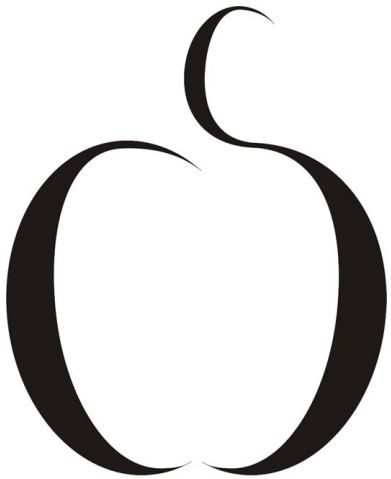
Алгоритмы для NP-трудных задач

Лекция 10: Точные и FPT-алгоритмы

А. Куликов

Computer Science клуб при ПОМИ
http://logic.pdmi.ras.ru/~infclub/



План лекции

1. Формула включений-исключений
   - Задача о гамильтоновом пути
   - Задача о количестве совершенных паросочетаний
   - Задача о количестве совершенных паросочетаний в двудольном графе

План лекции

1. Формула включений-исключений
   - Задача о гамильтоновом пути
   - Задача о количестве совершенных паросочетаний
   - Задача о количестве совершенных паросочетаний в двудольном графе
2. Сведение к простой задаче
   - Задача о сумме подмножества
   - Задачи о k-сумме и табличной k-сумме

План лекции

1. Формула включений-исключений
   - Задача о гамильтоновом пути
   - Задача о количестве совершенных паросочетаний
   - Задача о количестве совершенных паросочетаний в двудольном графе
2. Сведение к простой задаче
   - Задача о сумме подмножества
   - Задачи о k-сумме и табличной k-сумме
3. Алгебраизация
   - Задача максимальной 2-выполнимости

План лекции

Формула включений-исключений

Лемма

Пусть A — некоторое множество, $f, g\colon 2^A \to \mathbb{R}$, т.ч. $f(X) = \sum_{Y \subseteq X} g(Y)$. Тогда

$$g(X) = \sum_{Y \subseteq X} (-1)^{|X-Y|} f(Y)\,.$$

Формула включений-исключений

Лемма

Пусть A — некоторое множество, $f, g\colon 2^A \to \mathbb{R}$, т.ч. $f(X) = \sum_{Y \subseteq X} g(Y)$. Тогда

$$g(X) = \sum_{Y \subseteq X} (-1)^{|X-Y|} f(Y)\,.$$

Доказательство

$$\begin{aligned}\sum_{Y \subseteq X} (-1)^{|X-Y|} f(Y) &= \sum_{Y \subseteq X} \sum_{Z \subseteq Y} (-1)^{|X-Y|} g(Z) = \\ &= \sum_{Z \subseteq X} g(Z) \sum_{Z \subseteq Y \subseteq X} (-1)^{|X-Y|} = g(X)\end{aligned}$$

(последняя сумма равна 1, если $Z = X$, и нулю иначе). □

План лекции

Задача о гамильтоновом пути

Задача о гамильтоновом пути

- Формулировка задачи: необходимо проверить, есть ли в данном графе простой путь, проходящий через все вершины, начинающийся в заданной вершине s и заканчивающийся в заданной вершине t.

Задача о гамильтоновом пути

- Формулировка задачи: необходимо проверить, есть ли в данном графе простой путь, проходящий через все вершины, начинающийся в заданной вершине s и заканчивающийся в заданной вершине t.
- Для $\{s, t\} \subseteq X \subseteq V$ обозначим через $g(X)$ количество путей (не обязательно простых! путь может проходить по некоторым вершинам несколько раз, а по некоторым вообще не проходить) длины $n-1$ из s в t, проходящих только по вершинам множества X.

Задача о гамильтоновом пути

- Формулировка задачи: необходимо проверить, есть ли в данном графе простой путь, проходящий через все вершины, начинающийся в заданной вершине s и заканчивающийся в заданной вершине t.
- Для $\{s, t\} \subseteq X \subseteq V$ обозначим через $g(X)$ количество путей (не обязательно простых! путь может проходить по некоторым вершинам несколько раз, а по некоторым вообще не проходить) длины $n-1$ из s в t, проходящих только по вершинам множества X.
- Нетрудно видеть, что значение $g(X)$ содержится в строке s и столбце t матрицы A^{n-1}, где A — матрица смежности графа $G[X]$.

Задача о гамильтоновом пути (продолжение)

Задача о гамильтоновом пути (продолжение)

- Пусть теперь $f(X)$ есть количество путей длины $n-1$ из s в t, проходящих по всем вершинам множества X. В частности, $f(V)$ есть количество гамильтоновых путей из s в t.

Задача о гамильтоновом пути (продолжение)

- Пусть теперь $f(X)$ есть количество путей длины $n-1$ из s в t, проходящих по всем вершинам множества X. В частности, $f(V)$ есть количество гамильтоновых путей из s в t.
- Тогда

$$f(V) = \sum_{Y \subseteq V} (-1)^{|V-Y|} g(Y)\,.$$

Задача о гамильтоновом пути (продолжение)

- Пусть теперь $f(X)$ есть количество путей длины $n-1$ из s в t, проходящих по всем вершинам множества X. В частности, $f(V)$ есть количество гамильтоновых путей из s в t.
- Тогда

$$f(V) = \sum_{Y \subseteq V} (-1)^{|V-Y|} g(Y)\,.$$

- Таким образом, количество гамильтоновых путей в графе может быть найдено за время $O^*(2^n)$ и полиномиальную память.

План лекции

Задача о количестве совершенных паросочетаний

Задача о количестве совершенных паросочетаний

- Совершенным паросочетанием (perfect matching) графа $G = (V, E)$ называется множество из $n/2$ ребер, содержащее каждую вершину графа ровно один раз.

Задача о количестве совершенных паросочетаний

- Совершенным паросочетанием (perfect matching) графа $G = (V, E)$ называется множество из $n/2$ ребер, содержащее каждую вершину графа ровно один раз.
- Максимальное по размеру паросочетание в графе может быть найдено за полиномиальное время.

Задача о количестве совершенных паросочетаний

- Совершенным паросочетанием (perfect matching) графа $G = (V, E)$ называется множество из $n/2$ ребер, содержащее каждую вершину графа ровно один раз.
- Максимальное по размеру паросочетание в графе может быть найдено за полиномиальное время.
- Мы рассмотрим задачу нахождения количества совершенных паросочетаний графа. Наивный алгоритм требует времени $O^*(2^m)$.

Алгоритм

Алгоритм

- Пусть $f(Y)$ есть количество способов покрыть все вершины Y, используя ровно $n/2$ ребер.

Алгоритм

- Пусть $f(Y)$ есть количество способов покрыть все вершины Y, используя ровно $n/2$ ребер.
- Тогда $f(V)$ — количество совершенных паросочетаний.

Алгоритм

- Пусть $f(Y)$ есть количество способов покрыть все вершины Y, используя ровно $n/2$ ребер.
- Тогда $f(V)$ — количество совершенных паросочетаний.
- Пусть теперь $g(Y)$ есть количество способов выбрать $n/2$ рёбер графа, все концы которых принадлежат множеству Y (но при этом не обязательно покрывают все множество Y).

Алгоритм

- Пусть $f(Y)$ есть количество способов покрыть все вершины Y, используя ровно $n/2$ ребер.
- Тогда $f(V)$ — количество совершенных паросочетаний.
- Пусть теперь $g(Y)$ есть количество способов выбрать $n/2$ рёбер графа, все концы которых принадлежат множеству Y (но при этом не обязательно покрывают все множество Y).
- Поскольку для данного Y вычислить $g(Y)$ легко, получаем алгоритм, находящий за время $O^*(2^n)$ количество совершенных паросочетаний в графе.

План лекции

1. Формула включений-исключений
   - Задача о гамильтоновом пути
   - Задача о количестве совершенных паросочетаний
   - Задача о количестве совершенных паросочетаний в двудольном графе
2. Сведение к простой задаче
   - Задача о сумме подмножества
   - Задачи о k-сумме и табличной k-сумме
3. Алгебраизация
   - Задача максимальной 2-выполнимости

Задача о количестве совершенных паросочетаний в двудольном графе

Задача о количестве совершенных паросочетаний в двудольном графе

- Для двудольных графов алгоритм можно ускорить.

Задача о количестве совершенных паросочетаний в двудольном графе

- Для двудольных графов алгоритм можно ускорить.
- Пусть $V = L \cup R$ и $|L| = |R| = n/2$.

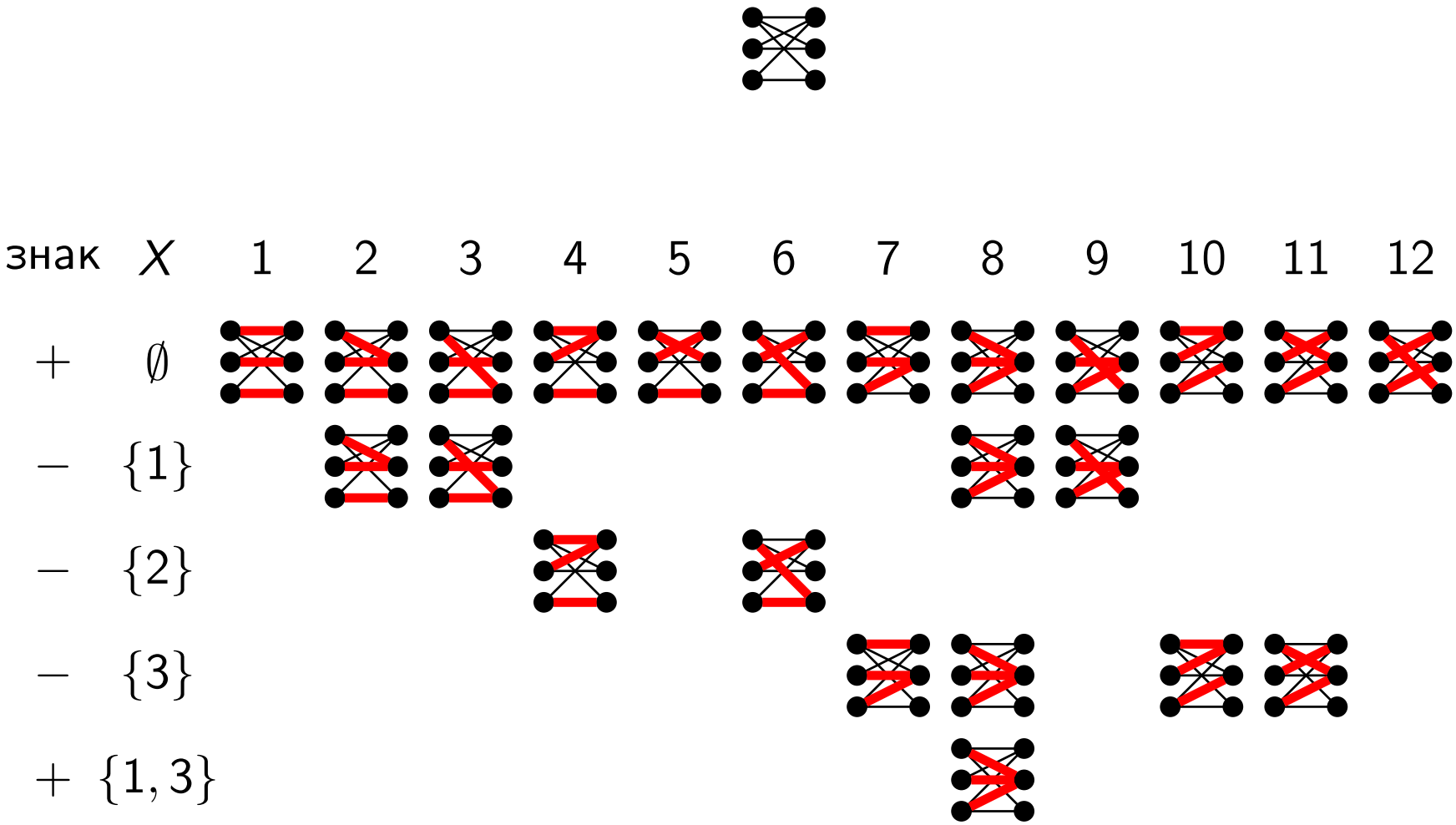
Задача о количестве совершенных паросочетаний в двудольном графе

- Для двудольных графов алгоритм можно ускорить.
- Пусть $V = L \cup R$ и $|L| = |R| = n/2$.
- Тогда для $Y \subseteq R$ (а не $Y \subseteq V$) определим $g(Y)$ как количество способов выбрать $n/2$ ребер, используя все вершиын L и некоторые вершины из Y.

Задача о количестве совершенных паросочетаний в двудольном графе

- Для двудольных графов алгоритм можно ускорить.
- Пусть $V = L \cup R$ и $|L| = |R| = n/2$.
- Тогда для $Y \subseteq R$ (а не $Y \subseteq V$) определим $g(Y)$ как количество способов выбрать $n/2$ ребер, используя все вершиын L и некоторые вершины из Y.
- Время работы будет $O^*(2^{n/2})$.

Пример



План лекции

План лекции

Задача о сумме подмножества

Определение

Задача о сумме подмножества (subset-sum problem) заключается в проверке существования среди данных n натуральных чисел $b_1, \ldots, b_n$ такого подмножества чисел, сумма которых равнялась бы заданному числу B.

Задача о сумме подмножества

Определение

Задача о сумме подмножества (subset-sum problem) заключается в проверке существования среди данных n натуральных чисел $b_1, \ldots, b_n$ такого подмножества чисел, сумма которых равнялась бы заданному числу B.

Замечание

Очевидный перебор требует времени 2^n.

Лучший известный алгоритм

Алгоритм

Лучший известный алгоритм

Алгоритм

- Запишем в массив размера $2^{n/2}$ все возможные суммы чисел среди $\{b_1, \ldots, b_{n/2}\}$.

Лучший известный алгоритм

Алгоритм

- Запишем в массив размера $2^{n/2}$ все возможные суммы чисел среди $\{b_1, \ldots, b_{n/2}\}$.
- Отсортируем полученный массив.

Лучший известный алгоритм

Алгоритм

- Запишем в массив размера $2^{n/2}$ все возможные суммы чисел среди $\{b_1, \ldots, b_{n/2}\}$.
- Отсортируем полученный массив.
- Для каждого $S \subseteq \{b_{n/2+1}, \ldots, b_n\}$ проверим бинарным поиском наличие числа $B - S$ в массиве.

Лучший известный алгоритм

Алгоритм

- Запишем в массив размера $2^{n/2}$ все возможные суммы чисел среди $\{b_1, \ldots, b_{n/2}\}$.
- Отсортируем полученный массив.
- Для каждого $S \subseteq \{b_{n/2+1}, \ldots, b_n\}$ проверим бинарным поиском наличие числа $B - S$ в массиве.

Замечание

Лучший известный алгоритм

Алгоритм

- Запишем в массив размера $2^{n/2}$ все возможные суммы чисел среди $\{b_1, \ldots, b_{n/2}\}$.
- Отсортируем полученный массив.
- Для каждого $S \subseteq \{b_{n/2+1}, \ldots, b_n\}$ проверим бинарным поиском наличие числа $B - S$ в массиве.

Замечание

- Время работы — $O^*(2^{n/2})$.

Лучший известный алгоритм

Алгоритм

- Запишем в массив размера $2^{n/2}$ все возможные суммы чисел среди $\{b_1, \ldots, b_{n/2}\}$.
- Отсортируем полученный массив.
- Для каждого $S \subseteq \{b_{n/2+1}, \ldots, b_n\}$ проверим бинарным поиском наличие числа $B - S$ в массиве.

Замечание

- Время работы — $O^*(2^{n/2})$.
- Быстрее алгоритма неизвестно.

Лучший известный алгоритм

Алгоритм

- Запишем в массив размера $2^{n/2}$ все возможные суммы чисел среди $\{b_1, \ldots, b_{n/2}\}$.
- Отсортируем полученный массив.
- Для каждого $S \subseteq \{b_{n/2+1}, \ldots, b_n\}$ проверим бинарным поиском наличие числа $B - S$ в массиве.

Замечание

- Время работы — $O^*(2^{n/2})$.
- Быстрее алгоритма неизвестно.
- Неизвестно и алгоритма, который бы работал быстрее чем 2^n и использовал лишь полиномиальную память.

План лекции

Задача о табличной сумме

Определение

В задаче о табличной k-сумме (table-k sum problem) дана матрица размера $k \times m$ натуральных чисел и число S и необходимо проверить, можно ли выбрать из каждой строки матрицы ровно по одному числу так, чтобы в сумме выбранные числа давали S.

Задача о табличной сумме

Определение

В задаче о табличной k-сумме (table-k sum problem) дана матрица размера $k \times m$ натуральных чисел и число S и необходимо проверить, можно ли выбрать из каждой строки матрицы ровно по одному числу так, чтобы в сумме выбранные числа давали S.

Замечание

Задача о табличной сумме

Определение

В задаче о табличной k-сумме (table-k sum problem) дана матрица размера $k \times m$ натуральных чисел и число S и необходимо проверить, можно ли выбрать из каждой строки матрицы ровно по одному числу так, чтобы в сумме выбранные числа давали S.

Замечание

- Очевидный перебор работает m^k.

Задача о табличной сумме

Определение

В задаче о табличной k-сумме (table-k sum problem) дана матрица размера $k \times m$ натуральных чисел и число S и необходимо проверить, можно ли выбрать из каждой строки матрицы ровно по одному числу так, чтобы в сумме выбранные числа давали S.

Замечание

- Очевидный перебор работает m^k.
- В алгоритме выше мы использовали алгоритм, решающий задачу о табличной 2-сумме за время $O(m \log m)$.

Задача о табличной сумме

Определение

В задаче о табличной k-сумме (table-k sum problem) дана матрица размера $k \times m$ натуральных чисел и число S и необходимо проверить, можно ли выбрать из каждой строки матрицы ровно по одному числу так, чтобы в сумме выбранные числа давали S.

Замечание

- Очевидный перебор работает m^k.
- В алгоритме выше мы использовали алгоритм, решающий задачу о табличной 2-сумме за время $O(m \log m)$.
- Этот же алгоритм можно использовать для решения задачи о табличной k-сумме за время $O(m^{\lceil k/2 \rceil} \log m)$ и память $O(m^{\lceil k/2 \rceil})$.

Задача о 3-сумме

Определение

В задаче о k-сумме (k-sum problem) даны m натуральных чисел и число S и требуется проверить, можно ли из них выбрать k так, чтобы их сумма равнялась S.

Задача о 3-сумме

Определение

В задаче о k-сумме (k-sum problem) даны m натуральных чисел и число S и требуется проверить, можно ли из них выбрать k так, чтобы их сумма равнялась S.

Факт

Задача о 3-сумме

Определение

В задаче о k-сумме (k-sum problem) даны m натуральных чисел и число S и требуется проверить, можно ли из них выбрать k так, чтобы их сумма равнялась S.

Факт

- Существует алгоритм, решающий задачу о 3-сумме за время $O(m^2)$.

Задача о 3-сумме

Определение

В задаче о k-сумме (k-sum problem) даны m натуральных чисел и число S и требуется проверить, можно ли из них выбрать k так, чтобы их сумма равнялась S.

Факт

- Существует алгоритм, решающий задачу о 3-сумме за время $O(m^2)$.
- Известно большое количество геометрических задач, для каждой из которых лучший известный алгоритм работает квадратичное время и каждая из которых, содержит задачу о 3-сумме, как частный случай, и этот частный случай интуитивно представляется самым трудным.

Пример геометрической задачи

Пример геометрической задачи

- Пусть даны m точек на плоскости и требуется проверить, лежат ли какие-нибудь три из них на одной прямой.

Пример геометрической задачи

- Пусть даны m точек на плоскости и требуется проверить, лежат ли какие-нибудь три из них на одной прямой.
- Пересечением прямой $y = ax + b$ и кривой $y = f(x) = x^3 - Sx^2$ являются корни уравнения $x^3 - Sx^2 - ax - b = 0$.

Пример геометрической задачи

- Пусть даны m точек на плоскости и требуется проверить, лежат ли какие-нибудь три из них на одной прямой.
- Пересечением прямой $y = ax + b$ и кривой $y = f(x) = x^3 - Sx^2$ являются корни уравнения $x^3 - Sx^2 - ax - b = 0$.
- Значит, множество точек $(a_1, f(a_1)), \ldots, (a_m, f(a_m))$ содержит три точки $(a_x, f(a_x)), (a_y, f(a_y)), (a_z, f(a_z))$ на одной прямой тогда и только тогда, когда $a_x + a_y + a_z = S$.

План лекции

1. Формула включений-исключений
   - Задача о гамильтоновом пути
   - Задача о количестве совершенных паросочетаний
   - Задача о количестве совершенных паросочетаний в двудольном графе
2. Сведение к простой задаче
   - Задача о сумме подмножества
   - Задачи о k-сумме и табличной k-сумме
3. Алгебраизация
   - Задача максимальной 2-выполнимости

План лекции

Алгебраизация

Алгебраизация

- Для набора значений переменным формулы $t = (t_1, \ldots, t_n) \in \{0,1\}^n$ обозначим через $N(t)$ количество выполненных клозов данной формулы в 2-КНФ.

Алгебраизация

- Для набора значений переменным формулы $t = (t_1, \ldots, t_n) \in \{0,1\}^n$ обозначим через $N(t)$ количество выполненных клозов данной формулы в 2-КНФ.
- Пусть

$$G(w) = \sum_{t \in \{0,1\}^n} w^{N(t)}.$$

Алгебраизация

- Для набора значений переменным формулы $t = (t_1, \ldots, t_n) \in \{0,1\}^n$ обозначим через $N(t)$ количество выполненных клозов данной формулы в 2-КНФ.
- Пусть

$$G(w) = \sum_{t \in \{0,1\}^n} w^{N(t)}.$$

- Нас, таким образом, интересует степень этого многочлена.

Построение матриц

Построение матриц

- Разобьем переменные формулы на три часта A, B, C размера $n/3$.

Построение матриц

- Разобьем переменные формулы на три часта A, B, C размера $n/3$.
- Для $t = (a, b, c)$, где $a, b, c \in \{0, 1\}^{n/3}$,

$$N(t) = N_C(a, b) + N_B(a, c) + N_A(b, c)\,,$$

где $N_C(a, b)$ есть количество клозов, которые не зависят от переменных множества C и выполнены наборами a, b (оставшиеся два слагаемых определяются аналогично).

Построение матриц

- Разобьем переменные формулы на три часта A, B, C размера $n/3$.
- Для $t = (a, b, c)$, где $a, b, c \in \{0, 1\}^{n/3}$,

$$N(t) = N_C(a, b) + N_B(a, c) + N_A(b, c)\,,$$

где $N_C(a, b)$ есть количество клозов, которые не зависят от переменных множества C и выполнены наборами a, b (оставшиеся два слагаемых определяются аналогично).

- Определим матрицы N_C^w, N_B^w, N_A^w размера $2^{n/3} \times 2^{n/3}$ следующим образом:

$$N_C^w(a, b) = w^{N_C(a,b)}, \quad N_B^w(b, c) = w^{N_B(b,c)}, \quad N_C^w(a, b) = w^{N_C(a,b)}\,.$$

Вычисление многочлена

$$G(w) = \sum_{a,b,c} w^{N_C(a,b)+N_B(a,c)+N_A(b,c)} =$$

Вычисление многочлена

$$
\begin{aligned}
G(w) &= \sum_{a,b,c} w^{N_C(a,b)+N_B(a,c)+N_A(b,c)} = \\
&= \sum_{a,b,c} w^{N_C(a,b)} w^{N_B(a,c)} w^{N_A(b,c)} =
\end{aligned}
$$

Вычисление многочлена

$$
\begin{aligned}
G(w) &= \sum_{a,b,c} w^{N_C(a,b)+N_B(a,c)+N_A(b,c)} = \\
&= \sum_{a,b,c} w^{N_C(a,b)} w^{N_B(a,c)} w^{N_A(b,c)} = \\
&= \sum_{a,c} w^{N_B(a,c)} \sum_{b} w^{N_C(a,b)} w^{N_A(b,c)} =
\end{aligned}
$$

Вычисление многочлена

$$
\begin{aligned}
G(w) &= \sum_{a,b,c} w^{N_C(a,b)+N_B(a,c)+N_A(b,c)} = \\
&= \sum_{a,b,c} w^{N_C(a,b)} w^{N_B(a,c)} w^{N_A(b,c)} = \\
&= \sum_{a,c} w^{N_B(a,c)} \sum_{b} w^{N_C(a,b)} w^{N_A(b,c)} = \\
&= \sum_{a,c} N_B^w(a,c)(N_C^w N_A^w)(a,c)\,.
\end{aligned}
$$

Итог

Итог

- Итак, полученный алгоритм требует времени $O^*2^{\omega n/3}$ и $O(2^{2n/3})$ памяти ($\omega \approx 2.38$ — экспонента умножения матриц).

Итог

- Итак, полученный алгоритм требует времени $O^*2^{\omega n/3}$ и $O(2^{2n/3})$ памяти ($\omega \approx 2.38$ — экспонента умножения матриц).
- Это лучше полного перебора по времени, но гораздо хуже по памяти (time-space tradeoff).

Спасибо за внимание!