 9, 9, 9),
(0, 0, 0, 0, 9, 9, 11),
(0, 0, 0, 0, 9, 11, 11),
(0, 0, 0, 0, 11, 11, 11),
(0, 0, 0, 2, 2, 2, 2),
(0, 0, 0, 7, 7, 7, 7),
(0, 0, 0, 7, 7, 7, 9),
(0, 0, 0, 7, 7, 9, 9),
(0, 0, 0, 7, 9, 9, 9),
(0, 0, 0, 9, 9, 9, 9),
(0, 0, 0, 9, 9, 9, 11),
(0, 0, 0, 9, 9, 11, 11),
(0, 0, 2, 2, 2, 2, 2),
(0, 0, 7, 7, 7, 7, 7),
(0, 0, 7, 7, 7, 7, 9),
(0, 0, 7, 7, 7, 9, 9),
(0, 0, 7, 7, 9, 9, 9),
(0, 0, 7, 9, 9, 9, 9),
(0, 0, 9, 9, 9, 9, 9),
(0, 0, 9, 9, 9, 9, 11),
(0, 2, 2, 2, 2, 2, 2),

(0, 7, 7, 7, 7, 7, 7),
(0, 7, 7, 7, 7, 7, 9),
(0, 7, 7, 7, 7, 9, 9),
(0, 7, 7, 7, 9, 9, 9),
(0, 7, 7, 9, 9, 9, 9),
(0, 7, 9, 9, 9, 9, 9),
(0, 9, 9, 9, 9, 9, 9)

$q = 8$

(0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 1),
(0, 0, 0, 0, 0, 0, 0, 2),
(0, 0, 0, 0, 0, 0, 0, 7),
(0, 0, 0, 0, 0, 0, 0, 9),
(0, 0, 0, 0, 0, 0, 0, 11),
(0, 0, 0, 0, 0, 0, 0, 13),
(0, 0, 0, 0, 0, 0, 0, 15),
(0, 0, 0, 0, 0, 0, 1, 2),
(0, 0, 0, 0, 0, 0, 2, 2),
(0, 0, 0, 0, 0, 0, 2, 15),
(0, 0, 0, 0, 0, 0, 7, 7),
(0, 0, 0, 0, 0, 0, 7, 9),
(0, 0, 0, 0, 0, 0, 9, 9),
(0, 0, 0, 0, 0, 0, 9, 11),
(0, 0, 0, 0, 0, 0, 11, 11),
(0, 0, 0, 0, 0, 0, 11, 13),
(0, 0, 0, 0, 0, 0, 13, 13),
(0, 0, 0, 0, 0, 1, 2, 2),
(0, 0, 0, 0, 0, 2, 2, 2),
(0, 0, 0, 0, 0, 2, 2, 15),
(0, 0, 0, 0, 0, 7, 7, 7),
(0, 0, 0, 0, 0, 7, 7, 9),
(0, 0, 0, 0, 0, 7, 9, 9),
(0, 0, 0, 0, 0, 9, 9, 9),
(0, 0, 0, 0, 0, 9, 9, 11),
(0, 0, 0, 0, 0, 9, 11, 11),
(0, 0, 0, 0, 0, 11, 11, 11),
(0, 0, 0, 0, 2, 2, 2, 2),
(0, 0, 0, 0, 7, 7, 7, 7),
(0, 0, 0, 0, 7, 7, 7, 9),
(0, 0, 0, 0, 7, 7, 9, 9),

(0, 0, 0, 0, 7, 9, 9, 9),
(0, 0, 0, 0, 9, 9, 9, 9),
(0, 0, 0, 0, 9, 9, 9, 11),
(0, 0, 0, 0, 9, 9, 11, 11),
(0, 0, 0, 2, 2, 2, 2, 2),
(0, 0, 0, 7, 7, 7, 7, 7),
(0, 0, 0, 7, 7, 7, 7, 9),
(0, 0, 0, 7, 7, 7, 9, 9),
(0, 0, 0, 7, 7, 9, 9, 9),
(0, 0, 0, 7, 9, 9, 9, 9),
(0, 0, 0, 9, 9, 9, 9, 9),
(0, 0, 0, 9, 9, 9, 9, 11),
(0, 0, 2, 2, 2, 2, 2, 2),
(0, 0, 7, 7, 7, 7, 7, 7),
(0, 0, 7, 7, 7, 7, 7, 9),
(0, 0, 7, 7, 7, 7, 9, 9),
(0, 0, 7, 7, 7, 9, 9, 9),
(0, 0, 7, 7, 9, 9, 9, 9),
(0, 0, 7, 9, 9, 9, 9, 9),
(0, 0, 9, 9, 9, 9, 9, 9),
(0, 7, 7, 7, 7, 7, 7, 7),
(0, 7, 7, 7, 7, 7, 7, 9),
(0, 7, 7, 7, 7, 7, 9, 9),
(0, 7, 7, 7, 7, 9, 9, 9),
(0, 7, 7, 7, 9, 9, 9, 9),
(0, 7, 7, 9, 9, 9, 9, 9),
(0, 7, 9, 9, 9, 9, 9, 9)

Замечание. Все ребра(замены) из не оптимальных кортежей удовлетворяют условию сильной достаточности при $k \geq k_{min} = 23$.

Дальнейшая задача привести набор *оптимальных* кортежей для произвольного q.

Определение. S_q - лексикографически упорядоченный набор упорядоченных *оптимальных* кортежей длины q.

Определение. $S_q(K)$ - $(N, 8, 6, q)$-деревья с *базой* K, со структурой поддерева $x \in S_q$, т.е.

$$\forall T_i \text{ - прикорневое поддерево} \quad size_{-}(T_i) = 7K + x_i$$

В $S_q(K)$ точно содержатся все $(N, 8, 6, q)$-минимальные деревья с базой K. Т.к. из них в принципе нет уменьшающего ребра.

Замечание.

$$\forall x \in S_q \quad x_1 < 7$$

Иначе можем вычесть из каждого x_i 7 и получить такой же кортеж с точки зрения структуры дерева (можно считать, что увеличили *базу* на 1).

Замечание. По заданной базе K и кортежу $x \in S_q$ можно однозначно восстановить $N = N(K, x)$ - размер $(N, 8, 6, q)$-дерева, соответствующему x.

Определение.

$$\forall K \forall x \in S_q(K) \quad x \text{ является } (N(K,x), 8, 6, q)\text{-минимальным}$$

Это равносильно

$$\forall N \geq (7k_{min} \cdot q + q + 1)\ \exists! K \geq k_{min}\ \exists! t \in S_q(K), \text{ т.ч. } t - (N(K,x), 8, 6, q)\text{-дерево} \tag{9}$$

Если S_q удовлетворяет условию 9, то назовем S_q - *минимальным набором*.

Определение. $q \geq 8$

$$\begin{aligned} A_q &= \{(0, x) \mid x \in S_{q-1}\} \\ B_q &= \{(0, 7, 7, ..., 7), (0, 7, 7, ..., 7, 9), ..., (0, 7, 7, ..., 7, 9, 9, 9, 9, 9, 9)\} \end{aligned} \tag{10}$$

Замечание. B_q - последние семь элементов в S_q.

Теорема 3. *1.* $\forall q > 7 \quad S(q) = A(q) \sqcup B(q)$

Доказательство. 1 выполнено для $q = 8$. Докажем по индукции для $q \geq 9$.

1. $S_q \subseteq A_q \sqcup B_q$. Предположим обратное, т.е.

$$\exists x \in S_q : \quad x \notin A(q) \sqcup B(q)$$

Тогда

(a) $x_1 \neq 0$. Тогда рассмотрим $x' = x_{1...(q-1)}$.

$$x_1' \neq 0 \Rightarrow x' \notin S_{q-1} \Rightarrow \exists y \in S_{q-1} : \ x_{1...(q-1)} = x' \to y \Rightarrow x \notin S_q$$

Противоречие.

(b) $x_1 = 0$. Тогда

i. $x_2 = 0$. Тогда

$$x' = x_{2\ldots q} \in S_{q-1} \Rightarrow x = (0, x') \in A_q.$$

Противоречие.

ii. $x_2 \neq 0 \Rightarrow \forall i \in [2, q]\ x_i \neq 0$. Тогда

$$\forall i \in [2, q]\ x^{(i)} = (x_{1\ldots(i-1)}, x_{(i+1)\ldots q}) \in S_{q-1} \Rightarrow x^{(i)} \in B_{q-1} \Rightarrow x \in B_q$$

Противоречие.

2. $A_q \sqcup B_q \subseteq S_q$. Предположим обратное.

(a) $x \in A_q$ и $x \notin S_q$. Тогда

$$\exists y\ :\ x \to y$$

i. $y \in S_q$. Тогда

$$y_1 = 0 \text{ но } x_1 = 0 \text{ тоже} \Rightarrow x_{2\ldots q} \to y_{2\ldots q} \Rightarrow x_{2\ldots q} \notin S_{q-1} \Rightarrow x \notin A_q$$

Противоречие.

ii. $y \notin S_q$. Тогда если $\exists z\ :\ y \to z$, то повторим рассуждения для $x \to z$. Иначе

$$\forall i \leq q\ y_i \geq 7 \Rightarrow sum(x) = sum(y) \geq 7q$$

Такого быть не может, т.к.

$$\max_{x \in A_q} sum(x) = \max_{x \in S_{q-1}} sum(x) = sum(0, 7, \ldots 7, 9, 9, 9, 9, 9, 9) = 7q - 2 < 7q$$

Следовательно ребра $x \to y$ быть не может. Противоречие.

(b) $x \in B_q$ и $x \notin S_q$. Тогда $\exists y\ :\ x \to y$.

i. $y \in S_q$. Тогда

$$y_1 = 0 \Rightarrow x_{2\ldots q} \to y_{2\ldots q}$$

Покажем, что $y \in B_q$. Для этого достаточно показать, что

$$\{sum(x) \mid x \in A_q\} \cap \{sum(x) \mid x \in B_q\} = \varnothing \tag{11}$$

Это выполнено, для $q \leq 8$. Для $q \geq 9$ надо заметить следующее[9]:

$$\min_{x \in B_q} sum(x) > \max_{x \in B_{q-2}} sum(x) = \max_{x \in S_{q-2}} sum(x) = \max_{x \in A_{q-1}} sum(x)$$

[9]Вывод для $q = 9$ см. в G

Отсюда

$$\{sum(x) \mid x \in B_q\} \cap \{sum(x) \mid x \in A_{q-1}\} = \varnothing \tag{12}$$

Так же выполнено, что все суммы элементов B_q дают различные остатки по модулю 7. Отсюда

$$\begin{aligned}
&|\{sum(x) \mid x \in B_q\} \cap \{sum(x) \mid x \in B_{q-1}\}| = \\
&|\{7 + sum(x) \mid x \in B_{q-1}\} \cap \{sum(x) \mid x \in B_{q-1}\}| = \\
&|\{7 + sum(x) = sum(y) \mid x, y \in B_{q-1}\}| \leq \\
&|\{sum(x) = sum(y) \pmod 7 \quad \mid x, y \in B_{q-1}, x \neq y\}| = 0
\end{aligned} \tag{13}$$

Из 12 и 13 следует

$$\{sum(x) \mid x \in B_q\} \cap \{sum(x) \mid x \in A_{q-1} \sqcup B_{q-1}\} = \varnothing$$

Отсюда уже следует 11. Таким образом мы выяснили, что $y \in B_q$ и $x \to y$. Такого быть не может, т.к. все остатки сумм из B_q различны по модулю 7.

ii. $y \notin S_q$. Тогда либо $y \to z$ и повторим рассуждение, либо $y_1 \geq 7$. Тогда

$$(\forall i \leq q \; y_i \geq 7) \Rightarrow sum(x) = sum(y) \geq 7q$$

Заметим, что подходящие x - это три последних элемента B_q. Их суммы равны $7q + 1$, $7q + 3$, $7q + 5$. Тогда, если увеличить *базу* на 1, то все элементы y уменьшаться на 7, и y уже можно начать рассматривать как элемент S_q, с суммой 1, 3 или 5[10]. Заметим, что в замене $x \to y$ участвовало не более 7 прикорневых поддеревьев(индексов) x (там где стоит 9 и 0). Т.к. *сильно достаточная* замена не зависит от F_0, то мы можем рассмотреть все эти замены для $q = 8$ (минимальное q для существования B_q). Для $q = 8$, такая замена *не сильно достаточна*, поэтому и для произвольного $q \geq 8$ ребра $x \to y$ быть не может. Противоречие.

$\square$

Замечание. В последнем рассуждении мы показали, что S_q - не *минимальный набор*. А именно привели все 3 кортежа, которые имеют парные, совпадающие с ними по сумме при увеличении *базы* на 1.

Следствие 1 (Алгоритм поиска структуры $(N, 8, 6)$-минимального дерева)**.** *Существует полиномиальный алгоритм поиска структуры $(N, 8, 6)$-минимального дерева.*

[10] Кортежи с такой суммой: (0,...0, 1); (0,...0,1,2); (0,...0,1,2,2)

Доказательство. Пусть дано N и нужно найти структуру $(N, 8, 6)$-минимального дерева. Для начала мы выберем максимальное возможное q - количество прикорневых деревьев, т.ч. $23q+q+1 \leq N$. Теперь мы можем построить S_q за линию[11]. Размер такого списка будет $|S_8|+7(q-8)$. Теперь нужно выбрать *базу* K. Мы знаем минимальный(m) и максимальный(M) по сумме элемент в S_q. Можно бинарным поиском подобрать базу K, чтобы $N(K,m) \leq N \leq N(K,M)$. При фиксированной базе проходимся[12] по $S_q(K)$. И находим там единственного кандидата на минимальное $(N, 8, 6, q)$-дерево при фиксированной базе K. По предыдущему замечанию далее надо проверить $S_q(K-1)$[13] и $S_q(K+1)$. Таким образом для фиксированного q время работы алгоритма не более $\mathcal{O}(|S_q|) = \mathcal{O}(N)$. Теперь уменьшаем q, отрезая от S_q B_q снизу, и 0 слева. И повторяем. Общее время работы $\mathcal{O}(N^2)$. $\square$

3 Выводы

Мы изучили структуру деревьев диаметра 8 с минимальным количеством независимых множеств и некоторыми ограничениями на поддеревья. Мы получили полиномиальный алгоритм поиска структуры. Дальнейшая работа может быть направлена на:

1. получение алгоритма поиска структуры в заданных ограничениях, работающего за $\mathcal{O}(1)$, это позволит без вычислений на компьютере получать структуру для произвольного N.

2. получение структуры $(N, 8)$-минимального дерева в общем случае. Для этого нужно проанализировать случаи прикорневых поддеревьев, не являющихся $(N_i, 6)$-минимальными или имеющих маленький размер.

[11] Дописываем к предыдущему слева 0, а снизу B_q

[12] Можно это делать за линию, можно, например, за логарифм, если упорядочивать S_q по сумме

[13] если $K-1 \geq 23$

Список литературы

1. *Prodinger H.*, *Tichy R. F.* Fibonacci numbers of graphs // Fibonacci Quart. — 1982. — T. 20, № 1. — C. 16—21. — ISSN 0015-0517.

2. *Pedersen A. S.*, *Vestergaard P. D.* An upper bound on the number of independent sets in a tree // Ars Combin. — 2007. — T. 84. — C. 85—96. — ISSN 0381-7032.

3. *Dainiak A.* Sharp bounds for the number of maximal independent sets in trees of fixed diameter. — 2008. — DOI: 10.48550/ARXIV.0812.4948. — URL: https://arxiv.org/abs/0812.4948.

4. Merrifield-Simmons index and minimum number of independent sets in short trees / A. Frendrup [и др.] // Ars Combin. — 2013. — T. 111. — C. 85—95. — ISSN 0381-7032.

5. *Dainyak A. B.* On the number of independent sets in the trees of a fixed diameter // Journal of Applied and Industrial Mathematics. — 2010. — T. 4, № 2. — C. 163—171. — ISSN 1990-4797. — DOI: 10.1134/S1990478910020043.

6. *Taletskii D. S.* Trees of Diameter 6 and 7 with Minimum Number of Independent Sets // Mathematical Notes. — 2021. — T. 109, № 1. — C. 280—291. — ISSN 1573-8876. — DOI: 10.1134/S0001434621010326.

A IndSetCntExpr

```python
class IndSetCntExpr:
    def __init__(self, power_bases: List[int], factors: List[Fraction]):
        assert len(power_bases) == len(factors)
        self.power_bases = power_bases.copy()
        self.factors = factors.copy()
        self.remove_zero_factors()

    def remove_zero_factors(self):
        self.power_bases = [self.power_bases[i] for i in range(len(self.factors)) if self.factors[i]!= 0]
        self.factors = [self.factors[i] for i in range(len(self.factors)) if self.factors[i] != 0]

    def is_number(self):
        if len(self.power_bases) > 1:
            return False
        if len(self.power_bases) == 1:
            return self.power_bases[0] == 1
        return True

    def substitute(self, val):
        ret = Fraction(0)
        for p, f in zip(self.power_bases, self.factors):
            ret += f * (p**val)
        return ret

    def __iadd__(self, other):
        for op, of in zip(other.power_bases, other.factors):
            if op in self.power_bases:
                i = self.power_bases.index(op)
                self.factors[i] += of
            else:
                self.power_bases.append(op)
                self.factors.append(of)
        assert len(self.power_bases) == len(self.factors)
        self.remove_zero_factors()
        return self

    def __add__(self, other):
        ret = IndSetCntExpr(self.power_bases, self.factors)
        ret += other
        return ret

    def __mul__(self, other):
        if isinstance(other, self.__class__):
            ret = IndSetCntExpr([], [])
            for (sp, sf) in zip(self.power_bases, self.factors):
                for (op, of) in zip(other.power_bases, other.factors):
                    ret += IndSetCntExpr([op * sp], [of * sf])
            return ret
```

```
        else:
            ret = IndSetCntExpr(self.power_bases, self.factors)
            other_num = Fraction(other)
            ret.factors = [f * other_num for f in ret.factors]
            ret.remove_zero_factors()
            return ret

    def __imul__(self, other):
        new_val = self * other
        self.power_bases = new_val.power_bases.copy()
        self.factors = new_val.factors.copy()
        return self

    def __sub__(self, other):
        return self + other * (-1)

    def __isub__(self, other):
        new_val = self - other
        self.power_bases = new_val.power_bases.copy()
        self.factors = new_val.factors.copy()
        return self

    def find_index_of_max_abs_power(self):
        m = max(self.power_bases, key=abs)
        return self.power_bases.index(m)

    def sign_on_limit(self):
        if len(self.power_bases) == 0:
            return 0, 0
        tmp = IndSetCntExpr(self.power_bases, self.factors)
        id = tmp.find_index_of_max_abs_power()

        assert len(tmp.factors) > 0
        if tmp.is_number():
            return (1 if tmp.factors[0] > 0 else 0 if tmp.factors[0] == 0 else -1), 0
        val_0 = tmp.substitute(0)
        last_sign = 1 if val_0 > 0 else 0 if val_0 == 0 else -1
        last_k = 0

        for k in itertools.count(start=1):
            sum = Fraction(0, 1)
            for i in range(len(tmp.factors)):
                if i != id:
                    sum += abs(tmp.factors[i])
            if sum < abs(tmp.factors[id]):
                return last_sign, last_k

            val_k = tmp.substitute(0)

            new_sign = 1 if val_k > 0 else 0 if val_k == 0 else -1
            if new_sign != last_sign:
```

```
                last_sign = new_sign
                last_k = k

            for i in range(len(tmp.power_bases)):
                tmp.factors[i] *= tmp.power_bases[i]

        raise NotImplementedError
```

B PreRootVertex

```python
class PreRootVertex:
def __init__(self, mod: int, delta: int = 0):
    assert 0 <= mod < 7

    # kd = k + delta
    self.mod = mod
    self.delta = delta
    if mod == 0:
        incl_power_base = 27
        incl_factor = Fraction(27**delta) if delta >= 0 else Fraction(1, 27**(-delta))
        self.incl_ind_cnt = IndSetCntExpr([incl_power_base], [incl_factor])
        # self.incl_ind_cnt = 27**kd
        excl_power_base = 35
        excl_factor = Fraction(35**delta) if delta > 0 else Fraction(1, 35**(-delta))
        self.excl_ind_cnt = IndSetCntExpr([excl_power_base], [excl_factor])
        # self.excl_ind_cnt = 35**kd
    elif mod == 1:
        incl_power_base = 27
        incl_factor = (Fraction(27 ** delta) if delta >= 0 else Fraction(1, 27 ** (-delta))) * 3
        self.incl_ind_cnt = IndSetCntExpr([incl_power_base], [incl_factor])
        # self.incl_ind_cnt = 27**kd * 3
        excl_power_base = 35
        excl_factor = (Fraction(35 ** (delta - 5)) if delta - 5 >= 0 else Fraction(1, 35 ** (-delta + 5))
        self.excl_ind_cnt = IndSetCntExpr([excl_power_base], [excl_factor])
        # self.excl_ind_cnt = 35**(kd - 5) * 97**4
    elif mod == 2:
        incl_power_base = 27
        incl_factor = (Fraction(27 ** delta) if delta >= 0 else Fraction(1, 27 ** (-delta))) * 3
        self.incl_ind_cnt = IndSetCntExpr([incl_power_base], [incl_factor])
        # self.incl_ind_cnt = 27**kd * 3
        excl_power_base = 35
        excl_factor = (Fraction(35 ** (delta - 1)) if delta - 1 >= 0 else Fraction(1, 35 ** (-delta + 1))
        self.excl_ind_cnt = IndSetCntExpr([excl_power_base], [excl_factor])
        # self.excl_ind_cnt = 35**(kd - 1) * 97
    elif mod == 3:
        incl_power_base = 27
        incl_factor = (Fraction(27 ** delta) if delta >= 0 else Fraction(1, 27 ** (-delta))) * 9
        self.incl_ind_cnt = IndSetCntExpr([incl_power_base], [incl_factor])
        # self.incl_ind_cnt = 27**kd * 9
        excl_power_base = 35
        excl_factor = (Fraction(35 ** (delta - 6)) if delta - 6 >= 0 else Fraction(1, 35 ** (-delta + 6))
        self.excl_ind_cnt = IndSetCntExpr([excl_power_base], [excl_factor])
        # self.excl_ind_cnt = 35**(kd - 6) * 97**5
    elif mod == 4:
        incl_power_base = 27
        incl_factor = (Fraction(27 ** delta) if delta >= 0 else Fraction(1, 27 ** (-delta))) * 9
        self.incl_ind_cnt = IndSetCntExpr([incl_power_base], [incl_factor])
        # self.incl_ind_cnt = 27**kd * 9
```

```
        excl_power_base = 35
        excl_factor = (Fraction(35 ** (delta - 2)) if delta - 2 >= 0 else Fraction(1, 35 ** (-delta + 2))
        self.excl_ind_cnt = IndSetCntExpr([excl_power_base], [excl_factor])
        # self.excl_ind_cnt = 35**(kd - 2) * 97**2
    elif mod == 5:
        incl_power_base = 27
        incl_factor = (Fraction(27 ** delta) if delta >= 0 else Fraction(1, 27 ** (-delta))) * 27
        self.incl_ind_cnt = IndSetCntExpr([incl_power_base], [incl_factor])
        # self.incl_ind_cnt = 27**(kd + 1)
        excl_power_base = 35
        excl_factor = (Fraction(35 ** (delta - 7)) if delta - 7 >= 0 else Fraction(1, 35 ** (-delta + 7))
        self.excl_ind_cnt = IndSetCntExpr([excl_power_base], [excl_factor])
        # self.excl_ind_cnt = 35**(kd - 7) * 97**6
    elif mod == 6:
        incl_power_base = 27
        incl_factor = (Fraction(27 ** delta) if delta >= 0 else Fraction(1, 27 ** (-delta))) * 27
        self.incl_ind_cnt = IndSetCntExpr([incl_power_base], [incl_factor])
        # self.incl_ind_cnt = 27**(kd + 1)
        excl_power_base = 35
        excl_factor = (Fraction(35 ** (delta - 3)) if delta - 3 >= 0 else Fraction(1, 35 ** (-delta + 3))
        self.excl_ind_cnt = IndSetCntExpr([excl_power_base], [excl_factor])
        # self.excl_ind_cnt = 35**(kd - 3) * 97**3
```

C SwapHelpers

```python
def get_forest_params(f: List[PreRootVertex]):
    f_ind_cnt = IndSetCntExpr([1], [Fraction(1)])
    f_excl_ind_cnt = IndSetCntExpr([1], [Fraction(1)])
    for v in f:
        f_excl_ind_cnt *= v.excl_ind_cnt
        f_ind_cnt *= (v.excl_ind_cnt + v.incl_ind_cnt)
    return f_excl_ind_cnt, f_ind_cnt


def swap_forest(f: List[PreRootVertex], t: List[PreRootVertex]):
    f_excl_ind_cnt, f_ind_cnt = get_forest_params(f)
    t_excl_ind_cnt, t_ind_cnt = get_forest_params(t)
    return f_ind_cnt - t_ind_cnt, t_excl_ind_cnt - f_excl_ind_cnt


def is_swap_edge(from_forest, to_forest):
    F12, F21_excl = swap_forest(from_forest, to_forest)
     # do not add edge which equalize independent sets count
    if F12.is_number() and F12.substitute(0)==0 and F21_excl.is_number() and F21_excl.substitute(0)==0:
        return False, -1, F12, F21_excl
    max_since = MIN_K_VAL
    sign, since = F12.sign_on_limit()
    max_since = max(max_since, since)
    if sign > 0:
        sign, since = F21_excl.sign_on_limit()
        max_since = max(max_since, since)
        if sign > 0:
            sign, since = (F12 - F21_excl).sign_on_limit()
            max_since = max(max_since, since)
            if sign > 0: # because i_(F0) / i(F0) could equals ONE (if F0 is empty)
                return True, max_since, F12, F21_excl
            else:
                return False, -1, F12, F21_excl
        else:
            return True, max_since, F12, F21_excl
    return False, -1, F12, F21_excl
```

D ForceM7

```python
r mod_from in range(7):
    from_tree = PreRootVertex(mod_from, -1) #-1 because we check transfer from_tree without M7
    for mod_to in range(7):
        delta = 1
        for d in itertools.count(1):
            to_tree = PreRootVertex(mod_to, -d)

```

```
          assert (from_tree.excl_ind_cnt * to_tree.incl_ind_cnt).power_bases == \
                 (from_tree.incl_ind_cnt * to_tree.excl_ind_cnt).power_bases
          criterion = (from_tree.excl_ind_cnt * to_tree.incl_ind_cnt).substitute(0) /\
                      (from_tree.incl_ind_cnt * to_tree.excl_ind_cnt).substitute(0)
          if criterion >= 1:
              str = "mods: {} -> {}; delta: {}; criterion: {}".format(mod_from, mod_to, d, criterion)
              print(str)
              break
```

E Two pre-root subtrees

```
ass Edge:
  def __init__(self, f: tuple, t: tuple, c, f12, f21_excl):
      self.f = f
      self.t = t
      self.c = c
      self.f12 = f12
      self.f21_excl = f21_excl

om collections import defaultdict

aph = [dict(), dict(), dict()]

x_delta = 2
x_dif = max_delta * 7 + 6

sz = [0, 0]
sz = [0, 0]
x_since = MIN_K_VAL
r f_sz[0] in range(7):
  for f_sz[1] in range(f_sz[0], max_dif + 1):
      from_forest = [PreRootVertex(f_sz[0] % 7, f_sz[0] // 7), PreRootVertex(f_sz[1] % 7, f_sz[1] // 7)]
      graph[2][tuple(f_sz)] = []
      cur_since = max_since
      for t_sz[0] in range(max_dif):
          transfer = t_sz[0] - f_sz[0]
          t_sz[1] = f_sz[1] - transfer
          if t_sz[1] < t_sz[0] or t_sz[1] > max_dif:
              continue
          to_forest = [PreRootVertex(t_sz[0] % 7, t_sz[0] // 7), PreRootVertex(t_sz[1] % 7, t_sz[1] // 7)]
          can_swap, since, F12, F21_excl = is_swap_edge(from_forest, to_forest)
          if can_swap:
              graph[2][tuple(f_sz)].append(Edge(tuple(f_sz), tuple(t_sz), since, F12, F21_excl))
              cur_since = min(cur_since, since)
      max_since = max(max_since, cur_since)

int("max k value is: {}".format(max_since))
```

```python
ans = [set(), set(), set()]
for f_sz[0] in range(7):
    for f_sz[1] in range(f_sz[0], max_dif + 1):
        if len(graph[2][tuple(f_sz)]) == 0:
            ans[2].add(tuple(f_sz))
            # print(bests[-1])
print("allowed pairs count of pre-root subtrees: {}".format(len(ans[2])))
print(*sorted(list(ans[2])), sep='\n')
```

F More subtrees

```python
f is_possible_forest(tup, prev_tups):
    for i in range(len(tup)):
        new_tup = tup[:i] + tup[i + 1:]
        min_elem = min(new_tup)
        while min_elem > 6:
            new_tup = tuple(el - 7 for el in new_tup)
            min_elem -= 7
        if new_tup not in prev_tups:
            return False
    return True

f_set = set()

r subtree_cnt in itertools.count(start=3):
    graph.append(dict())
    dif_set.clear()
    for prev_from in ans[subtree_cnt - 1]:
        for last_from in range(max(prev_from), max_dif + 1):
            from_sizes = prev_from + (last_from, )
            if is_possible_forest(from_sizes, ans[subtree_cnt - 1]):
                graph[subtree_cnt][from_sizes] = []

    for from_sizes in graph[subtree_cnt].keys():
        from_sz = 0
        for sz in from_sizes:
            from_sz += sz
        from_forest = [PreRootVertex(sz % 7, sz // 7) for sz in from_sizes]
        cur_since = max_since
        for prev_to in ans[subtree_cnt - 1]:
            prev_to_sz = 0
            for sz in prev_to:
                prev_to_sz += sz
            last_to = from_sz - prev_to_sz
            if last_to < max(prev_to): # subtree tuple is ordered
                continue
            to_sizes = prev_to + (last_to, )

```

```python
        to_forest = [PreRootVertex(sz % 7, sz // 7) for sz in to_sizes]
        can_swap, since, F12, F21_excl = is_swap_edge(from_forest, to_forest)
        if can_swap:
            graph[subtree_cnt][from_sizes].append(Edge(from_sizes, to_sizes, since, F12, F21_excl))
            cur_since = min(cur_since, since)
    max_since = max(max_since, cur_since)

ans.append(set())
for forest, edge_list in graph[subtree_cnt].items():
    if len(edge_list) == 0:
        ans[subtree_cnt].add(forest)
        dif_set.update(forest)

print(”subtree_cnt is {}, size is {}”.format(subtree_cnt, len(ans[subtree_cnt])))
# print(*sorted(list(ans[subtree_cnt])), sep='\n')
if subtree_cnt == 8:
    break
```

G Nine subtrees

```
(0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 1),
(0, 0, 0, 0, 0, 0, 0, 0, 2),
(0, 0, 0, 0, 0, 0, 0, 0, 7),
(0, 0, 0, 0, 0, 0, 0, 0, 9),
(0, 0, 0, 0, 0, 0, 0, 0, 11),
(0, 0, 0, 0, 0, 0, 0, 0, 13),
(0, 0, 0, 0, 0, 0, 0, 0, 15),
(0, 0, 0, 0, 0, 0, 0, 1, 2),
(0, 0, 0, 0, 0, 0, 0, 2, 2),
(0, 0, 0, 0, 0, 0, 0, 2, 15),
(0, 0, 0, 0, 0, 0, 0, 7, 7),
(0, 0, 0, 0, 0, 0, 0, 7, 9),
(0, 0, 0, 0, 0, 0, 0, 9, 9),
(0, 0, 0, 0, 0, 0, 0, 9, 11),
(0, 0, 0, 0, 0, 0, 0, 11, 11),
(0, 0, 0, 0, 0, 0, 0, 11, 13),
(0, 0, 0, 0, 0, 0, 0, 13, 13),
(0, 0, 0, 0, 0, 0, 1, 2, 2),
(0, 0, 0, 0, 0, 0, 2, 2, 2),
(0, 0, 0, 0, 0, 0, 2, 2, 15),
(0, 0, 0, 0, 0, 0, 7, 7, 7),
(0, 0, 0, 0, 0, 0, 7, 7, 9),
```

(0, 0, 0, 0, 0, 0, 7, 9, 9),
(0, 0, 0, 0, 0, 0, 9, 9, 9),
(0, 0, 0, 0, 0, 0, 9, 9, 11),
(0, 0, 0, 0, 0, 0, 9, 11, 11),
(0, 0, 0, 0, 0, 0, 11, 11, 11),
(0, 0, 0, 0, 0, 2, 2, 2, 2),
(0, 0, 0, 0, 0, 7, 7, 7, 7),
(0, 0, 0, 0, 0, 7, 7, 7, 9),
(0, 0, 0, 0, 0, 7, 7, 9, 9),
(0, 0, 0, 0, 0, 7, 9, 9, 9),
(0, 0, 0, 0, 0, 9, 9, 9, 9),
(0, 0, 0, 0, 0, 9, 9, 9, 11),
(0, 0, 0, 0, 0, 9, 9, 11, 11),
(0, 0, 0, 0, 2, 2, 2, 2, 2),
(0, 0, 0, 0, 7, 7, 7, 7, 7),
(0, 0, 0, 0, 7, 7, 7, 7, 9),
(0, 0, 0, 0, 7, 7, 7, 9, 9),
(0, 0, 0, 0, 7, 7, 9, 9, 9),
(0, 0, 0, 0, 7, 9, 9, 9, 9),
(0, 0, 0, 0, 9, 9, 9, 9, 9),
(0, 0, 0, 0, 9, 9, 9, 9, 11),
(0, 0, 0, 2, 2, 2, 2, 2, 2),
(0, 0, 0, 7, 7, 7, 7, 7, 7),
(0, 0, 0, 7, 7, 7, 7, 7, 9),
(0, 0, 0, 7, 7, 7, 7, 9, 9),
(0, 0, 0, 7, 7, 7, 9, 9, 9),
(0, 0, 0, 7, 7, 9, 9, 9, 9),
(0, 0, 0, 7, 9, 9, 9, 9, 9),
(0, 0, 0, 9, 9, 9, 9, 9, 9),
(0, 0, 7, 7, 7, 7, 7, 7, 7),
(0, 0, 7, 7, 7, 7, 7, 7, 9),
(0, 0, 7, 7, 7, 7, 7, 9, 9),
(0, 0, 7, 7, 7, 7, 9, 9, 9),
(0, 0, 7, 7, 7, 9, 9, 9, 9),
(0, 0, 7, 7, 9, 9, 9, 9, 9),
(0, 0, 7, 9, 9, 9, 9, 9, 9),
(0, 7, 7, 7, 7, 7, 7, 7, 7),
(0, 7, 7, 7, 7, 7, 7, 7, 9),
(0, 7, 7, 7, 7, 7, 7, 9, 9),
(0, 7, 7, 7, 7, 7, 9, 9, 9),

(0, 7, 7, 7, 7, 9, 9, 9, 9),
(0, 7, 7, 7, 9, 9, 9, 9, 9),
(0, 7, 7, 9, 9, 9, 9, 9, 9)