Дискретная оптимизация
Сложные комбинаторные задачи

Роланд Хильдебранд

Институт Вычислительной Математики им. Г.И. Марчука РАН

31 марта 2026 г.

План

сложные комбинаторные задачи

- **наименьшее хордальное дополнение**
- выполнимость булевых формул
- задача о сумме подмножества
- задача о рюкзаке

Хордальные дополнения

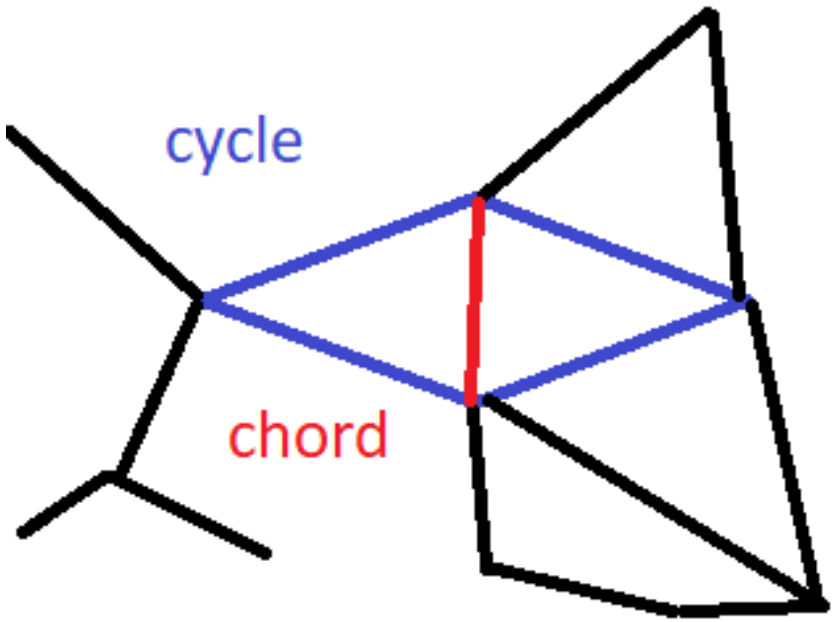
Определение

Пусть G — ненаправленный граф. Граф называется хордальным если каждый цикл длины ≥ 4 имеет хорду.

Определение

Пусть $G = (V, E)$ — ненаправленный граф. Множество $E' \subset (V \times V) \setminus E$ называется хордальным дополнением если $G' = (V, E \cup E')$ — хордальный.
Дополнение E' называется минимальным если для любого $E'' \subsetneq E'$ граф $G'' = (V, E'')$ — не хордальный.
Дополнение E' называется наименьшим, если для любого дополнения E'' имеем $|E''| \geq E'$.

Хордальные дополнения



цикл с хордой

Хордальные дополнения

Определение

Пусть $G = (V, E)$ — ненаправленный граф с n вершинами. Биективное отображение $\alpha : V \to \{1, \ldots, n\}$ называется порядком совершенного исключения если из $\alpha(u) < \min(\alpha(v), \alpha(w))$, $(u, v), (u, w) \in E$ следует $(v, w) \in E$.

интерпретация: при факторизации матрицы с симметричным шаблоном заполнения, заданным G, и если строки и столбцы матрицы ранжированы в порядке совершенного исключения, не происходит заполнения треугольного фактора

Теорема

Граф обладает порядком совершенного исключения тогда и только тогда, когда он хордальный.

Хордальные дополнения

простые задачи

- проверить хордальность графа
- найти порядок совершенного исключения для хордального графа
- найти минимальное хордальное дополнение для произвольного графа

поиск порядка совершенного исключения: лексикографический поиск

сложная задача: найти наименьшее хордальное дополнение

приложение задачи о наименьшем хордальном дополнении: минимизировать заполнение при факторизации матрицы

Лексикографический поиск

построение порядка вершин

маркируем вершины числом m в системе чисел по модулю $n+1$ вида $m = 0, k_1 k_2 \ldots,\ k_j \in \{1, \ldots, n\}$

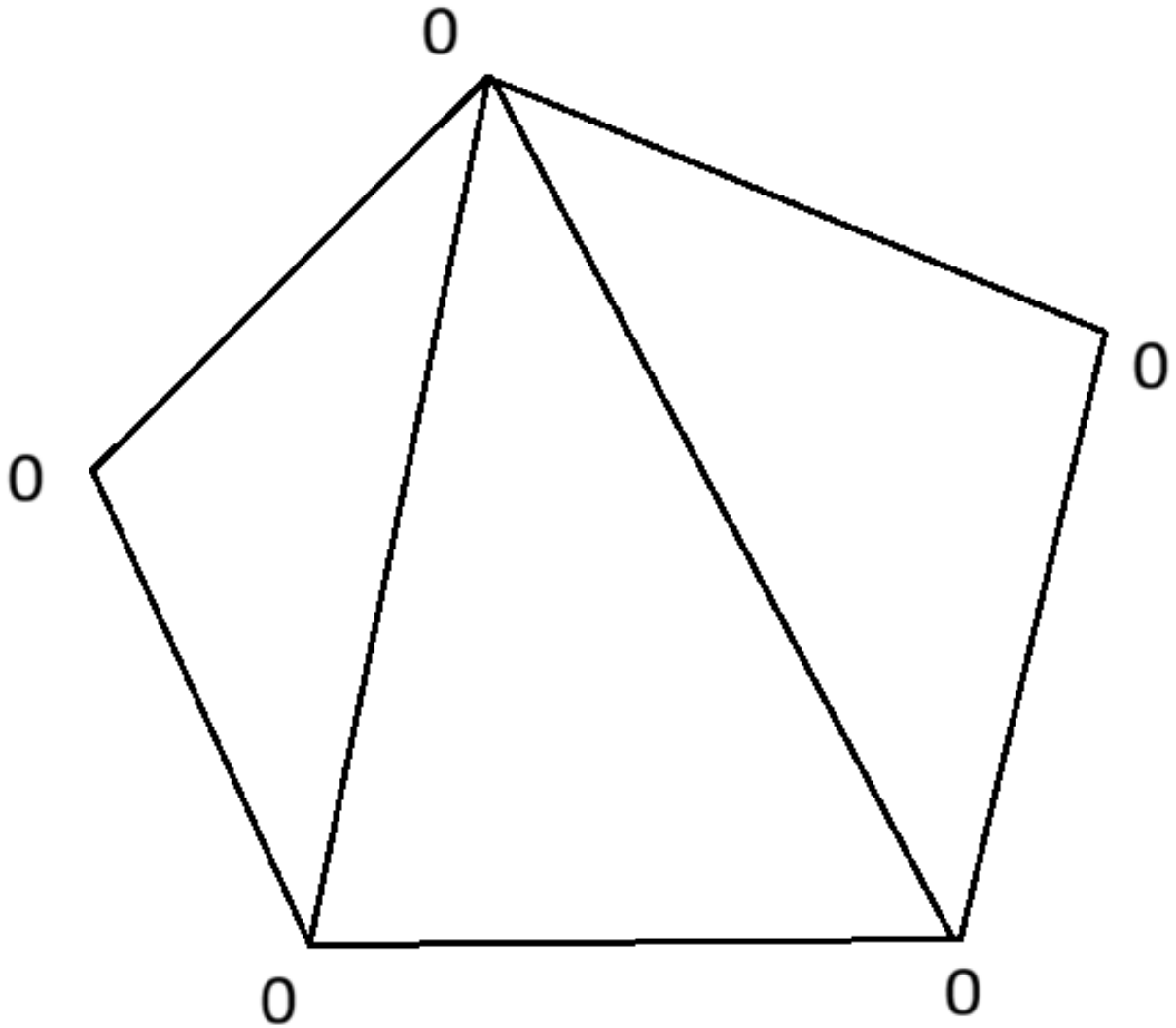
- все вершины получают маркировку $m(u) = 0$
- for $i = n : -1 : 1$
  - выбираем неупорядоченную вершину v с наибольшей маркировкой $m(v)$
  - присваиваем ей порядок $\alpha(v) = i$
  - у всех смежных с v неупорядоченных вершин добавляем цифру i в маркировку

после завершнения построения порядок проверяется на выполнение свойства совершенного исключения

- если свойство выполнено, то порядок построен
- если свойство невыполнено, то граф не хордальный

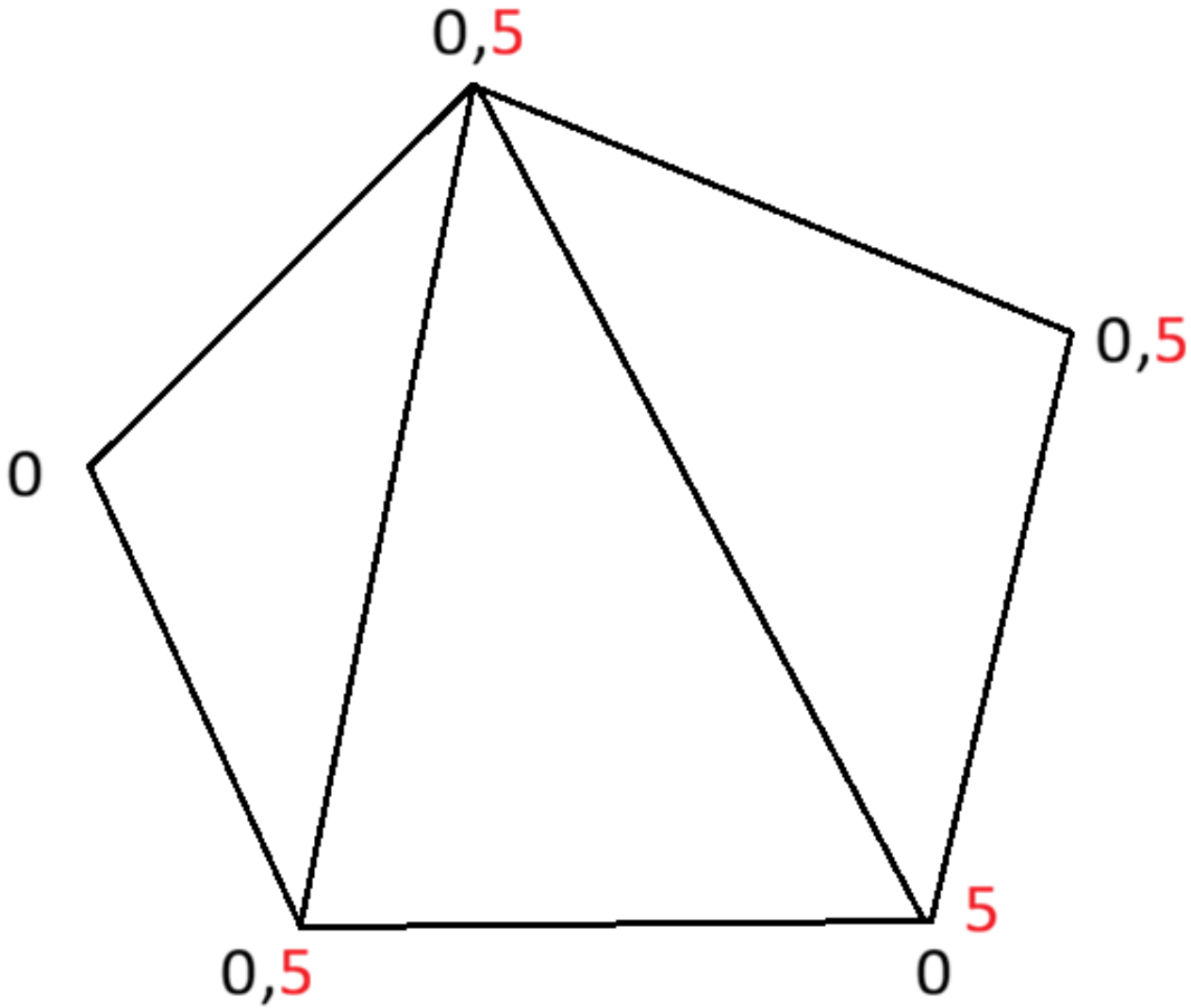
Лексикографический поиск

пример



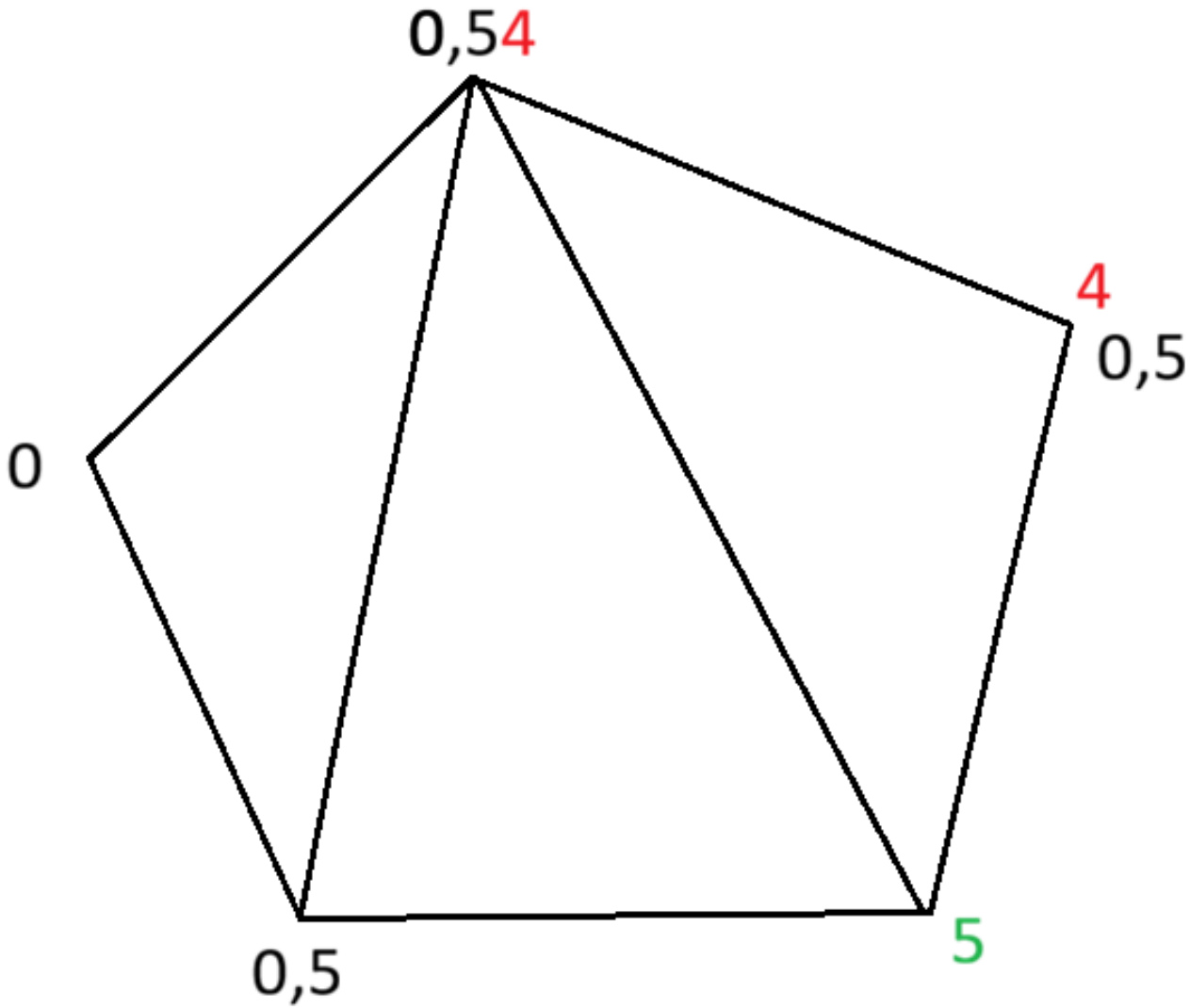
Лексикографический поиск

пример



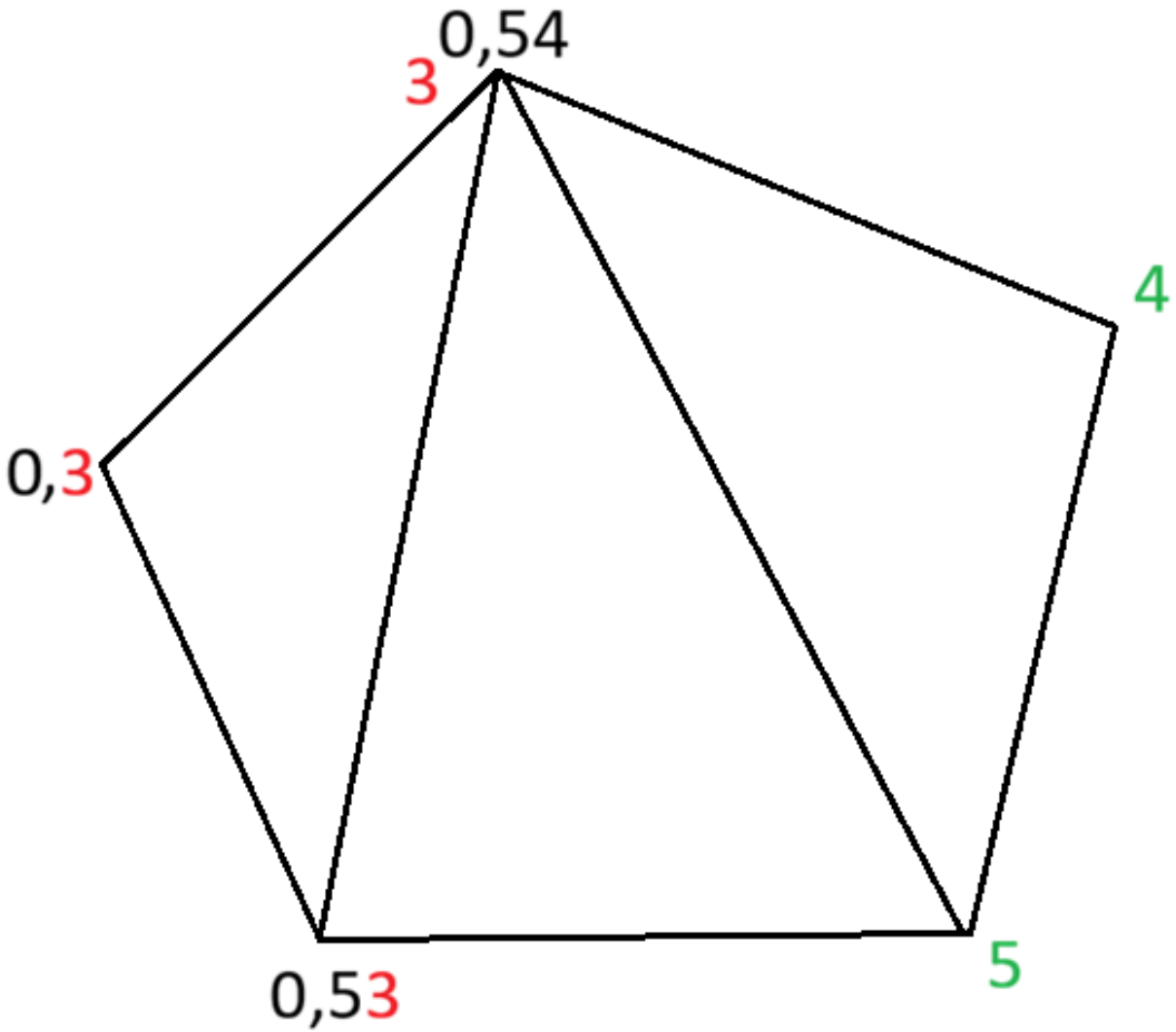
Лексикографический поиск

пример



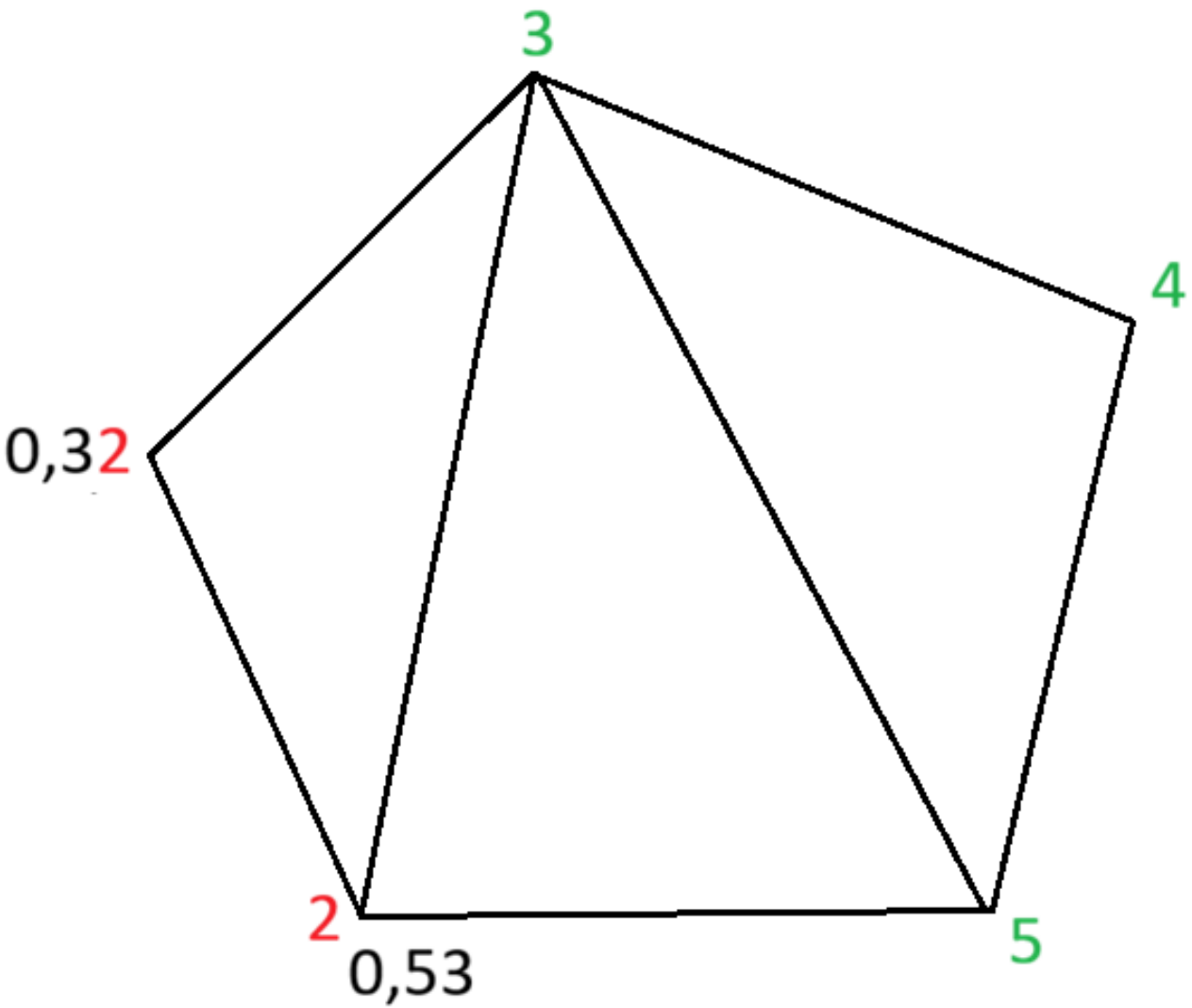
Лексикографический поиск

пример



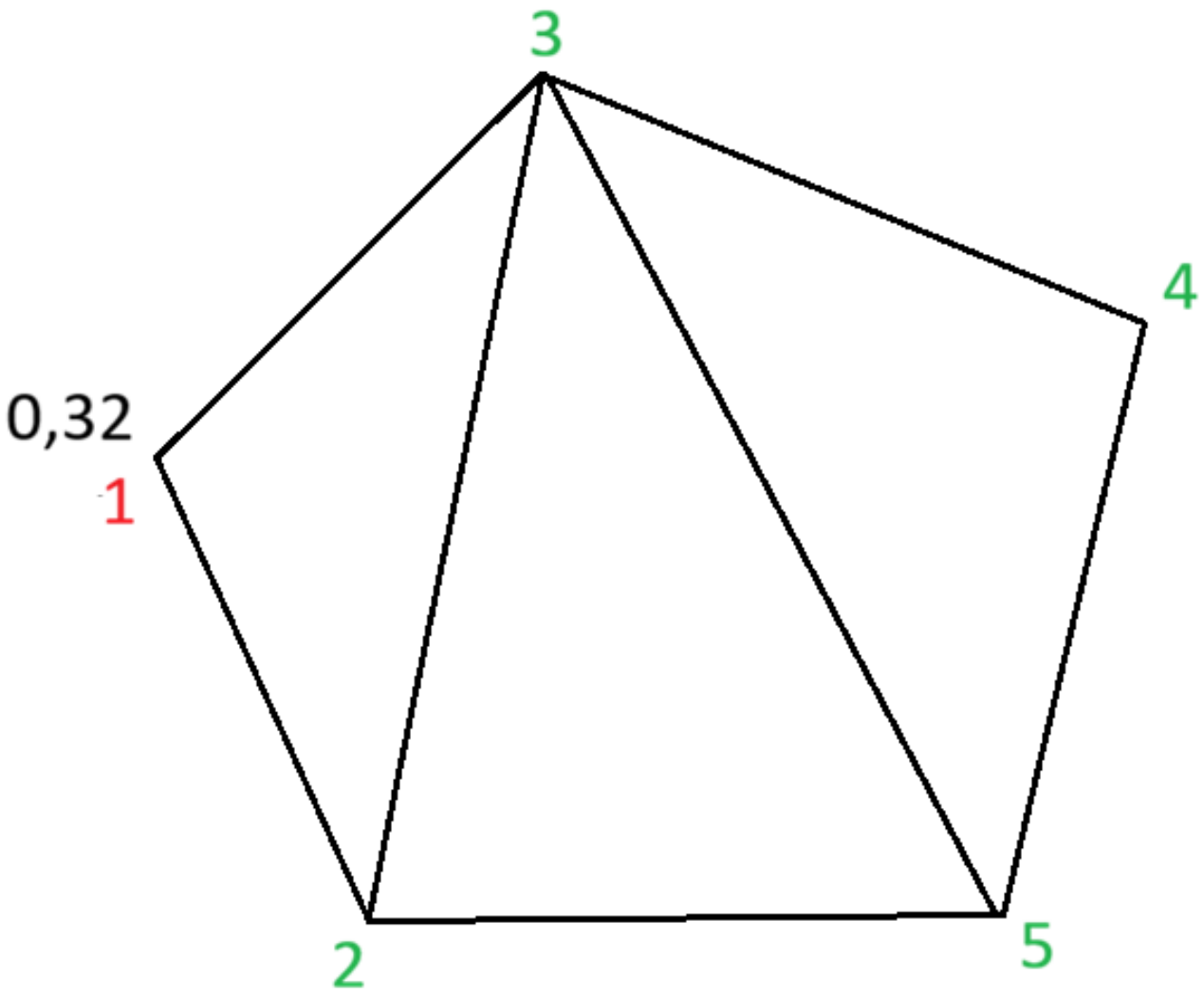
Лексикографический поиск

пример



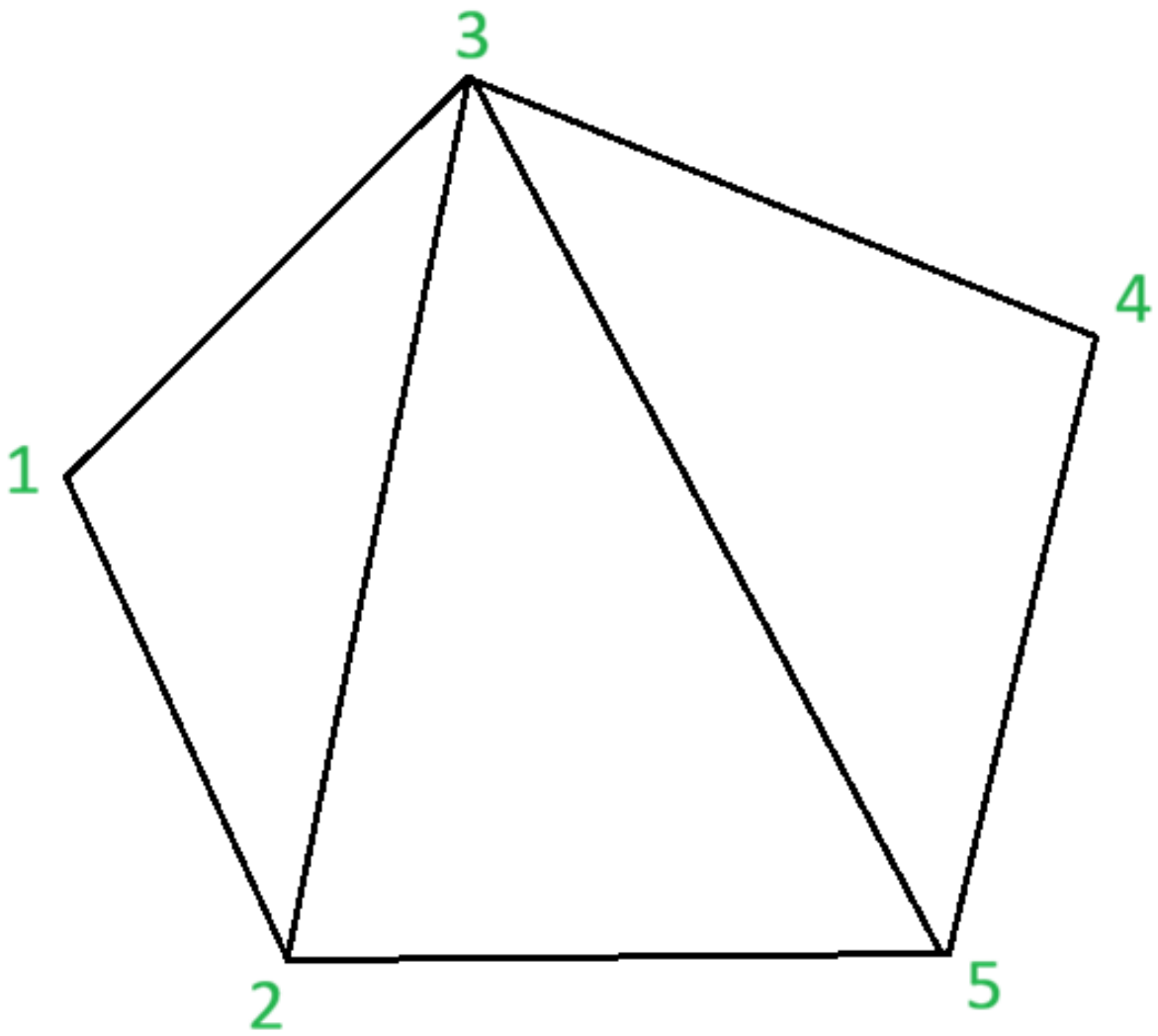
Лексикографический поиск

пример



Лексикографический поиск

пример



Минимальные хордальные дополнения

Лемма

Пусть $G = (V, E)$ — граф. Тогда E' — минимальное хордальное дополнение графа G тогда и только тогда, когда для любого ребра $e \in E'$ граф $G_e = (V, E \cup E' \setminus \{e\})$ — не хордальный.

если изъятие рёбер $e \in E'$ по отдельности разрушает хордальность, то изъятие нескольких рёбер одновременно также разрушает хордальность

Лемма

Пусть $G = (V, E)$ — хордальный граф, $e \in E$ — произвольное ребро. Тогда следующие условия эквивалентны:

- *граф $G_e = (V, E \setminus \{e\})$ — не хордальный*
- *e является единственной хордой в цикле длины 4*

Минимальные хордальные дополнения

отсюда получаем простую характеризацию минимальных дополнений

E' — минимальное хордальное дополнение если любое ребро $e \in E'$ — единственная хорда цикла длины 4 в хордальном графе $G' = (V, E \cup E')$

приложение минимальных хордальных дополнений — построение сепарирующих клик

Как строить минимальные хордальные дополнения?

среди минимальных дополнений всегда найдётся дополнение, индуцированное заполнением при работе алгоритма Гаусса над факторизацией матрицы для некоторого порядка вершин

Минимальные хордальные дополнения

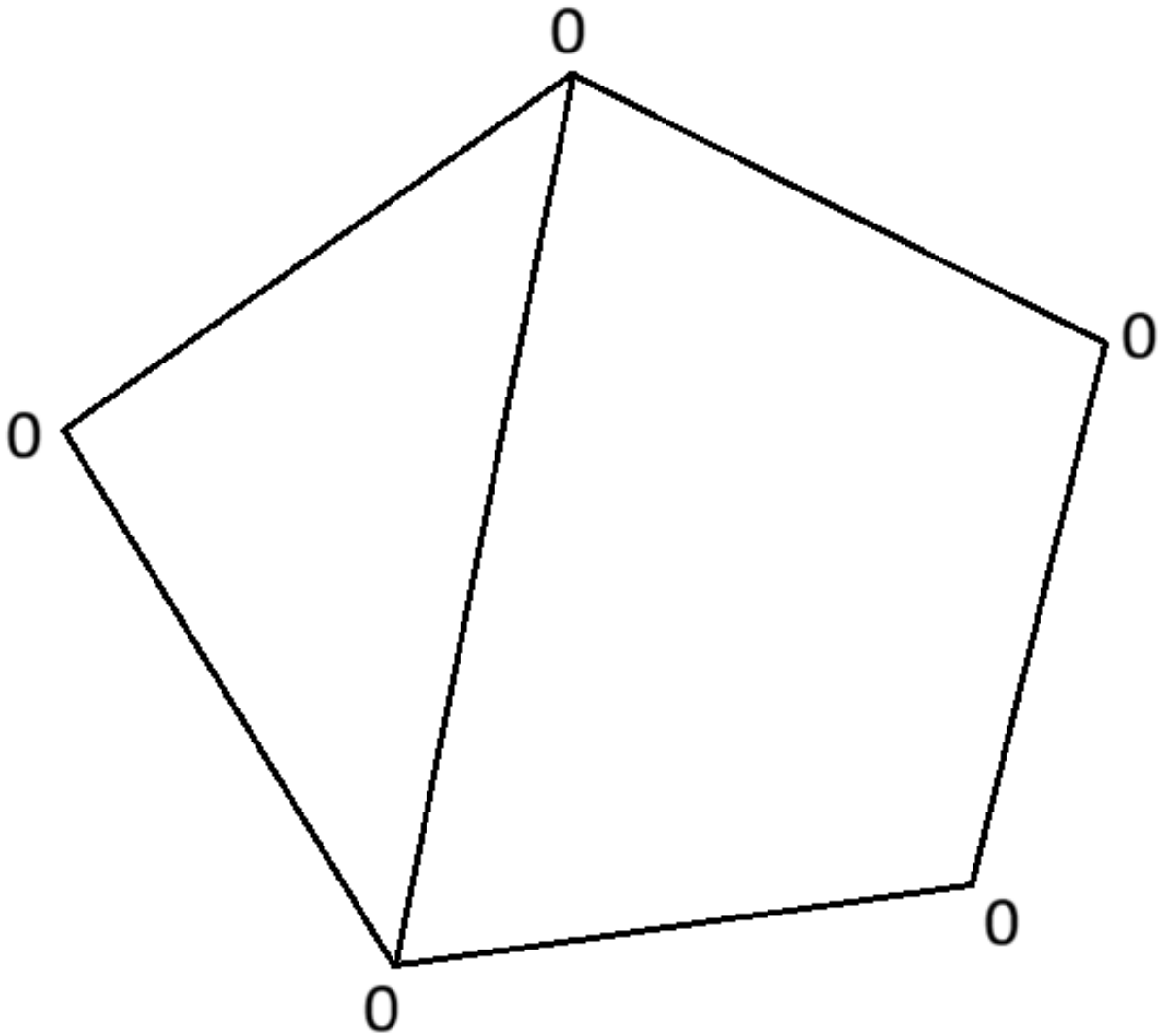
такой порядок вершин находится модификацией лексикографического поиска

маркируем вершины числом m в системе чисел по модулю $n+1$ вида $m = 0, k_1 k_2 \ldots,\ k_j \in \{1, \ldots, n\}$

- все вершины получают маркировку $m(u) = 0$
- for $i = n : -1 : 1$
  - выбираем неупорядоченную вершину v с наибольшей маркировкой $m(v)$
  - присваиваем ей порядок $\alpha(v) = i$
  - у всех неупорядоченных вершин u, связанных с v путём через неупорядоченные вершины w_j с $m(w_j) < m(u)$, добавляем цифру i в маркировку

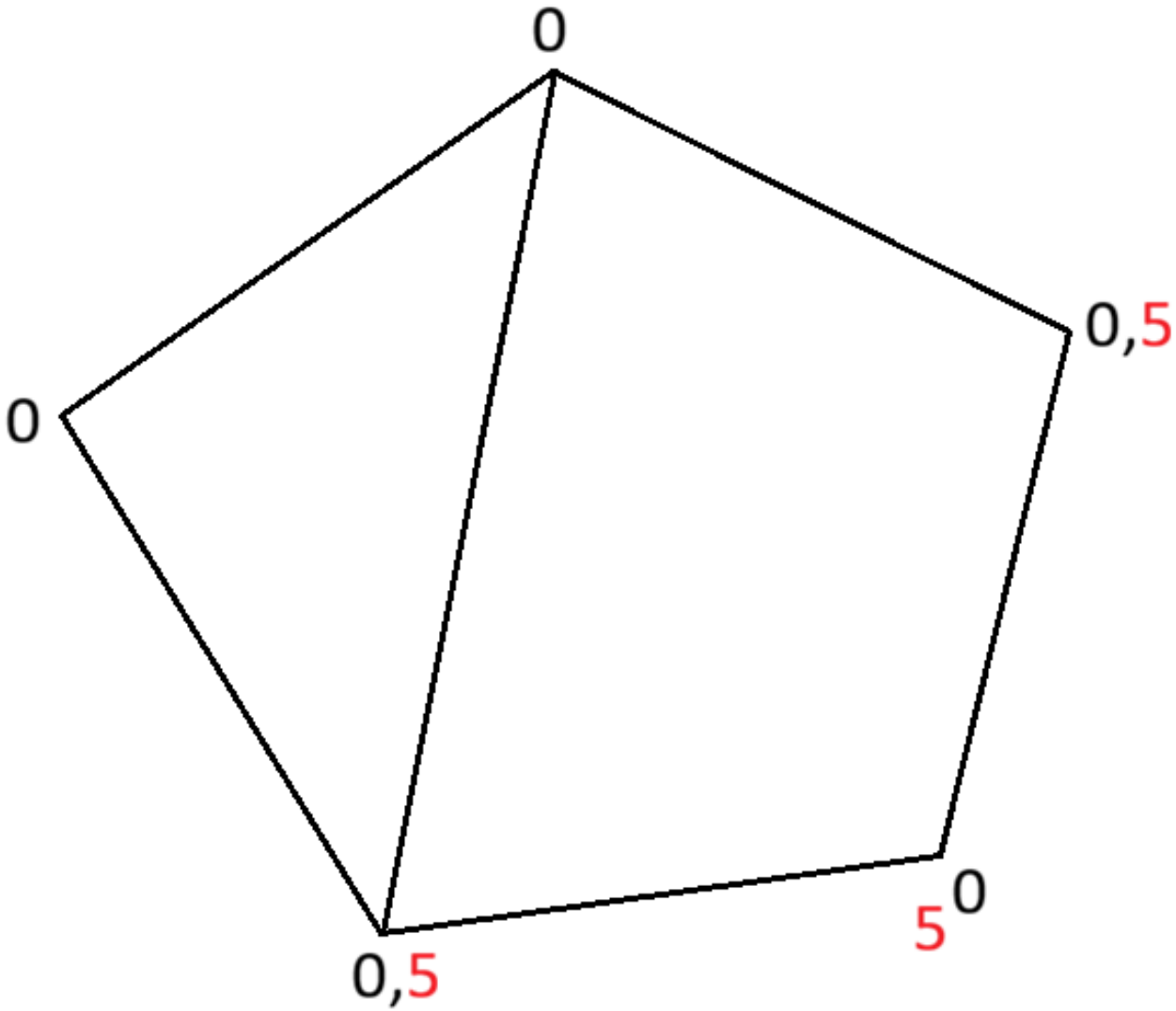
Модифицированный лексикографический поиск

пример



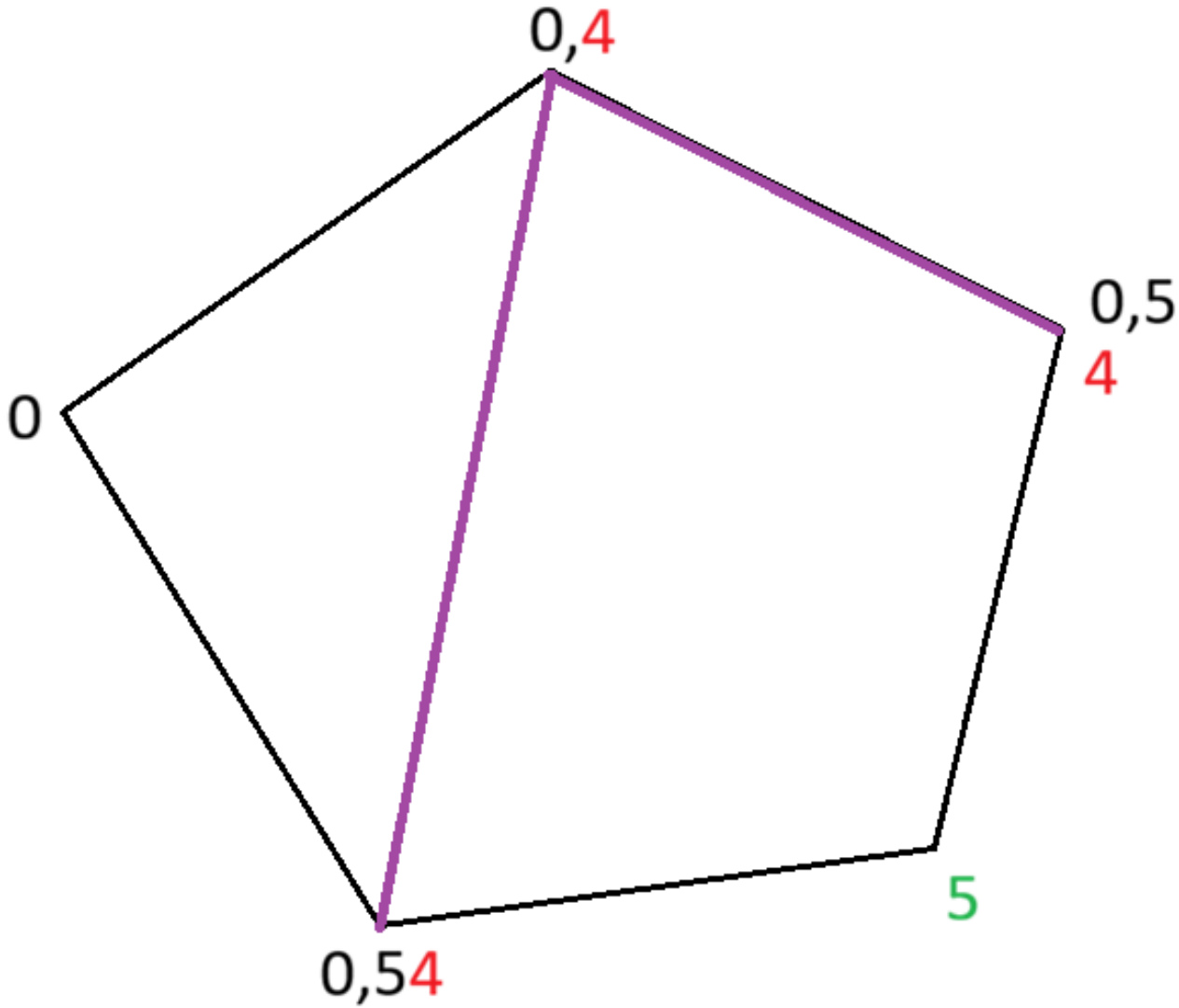
Модифицированный лексикографический поиск

пример



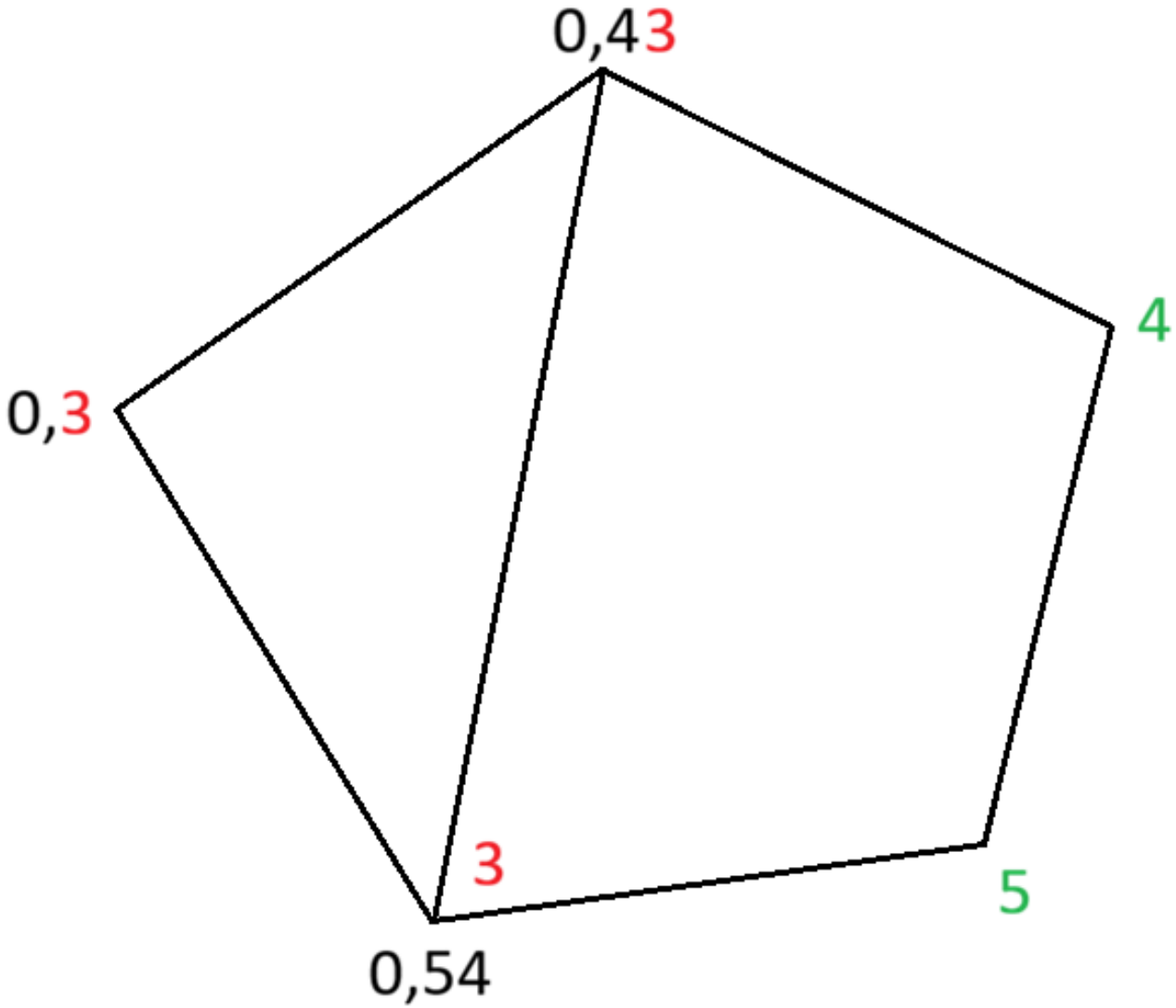
Модифицированный лексикографический поиск

пример



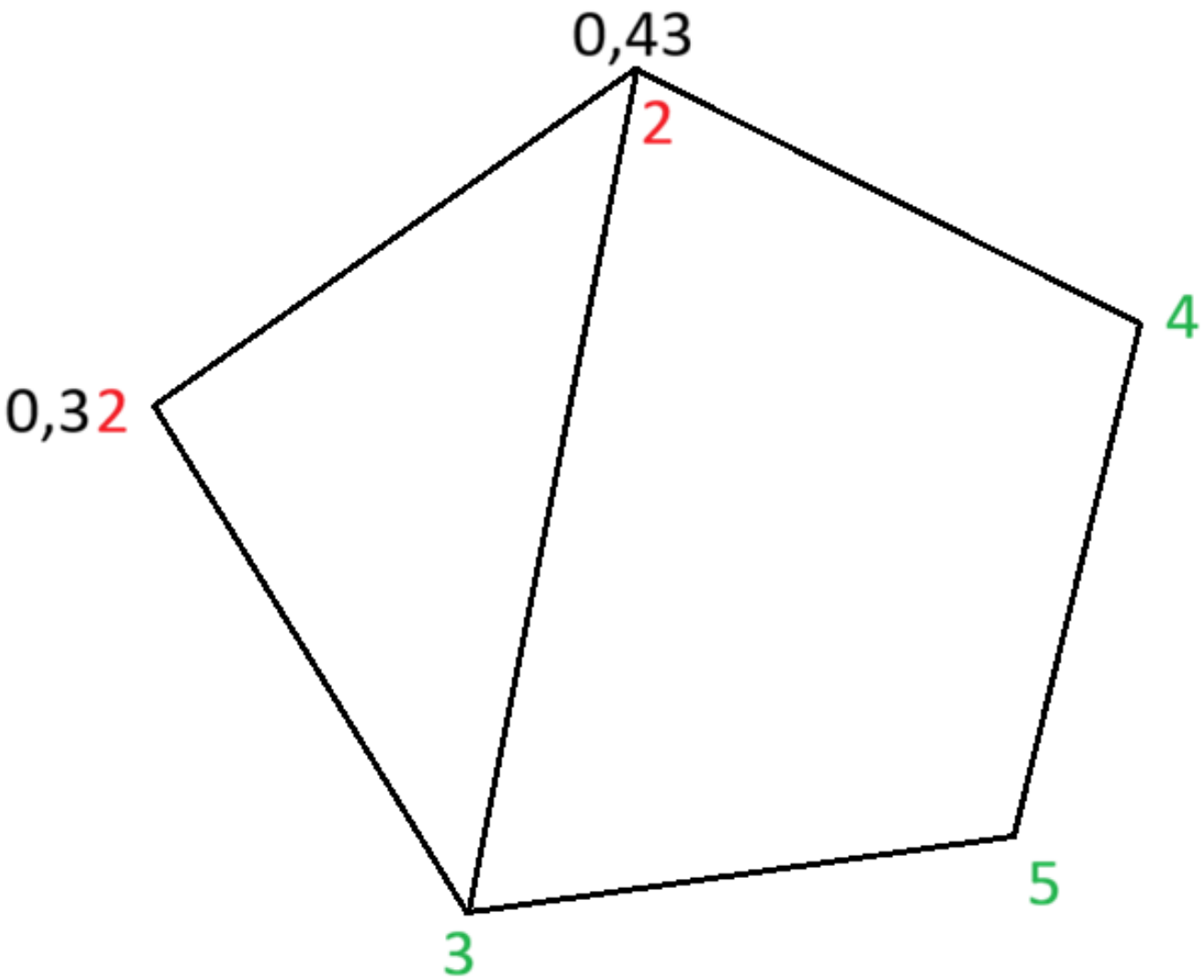
Модифицированный лексикографический поиск

пример



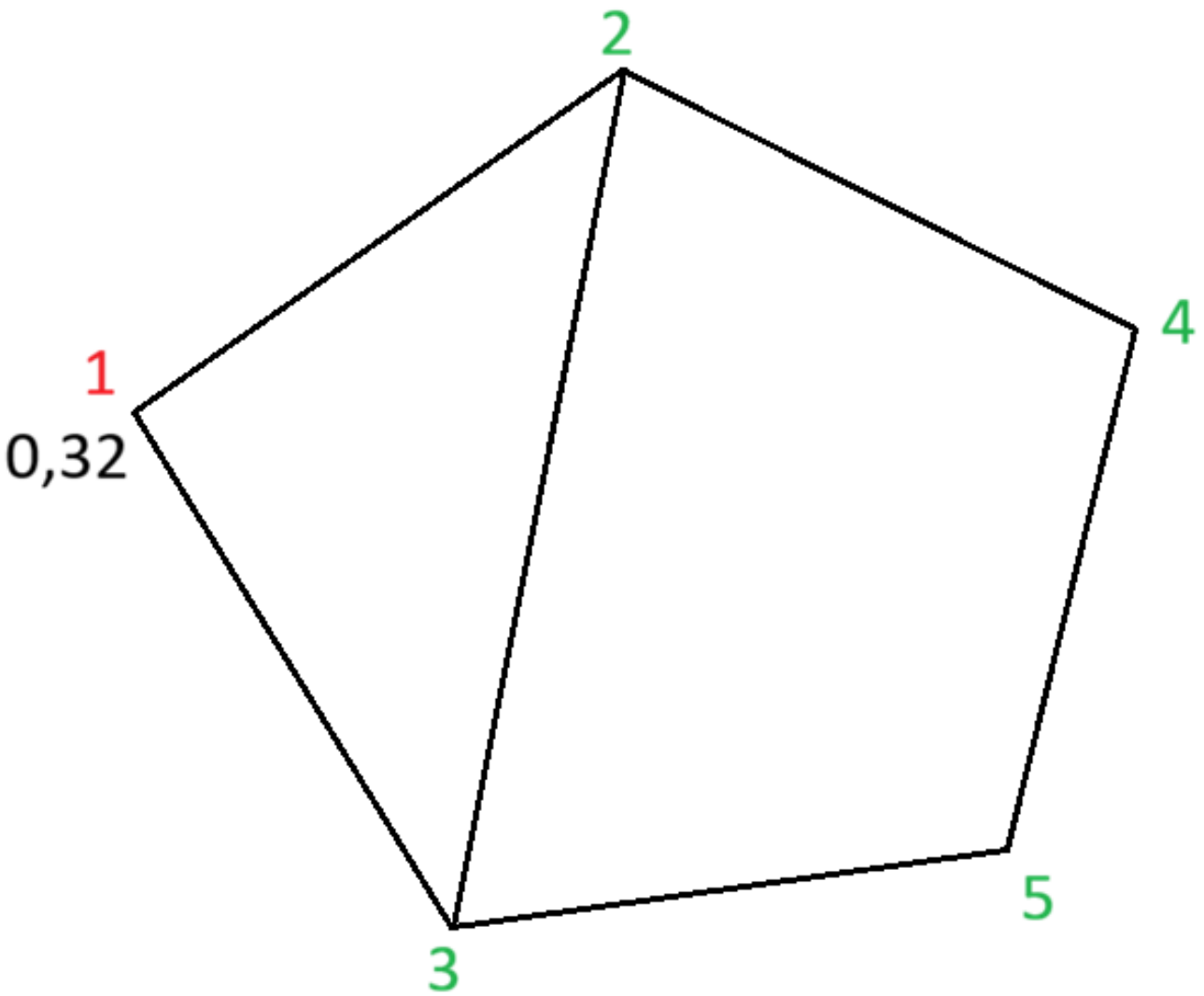
Модифицированный лексикографический поиск

пример



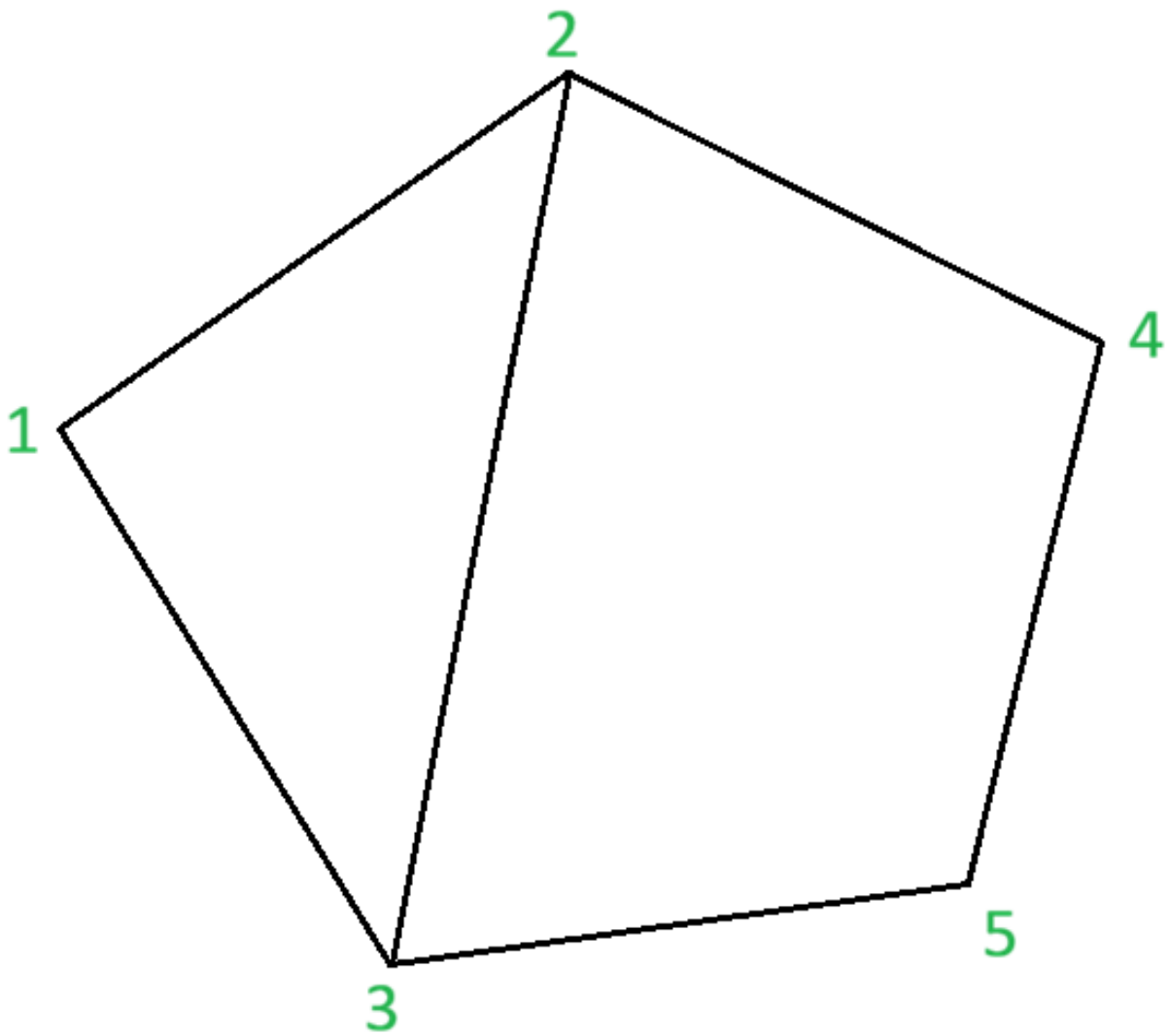
Модифицированный лексикографический поиск

пример



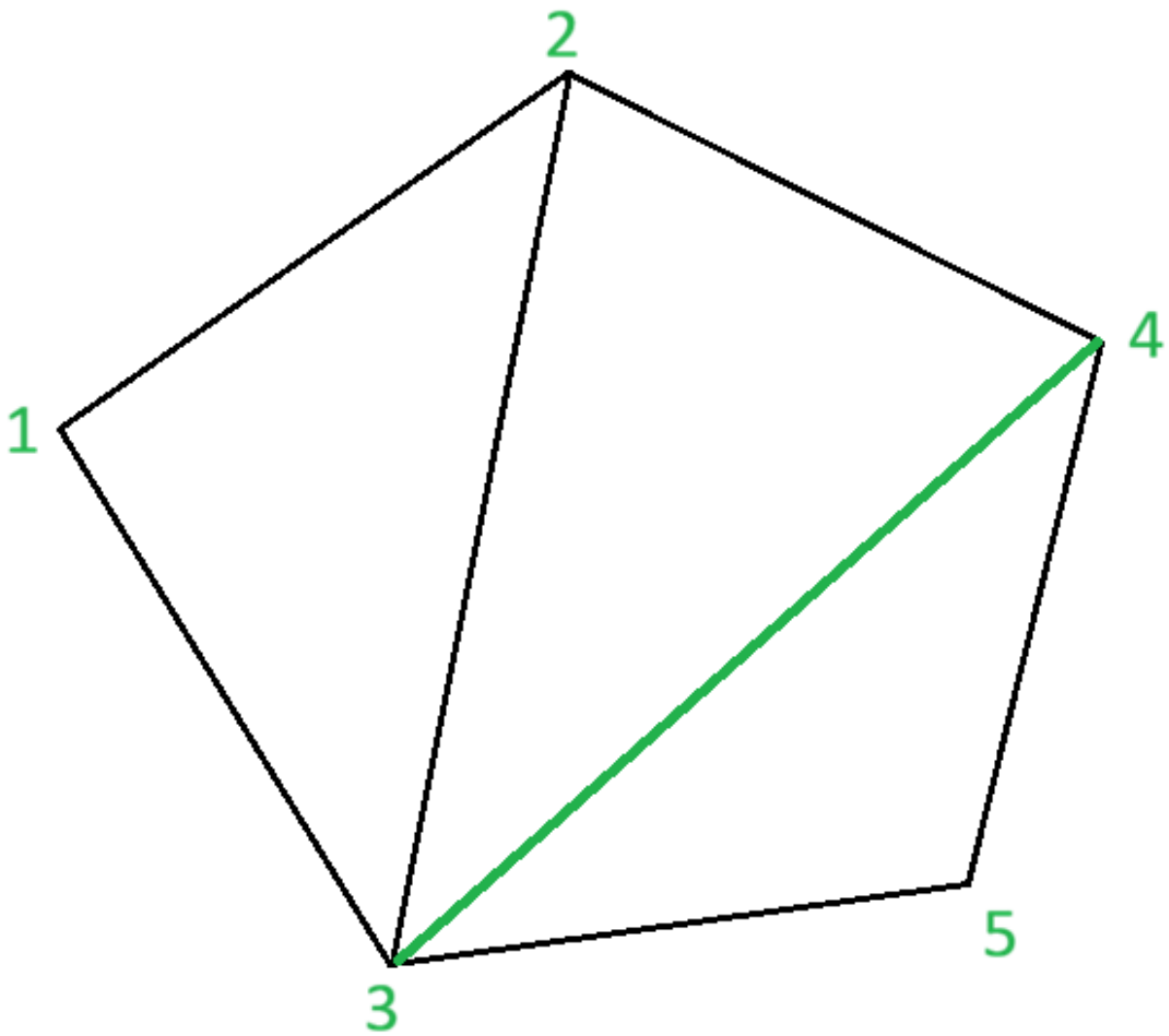
Модифицированный лексикографический поиск

пример



Модифицированный лексикографический поиск

пример



Разложение сепарирующими кликами

Определение

Пусть $G = (V, E)$ — граф. Клика $C \subset V$ называется сепарирующей если индуцированный на $V \setminus C$ подграф — несвязный.

пусть V_i — компоненты связности $V \setminus C$

тогда хордальные дополнения можно искать для подграфов, индуцированных над $V_i \cup C$

эти подграфы могут дальше раскладываться

Разложение сепарирующими кликами

алгоритм Тарьяна

- найти минимиальное хордальное дополнение G' графа G
- найти совершенный порядок исключения α для G'
- for $i = 1 : n$
  - $v = \alpha^{-1}(i)$
  - если клика $C(v) = \{u \mid (u, v) \in E \cup E',\ \alpha(u) > \alpha(v)\}$ — сепарирующая для G, то
    - запомнить компоненту вершины v в $V \setminus C(v)$
    - удалить компоненту из V
- объединить компоненты с соответствующими кликами

алгоритм разлагает граф на неразложимые далее компоненты

Наименьшее хордальное дополнение

алгоритм для нахождения маленького хордального дополнения с помощью разложения сепарирующими кликами

- разложить граф на компоненты сепарирующими кликами
- найти хордальное дополнение для каждой компоненты
- добавить в граф все дополнения — граф станет хордальным
- найти порядок совершенного исключения α для глобального хордального дополнения
- упорядочить исходный граф в порядке α и найти дополнение алгоритмом Гаусса

- итоговое дополнение не больше, чем сумма дополнений на компонентах
- если дополнения на компонентах наименьшие, то итоговое дополнение — наименьшее

План

сложные комбинаторные задачи

- наименьшее хордальное дополнение
- **выполнимость булевых формул**
- задача о сумме подмножества
- задача о рюкзаке

k-Sat

коньюнктивно нормальная форма (КНФ) булевой функции

$$f(x) = (a_{i_1} \vee \cdots \vee a_{i_k}) \wedge \cdots \wedge (a_{j_1} \vee \cdots \vee a_{j_l})$$

где $a_i = \neg x_i$ или $a_i = x_i$

если число литералов в каждой скобке — k, то форма k-КНФ

задача k-SAT состоит в нахождении $x \in \{0,1\}^n$ таких, что $f(x) = 1$, или в доказательстве невозможности выполнить формулу

задача 2-SAT решается за полиномиальное время

задача 3-SAT уже NP сложная

Определение

Пусть $f(x)$ — функция от булевых переменных $x_1, \ldots, x_n$ в коньюнктивно нормальной форме. Пусть каждая скобка c обладает весом $w_c > 0$.
Задача MaxSat состоит в нахождении x такого, что суммарный вес скобок, которые становятся истинными, был максимален.
Если функция в k-КНФ, то задача называется Max-k-Sat.

задача Max-2-Sat — NP трудная

если в скобке встречаются x_i и $\neg x_i$, то она всегда истинна

пусть C_0 — множество таких скобок

если в скобке k литералов a_{i_j} с попарно различными i_j, то при *случайном* выборе x_i (с равной вероятностью $\frac{1}{2}$) вероятность выполнения скобки будет равна $1 - 2^{-k}$

мат. ожидание вклада скобки c с $k(c)$ литералами имеет вид

$$(1 - 2^{-k(c)})w_c$$

итоговое мат. ожидание равно

$$\mathbb{E}W = \sum_{c \in C_0} w_c + \sum_{c \notin C_0} (1 - 2^{-k(c)})w_c$$

MaxSat

рандомизированный алгоритм:

- генерируются случайные значения x, $P(x_i = 0) = P(x_i = 1) = \frac{1}{2}$
- подставляются в скобки формулы
- считается суммарный вес выполненных скобок
- выдаётся x, дающий максимальный вес

можно дерандомизировать алгоритм и гарантировано получить x со значением не менее $\mathbb{E}W$

- for $i = 1 : n$
  - полагаем поочерёдно $x_i = 0$ и $x_i = 1$
  - для каждого значения считаем условное мат. ожидание по $x_{i+1}, \ldots, x_n$
  - фиксируем x_i к тому значению, для которого результат больше

если длина скобок не менее k, то точность аппроксимации алгоритма — не менее $1 - 2^{-k}$

задача формулируется в виде ЦЛП

определим для $c \notin C_0$ множества индексов $S_{c\pm} \subset \{1, \ldots, n\}$

- $i \in S_{c+}$ если x_i присутствует в c
- $i \in S_{c-}$ если $\neg x_i$ присутствует в c

получаем задачу

$$\max_{y,z} \sum_{c \notin C_0} z_c w_c : \quad z_c \geq \sum_{i \in S_{c+}} y_i + \sum_{i \in S_{c-}} (1 - y_i) \; \forall \; c \notin C_0,$$

$$y \in \{0,1\}^n, \quad z \in \{0,1\}^{|C|-|C_0|}$$

заменой включений на $y \in [0,1]^n$, $z \in [0,1]^{|C|-|C_0|}$ получаем линейную релаксацию

рандомизированная процедура

- решается линейная релаксация, оптимальная точка (y^*, z^*)
- генерируются случайные значения x, $P(x_i = 1) = y_i^*$
- подставляются в скобки формулы
- считается суммарный вес выполненных скобок
- выдаётся x, дающий максимальный вес

мат. ожидание веса скобки $c \notin C_0$ с k литералами

$$\mathbb{E}W_c \geq \left(1 - \left(\frac{k-1}{k}\right)^k\right) w_c z_c^*$$

сравнением условных мат. ожиданий по $x_{i+1}, \ldots, x_n$ и фиксацией x_i можно найти значение x такое, что функционал $\geq \mathbb{E}W$

действительно

$$\frac{\mathbb{E}W_c}{w_c} = 1 - \prod_{i \in S_{c-}} y_i^* \cdot \prod_{i \in S_{c+}} (1 - y_i^*)$$

$$\geq 1 - \left(\frac{\sum_{i \in S_{c+}} y_i^* + \sum_{i \in S_{c-}} (1 - y_i^*)}{k} \right)^k$$

$$\geq 1 - \left(\frac{k - z_c^*}{k} \right)^k \geq \left(1 - \left(1 - \frac{1}{k} \right)^k \right) \cdot z_c^*$$

последнее неравенство: в силу вогнутости функции $f(t) = 1 - \left(1 - \frac{t}{k}\right)^k$ и $f(0) = 0$ имеем

$$f(t) \geq t \cdot f(1), \quad t \in [0, 1]$$

гарантия оптимальности с точностью до множителя

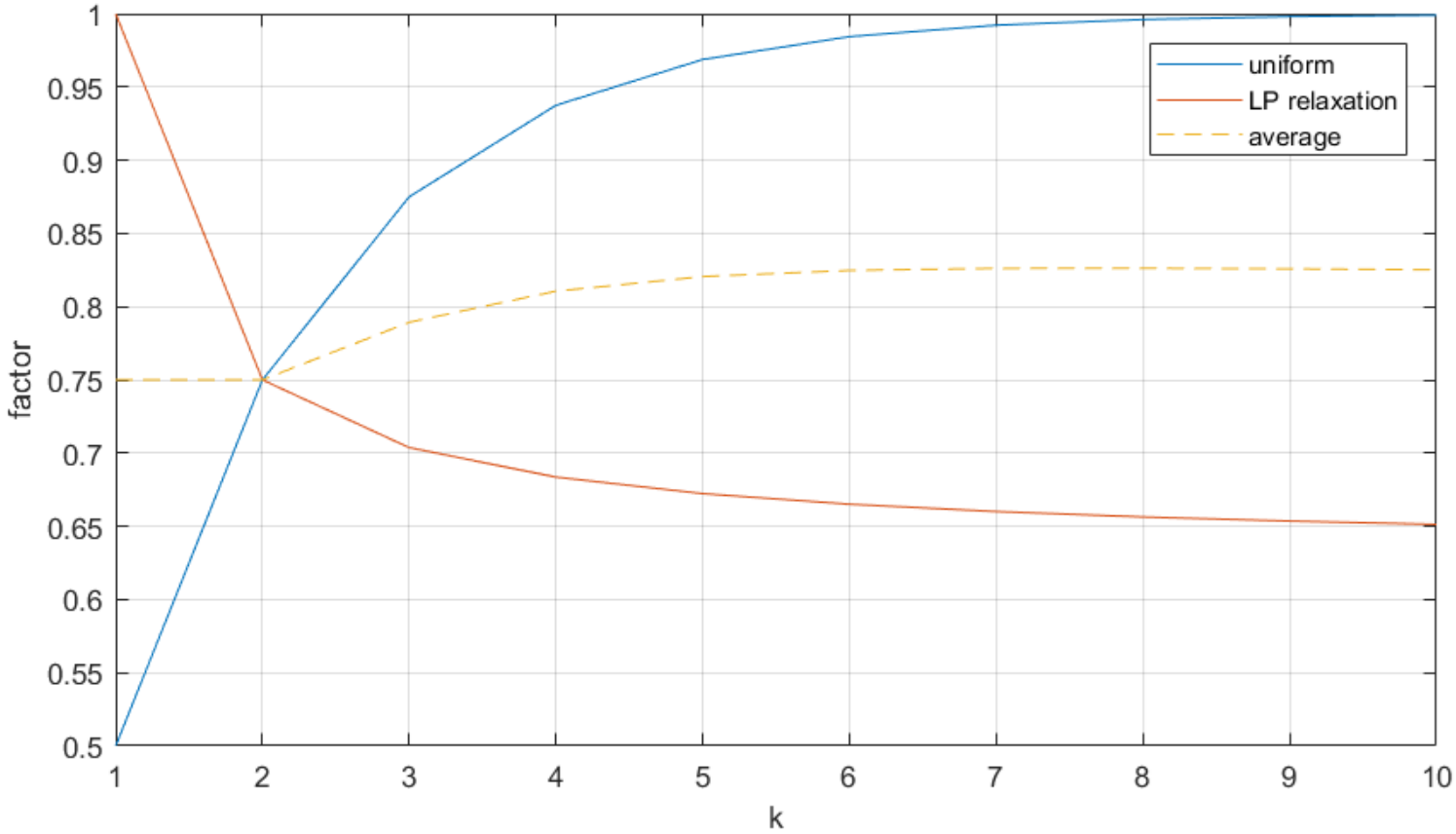
k	$1-2^{-k}$	$1-(1-k^{-1})^k$
1	0,5	1
2	0,75	0,75
3	0,875	0,7037
4	0,9375	0,6836
∞	1	$1-e^{-1}$

равномерное распределение гарантирует фактор $1-2^{-k}$ если длина скобок $\geq k$

ЛП релаксация гарантирует фактор $1-(1-k^{-1})^k$ если длина скобок $\leq k$

факторы гарантированы для каждой скобки в отдельности

MaxSat



равномерное распределение работает хорошо для больших k,
релаксация для маленьких k
среднее для любого k не меньше $\frac{3}{4}$

комбинированный алгоритм

- сгенерировать $\xi = \pm 1$ с равной вероятностью
- в случае $\xi = -1$ запустить рандомизированный алгоритм с равномерным распределением
- в случае $\xi = 1$ запустить рандомизированный алгоритм с распределением, заданным решением релаксации

тогда

$$\mathbb{E}W = \frac{\mathbb{E}_{uni}W + \mathbb{E}_{LP}W}{2} \leq \max(\mathbb{E}_{uni}W, \mathbb{E}_{LP}W)$$

в итоге получаем детерминированный алгоритм

- найти x_{uni} со значением $\geq \mathbb{E}_{uni}W$
- найти x_{LP} со значением $\geq \mathbb{E}_{LP}W$
- выбрать x с лучшим значением

гарантированный фактор точности — $\frac{3}{4}$

неулучшаемость результатов

Теорема (Hastad 1997)

Если $P \neq NP$, $\epsilon > 0$, то нет алгоритма полиномиальной сложности, решающий задачу Max2Sat с точностью $\geq \frac{21}{22} + \epsilon$.

Теорема (Hastad 1997)

Если $P \neq NP$, $\epsilon > 0$, то нет алгоритма полиномиальной сложности, решающий задачу Max3Sat с точностью $\geq \frac{7}{8} + \epsilon$.

точность $\frac{7}{8}$ достигается подстановкой равномерно распределённых случайных значений

План

сложные комбинаторные задачи

- наименьшее хордальное дополнение
- выполнимость булевых формул
- **задача о сумме подмножества**
- задача о рюкзаке

Задача о сумме подмножества

пусть даны n объектов с весами $w_i \in \mathbb{N}_+$ и суммой $W = \sum_{i=1}^{n} w_i$

задача о сумме подмножества состоит в том, чтобы разделить объекты на два дизъюнктных подмножества S, T с равными суммарными весами в случае чётного W и разницей 1 в случае нечётного W

формулировка ЦЛП

$$\min_{x \in \{-1,+1\}^n, t} t : \quad -t \leq \sum_{i=1}^{n} x_i w_i \leq t$$

если оптимальное значение задачи — 0 или 1, то разложение существует и задаётся

$$S = \{i \mid x_i = -1\}, \quad T = \{i \mid x_i = +1\}$$

Задача о сумме подмножества

фазовый переход

- если $w_i \sim a$ и $n \gg \log_2 a$, то решение с высокой вероятностью существует
- если $w_i \sim a$ и $n \ll \log_2 a$, то решения с высокой вероятностью не существует

задачи, в которых $\log_2 a \approx n$, самые сложные

Теорема (Borgs, Chayes, Pittel 2001)

Пусть w_i генерируются независимо с равномерным распределением на интервале $[1, 2^m]$. Пусть P — вероятность существования разбиения. Пусть $m, n \to +\infty$ и $\kappa = \lim_{n\to+\infty} \frac{m}{n}$. Тогда

$$\lim_{n\to+\infty} P = \begin{cases} 1, & \kappa < 1 \\ 0, & \kappa > 1 \end{cases}$$

Задача о сумме подмножества

можно более точно ограничить размер окна

Теорема (Borgs, Chayes, Pittel 2001)

Пусть w_i генерируются независимо с равномерным распределением на интервале $[1, 2^m]$. Пусть P — вероятность существования разбиения. Пусть $m, n \to +\infty$ и $m = n - \frac{\log_2 n}{2} + \lambda_n$, $\lambda = \lim_{n\to+\infty} \lambda_n$. Тогда

$$\lim_{n\to+\infty} P = \begin{cases} 1, & \lambda = -\infty \\ 1 - \frac{r(r+1)}{2}, & \lambda \in \mathbb{R} \\ 0, & \lambda = +\infty \end{cases}$$

где $r = \exp(-\sqrt{\frac{3}{2\pi}} 2^{-\lambda})$.

Экскурс: фазовый переход

пусть Z — число *оптимальных* разбиений (т.е., минимизирующих функционал в формулировке ЦЛП)

энтропия определяется как $\log_2 Z$, энтропия на переменную как $s = \frac{\log_2 Z}{n}$

тогда, если $\kappa = \lim_{n \to +\infty} \frac{m}{n}$ существует, то

$$\lim_{n \to +\infty} s = \max(0, 1 - \kappa)$$

по распределению

производная по κ разрывна — фазовый переход 1-го порядка

Экскурс: фазовый переход

рассмотрим задачу 2-SAT с n переменными и m скобками

литералы в скобках генерируются случайно с равномерным распределением по $x_i, \neg x_i$

пусть $m, n \to +\infty$, $\frac{m}{n} \to \alpha$

тогда с вероятностью $\to 1$

- формула не выполняется если $\alpha > 1$
- формула выполняется если $\alpha < 1$

более точный результат: $m = n + \lambda n^{2/3}$

- при $\lambda \ll 0$ вероятность невыполнения убывает как $O(|\lambda|^{-3})$
- при $\lambda \gg 0$ вероятность выполнения убывает как $e^{-O(\lambda^3)}$

Задача о сумме подмножества

алгоритм Хоровица-Сани (чётные суммы)

- разбить множество $\{1, \ldots, n\}$ на два подмножества I_1, I_2 по $\approx \frac{n}{2}$ элементов
- посчитать все суммы вида $\sum_{i \in I \subset I_1} w_i$, $\sum_{i \in I \subset I_2} w_i$
- упорядочить первое множество сумм s_k^1 по возрастанию в список S_1
- упорядочить второе множество сумм s_k^2 по убыванию в список S_2
- начиная с $k_1 = k_2 = 1$, сравнивать сумму $s = s_{k_1}^1 + s_{k_2}^2$ с $\overline{w}$:
- если $s < S$, $k_1 \leftarrow k_1 + 1$
- если $s > S$, $k_2 \leftarrow k_2 + 1$
- если $s = S$, решение найдено

Задача о сумме подмножества

пример: $n = 4,\ w = (13, 6, 11, 2),\ \frac{W}{2} = 16$

- разбиваем на два подмножества: $w_{I_1} = (13, 6),\ w_{I_2} = (11, 2)$
- суммы в первом подмножестве: 0,13,6,19
- суммы во втором подмножестве: 0,11,2,13
- упорядочиваем первое множество сумм по возрастанию: $S_1 = (0, 6, 13, 19)$
- упорядочиваем второе множество сумм по убыванию: $S_2 = (13, 11, 2, 0)$

Задача о сумме подмножества

16	0	6	13	19
13				
11				
2				
0				

Задача о сумме подмножества

16	0	6	13	19
13	13			
11				
2				
0				

Задача о сумме подмножества

16	0	6	13	19
13	13	19		
11				
2				
0				

Задача о сумме подмножества

16	0	6	13	19
13	13	19		
11		17		
2				
0				

Задача о сумме подмножества

16	0	6	13	19
13	13	19		
11		17		
2		8		
0				

Задача о сумме подмножества

16	0	6	13	19
13	13	19		
11		17		
2		8	15	
0				

Задача о сумме подмножества

16	0	6	13	19
13	13	19		
11		17		
2		8	15	21
0				

Задача о сумме подмножества

16	0	6	13	19
13	13	19		
11		17		
2		8	15	21
0				19

Задача о сумме подмножества

$S_1 = (0, 6, 13, 19)$, $S_2 = (13, 11, 2, 0)$

- в S_1 найти наименьшее число, не меньшее, чем $16 - 13 = 3$
- в S_2 найти наибольшее число, не превышающее $16 - 6 = 10$
- в S_1 найти наименьшее число, не меньшее, чем $16 - 2 = 14$
- в S_2 найти наибольшее число, не превышающее $16 - 19 = -3$

для S_1 целевые пороги растут, для S_2 убывают

16	0	6	13	19
13	13	19		
11		17		
2		8	15	21
0				19

Задача о сумме подмножества

алгоритм Шрёппеля-Шамира

ищет нужные элементы в S_1, S_2, не строя всё множество S_i

разделим I_1 на подмножества I_{11}, I_{12} по $\approx \frac{n}{4}$ элемента

пусть S_{1j} содержит все суммы подмножеств элементов из I_{1j}

тогда элемент из S_1 есть сумма одного элемента из S_{11} и одного элемента из S_{12}

- упорядочиваем только элементы из S_{11}
- для каждого $r \in S_{12}$ сканируем последовательность $r + S_{11}$

требует $O(2^{n/4})$ памяти вместо $O(2^{n/2})$

Задача о сумме подмножества

пример:

$w_{l_{11}} = \{13\}$, $w_{l_{12}} = \{6\}$
$S_{11} = (0, 13)$, $S_{12} = \{0, 6\}$
целевые пороги: 3,14

3	0	13
0		13
6	6	

14	0	13
0		13
6	6	19

каждую строку сканируем только с предыдущего элемента

План

сложные комбинаторные задачи

- наименьшее хордальное дополнение
- выполнимость булевых формул
- задача о сумме подмножества
- **задача о рюкзаке**

Задача о рюкзаке

даны объекты $1, \ldots, n$ с весами w_i и стоимостями v_i

задача о рюкзаке состоит в нахождении подмножества объектов (рюкзака) с максимальной суммарной стоимостью и с суммарным весом, не превосходящим порог $\bar{w}$

формулировка в виде ЦЛП:

$$\min_{x \in \{0,1\}^n} \langle v, x \rangle : \quad \langle w, x \rangle \leq \bar{w}$$

бинарная переменная $x_i \in \{0, 1\}$ индицирует принадлежность объекта i к рюкзаку

Задача о рюкзаке: жадный алгоритм

жадный алгоритм:

- упорядочить объекты по убывающей удельной стоимости $\frac{v_i}{w_i}$
- инициализировать пустой рюкзак
- по одному добавлять объекты, если они умещаются

первый объект, который *не умещается*, называется *критическим*

в конце выбирается наиболее ценная из альтернатив: построенный рюкзак — только критический объект

гарантирована точность с множителем 2: оптимальное значение строго меньше суммы значений двух альтернатив

Задача о рюкзаке: алгоритм Сани

для любого $k \in \mathbb{N}_+$ существует алгоритм со сложностью $O(n^{k+1})$ гарантирующий точность с множителем лучше $\frac{k}{k+1}$ [Sahni 1975]

пусть I_{opt} — оптимальный рюкзак, а I_1 — множество индексов k объектов в нём с максимальными стоимостями v_i

построение субоптимального рюкзака $\hat{I}$

- положим объекты из I_1 в рюкзак $\hat{I}$
- упорядочим остальные по удельной стоимости
- добавляем один за другим объекты, пока они входят

пусть j — индекс первого объекта из $I_{opt} \setminus \hat{I}$

определим индексные подмножества

- I_2 — объекты из оптимального рюкзака, попавшие в $\hat{I}$ до j
- $I_3 = I_{opt} \setminus (I_1 \cup I_2)$
- I_4 — объекты не из I_{opt}, попавшие в $\hat{I}$ до j

Задача о рюкзаке: алгоритм Сани

пусть

- v_{opt} — оптимальное значение
- $\hat{v}$ — значение построенного рюкзака
- v^j — стоимость объектов из I_j
- w^j — вес объектов из I_j

ошибка ограничивается v_j, но это — малая доля общей стоимости

а именно,

- $v_{opt} \geq (k+1)v_j$
- $w^1 + w^2 + w^4 + w_j > \bar{w} \geq w^1 + w^2 + w^3$
- $\frac{v^4}{w^4} \geq u_j \geq \frac{v^3}{w^3}$
- $v_{opt} - \hat{v} \leq v^3 - v^4 \leq u_j(w^3 - w^4) < u_j w_j = v_j$
- $\frac{v_{opt} - \hat{v}}{v_{opt}} < \frac{1}{k+1}$

Задача о рюкзаке: динамическое программирование

подход с помощью динамического программирования

пусть $v(w, k)$ — оптимальное значение рюкзака при пороге $\bar{w} = w$ и наличии только первых k объектов

тогда

$$v(w, k) = \max(v(w, k-1), v(w - w_k, k-1) + v_k)$$

с начальным условием

$$v(w, 0) = \begin{cases} 0, & w \geq 0 \\ -\infty, & w < 0 \end{cases}$$

(либо кладём объект $k+1$, либо нет)

функция $v(w, k)$ — кусочно-постоянна

достаточно посчитать функцию на интервале $[0, \bar{w}]$

Задача о рюкзаке: гибридный алгоритм

алгоритм Ибарры-Кима использует жадный подход и динамическое программирование

пусть $\epsilon > 0$, $v_{opt} \leq \overline{v} \leq 2v_{opt}$

поделим объекты на

- дорогие: $v_i \geq \frac{\epsilon}{3}\overline{v}$
- дешёвые: $v_i < \frac{\epsilon}{3}\overline{v}$

Задача о рюкзаке: гибридный алгоритм

пусть

$$\delta = (\frac{\epsilon}{3})^2 \overline{v}, \quad q = \left\lfloor (\frac{3}{\epsilon})^2 \right\rfloor = \left\lfloor \frac{\overline{v}}{\delta} \right\rfloor$$

стоимость дорогих объектов отшкалируем и округлим вверх:

$$v_i \leftarrow \xi_j = \left\lfloor \frac{v_j}{\delta} \right\rfloor$$

тогда $\delta \cdot \xi_j \leq v_j < \delta \cdot (\xi_j + 1)$

любой допустимый рюкзак имеет суммарную стоимость $\leq v_{opt}$

стоимость дорогих объектов в нём $\leq v_{opt} \leq \overline{v}$
их не более $\frac{3}{\epsilon} \cdot \frac{v_{opt}}{\overline{v}} \leq \frac{3}{\epsilon}$

тогда

$$\sum_i \xi_i \leq \lfloor \frac{\overline{v}}{\delta} \rfloor = q$$

суммарная ошибка округления $\leq \frac{\epsilon}{3}$

Задача о рюкзаке: гибридный алгоритм

схема алгоритма

- упорядочивание объектов по удельной стоимости u_i
- фаза динамического программирования — оптимизация дорогих объектов
- фаза жадного алгоритма — оптимизация дешёвых объектов
- выбор лучшего из найденных решений

динамическое программирование: для каждой стоимости $\sum_i \xi_i = \xi \in \{0, \ldots, q\}$ определяется рюкзак наименьшего веса — $O(nq)$ операций

к каждому полученному рюкзаку добавляются дешёвые объекты жадным алгоритмом — $O(nq)$ операций

ошибка при распределении дешёвых объектов ограничивается критическим объектом — будет меньше $\frac{2\epsilon}{3}$