Дискретная оптимизация. Осень 2015.[1]

Введение

На лекции мы вспомнили/узнали о следующих классических задачах дискретной оптимизации:

(1) покрытие множествами,
(2) вершинное покрытие,
(3) кратчайший путь,
(4) минимальное остовное дерево (MST),
(5) паросочетание наибольшей мощности,
(6) паросочетание наибольшего веса,
(7) паросочетание наибольшей мощности и наименьшей стоимости,
(8) задача о назначениях,
(9) (одномерная) задача об упаковке в контейнеры (bin packing),
(10) задача о 0,1-рюкзаке,
(11) нахождение потока наибольшей величины,
(12) нахождение потока заданной величины и наименьшей стоимости,
(13) мультипродуктовый поток максимальной суммарной величины,
(14) задача коммивояжёра (TSP).

Далее будем считать, что все веса и стоимости в задачах (1)–(14) суть целые числа.

Задача 1. Задачу дискретной оптимизации в общем виде можно представить в форме: «найти точку максимума/минимума функционала f на конечном множестве $\mathcal{S}$». Для каждой из задач (1)–(14) проделайте следующее.

(a) Укажите соответствующие «разумные» множество $\mathcal{S}$ и функционал f (ответ может быть неоднозначным). Обратите внимание, что выбор пары $\mathcal{S}$, f неоднозначен, его можно осуществлять по-разному: ценой более сложного устройства $\mathcal{S}$ можно уменьшить сложность f и наоборот.

(b) Определите, сколько асимптотически времени требуется, чтобы вычислить значение функционала f в фиксированной точке множества $\mathcal{S}$.

(c) Определите, как мощность множества $\mathcal{S}$ зависит от «размера» содержательной задачи (например, в задаче TSP это количества вершин и рёбер графа, а также диапазон весов рёбер (ведь на запись чисел тоже требуется место)), — полиномиально/линейно/экспоненциально/сверхэкспоненциально.

(d) Оцените максимальный размер задачи, которую можно решить за сутки полным перебором, если на рассмотрение каждого элемента множества $\mathcal{S}$ требуется 10^{-9} с.

Задача 2. Рассмотрим следующие задачи:

1. Для заданного полного n-вершинного графа G с целочисленными положительными весами на рёбрах *найти гамильтонов цикл* наименьшей стоимости в G. (Стандартная формулировка оптимизационной задачи TSP. Это т. н. *search variant* задачи.)
2. Для заданного полного n-вершинного графа G с целочисленными положительными весами на рёбрах *найти минимальную стоимость* гамильтонова цикла в G (при этом сам цикл, имеющий такую стоимость, указывать необязательно). (Вместо поиска точки минимума функционала ищем значение. Это т. н. *evaluation variant* задачи.)
3. Для заданного полного n-вершинного графа G с целочисленными положительными весами на рёбрах и *заданного* числа $C_0 \in \mathbb{Z}$ определить, *существует ли* в G гамильтонов цикл, имеющий стоимость не более C_0. (Задача с ответом «да/нет», так называемая *задача распознавания*. Это т. н. *decision variant* задачи.)

Докажите следующие импликации.

(a) Пусть существует алгоритм, решающий третью задачу за время $T(n)$. Тогда существует алгоритм, решающий вторую задачу за время $O\left(T(n) \cdot \log C_{\min}\right)$, где $C_{\min}$ — минимальная стоимость гамильтонова цикла в G.

(b) Пусть существует алгоритм, решающий вторую задачу за время $T(n)$. Тогда существует алгоритм, решающий первую задачу за время $O(T(n) \cdot n^2)$.

Предложите аналогичные постановки для задач о 0,1-рюкзаке, упаковке в контейнеры и вершинном покрытии; докажите аналогичные импликации.

Хотя множество $\mathcal{S}$ в оптимизационной задаче может быть очень большим, описать его обычно можно компактно (например, в задаче TSP множество $\mathcal{S}$ — все гамильтоновы циклы графа, но чтобы полностью определить $\mathcal{S}$, достаточно задать только сам граф). Под входными данными [алгоритма решения] оптимизационной задачи будем подразумевать именно их компактное описание.

Задача 3. Будем говорить, что задача $\mathcal{A}$ *полиномиально сводится* к задаче $\mathcal{B}$, если существует алгоритм, который преобразует за полиномиальное время входные данные задачи $\mathcal{A}$ во входные данные задачи $\mathcal{B}$, так, что по ответу в задаче $\mathcal{B}$ можно за полиномиальное время вычислить ответ задачи $\mathcal{A}$. Определите, какие из задач (1)–(14) полиномиально сводятся друг к другу. Этот вид сводимости является промежуточным между *сводимостью по Карпу*, при которой ответ в задаче $\mathcal{B}$ должен *совпадать* с ответом в задаче $\mathcal{A}$, и *сводимостью по Тьюрингу*, она же *сводимость по Куку*. Последняя определяется так: говорят, что задача $\mathcal{A}$ *полиномиально сводится (по Тьюрингу)* к задаче $\mathcal{B}$, если существует полиномиальный алгоритм, решающий задачу $\mathcal{A}$, который может обращаться к волшебной процедуре-оракулу, разрешающей задачу $\mathcal{B}$ за один такт времени. Поясните, почему из сводимости по Карпу следует сводимость по Тьюрингу.

Задача 4. Напомним, что в задаче о 0,1-рюкзаке для каждого из предметов нужно определить, включать его в рюкзак или нет. В более общей задаче о целочисленном рюкзаке имеется несколько различных наименований предметов и известно, сколько предметов каждого наименования у нас есть (максимально возможные «кратности»). Требуется, как и в задаче о 0,1-рюкзаке, максимизировать суммарную ценность попавших в рюкзак предметов. *(a)* Поставьте эту задачу математически. *(b)* Докажите, что задача о целочисленном рюкзаке полиномиально сводится к задаче о 0,1-рюкзаке.

Локальный поиск и модификация Кернигана—Лина

Задача 5. *(a)* При рассмотрении 3-окрестности гамильтонова цикла при локальном поиске в задаче TSP мы удаляем 3 ребра из текущего цикла и пытаемся провести три новых ребра. Перечислите все возможные варианты добавления новых рёбер, так, что снова получается гамильтонов цикл.

[1]Автор: А. Б. Дайняк. Дата последнего обновления: 20 декабря 2015 г. в 23:21:00.

(b) Пусть теперь мы используем в локальном поиске k-окрестности. Сколькими (в точности) способами можно провести k новых рёбер для получения нового цикла (при условии, что мы уже удалили из текущего цикла k фиксированных рёбер)?

Задача 6. Рассмотрим задачу о вершинном покрытии. Пусть $\mathcal{S}$ — семейство всех подмножеств вершин графа G, образующих в G вершинное покрытие, и мы ищем множество из $\mathcal{S}$, имеющее минимальную мощность. Для $x \in \mathcal{S}$ положим $\mathcal{N}(x) := \{x' \in \mathcal{S} \mid x' \subset x \text{ и } |x'| = |x| - 1\}$. Докажите, что относительно такой системы окрестностей может экспоненциально много локальных оптимумов, например, когда G является простой цепью. Сколько в этом случае глобальных оптимумов?

Задача 7. Приведите примеры сильно связных полиномиально обозримых систем окрестностей для следующих задач: *(a)* вершинное покрытие, *(b)* сбалансированное разбиение графа, *(c)* покрытие множествами, *(d)* 0,1-рюкзак.

Задача 8. Докажите сильную связность системы 2-окрестностей в задаче TSP: покажите, что для любых двух гамильтоновых циклов C', C'' в полном графе можно найти последовательность гамильтоновых циклов $C_1, C_2, \ldots, C_k$, такую, что $C_1 = C'$, $C_k = C''$, и для каждого i цикл C_{i+1} принадлежит 2-окрестности цикла C_i.

Задача 9. Определим k-окрестность гамильтонова цикла C в графе как множество всех циклов, которые можно получить удалением k рёбер из C и добавлением k рёбер (необязательно отличных от бывших). На лекции было показано, что рассмотрение 2-окрестностей не гарантирует достижение глобального оптимума в задаче коммивояжёра при локальном поиске.

(a) Докажите, что система 3-окрестностей не является точной (существуют локальные оптимумы, не являющиеся глобальными). (Внимание! В примерах-картинках вес ребра вовсе не обязан быть пропорциональным длине ребра.)

(b) Докажите, что система $\lceil n/2 \rceil$-окрестностей не является точной, где n — количество вершин в графе.

(c) Докажите, что система $(n-1)$-окрестностей является точной.

Задача 10. Пусть t — фиксированное натуральное число. Будем рассматривать в локальном поиске в задаче TSP уменьшенные 2-окрестности, в которых из цикла удаляется-добавляется только пара рёбер, удалённых друг от друга по циклу на расстояние не больше t. Будет ли такая система окрестностей слабее, чем система обычных 2-окрестностей (то есть, может ли цикл быть локально-оптимальным относительно системы уменьшенных 2-окрестностей, но не являться при этом локально-оптимальным относительно системы обычных 2-окрестностей)?

Задача 11. Оцените время работы итерации обычного локального поиска в задаче о сбалансированном разбиении графа при использовании k-окрестностей (обмен не более k пар вершин между частями разбиения) и сравните это со временем работы локального поиска переменной глубины (Керниган–Лин) с использованием на каждом элементарном шаге замен одной вершины на одну вершину и глубиной не более k.

Задача 12. Рассмотрим задачу о сбалансированном разбиении графа с весами на рёбрах, в которой части разбиения должны быть равными ($\alpha = \frac{1}{2}$). Напомним, что в этой задаче k-окрестностью разбиения мы называем семейство всех таких разбиений, которые можно получить перебросом в общей сложности не более чем $2k$ вершин между частями.

(a) Верно ли, что для любого $4n$-вершинного графа обычный локальный поиск при использовании n-окрестностей всегда найдёт оптимальное разбиение?

(b) Придумайте пример такого $4n$-вершинного графа и такого начального разбиения этого графа, на котором поиск Кернигана—Лина глубины не более n находит оптимальное решение, а обычный локальный поиск при использовании $(n-1)$-окрестностей не найдёт оптимума.

(c) Придумайте пример $4n$-вершинного графа, на котором поиск Кернигана–Лина глубины не более n может не найти оптимум, а обычный локальный поиск при использовании n-окрестностей всегда находит оптимум.

(d) Придумайте пример $4n$-вершинного графа, на котором поиск Кернигана–Лина глубины не более $2n$ (который при построении цепочек переходов использует 1-окрестности) всегда находит оптимальное решение, а обычный локальный поиск при использовании $(n-1)$-окрестностей может не найти оптимума. Сравните вычислительную сложность соответствующих алгоритмов.

Задача 13. Рассмотрим задачу о сбалансированном разбиении графа с весами на рёбрах, в которой части разбиения должны быть равными ($\alpha = \frac{1}{2}$). Напомним, что в этой задаче k-окрестностью разбиения мы называем семейство всех таких разбиений, которые можно получить перебросом в общей сложности не более чем $2k$ вершин между частями. Напомним также, что эвристика Кернигана—Лина (далее KL) состоит в том, что один шаг локального поиска по k-окрестности заменяется на один «макрошаг» алгорита KL. Этот макрошаг состоит в свою очередь из не более чем k «микрошагов». Каждый микрошаг состоит в поиске одной пары вершин, переброс которых между частями приводит к самому сильному уменьшению веса разбиения (или, когда *уменьшения* добиться невозможно, к *минимально возможному увеличению* веса). Если, сделав не более k микрошагов, алгоритм KL приходит к некоторому разбиению, улучшающее то, с которого стартовал текущий макрошаг, алгоритм KL фиксирует это разбиение и переходит к очередному макрошагу. В противном случае алгоритм KL останавливает поиск. Для того, чтобы в ходе выполнения микрошагов можно было достаточно далеко отойти от начального разбиения («прозондировать» бóльшую область), алгоритм KL в рамках каждого отдельного макрошага запоминает, какие изменения он делал на микрошагах, и запрещает на оставшихся микрошагах производить изменения, которые могут нивелировать уже сделанные. Это можно конкретизировать по-разному и будет приводить к разным результатам. Рассмотрим в качестве графа G, для которого строится сбалансированное разбиение, цепь на $4n$ вершинах.

(a) Пусть алгоритм KL при обмене на микрошаге пары вершин u, v запрещает на дальнейших микрошагах обмен той же самой пары вершин u, v. Покажите, что при всех достоточно больших n алгоритм KL глубины $2n$ может не найти оптимального разбиения для G при «неудачном» начальном разбиении, а именно, алгоритм KL глубины $2n$ может зациклиться в следующем смысле: сделав не более чем $2n$ микрошагов алгоритм придёт к тому же точно разбиению, с которого начинался текущий макрошаг, и ни на одном промежуточном микрошаге алгоритм не будет иметь разбиения, улучшающего начальное.

(b) Пусть алгоритм KL при обмене на микрошаге пары вершин u, v запрещает на дальнейших микрошагах обмен любых пар вершин вида u, w и v, w. Иначе говоря, если какая-то вершина меняет свою долю в разбиении, то она должна остаться в этой доле до конца макрошага. Покажите, что в этом случае алгоритм KL глубины $2n$ найдёт оптимальное разбиение для G при любом начальном разбиении.

Придумайте пример $4n$-вершинного графа, на котором поиск Кернигана–Лина глубины не более $2n$ (который при построении цепочек переходов использует 1-окрестности) всегда находит оптимальное решение, а обычный локальный поиск при использовании $(n-1)$-окрестностей может не найти оптимума. Сравните вычислительную сложность соответствующих алгоритмов.

NP-трудность решения некоторых оптимизационных задач

Задача 14. Известно, что следующая задача SET PARTITION является NP-полной:

- Дано конечное множество $M \subset \mathbb{N}$.
- Вопрос: есть ли такое $M' \subseteq M$, что $\sum_{x \in M'} x = \sum_{x \in M \setminus M'} x$?

Докажите NP-трудность задачи bin packing. Является ли эта задача NP-полной?

Задача 15. Известно, что следующая задача MAX-CUT является NP-трудной:

- Дан граф $G = (V, E)$ и число K.
- Вопрос: есть ли такое разбиение $V = V' \sqcup V''$, что $\#\{uv \in E \mid u \in V' \text{ и } v \in V''\} \geqslant K$?

Докажите NP-трудность задачи о равномерном разбиении графа (с $\alpha = \frac{1}{2}$). Является ли эта задача NP-полной?

Задача 16. Пусть G — полный граф с неотрицательными весами на рёбрах.

(a) Докажите, что вес любого гамильтонова цикла в G не меньше веса минимального остовного дерева в G.

(b) Докажите, что если число вершин в G чётно, то вес любого гамильтонова цикла в G не меньше удвоенного веса минимального совершенного паросочетания в G.

(c) Докажите, что если веса рёбер графа удовлетворяют неравенству треугольника, то вес оптимального гамильтонова цикла не больше, чем вес любого другого цикла, посещающего все вершины (возможно, не единожды). Приведите пример, показывающий, что требование выполнения неравенства треугольника существенно.

(d) Докажите, что если веса рёбер графа удовлетворяют неравенству треугольника, то вес оптимального гамильтонова цикла в любом порождённом подграфе G не превосходит веса оптимального цикла в самом G. Приведите пример, показывающий, что требование выполнения неравенства треугольника существенно.

Задача 17. В этой задаче будем работать с полным графом G с неотрицательными весами, удовлетворяющими неравенству треугольника.

(a) Пусть T — минимальное остовное дерево в G. Покажите, как, используя обход дерева T в глубину, построить гамильтонов цикл в G, вес которого не больше удвоенного веса *оптимального* гамильтонова цикла. Какие пункты задачи 16 Вы при этом использовали?

(b) Приведите пример, показывающий, что метод п. a действительно может приводить к циклам, которые в $(2-\varepsilon)$ раз хуже оптимальных, для любого $\varepsilon \in (0, 1)$. (Можно использовать пару концентрических правильных многоугольников.)

Задача 18. Алгоритм Кристофидеса (N. Christofides, 1976) развивает подход задачи 17. Пусть G — граф с весами, удовлетворяющими неравенству треугольника. Построим T — минимальное остовное дерево в G. Пусть G' — подграф G, порождённый всеми теми вершинами, у которых степень в T оказалась нечётной. Пусть M — минимальное совершенное паросочетание в G'. Граф $T \cup M$ связен и все вершины в нём имеют чётные степени (если какое-то ребро входило и в T, и в M, то мы это ребро считаем кратным в $T \cup M$), значит, он эйлеров. Пусть C' — соответствующий эйлеров обход. Пусть C — гамильтонов цикл в G, полученный, если пройти по C', пропуская, при необходимости, уже посещённые ранее вершины. Алгоритм Кристофидеса выдаёт в качестве ответа C.

(a) Докажите, что построенный алгоритмом Кристофидеса цикл не более чем в полтора раза длиннее оптимального. Какие пункты задачи 16 Вы при этом использовали?

(b) Приведите пример, показывающий, что алгоритм Кристофидеса действительно может ошибаться в полтора раза (асимптотически). (Можно использовать длинную пилу.)

Задача 19. Опишем алгоритм кратчайших вставок (Nearest Insertion) для решения задачи TSP. Алгоритм на каждом шаге имеет дело с гамильтоновым циклом через некоторое подмножество вершин графа. Начинает алгоритм с ребра минимального веса в графе. Затем на каждом шаге алгоритм ищет вершину z, ближайшую к уже обойденным вершинам. Далее алгоритм выбирает в имеющемся маршруте такое ребро xy, для которого величина $(w(xz) + w(yz) - w(xy))$ минимальна, и вставляет вершину z в текущий маршрут (удаляет ребро xy и добавляет рёбра xz и yz; на самом первом шаге ребро xy не удаляется).

(a) Докажите, что если веса рёбер графа удовлетворяют неравенству треугольника, то длина цикла, построенного алгоритмом кратчайших вставок, не больше удвоенной длины оптимального гамильтонова цикла. (Можно использовать тот факт, что *алгоритм Прима* построения MST корректен, см. задачу 35.)

(b) Приведите пример, показывающий, что алгоритм кратчайших вставок действительно может ошибиться вдвое.

Задача 20. Цель этой задачи — показать, что вряд ли можно надеяться на алгоритмы типа Nearest Neighbor в случае, когда неравенство треугольника не выполняется. Пусть $n \geqslant 4$. Определим полный граф G на множестве вершин $\{1, 2, \ldots, n\}$, приписав рёбрам следующие веса:

- $w(i, i+1) := 2ni$ при $i \in \{1, 2, \ldots, n-1\}$,
- $w(i, i+2) := 2ni + 1$ при $i \in \{1, 2, \ldots, n-2\}$,
- $w(i, j) := 2ni + 2$ для всех пар i, j, таких, что $i + 3 \leqslant j$.

Докажем, что на таком графе алгоритм Nearest Neighbor выдаст г. ц., который оказывается хуже почти всех возможных г. ц. в данном графе.

(a) Докажите, что г. ц., построенный алгоритмом ближайшего соседа, будет иметь вес не меньше $\sum_{i=1}^{n-1} 2ni + 2n + 2$.

(b) Будем при записи циклов считать, что они «начинаются» с вершины 1. Назовём г. ц. $x_1 x_2 \ldots x_n$ в графе G *пирамидальным*, если он имеет вид $1 = x_1 < x_2 < x_3 < \ldots <$

$x_k > x_{k+1} > \ldots > x_n$ для некоторого k. Покажите, что количество пирамидальных г. ц. не превосходит 2^{n-2} (и, как следствие, составляет стремящуюся к нулю долю от числа всех г. ц.).

(c) Покажем, что любой *не*пирамидальный г. ц. будет иметь меньший вес, чем цикл алгоритма ближайшего соседа. Пусть C — непирамидальный г. ц.. Запишем его в виде $y_1 y_2 \ldots y_n$, где $y_1 = 1$. Будем называть ребро $y_k y_{k+1}$ цикла C *прямым*, если $y_k < y_{k+1}$ и *обратным* в противном случае. Для обратного ребра $e = y_k y_{k+1}$ обозначим через $q(e)$ разность $(y_k - y_{k+1})$. Докажите, что у любого непирамидального г. ц. сумма $\sum_{e \text{ обратное}} q(e)$ не меньше n.

(d) У любого прямого ребра $y_k y_{k+1}$ непирамидального цикла вес может быть оценён сверху величиной $y_k \cdot 2n + 2$, а вес обратного ребра оценён величиной $2ny_{k+1} + 2 - 2nq(y_k y_{k+1})$. Выведите отсюда, что вес непирамидального г. ц. не превышает $\sum_{i=1}^{n-1} 2ni + 2n$.

Наследственные системы и матроиды

Задача 21. При разработке алгоритмов на матроидах предполагать явное задание семейства независимых множеств матроида обычно нельзя: его размер часто экспоненциален. Поэтому считают, что матроид задаётся неявно, *оракулом* (функцией, устройство которой в данном контексте не важно, и которая, по предположению, всегда выдаёт ответ за один условный такт времени). Например, для полиномиальности жадного алгоритма решения DLS-задачи достаточно иметь функцию `isIndependent`, которая по произвольному множеству A элементов матроида быстро определяет, независимо ли оно. Но можно задать матроид и другим оракулом: например, `isCircuit` (отвечает на вопрос «верно ли, что заданное множество — цикл матроида»), `isBasis` («верно ли, что заданное множество — база матроида»), `getRank` (вычисляет ранг заданного множества), `isBasisSuperset` (верно ли, что заданное множество содержит одну из баз матроида).

(a) Докажите, что если матроид задан одним из оракулов `isIndependent`, `isCircuit`, `isBasis`, `getRank`, `isBasisSuperset`, то можно вычислить значение любого другого оракула на любом заданном множестве (пусть даже и за экспоненциальное время).

(b) Докажите, что если матроид задан оракулом `isCircuit`, то за полиномиальное время не всегда можно вычислить значение оракула `isIndependent`. Зафиксируем множества E и $A \subseteq E$. Рассмотрим матроид $(E, \mathscr{F}_A)$, в котором A — единственный цикл. Пусть теперь нам задан некоторый матроид $\mathscr{M}$, и мы хотим понять, совпадает ли он с тривиальным матроидом $(E, 2^E)$. Это равносильно вычислению значения `isIndependent`(E). Если же в распоряжении у нас есть лишь функция `isCircuit`, нам придётся вызвать её на *всех* подмножествах E: не сделав для некоторого A вызов `isCircuit`(A), можно ошибочно принять за $(E, 2^E)$ матроид $(E, \mathscr{F}_A)$.

Задача 22. Докажите, что следующие системы $(E, \mathscr{F})$ являются матроидами.

(a) Пусть $(E', \mathscr{F}')$ и $(E'', \mathscr{F}'')$ — матроиды, причём $E' \cap E'' = \varnothing$. Положим $E := E' \cup E''$, $\mathscr{F} := \{A' \cup A'' \mid A' \in \mathscr{F}',\ A'' \in \mathscr{F}''\}$.

(b) Пусть $E = E_1 \sqcup E_2 \sqcup \ldots \sqcup E_m$ и $\mathscr{F} := \{E' \subseteq E \mid \forall i\, |E' \cap E_i| \leqslant 1\}$. Такой матроид $(E, \mathscr{F})$ называется *матроидом разбиения*.

(c) Пусть $G = (V, E)$ — простой граф, и $V' \subset V$ — независимое множество в G. Зафиксируем для каждого $v \in V'$ число $k_v \in \mathbb{N}$ и положим

$$\mathscr{F} := \{E' \subseteq E \mid \forall v \in V' \text{ не более } k_v \text{ рёбер из } E' \text{ инцидентны } v\}.$$

(d) Пусть $G = (V, E)$ — ориентированный граф, и $V' \subseteq V$. Зафиксируем для каждого $v \in V'$ число $k_v \in \mathbb{N}$ и положим

$$\mathscr{F} := \{E' \subseteq E \mid \forall v \in V' \text{ не более } k_v \text{ рёбер из } E' \text{ выходят из } v\}.$$

Задача 23. Пусть $(E', \mathscr{F}')$, $(E'', \mathscr{F}'')$ — матроиды. Убедитесь, что $(E' \cup E'', \mathscr{F}' \cup \mathscr{F}'')$ не всегда является матроидом (ср. п. a задачи 22).

Задача 24. Найдите достижимые нижние оценки величины рангового разброса наследственных систем, возникающих в следующих задачах дискретной оптимизации.

(a) Задача о клике максимального веса (в заданном графе с весами на *вершинах* найти клику максимального суммарного веса).

(b) Задача о 0,1-рюкзаке.

Задача 25. *(a)* Рассмотрим наследственную систему, возникающую в задаче TSP на полном графе K_n. Допустимые множества в этой системе — это любые подмножества рёбер гамильтоновых циклов (то есть, по сути, это либо сами гамильтоновы циклы, либо объединения непересекающихся цепей). Докажите, что ранговый разброс такой системы не меньше $\frac{1}{2}$. Рассмотрите две базы и оцените размер одной из них сверху через размер другой.

(b) Найдите точное значение рангового разброса системы п. a как формулу от n.

(c) Рассмотрим наследственную систему, возникающую в задаче TSP на произвольном n-вершинном графе G. Допустимые множества в этой системе — это любые подмножества рёбер гамильтоновых циклов в G. Докажите, что, в отличие от п. a, ранговый разброс такой системы может быть равен $\frac{1}{n}$. Можно, но вовсе необязательно, использовать в своей конструкции подграфы-алмазы.

Задача 26. *(a)* Докажите, что если $\mathrm{P} \neq \mathrm{NP}$, то для системы из задачи 25c за полиномиальное время в общем случае не получится вычислять значения функции `isIndependent` (см. задачу 21). Можно ли вычислять за полиномиальное время значения функции `isIndependent` для системы из задачи 25a?

(b) Отметим следующие факты: (1) мы знаем, что за счёт трюка с изменением весов можно свести задачу поиска гамильтонова цикла минимального веса к поиску цикла максимального веса; (2) ранговый разброс системы в задаче 25a ограничен снизу константой $1/2$; (3) у нас есть теорема Дженкинса—Корте—Хаусмана о том, что в DLS-задаче максимизации на системах с ранговым разбросом q жадный алгоритм имеет показатель аппроксимации не хуже q. Казалось бы, из трёх приведённых фактов вытекает, что у нас в кармане простой полиномиальный жадный алгоритм решения задачи коммивояжёра на полном графе, имеющий константный показатель аппроксимации. В то же время, известно, что если $\mathrm{P} \neq \mathrm{NP}$, то для задачи TSP на полном графе общего вида (без требования выполнения неравенства треугольника для весов) *не существует* полиномиальных алгоритмов, которые могли бы приближать оптимальное решение с константной мультипликативной погрешностью. Разрешите это кажущееся «противоречие»!

Задача 27. Пусть E — множество мощности n, а $\mathcal{F}$ — наследственная система подмножеств E. Пусть веса элементов множества E принадлежат множеству $\{1, \ldots, N\}$. Допустим, что пара $(E, \mathcal{F})$ не является матроидом. Какое максимальное значение может принимать разность $\max_{A \in \mathcal{F}} w(A) - w(A')$, где A' — множество, построенное жадным алгоритмом? Дайте ответ в виде функции от величин n, N. Заметьте, что в описании жадного алгоритма не сказано, как он выбирает элемент в случае, когда ему доступно для выбора несколько самых «тяжёлых» элементов. Поэтому рассмотрите две постановки указанной задачи: первая, когда жадному алгоритму «везёт» в таких случаях, и вторую, когда «не везёт».

Задача 28. Пусть $(E, \mathcal{F})$ — наследственная система, ранговый разброс которой равен $1/t$. Постройте для каждого ε набор весов, на котором жадный алгоритм ошибётся не менее чем в $(t - \varepsilon)$ раз даже при условии, что ему «везёт» в случаях, когда можно добавить несколько разных элементов, имеющих максимальный вес.

Задача 29. Докажите, что на любом матроиде $(E, \mathcal{F})$ будет давать точное решение максимизационной DLS-задачи стандартный алгоритм локального поиска, в котором пространство решений (область поиска) — множество $\mathcal{S}$ всех баз матроида, а окрестность решения $X \in \mathcal{S}$ определена так: $\mathcal{N}(X) := \{X' \in \mathcal{S} \mid |X \triangle X'| = 2\}$.

Задача 30. *Двойственной* наследственной системой к системе $(E, \mathcal{F})$ называется система $(E, \mathcal{F}^*)$, в которой

$$\mathcal{F}^* := \bigcup_{B - \text{база } (E, \mathcal{F})} 2^{E \setminus B}.$$

(a) Убедитесь, что $(E, (\mathcal{F}^*)^*) = (E, \mathcal{F})$ для любой наследственной системы $(E, \mathcal{F})$.

(b) Докажите, что если $(E, \mathcal{F})$ матроид, то $(E, \mathcal{F}^*)$ также матроид. Пусть $A \subseteq E$ и пусть X — база A в системе $\mathcal{F}^*$. Пусть $B_{E \setminus A}$ — база множества $E \setminus A$ в матроиде $\mathcal{F}$. Множество $B_{E \setminus A}$ можно расширить до базы B множества $E \setminus X$ в $\mathcal{F}$ (почему?). Из того, что $X \in \mathcal{F}^*$, следует, что $|B| = \mathrm{rk}(E)$ (везде rk означает ранг относительно матроида $\mathcal{F}$). Докажите, что $B \supseteq A \setminus X$. Выведите отсюда равенство $|X| = |A| - \mathrm{rk}(E) + \mathrm{rk}(E \setminus A)$. Оно показывает, что $|X|$ не зависит от выбора X, что и требовалось.

Задача 31. *(a)* На лекции мы рассмотрели алгоритм поиска допустимого множества максимального веса (естественно, при положительных весах такое множество всегда будет базой): добавляем каждый раз элемент максимального веса, так, чтобы сохранить независимость получаемого множества. Условно такой алгоритм можно назвать Best-In. Рассмотрим обратный алгоритм (условно называемый Worst-Out): начав со множества всех элементов, последовательно удаляем на каждом шаге такой элемент минимального веса, что среди остающихся элементов всё ещё содержится хотя бы одна база матроида. Докажите, что для матроидов такой алгоритм даёт точное решение задачи поиска базы максимального веса. (Можно рассмотреть двойственный матроид.)

(b) Приведите пример наследственной системы (не являющейся матроидом) и набора весов на её элементах, таких, что алгоритм Worst-Out выдаёт базу, вес которой в любое наперёд заданное число раз отличается от оптимального. Получается, что, в отличие от алгоритма Best-In, мы не можем дать в общем случае никакой оценки качества работы алгоритма Worst-Out через ранговый разброс.

Задача 32. Пусть E — произвольное конечное множество мощности n, и пусть k — фиксированное натуральное число, $1 \leqslant k \leqslant |E| - 1$. Положим $\mathcal{F} := \{A \subseteq E \mid |A| \leqslant k\}$ (однородный матроид). Пусть $P(n, k, N)$ — вероятность того, что в матроиде $(E, \mathcal{F})$ при случайном выборе весов из множества $\{1, \ldots, N\}$ в задаче поиска допустимого множества максимального веса будет единственное решение. Докажите равенство

$$P(n, k, N) = \binom{n}{k} \cdot \sum_{i=1}^{N-1} \left(\left(1 - \frac{i}{N}\right)^k - \left(1 - \frac{i+1}{N}\right)^k \right) \left(\frac{i}{N}\right)^{n-k}.$$

Задачи MST и Steiner Tree

Задача 33. Рассмотрим следующий алгоритм поиска минимального остовного дерева в полном графе («обратный алгоритм Краскала»). На каждом очередном шаге удаляем из графа ребро максимального веса, такое, что во множестве остающихся рёбер всё ещё можно выделить остовное дерево. Когда ничего уже удалить не получается (а значит, оставшиеся рёбра как раз и образуют остовное дерево), выдаём ответ. Обоснуйте корректность этого алгоритма. (Можно использовать результат задачи 31.)

Задача 34. Пусть T — произвольное минимальное остовное дерево графа G, пусть $e \in E(G) \setminus E(T)$ — произвольное ребро. Пусть C — цикл (единственный) в графе $(T + e)$. Докажите, что вес ребра e максимален среди весов рёбер C. (Это утверждение может быть использовано либо напрямую, либо косвенно, в доказательстве корректности большинства алгоритмов решения задачи MST.)

Задача 35. Алгоритм Прима (Robert C. Prim, 1957; Vojtěch Jarník, 1930; Edsger W. Dijkstra, 1959) решения задачи MST действует следующим образом: начав с произвольной вершины графа, алгоритм строит минимальное остовное дерево ребро за ребром, добавляя каждый раз легчайшее из рёбер, соединяющих вершины текущего дерева с вершинами вне его. Таким образом, в отличие от алгоритма Дейкстры, работающего с лесом и сокращающего число компонент связности в лесе, алгоритм Прима на каждом шаге работает с деревом. Докажите, что дерево, выдаваемое алгоритмом Прима, оптимально. (Зафиксируйте м. о. д. T_0 и дерево, построенное алгоритмом Прима T_P. Рёбра T_P считаются упорядоченными в том порядке, в котором их добавлял алгоритм. Рассмотрите первое по порядку ребро e дерева T_P, не лежащее в T_0. Пусть k — порядковый номер e. Покажите, что можно перестроить дерево T_0 в дерево T_1, вес которого не больше веса T_0 (а значит, оптимальное), и такое, что первые k рёбер T_P лежат в T_1. Продолжая процесс, придите к оптимальному дереву, совпадающему с T_P.)

Задача 36. Алгоритм Борувки (Otakar Borůvka, 1926) решения задачи MST для графа, в котором *веса всех рёбер различны*, действует следующим образом. Пусть $G = (V, E)$ — граф, на котором запускается алгоритм. Алгорим строит остовный лес, работая со множеством C компонент связности, последовательно объединяя эти компоненты, пока не получится одна компонента. Изначально алгоритм полагает $C := \{\{v\} \mid v \in V\}$. На каждой итерации алгоритм для каждой компоненты из множества C (независимо от остальных компонент) ищет ребро минимального веса среди рёбер, ведущих из компоненты вовне.

Затем алгоритм одномоментно добавляет все найденные рёбра, в результате чего некоторые из компонент сливаются друг с другом. Алгоритм действует так, пока не останется единственная компонента. Докажите корректность этого алгоритма. Можно использовать результат задачи 34.

Задача 37. Рассмотрим задачу MinMax Spanning Tree (MinMaxST) — поиск остовного дерева, минимизирующего не сумму весов рёбер, а вес самого тяжёлого ребра.

(a) Докажите, что любое решение задачи MST является также и решением задачи MinMaxST.

(b) Верно ли утверждение, обратное утверждению п. a?

Задача о *дереве Штейнера* (Steiner Tree problem) состоит в отыскании для заданного графа $G = (V, E)$ с весами на рёбрах и заданного множества $V' \subseteq V$ дерева $T \subseteq G$, такого, что все вершины из V' попадают в T и сумма весов рёбер в T минимальна. При этом иногда говорят о вершинах из V' как об *обязательных*, а об остальных вершинах как о *вспомогательных*. Содержательно, речь в задаче идёт о построении оптимальной сети связей между обязательными вершинами с использованием вспомогательных вершин. Очевидно, что если $V' = V$, задача вырождается в задачу о минимальном остовном дереве, а если $|V'| = 2$, то в задачу о кратчайшем пути.

Задача 38. Известно, что задача о вершинном покрытии NP-трудна (к ней легко сводится NP-трудная задача КЛИКА). Докажем NP-трудность задачи SteinerTree. Пусть $G = (V, E)$ — граф в задаче о вершинном покрытии. Добавим на каждое ребро $e \in E$ по одной новой вершине u_e (то есть подразобьем каждое ребро), а также добавим одну новую вершину u, соединив её со всеми вершинами множества V. Получится граф G'. Как связаны размер вершинного покрытия графа G и размер дерева Штейнера для графа G' (обязательными считаются вершины $\{u\} \cup \{u_e \mid e \in E\}$)?

Задача 39. Рассмотрим задачу MetricSteinerTree, вариант задачи SteinerTree для полного графа, веса рёбер которого удовлетворяют неравенству треугольника.

(a) Докажите, что для задачи MetricSteinerTree можно легко построить дерево, вес которого не более чем вдвое превосходит вес оптимального дерева: просто взять минимальное остовное дерево в подграфе, порождённом обязательными вершинами. Обоснуйте, что вес такого дерева действительно ограничен удвоенным весом оптимального. (Указание. Пусть T — оптимальное дерево Штейнера. Рассмотрим обход T в глубину и «срежем в нём углы», оставляя только обязательные вершины, пропуская вспомогательные. См. также задачу 17).

(b) Приведите пример последовательности графов, веса рёбер которых удовлетворяют неравенству треугольника, и для которых подход из указания к п. a действительно может ошибиться в $(2 - \varepsilon)$ раз для любого ε. (Можно взять цикл с весами рёбер $(2 - \varepsilon)$ и соединить все вершины цикла с одной новой вершиной рёбрами веса 1.)

(c) Приведите для любого K пример графа, веса рёбер которого *не* удовлетворяют неравенству треугольника, и для которого алгоритм из п. a может ошибиться более чем в K раз.

Задача 40. Пусть G — связный граф (необязательно полный) с неотрицательными весами на рёбрах, соответствующую весовую функцию обозначим w. *Метрическим замыканием* графа G называется полный граф G', для которого $V(G') = V(G)$, в котором вес каждого ребра uv полагается равным минимальному весу пути, соединяющего u и v в G. Соответствующую весовую функцию обозначим w'.

(a) Докажите, что веса рёбер графа G' удовлетворяют неравенству треугольника.

(b) Докажите, что для любого подграфа $H' \subseteq G'$ выполнено неравенство $w'(H') \leqslant w(H')$. (Если некоторое ребро e отсутствует в G, считаем $w(e) = +\infty$.)

(c) Докажите, что для любого подграфа $H' \subseteq G'$ существует подграф $H \subseteq G$, такой, что $w(H) \leqslant w'(H')$ и каждая компонента связности графа H' является подмножеством компоненты связности графа H.

Задача 41. В этой задаче мы докажем, что если для задачи MetricSteinerTree есть полиномиальный алгоритм $\mathcal{A}$ с показателем аппроксимации c, то и для задачи SteinerTree существует полиномиальный алгоритм с тем же показателем аппроксимации c. Пусть G — произвольный граф с весами на рёбрах (неравенство треугольника для G может не выполняться), соответствующую весовую функцию обозначим w. Пусть $T_{\text{опт}}$ — дерево Штейнера в графе G, оптимальное относительно w. Докажите, что следующая процедура даёт искомое приближение:

- Строим метрическое замыкание G' графа G (см. задачу 40). Соответствующую весовую функцию обозначим w'. Пусть $T'_{\text{опт}}$ — оптимальное дерево Штейнера в G'.
- Строим в графе G' с весами w' при помощи алгоритма $\mathcal{A}$ приближение T' для дерева Штейнера $T'_{\text{опт}}$, такое, что $w'(T') \leqslant c \cdot w'(T'_{\text{опт}})$.
- Рассматриваем остовный связный подграф $\widetilde{T}$ в G, такой, что $w(\widetilde{T}) \leqslant w'(T')$. (См. п. c задачи 40.)
- Строим в графе $\widetilde{T}$ минимальное остовное дерево T. Утверждается, что T будет искомым c-приближением дерева Штейнера графа G.

Линейное программирование

Задача 42. Есть 10 видов продуктов. Для каждого из них известно, сколько белков, жиров и углеводов содержится в единице продукта. Также известна стоимость единицы каждого продукта. Известно, сколько белков, жиров и углеводов требуется взрослому человеку в сутки. Задача: составить план закупки продуктов на неделю, так, чтобы удовлетворять суточную потребность организма и при этом минимизировать суммарную стоимость закупок. Сформулируйте соответствующую задачу линейного программирования и укажите, сколько неравенств в полученной системе.

Задача 43. Пусть $x_1, \ldots, x_n$ и $y_1, \ldots, y_m$ — переменные. Составьте систему линейных неравенств, которая эквивалентна неравенству $\max\{x_1, \ldots, x_n\} \leqslant \min\{y_1, \ldots, y_m\}$,

(a) без использования дополнительной переменной;

(b) используя одну дополнительную переменную.

(c) Докажите, что в п. a не получится обойтись меньшим количеством неравенств.

Задача 44. Докажите, что не существует системы линейных неравенств, эквивалентной неравенству $\min\{x_1, x_2\} \leqslant 2015$.

Задача 45. Пусть задача ЛП задана в форме:

$$\begin{cases} \mathbf{c}^T x \to \min, \\ \mathbf{A}\mathbf{x} = \mathbf{b}, \\ \mathbf{x} \geqslant \mathbf{0}, \end{cases}$$

где $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$. Пусть α — максимум из абсолютных значений компонент вектора $\mathbf{b}$ и матрицы $\mathbf{A}$. Докажите, что если указанная задача ЛП имеет решение, то она имеет и такое решение, все компоненты которого ограничены сверху величиной $m! \cdot \alpha^m$.

Задача 46. Задача *целочисленного линейного программирования* (ЦЛП) ставится так же, как и обычная задача линейного программирования, но с дополнительным условием целочисленности значений всех переменных (коэффициенты всех неравенств и оптимизируемой линейной формы также считаются целыми). Если условия целочисленности наложены не на все переменные, то говорят о задаче *смешанного целочисленного линейного программирования* (СЦЛП). Если в задаче ЦЛП дополнительно наложено ограничение, что для каждой переменной допускаются лишь два значения, 0 и 1, то говорят о задаче *булева линейного программирования* (БЛП).

(a) Верно ли, что БЛП — это частный случай ЦЛП?

(b) Покажите, что для любой задачи ЦЛП, имеющей конечное решение, можно построить эквивалентную ей задачу БЛП. Причём так, чтобы размер полученной задачи (количество памяти, в которую можно записать все коэффициенты и ограничения) не превосходил полинома от размера исходной задачи. (См. задачи 4 и 45.)

(c) Докажите NP-трудность задачи БЛП, сведя к ней NP-трудную задачу SAT.

Задача 47. Докажите, что для любой задачи СЦЛП, в которой присутствует *одно* из следующих нестандартных условий, можно построить эквивалентную задачу СЦЛП, где в качестве ограничений на значения переменных присутствуют лишь стандартные линейные неравенства и, возможно, требование целочисленности.

(a) «$x = 0$ или $x = 2$»,

(b) «$x = 1$ или $x = 18$»,

(c) «$x \in \{0, 3, 5, 8\}$»,

(d) «$x \in \{0, 3, 5\}$»,

(e) «$x \in S$», где S — произвольное фиксированное конечное множество целых чисел,

(f) «$\max\{x_1, \ldots, x_m\} \leqslant a$ и одновременно $\min\{x_1, \ldots, x_m\} \leqslant b$», где a и b — фиксированные целые числа и $b < a$.

Задача 48. Рассмотрим граф G и *фиксированное* натуральное k. Задача MAX-COVER состоит в том, чтобы выбрать в G множество вершин мощности k, покрывающее как можно больше рёбер. Поставьте эту задачу как задачу (целочисленного) линейного программирования.

Задача 49. Задача размещения ресурсов (facility location) ставится следующим образом. Есть набор *пунктов возможного размещения* ресурсов (например, места, в которых можно открыть продуктовые магазины шаговой доступности) и *набор пользователей* ресурсов (например, жители города). Для каждого пункта j возможного размещения ресурса задана стоимость размещения ресурса f_j в этом пункте. Также для каждого пользователя i и каждой точки j задана стоимость c_{ij} использования i-м пользователем ресурса, находящегося в j-м пункте (например, время, которое горожанин тратит на то, чтобы дойти до соответствующего магазина). Можно считать, что один и тот же пункт доступа к ресурсу (при его открытии) могут использовать неограниченное число пользователей. Задача в том, чтобы разместить в некоторых из допустимых пунктов ресурс, так, чтобы минимизировать суммарные стоимости (т.е. затраты на размещение плюс затраты на пользование). Поставьте эту задачу как задачу ЦЛП.

Двойственность в линейном программировании

Задача 50. Докажите, что система неравенств $\mathbf{A}\mathbf{x} < \mathbf{b}$ имеет решение тогда и только тогда, когда условиям $\mathbf{y} \geqslant \mathbf{0}$, $\mathbf{y}^{\mathrm{T}}\mathbf{A} = \mathbf{0}$, $\mathbf{y}^{\mathrm{T}}\mathbf{b} \leqslant 0$ удовлетворяет только нулевой вектор.

Задача 51. Для следующих задач линейного программирования выпишите двойственные задачи:

(a)

$$\begin{cases} x_1 - 6x_2 - 6x_3 & \to & \min \\ 2x_1 + 5x_2 - x_3 & \leqslant & 13 \\ x_1 - 2x_2 - 3x_3 & \geqslant & 0 \\ 9x_1 + x_2 + x_3 & \leqslant & 7 \\ 4x_1 - x_2 & \leqslant & 20 \\ x_1,\, x_2,\, x_3 & \geqslant & 0 \end{cases}$$

(b)

$$\begin{cases} x_1 - 6x_2 - 6x_3 & \to & \max \\ 2x_1 + 5x_2 - x_3 & = & 13 \\ x_1 - 2x_2 - 3x_3 & = & 0 \\ x_1,\, x_2,\, x_3 & \geqslant & 0 \end{cases}$$

(c) Та же система, что в п. b, но *без* условий неотрицательности переменных.

Задача 52. *(a)* Некая компания добывает m видов сырья в количестве $b_1, \ldots, b_m$ и производит n видов продукции. На производство единицы j-го вида продукции идёт $a_{i,j}$ единиц i-го вида сырья. От реализации единицы i-го вида продукции компания получает прибыль c_i. Поставьте задачу максимизации прибыли добывающей компании как задачу ЛП.

(b) Некая фирма-покупатель предлагает выкупить сырьё у добывающей компании, платя y_i за единицу i-го сырья. При этом фирма рассчитывает выторговать как можно меньшую совокупную цену на приобретаемое сырьё. Добывающая компания будет готова продавать сырьё только в том случае, если для каждого вида продукции выручка от продажи фирме сырья, идущего на производство продукции, будет не меньше, чем прибыль от реализации этой продукции. Поставьте задачу минимизации затрат фирмы-покупателя как задачу ЛП. Убедитесь, что получается задача, двойственная к задаче максимизации прибыли добывающей компании.

(Вместе указанные задачи ЛП называются «задачей торга».)

Задача 53. Дан неориентированный граф с «длинами» рёбер и зафиксирована пара вершин s и t в нём. Представим себе, что все вершины графа суть точки на прямой, а каждое

ребро — нить, концы которой привязаны к соответствующим точкам. На каком расстоянии окажутся вершины s и t, если попытаться максимально далеко оттянуть их друг от друга? Поставьте эту задачу как задачу линейного программирования, в которой переменные соответствуют координатам точек на прямой. Выпишите двойственную задачу. Какой «физический смысл» у полученной двойственной задачи?

Задачи о покрытии

Задача 54. *(a)* Рассмотрим задачу «Анти-Треугольник»: для заданного графа G с весами на рёбрах удалить минимальное число рёбер, так, чтобы в полученном графе не осталось треугольников. Предложите полиномиальный алгоритм «избавления от треугольников», который бы удалял рёбра, число которых не более чем втрое превышает оптимальное.

(b) Рассмотрим взвешенный вариант задачи Анти-Треугольник: граф с весами, требуется удалить множество рёбер с минимально возможным суммарным весом. Предложите полиномиальный алгоритм для этой задачи с показателем аппроксимации 3.

Задача 55. Задача «покрытие множеств» SET-COVER состоит в том, чтобы из заданного набора множеств $S_1, \ldots, S_n \subseteq \{e_1, \ldots, e_m\}$ выделить поднабор $S_{i_1}, \ldots, S_{i_k}$, покрывающий всю совокупность $\{e_1, \ldots, e_m\}$. В задаче без весов требуется минимизировать k, а во взвешенной задаче требуется минимизировать $w_{i_1} + \ldots + w_{i_k}$, где $w_{\ldots}$ — стоимости/веса соответствующих множеств.

(a) Сформулируйте задачу SET-COVER как задачу ЦЛП.

(b) Сформулируйте двойственную задачу.

(c) Объясните, почему для решения задачи о покрытии множеств не работает такая идея: решим соответствующую задачу линейного программирования («забыв» про целочисленность переменных), а затем округлим значения переменных до ближайших целых чисел.

(d) Предложите полиномиальный алгоритм на основе линейного программирования, который строит покрытие, не более чем в k раз превосходящее по весу оптимальное, где

$$k := \max_{1 \leqslant j \leqslant m} |\{i \mid S_i \ni e_j\}|.$$

Задача 56. Пусть $x_1, \ldots, x_n \in [0, 1]$ — решение задачи ЛП без ограничения на целочисленность, построенной по задаче SET-COVER (см. задачу 55). Рассмотрим следующую процедуру *вероятностного округления*. Изначально полагаем $\widetilde{x}_1, \ldots, \widetilde{x}_n = 0$. Далее на каждом шаге (число шагов потенциально неограниченное) делаем следующее. Если $\widetilde{x}_1, \ldots, \widetilde{x}_n$ ещё не задаёт покрытие, то просматриваем по одному разу каждое значение x_i и с вероятностью x_i полагаем $\widetilde{x}_i$ равным единице, а с вероятностью $(1 - x_i)$ оставляем старое значение $\widetilde{x}_i$ неизменным. Везде ниже e (без нижнего индекса) означает основание натурального логарифма.

(a) Пусть e_j — произвольный элемент. Тогда с вероятностью не менее $1 - e^{-1}$ этот элемент окажется покрытым за один шаг процедуры вероятностного округления (один шаг — это один просмотр всей последовательности $x_1, \ldots, x_n$). (Можно воспользоваться неравенством $1 - \gamma \leqslant e^{-\gamma}$ для $\gamma \in [0, 1]$.)

(b) Покажите, что вероятность того, что общее число шагов алгоритма превзойдёт $(\ln m + t)$, можно оценить сверху величиной e^{-t}.

(c) Пусть алгоритм остановился, проделав t шагов. Тогда матожидание веса полученного покрытия не более чем в t раз превосходит вес оптимального покрытия.

(d) Докажите, что с вероятностью не менее 0.45 алгоритм остановится за полиномиальное число шагов и выдаст покрытие, вес которого превосходит оптимальный не более чем в $(2 \ln m + 6)$ раз. (Пригодится неравенство Маркова.)

Особенности задач целочисленного программирования

Рассмотрим задачи

$$\begin{cases} \mathbf{A}\mathbf{x} = \mathbf{b}, \\ \mathbf{x} \geqslant \mathbf{0}, \\ \mathbf{c}^{\mathrm{T}}\mathbf{x} \to \max \end{cases} \tag{1}$$

и

$$\begin{cases} \mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \\ \mathbf{x} \geqslant \mathbf{0}, \\ \mathbf{c}^{\mathrm{T}}\mathbf{x} \to \max \end{cases} \tag{2}$$

Квадратная матрица называется *унимодулярной*, если её определитель равен 1 либо -1. Матрица называется *вполне унимодулярной*, если значение каждого её минора принадлежит множеству $\{-1, 0, 1\}$ (в частности, каждый элемент матрицы также обязан принадлежать этому множеству).

Задача 57. Обозначим через $\mathbf{I}$ единичную матрицу. Докажите, что если одна из матриц $\mathbf{A}$, $\mathbf{A}^{\mathrm{T}}$, $-\mathbf{A}$, $(\mathbf{A}|\mathbf{A})$, $(\mathbf{A}|\mathbf{I})$ вполне унимодулярна, то и все остальные также вполне унимодулярны.

Задача 58. *(a)* Докажите, что если матрица $\mathbf{A}$ вполне унимодулярна, то для любого целочисленного вектора $\mathbf{b}$ все вершины многогранника, определяемого ограничениями задачи (1), имеют целочисленные координаты. (Вспомните формулу Крамера.) Докажите, что при целочисленном $\mathbf{b}$ и вполне унимодулярной матрице $\mathbf{A}$ решение задачи (1), полученное симплекс-методом (если оно существует), целочисленно.

(b) Докажите утверждение п. a для задач ЛП в форме (2).

Задача 59. Докажите, что следующее условие является *достаточным* для полной унимодулярности матрицы $\mathbf{A} \in \{-1, 0, 1\}^{m \times n}$. В каждом столбце матрицы $\mathbf{A}$ содержится не более двух ненулевых элементов, а строки $\mathbf{A}$ можно разбить на два класса, так, чтобы

- если в столбце матрицы есть пара ненулевых элементов одного знака, то соответствующие им строки лежат в разных классах,
- если в столбце матрицы есть пара ненулевых элементов противоположных знаков, то соответствующие им строки лежат в одном и том же классе.

(Всё, сформулированное выше, — это одно нераздельное условие!) Можно использовать индукцию по размеру миноров.

Отметим, что на сегодня не известно никаких легко (за полиномиальное время) проверяемых *критериальных* условий полной унимодулярности.

Задача 60. *(a)* Докажите, что матрица инциденций вершин и дуг *ориентированного* графа является вполне унимодулярной.

(b) Докажите, что матрица инциденций вершин и рёбер двудольного графа является вполне унимодулярной.

(c) Докажите, что матрица ограничений в задаче о потоке фиксированной величины и наименьшей стоимости является вполне унимодулярной.

Потоки фиксированной величины и наименьшей стоимости

Дан ориентированный граф $G = (V, E)$, в котором каждой дуге e приписана *пропускная способность* $\mathrm{cap}(e)$ и *стоимость* перемещения единицы продукта по этому ребру $\mathrm{cost}(e)$. *Циркуляцией* в графе G называется набор величин $f(e)$ на дугах, такой, что $f(e) \in [0, \mathrm{cap}(e)]$ и $\sum_{u:\, uv \in E} f(e) = \sum_{w:\, vw \in E} f(e)$. Иначе говоря, циркуляция — это поток нулевой величины. *Стоимость циркуляции* определяется суммой $\sum_{e \in E} f(e)\mathrm{cost}(e)$.

Остаточной сетью или *сетью приращения* для сети G и потока f называется граф G_f, в котором каждой дуге $e \in E$ соответсвуют две дуги. Первая из них направлена так же, как e и имеет стоимость $\mathrm{cost}(e)$ и пропускную способность $\mathrm{cap}(e) - f(e)$, а вторая направлена противоположно e и имеет стоимость $-\mathrm{cost}(e)$ и пропускную способность $f(e)$.

Задача 61. *(a)* Докажите, что в любой циркуляции отрицательной стоимости можно выделить простой цикл отрицательной стоимости.

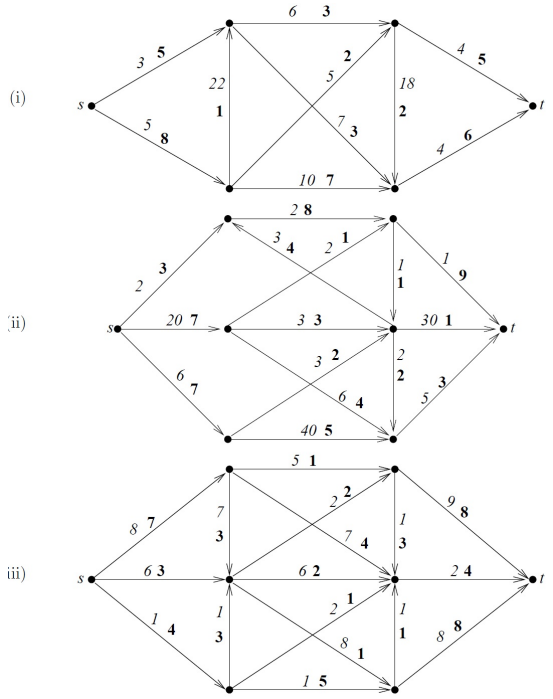
(b) Докажите, что поток в сети является оптимальным по стоимости тогда и только тогда, когда в остаточной сети нет циклов отрицательной стоимости по дугам с ненулевыми пропускными способностями.

(c) Предложите алгоритм поиска потока фиксированной величины и наименьшей стоимости в заданной сети. Оцените вычислительную сложность предложенного алгоритма.

Задача 62. *(a)* Пусть f — самый «дешёвый» поток величины t в сети G. Пусть P — путь по дугам с ненулевыми пропускными способностями в сети G_f от источника к стоку, такой, что сумма стоимостей его дуг минимально возможна. Докажите, что если увеличить поток f вдоль пути P на любую допускаемую пропускными способностями величину Δ, то полученный поток будет самым дешёвым из потоков величины $(t + \Delta)$ в сети G.

(b) Предложите алгоритм поиска потока фиксированной величины и наименьшей стоимости в заданной сети. Оцените вычислительную сложность предложенного алгоритма.

Задача 63. Для следующих сетей найдите потоки наибольшей величины и наименьшей стоимости (жирным шрифтом указаны пропускные способности, курсивом — стоимости):



Алгоритмы упаковки и online-алгоритмы

Напомним задачу Bin Packing: требуется разбить набор чисел $(a_1, \ldots, a_n)$ на минимально возможное число поднаборов, так, чтобы в каждом поднаборе сумма всех чисел не превышала 1.

Задача 64. Рассмотрим алгоритм Next Fit для решения задачи Bin Packing. Алгоритм просматривает последовательность $a_1, a_2, \ldots$ по очереди, и если добавление очередного элемента a_i к текущему поднабору приводит к превышению суммы чисел в поднаборе, то текущий поднабор фиксируется (и добавление в него больше не производится) и создаётся новый текущий поднабор, в который вносится a_i. Пусть $\mathrm{NF}(a_1, \ldots, a_n)$ — количество поднаборов, которые построил алгоритм Next Fit на наборе $(a_1, \ldots, a_n)$. Пусть $\mathrm{OPT}(a_1, \ldots, a_n)$ — оптимальное количество поднаборов.

(a) Докажите, что любую последовательность чисел $(a_1, \ldots, a_n)$ можно так переупорядочить, что будет выполняться равенство $\mathrm{NF}(a_{\pi(1)}, \ldots, a_{\pi(n)}) = \mathrm{OPT}(a_{\pi(1)}, \ldots, a_{\pi(n)})$. Обязательно ли при этом разбиение, построенное алгоритмом Next Fit, совпадёт с тем оптимальным разбиением, которое мы изначально рассматриваем?

(b) Докажите, что для любого $t \in [0, \min\{a_r, 1 - a_{r+1}\}]$ выполняется неравенство $\mathrm{NF}(a_1, \ldots, a_n) \leqslant \mathrm{NF}(a_1, \ldots, a_r - t, a_{r+1} + t, \ldots, a_n)$.

(c) Докажите оценку $\mathrm{NF} \leqslant 2 \cdot \mathrm{OPT} - 1$.

(d) Приведите пример последовательности $a_1, \ldots, a_n$, для которой $\mathrm{NF} = 2 \cdot \mathrm{OPT} - 1$.

(e) Докажите, что если $0 \leqslant a_1, a_2, \ldots, a_n \leqslant \gamma$ для некоторого $\gamma \in (0, 1)$, то $\mathrm{NF} \leqslant$

$\left\lceil \frac{\sum_i a_i}{1-\gamma} \right\rceil$.

Задача 65. Рассмотрим алгоритм First Fit для решения задачи Bin Packing. Алгоритм просматривает последовательность $a_1, a_2, \ldots$ по очереди, и добавляет очередной элемент a_i к поднабору с минимально возможным номером, так, чтобы не возникало переполнения. Пусть FF — количество поднаборов, которые построил алгоритм First Fit. Пусть OPT — оптимальное количество поднаборов.

Также рассмотрим ещё величину FFD — количество поднаборов, которые построит алгоритм First Fit Decreasing. Алгоритм First Fit Decreasing состоит в том, что вначале набор всех весов упорядочивается по убыванию, а затем на полученном наборе запускается стандартный алгоритм First Fit.

Отметим, что алгоритм First Fit является примером онлайн-алгоритма, распределяющего каждый вес «по мере поступления» и не пересматривающий свои решения. Алгоритм First Fit Decreasing требует наличия информации обо всех весах до начала своей работы, и поэтому не может называться онлайновым.

(a) Докажите оценку $\mathrm{FF} \leqslant 2 \cdot \mathrm{OPT} - 1$.

(b) Докажите оценку $\mathrm{FFD} \leqslant \frac{3}{2}\mathrm{OPT}$. (Пусть алгоритм отработал и выдал разбиение на поднаборы. Рассмотрите поднабор с номером $\left\lceil \frac{2}{3}\mathrm{FF} \right\rceil$ и разберите случаи присутствия/отсутствия в нём числа, большего $\frac{1}{2}$.)

(c) Докажите, что если $\mathrm{P} \neq \mathrm{NP}$, то не существует полиномиального алгоритма решения задачи Bin Packing, имеющего показатель аппроксимации строго меньший $\frac{3}{2}$. (Можно воспользоваться идеей из задачи 14.)

(d) Докажите, что при сколь угодно больших n существуют наборы $(a_1, \ldots, a_n)$, для которых $\mathrm{FF} \geqslant \frac{3}{2}\mathrm{OPT}$.

(e) Докажите, что при сколь угодно больших n существуют наборы $(a_1, \ldots, a_n)$, для которых $\mathrm{FF} \geqslant \frac{5}{3}\mathrm{OPT}$.

(f) Докажите, что при сколь угодно больших n существуют наборы $(a_1, \ldots, a_n)$, для которых $\mathrm{FF} \geqslant 1.67\mathrm{OPT}$.

Любопытно, насколько можно улучшить оценки, которые Вам предлагалось доказать в этой задаче. Во-первых, известно, что $\mathrm{FFD} \leqslant \frac{11}{9}\mathrm{OPT} + \frac{2}{3}$, и это неравенство в общем случае неулучшаемо. Во-вторых, $\mathrm{FFD} \leqslant \left\lceil \frac{17}{10}\mathrm{OPT} \right\rceil$, причём константа $\frac{17}{10}$ также неулучшаема. Известно также, что никакой *онлайновый* алгоритм решения задачи Bin Packing не может иметь показатель аппроксимации лучше, чем 1.54.

Задача 66. Пусть в последовательности $a_1, \ldots, a_n$ встречаются всего m различных значений $s_1, \ldots, s_m$. Пусть b_i — количество раз, которое значение s_i встречается. Пусть

$$\{T_1, \ldots, T_N\} := \left\{ (k_1, \ldots, k_m) \in \mathbb{N}_0^m \mid \sum_{i=1}^{m} k_i s_i \leqslant 1 \right\}$$

— способы, которыми можно сформировать один корректный поднабор. Пусть $T_j = (t_{j,1}, t_{j,2}, \ldots, t_{j,m})$. Докажите, что решение задачи Bin Packing эквивалентно решению следующей задачи ЦЛП:

$$\begin{cases} \sum_{j=1}^{N} x_j \to \min \\ \sum_{j=1}^{N} t_{j,i} x_j \geqslant b_i \quad (i = 1, \ldots, m) \\ x_j \in \mathbb{N}_0 \end{cases}$$

Задача 67. Переформулируйте задачу Bin Packing как частный случай задачи о покрытии множеств (возможно, с экспоненциальным числом множеств).

Задача 68. Напомним, что в задаче о рюкзаке требуется из множества объектов с заданными целочисленными весами и целочисленными стоимостями выбрать подмножество, вес которого ограничен заданным числом $W_{\max}$, а стоимость — максимально возможная. Пусть $w_1, \ldots, w_n$ — веса объектов, а $c_1, \ldots, c_n$ — стоимости. Убедитесь, что следующий алгоритм динамического программирования решает задачу о рюкзаке за время $O(n^2 c_{\mathrm{OPT}})$, где c_{OPT} — оптимальное значение стоимости.

- Положить $S := \{(0, 0)\}$ и $j := 1$.
- Положить $S_{\mathrm{new}} := S \cup \{(w + w_j, c + c_j) \mid (w, c) \in S,\ w + w_j \leqslant W_{\max}\}$.
- Удалить из S_{new} все такие пары (w, c), для которых существует пара (w, c') с $c' > c$ либо существует пара (w', c) с $w' < w$.
- Положить $j := j + 1$, $S := S_{\mathrm{new}}$ и, если $j \leqslant n$, то перейти к шагу 2.

(Считаем, что алгоритм для каждой пары (w, c) сохраняет подмножество объектов, на котором достигаются эти значения веса и стоимости.)

Как изменится время работы алгоритма, если при вычислениях округлять стоимости до чисел, кратных 10? Как при этом соотносятся стоимость полученного алгоритмом решения и оптимальная стоимость?

Паросочетания

Задача 69. Покажите, что задачу bin packing можно свести к задаче о паросочетании, если размеры всех грузов строго больше одной трети от размера контейнера.

Задача 70. Назовём *ориентированным паросочетанием* в орграфе такой набор дуг M, что из каждой вершины графа выходит ровно одна дуга M и заходит ровно одна дуга M. Придумайте сведение задачи поиска ориентированного паросочетания в произвольном орграфе к поиску совершенного паросочетания в двудольном графе.

Задача 71. *k-фактором* простого неориентированного графа называется его k-регулярный остовный подграф (частным случаем этого понятия при $k = 1$ является совершенное паросочетание). Для *фиксированного* k постройте полиномиальный алгоритм, который для заданного графа G искал бы k-фактор (или удостоверялся, что такового нет). Указание: рассмотрите вспомогательный граф G_k, который для каждой вершины $v \in V(G)$ содержит k её копий, и к тому же содержит по две дополнительные вершины для каждого ребра $e \in E(G)$ (итого $k \cdot |V(G)| + 2 \cdot |E(G)|$ вершин). Как в графе G' проводятся рёбра, догадайтесь самостоятельно. Далее запустите на G_k поиск совершенного паросочетания.

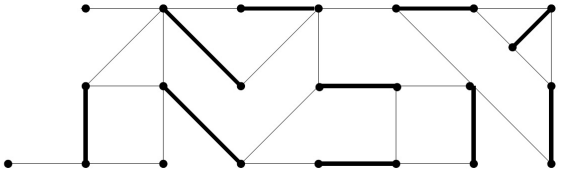
Останется ли этот алгоритм полиномиальным, если считать k частью входных данных алгоритма?

Задача 72. Пусть в графе G выбрано паросочетание M с m рёбрами, но при этом в G существует паросочетание с n ($n > m$) рёбрами. Покажите, что в графе G можно выбрать $(n - m)$ не пересекающихся по вершинам увеличивающих путей относительно M.

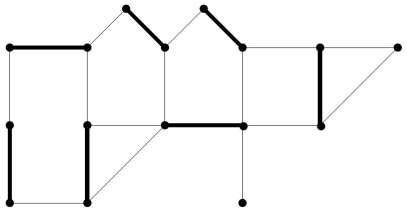
Задача 73. Пусть G — простой граф. Постройте граф G', такой, что в G' есть совершенное паросочетание тогда и только тогда, когда в G есть паросочетание мощности k.

Задача 74. В следующих графах, отталкиваясь от указанных первоначальных паросочетаний, найдите паросочетания большей мощности при помощи алгоритма Эдмондса:

(i)



(ii)



(iii)