Московский физико-технический институт (НИУ)
Физтех-школа прикладной математики и информатики
Сложность вычислений, осень 2025
Темы индивидуальных проектов

Темы проектов не могут повторяться внутри одной группы, при этом каждая тема может быть выбрана не более, чем **двумя** студентами с курса. Данное ограничение не распространяется на существенно разные проекты по одной теме, такие как обзоры одной области с подробным изложением разных теорем, реализации разных алгоритмов для одной задачи и т. д. В этом случае надо заранее указать спецификацию. Также возможно выполнение проекта в парах, в таком случае ограничение также не действует, но ожидается больший объём работы.

При выполнении любого проекта требуется написать оригинальный самодостаточный текст на русском языке на заданную тему. Принимаемые форматы уточняйте у своего семинариста, заведомо принимается PDF-файл, свёрстанный в LaTeX, также можно создать проект в OverLeaf и прислать не текст, а ссылку. Текст обязательно должен содержать:

1. Аннотацию (в LaTeX — окружение abstract), в которой кратко описано, о чём текст и что в нём самое главное.

2. Введение, в котором более подробно описано содержание текста и его место в литературе по теме. Здесь нужно сформулировать основные понятия и результаты работы, по возможности, без технических подробностей, рассказать об истории вопроса и значении результатов, описать связи работы с другими, привести близкие результаты из литературы.

3. Техническую часть, содержащую строгие определения всех использованных понятий (кроме общеизвестных), формулировки теорем и вспомогательных утверждений и т. п.

4. Основную часть, содержащую полное доказательство хотя бы одной нетривиальной теоремы и анализ программы для алгоритмических проектов.

5. Список источников хотя бы из двух наименований. На каждый источник должна быть хотя бы одна явная ссылка в тексте. Многие источники размещены в свободном доступе. Если вам нужна статья из платной библиотеки, обращайтесь к преподавателям. Будьте внимательны: в источниках могут быть ошибки. Также старайтесь расписывать более подробно по сравнению с источниками неочевидные переходы в доказательствах. Не допускаются прямые заимствования из других текстов, в том числе иностранных, без явного указания источника (в случае обнаружения возможны санкции вплоть до нулевой оценки за проект).

6. **Политика по использованию инструментов искусственного интеллекта.** Если вы используете какие-либо программы для обработки или генерации текста, то необходимо явно указывать, какие программы, в каком объёме и для чего именно вы использовали. При этом не допускается использовать генеративные модели

для непосредственного написания больших кусков текста, однако можно их использовать для редактуры, составления плана, изначального поиска информации и тому подобных задач. Ответственность за корректность представленной информации несёт автор текста, а не использованная программа. Если текст похож на машинно сгенерированный, то оценка может быть понижена вплоть до нуля, независимо от того, был ли он сгенерирован на самом деле (т.е. не нужно писать текст в стиле выдачи LLM).

7. Желательно соответствовать техническим требованиям, перечисленным в конце файла.

8. Какие-то вещи, специфичные для конкретных видов проектов, указаны ниже.

Предельные сроки выполнения работы: выбор темы проекта — **5 ноября 2025 г.**, сдача аннотации и плана текста — **21 ноября 2025 г.**, сдача предварительной версии — **3 декабря 2025 г.**, сдача окончательной работы — **17 декабря 2025 г.** При сдаче аннотации, плана текста и предварительной версии в срок гарантируется возможность исправить замечания и повысить оценку.

За проект ставится от 0 до 12 баллов, эта оценка входит в итоговую оценку с весом 0.2, при этом для получения итоговой оценки отлично(8–10) необходимо сдать проект хотя бы на 5. За аннотацию и план без самого проекта может быть поставлено до 3 баллов. В качестве проекта допускается сделать перевод реферата или обзора с английского (или другого) языка, но в таком случае источник должен быть указан явно, а проект оценивается не больше, чем на 6 баллов. (В этом случае, разумеется, заимствования возможны). Оценки 11 или 12 ставятся при наличии научной новизны, большого объёма и хорошей структурированности материала, либо других выдающихся достоинств. На одном из последних семинаров, либо на зачётной неделе, возможен устный доклад по проекту, который не исключает необходимости написания текста, но может повысить оценку.

Предпочтительный вариант.

0. Придумайте тему самостоятельно и согласуйте её с семинаристом. Особенно если все подходящие алгоритмические проекты уже разобрали, а вам тоже интереснее писать программу, а не доказательство сложной теоремы.

Обзорные проекты по теории. Нужно сделать обзор определений и результатов и хотя бы один из нетривиальных результатов доказать.

1. Виды полиномиальных сводимостей вычислительных задач и соотношения между разными сводимостями. (Можно оттолкнуться от [32, прил. B]).

2. **EXP**-полные задачи.

3. Задачи из $\mathbf{NP} \cap \mathbf{coNP}$, про которые неизвестно полиномиальных алгоритмов.

4. Задача об изоморфизме графов: полиномиальные алгоритмы для отдельных классов, лучшие алгоритмы для общего случая (можно без разбора квазиполиномиального алгоритма Бабаи).

5. Задача об изоморфизме графов: класс **GI** и полные задачи в нём. (Можно оттолкнуться от книги [38]).

6. Задача об изоморфизме графов: соотношения класса **GI** с другими классами (вложимость в **SPP** [11], $\mathbf{ZPP}^{\mathbf{NP}}$ [10] и др.).

7. Релятивизация вопросов $\mathbf{NP} \stackrel{?}{=} \mathbf{PSPACE}$, $\mathbf{PSPACE} \stackrel{?}{=} \mathbf{EXP}$, $\mathbf{P} \stackrel{?}{=} \mathbf{NP} \cap \mathbf{coNP}$, коллапсирование полиномиальной иерархии и т. д. (Построение оракулов с теми или иными ответами).

8. Предположительно псевдослучайные семейства функций в классе $\mathbf{TC}^0$. (Напр., по работам [12, 44])

9. Временны́е классы с очень большими ограничениями на время (галактические классы): **ELEMENTARY**, **Tower** и т. д. (Можно опереться на статью [53]).

10. Аналоги полиномиальной иерархии для других временны́х границ, например, экспоненциальная иерархия.

11. Классы, связанные с чётностью числа сертификатов: $\oplus\mathbf{P}$, $\oplus$Ł, $\oplus\mathbf{EXP}$ и др.

12. Классы, связанные с малым числом сертификатов: **Few**, **FewP**, **FewEXP** и др.

13. Классы, связанные с разрывом между числами сертификатов: **GapL**, **GapP** и др.

14. Иерархия подсчёта (**CH**).

15. «Стивов класс» **SC** языков, распознаваемых за полиномиальное время на полилогарифмической памяти. Теорема: $\mathbf{RL} \subset \mathbf{SC}$.

16. Колмогоровская сложность с ограничением на вычислительные ресурсы.

17. Сложностные классы с оракулом, вычисляющим колмогоровскую сложность ($\mathbf{P}^K$, $\mathbf{BPP}^{KT}$).

18. Коммуникационная сложность: классы $\mathbf{P}^{\text{cc}}$, $\mathbf{NP}^{\text{cc}}$, $\mathbf{BPP}^{\text{cc}}$ и т. д.

19. Алгебраическая сложность (можно оттолкнуться от книг Разборова [66] или Арора–Барака [8, гл. 16]).

20. Классы $\mathbf{SAC}^n$ и **LOGCFL**.

21. Однозначные вычисления: классы **UP**, **UL** и т. д.

22. Сложность сжато представленных задач (Succinctly Encoded Problems).

23. Классы задач поиска с гарантированным результатом (класс **TFNP** и его подклассы **PPP**, **PPAD** и др.)

24. Сложность поиска неподвижных точек (класс **FIXP** и др., можно оттолкнуться от [21]).

25. Алгоритмы для локальных вычислений (по работам Р. Рубинфельд и др.)

26. Параметризованная сложность: класс **FPT**, параметризованная сводимость, **W**-иерархия. (Можно искать информацию в книге [19]).

27. Задачи с предусловием (Promise problems) и основанные на них сложностные классы. (Можно оттолкнуться от [27]).

28. Эвристически решаемые задачи и соответствующие сложностные классы: **HeurP**, **HeurBPP** и др.

29. Fine-grained complexity: точные оценки времени для решения задач внутри **P**.

30. Различные варианты экспоненциальной гипотезы (ETH, SETH и т.п.) и следствия из них.

31. Аксиоматизация вопроса о $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ (Можно оттолкнуться от [1]).

Изложение доказательства некоторой теоремы. Нужно записать формулировку и доказательство выбранной теоремы, а также рассказать о её значении и связи с различными утверждениями. Разумеется, не требуется доказывать теорему самостоятельно или придумывать новое доказательство, но сам текст доказательства должен быть оригинальным. Отдельные технические леммы можно не доказывать, приводя точную ссылку на доказательство (не только источник, но и конкретные страницы или номер утверждения в нём). Подобные пропуски не должны влиять на восприятие сути доказательства. В списке приведены ссылки на первоисточники, но необязательно использовать именно их при подготовке текста.

32. Теорема Блюма [17]: для некоторых задач не существует оптимальных программ, как бы ни мерялась сложность.

33. Теорема Рейнгольда [49]: проверка на наличие пути в неориентированном графе лежит в **L**.

34. Теорема Разборова о размере монотонной схемы для решения задачи о клике [67].

35. Теорема Айтаи–Фюрста–Сакса–Сипсера [5, 23]: $\oplus \notin \mathbf{AC}^0$.

36. Теорема Алона–Боппаны о размере монотонной схемы для решения задачи о клике [6] (усиление теоремы Разборова).

37. Теоремы Разборова [67] и Тардош [57] о размере монотонной схемы для решения задачи о совершенном паросочетании.

38. Теорема Раца–Вигдерсона о глубине монотонной схемы для решения задачи о совершенном паросочетании [47].

39. Теоремы Яо [64] и Хостада [29] об экспоненциальной нижней оценке на размер схем константной глубины для функций чётности и большинства.

40. Теорема Линиаля–Мансура–Нисана о малом отклике функций из $\mathbf{AC}^0$ [43].

41. Теорема Разборова–Смоленского [55, 68]: $\mathrm{mod}_p \notin \mathbf{ACC}^0(q)$ для различных простых p и q.

42. Теорема Разборова–Рудича о естественных доказательствах [48].

43. Теорема Уильямса [61]: $\mathbf{NEXP} \not\subset \mathbf{ACC}^0$.

44. Теорема Ааронсона–Вигдерсона о барьере алгебризации [2].

45. Теорема Импальяццо–Вигдерсона [34]: если в классе $\mathbf{E}$ есть язык, не распознаваемый схемами субэкспоненциального размера, то $\mathbf{P} = \mathbf{BPP}$.

46. Теорема Кабанеца–Импальяццо [35]: если задачу об эквивалентности схем можно решить за субэкспоненциальное время, то либо $\mathbf{NEXP} \not\subset \mathbf{P}/_{\mathrm{poly}}$, либо перманент нельзя вычислить экспоненциальными схемами.

47. Теорема Хартманиса–Сьюэльсона–Иммермана [28]: $\mathbf{E} \neq \mathbf{NE}$ тогда и только тогда, когда $\mathbf{NP} \setminus \mathbf{P}$ содержит разреженные языки.

48. Теорема Бейгеля–Рейнгольда–Шпилмана [13] о замкнутости $\mathbf{PP}$ относительно пересечения и различных сводимостей.

49. Теорема Айелло–Голдвассер–Хостада [4]: для некоторого оракула B выполнено $\mathbf{IP}^B \not\subset \mathbf{PH}^B$.

50. Теорема Фортноу–Сипсера [22]: для некоторого оракула C выполнено $\mathbf{coNP}^C \not\subset \mathbf{IP}^C$.

51. $\mathbf{NP}$-трудность приближённого решения метрической задачи коммивояжёра с точностью выше 1.001 (годятся конструкции Пападимитриу–Вемпалы [45], Ламписа [40], Карпинского–Ламписа–Шмида [36] и др.)

52. $\mathbf{NP}$-трудность приближённого решения задачи о вершинном покрытии с точностью выше 1.05 (годятся конструкции Белларе–Голдрайха–Судана [14], Хостада [31], Динур–Сафры [18] и др.)

53. Трудность приближённого решения задачи о клике с точностью n^α для какого-нибудь $\alpha \in (0, 1)$ в каком-нибудь сложностном предположении (возможно, более слабом, чем $\mathbf{P} \neq \mathbf{NP}$). (Например, годятся результаты Белларе–Голдрайха–Судана [14], Хостада [30], Энгебретсена–Хольмерина [20], Хота [37] и Цукермана [65]).

54. Теорема Яннакакиса [63] об $\mathbf{NP}$-полноте задачи поиска обхода графа с минимальным наполнением. В работе требуется ясно объяснить, как эта задача связана с LU-разложением.

55. Теорема Уильямса [60] о моделировании вычислений за время t на памяти $O(\sqrt{t}\log t)$.

Обзоры сложности задач в некоторой области математики. Дана некоторая область математики. Нужно составить выборку задач из этой области и классифицировать их насколько возможно по различным сложностным классам: **P**, **NP**, **coNP**, **PH**, **PSPACE** и т. д. Хотя бы для одной из задач нужно доказать полноту в соответствующем классе. Хорошим стартовым пунктом может служить классификация задач в [24, 52].

56. Частные случаи задач из теории графов (для планарных графов, двудольных графов и т.д.)

57. Теория гиперграфов.

58. Теория расписаний.

59. Биматричные игры.

60. Расположение ферзей на шахматной доске.

61. Различные головоломки и настольные или видеоигры. (Хорошим источником будут труды конференции FUN with Algorithms).

62. Алгебра и теория чисел.

63. Теория решёток.

64. Теория кодирования.

65. Конечные автоматы и формальные языки.

66. Запросы к базам данным.

67. Пропозициональные модальные логики.

68. Многозначные логики.

Исследовательские проекты. Требуется провести мини-исследование и изложить его результаты.

69. Нахождение ошибки в одном из «доказательств» утверждения о (не)равенстве **P** и **NP**, например, приведённых на странице `http://www.win.tue.nl/~gwoegi/P-versus-NP.htm`. Можно также в качестве проекта составить по возможности полный список таких попыток после 2016 года.

70. Исследование сложности какой-либо компьютерной или настольной игры. Игра не должна содержать случайности, либо алгоритм должен заранее знать все розыгрыши случайных величин, либо сама классификация должна быть связана с каким-либо вероятностным классом. Желательно, чтобы игрок был один, т. е. чтобы в игре требовалось выполнить некоторое задание. Пример такой игры — «Лягушки-непоседы» (“Hoppers”). Если в игре зафиксирован размер поля, то её нужно обобщить на произвольный размер.

71. Поиск запрещённых миноров для какого-нибудь топологического свойства графов, сохраняющегося при стягивании и удалении рёбер.

72. Исследование по истории науки: рассказ об истории продвижений в какой-либо области сложности вычислений. В исследовании должно быть рассказано о содержании идей, их развитии и сравнительной важности, но не требуется излагать детали доказательств.

Методические проекты. Требуется придумать набор разнообразных задач, лежащих в заданном сложностном классе, либо семействе похожих друг на друга классов. Классификация задач не должна быть ни тривиальной, ни слишком сложной, но сами задачи должны быть не заимствованы из стандартных учебников и не похожи одна на другую. Возможны заимствования из малоизвестных книг, лекционных конспектов или домашек курсов различных университетов, разумеется, с указанием источника. Необходимое количество задач заранее не фиксируется.

Возможные списки классов:

73. **L** и **NL**.

74. **BPP**, **RP**, **coRP**, **ZPP**.

75. #**P** (не полученные из **NP**-полных).

76. **PP**.

77. Σ_k^p и Π_k^p для $k \geqslant 2$ (не из задач **NP**-оптимизации и не из статьи [52]).

78. Любые классы, не изученные на курсе.

Алгоритмические проекты. Требуется описать некоторый алгоритм, доказать его корректность и имплементировать его (возможно, в частном случае) на любом языке программирования. Текст отчёта должен включать в себя описание алгоритма, его анализ и результаты тестовых запусков. В частности, нужно описать, в каких случаях алгоритм работает хорошо, а в каких — плохо. Оценивается прежде всего текст отчёта, а не текст программы. Теоретическая и практическая части оцениваются из 5 баллов каждая, 2 бонусных балла могут быть даны за любую из них.

Если доказательство корректности алгоритма очень сложное, допускается представить его как цепочку лемм и доказать только некоторые из них, либо, наоборот, написать полностью теоретический проект по теме.

Многие темы связаны с построением приближённых алгоритмов, хорошими источниками по ним являются книги [33, 58, 62], там же можно найти и альтернативную тему, не указанную в этом списке.

Отчет о тестовых запусках должен включать в себя:

- **Описание наборов тестов.**
  Делать выводы об алгоритме по его работе на одном тесте невозможно. Поэтому тестирование ваших алгоритмов должно производиться на наборах тестов. Тест-сет должен покрывать некоторый класс ситуаций. Например, для алгоритмов на графах можно взять подобные наборы тестов:

– Все простые графы на не более, чем 8 вершинами.
– Случайные графы $G(n, p)$.
– Дистанционные, планарные, кнезеровские графы.
– Реальные графы дорог, web-графы, любые другие реальные данные.

Какие-то наборы тестов можно взять из `http://users.cecs.anu.edu.au/~bdm/data/graphs.html`. В задачах с более сложными входными данными рекомендуется рассмотреть случайно сгенерированные различными способами данные. Размер набора тестов должен быть достаточным для стабильности измеряемых статистик. Либо должна быть описана существенная причина, почему нельзя увеличить тест-сет до адекватного размера. Наборы тестов должны быть сгруппированы по существенным для алгоритма признакам. Тест-сет «Графы с нечетным количеством ребер» — плохой набор почти для любого алгоритма, «$G(n, \frac{10}{n})$» и «Прямое произведение двух небольших графов» могут быть адекватными тест-сетами.

- **Анализ запусков.**
Агрегированные по тест-сетам статистики данных более значимы, чем точные значения по конкретным запускам. В первую очередь нужно вести их анализ (считать минимум / максимум / среднее значение / медиану / различные квантили / эмпирическое распределение таких величин, как время / память / точность / вероятность). Работу на конкретных тестах имеет смысл рассматривать, если результаты сильно выбиваются или приближаются к теоретическим оценкам, или же в случае проведения визуализации алгоритма или подробного анализа его шагов. Если есть возможность реализовать наивный/оптимальный/детерминированный алгоритм для решения той же задачи, то это нужно сделать и сравнить результаты разных алгоритмов. Отображение данных на графиках оценивается лучше, чем просто набор табличек с данными.

- **Выводы, сделанные по полученным данным.**
На каких данных алгоритм работает хорошо или плохо. Сравнение с работой базового алгоритма. К каким параметрам или ситуациям наиболее чуствительны интересующие вас величины. Достигаются ли теоретические оценки на тест-сетах или дело обстоит лучше/хуже, чем в теории. Попытки объяснения замеченных аномалий.

Список тем для алгоритмических проектов

79. **Вероятностная проверка простоты без ошибок**

Требуется проверить простоту числа **ZPP**-алгоритмом. Используйте любые два теста простоты с ошибками в противоположные стороны. (Например, тесты Миллера–Рабина и Эйдлмана–Хуана [3]).

80. **Покрытие множества**

Дано множество M, а также набор его подмножеств $\{S_1, S_2, \ldots, S_n\}$. Требуется найти минимальный набор из заданных подмножеств, такой что их объединение равно (покрывает) M.

Например, если $M = \{1, 2, 3, 4, 5\}$ и $S_1 = \{1, 2, 3\}$, $S_2 = \{2, 3, 4\}$, $S_3 = \{1, 3, 5\}$, то минимальный покрывающий набор состоит из множеств S_2 и S_3.

Задание

(а) Докажите, что проверка существования покрытия из не более чем k множеств является **NP**-полной.

(б) Постройте алгоритм, дающий $\log n$-приближение, и имплементируйте его.

(в) Предположим, что каждый элемент множества M присутствует не более чем в k подмножествах. Постройте алгоритм, дающий k-приближение (подсказка: сведите к задаче линейного программирования).

81. **Упаковка по корзинам**

Даны числа $a_1, \ldots, a_n \in (0, 1]$. Требуется разбить их на минимальное число групп, так чтобы сумма чисел в каждой группе не превышала 1.

Задание

(а) Докажите, что для любого $\varepsilon > 0$ задача о $(\frac{3}{2} - \varepsilon)$-приближении является **NP**-трудной.

(б) Для любого $\varepsilon > 0$ постройте алгоритм, дающий $(1 + \varepsilon)OPT + 1$ корзин для оптимального значения OPT.

(в) Имплементируйте этот алгоритм.

82. **Дерево Штейнера**

Дан ненаправленный связный граф $G = (V, E)$, в нём выделено множество вершин V_0. Также имеются веса на рёбрах $w : E \to \mathbb{R}_+$. Требуется найти дерево T минимального веса, покрывающее все вершины V_0.

Задание

(а) Докажите, что проверка существования дерева веса не более k является **NP**-полной.

(б) Сведите задачу к метрической версии (в метрической версии исходный граф является полным и верно $w(x, z) \leqslant w(x, y) + w(y, z)$.

(в) Постройте алгоритм, дающий 2-приближение для метрической версии задачи, а также имплементируйте его.

83. **Дерево Штейнера: покупка или аренда рёбер**

Дан ненаправленный связный граф $G = (V, E)$, в нём выделена вершина r и множество вершин V_0. Также имеются веса на рёбрах $w : E \to \mathbb{R}_+$. Для дерева T, покрывающего r и все вершины V_0, издержки определяются так:

(а) Если через ребро e проходит $k < M$ кратчайших путей от вершин V_0 к корню r, то ребро даёт вклад kw_e (каждая из вершин «арендует» ребро).

(б) Если через ребро e проходит $k \geqslant M$ кратчайших путей от вершин V_0 к корню r, то ребро даёт вклад Mw_e (вершины в совокупности «покупают» ребро).

Задание

(а) Докажите, что проверка существования дерева веса не более k является **NP**-полной.

(б) Сведите задачу к метрической версии.

(в) Опираясь на алгоритм для поиска дерева Штейнера, постройте алгоритм, дающий 4-приближение для метрической версии задачи, а также имплементируйте его (для поиска дерева можно использовать библиотечную функцию).

84. **Остовное дерево минимальной степени**

Дан ненаправленный связный граф $G = (V, E)$, требуется найти его остовное дерево, т. е. подграф, который является деревом и в котором участвуют все вершины, такое что максимальная степень вершины в этом дереве минимально возможна.

Задание

(а) Докажите, что проверка существования остовного дерева максимальной степени не более k является **NP**-полной для любого фиксированного $k \geqslant 2$.

(б) Пусть минимально возможная степень остовного дерева равна Δ. Постройте полиномиальный алгоритм поиска остовного дерева степени Δ или $\Delta + 1$, и имплементируйте его.

85. **Метрическая задача коммивояжёра-1**

В общем случае задача коммивояжёра ставится так: дан граф $G = (V, E)$ и веса на рёбрах $w : E \to \mathbb{R}_+$. Требуется найти гамильтонов цикл минимального веса.

В метрическом случае граф полный, а функция весов метрическая (то есть для любых трёх вершин x, y и z выполнено $w(x, z) \leqslant w(x, y) + w(y, z)$).

Задание

(а) Докажите, что для стандартной задачи коммивояжёра не существует константных алгоритмов приближения, если $\mathbf{P} \neq \mathbf{NP}$.

(б) Постройте алгоритм, дающий 2-приближение для метрической задачи коммивояжёра на основе остовного дерева.

(в) Постройте алгоритм, дающий 1.5-приближение для метрической задачи коммивояжёра на основе остовного дерева и паросочетания.

(г) Имплементируйте второй алгоритм.

86. **Метрическая задача коммивояжёра-2**

Пусть теперь надо найти гамильтонов путь между двумя фиксированными вершинами минимального веса.

Задание

(а) Постройте алгоритм, дающий $\frac{5}{3}$-приближение для указанной задачи, также на основе остовного дерева и паросочетания.

(б) Имплементируйте этот алгоритм.

87. **(1,2)-Задача коммивояжёра-1**

Пусть все веса на рёбрах в задаче коммивояжёра равны 1 или 2. Имплементируйте аппроксимационный алгоритм Пападимитриу–Яннакакиса [46] с точностью приближения $\frac{7}{6}$.

88. **(1,2)-Задача коммивояжёра-2**

Пусть все веса на рёбрах в задаче коммивояжёра равны 1 или 2. Имплементируйте аппроксимационный алгоритм Бермана–Карпински [15] с точностью приближения $\frac{8}{7}$.

89. **Евклидова задача коммивояжёра**

Пусть все вершины графа помещены в евклидово пространство, а веса на рёбрах получены как евклидовы расстояния между вершинами. Имплементируйте какой-нибудь алгоритм приближённого решения с точностью выше 1.25. (Согласно теореме Ароры [7], существует полиномиальный алгоритм для любой точности $1 + \varepsilon$).

90. **Задача о деревенском почтальоне-1**

Задача о деревенском почтальоне (RURAL POSTMAN, RPP) ставится так: дан граф $G = (V, E)$, веса рёбер $w : E \to \mathbb{R}_+$ и множество рёбер $R \subset E$. Требуется найти цикл минимального суммарного веса, хотя бы один раз проходящий через каждое ребро из R.

Задание

(а) Докажите **NP**-полноту RPP.

(б) Докажите полиномиальную разрешимость RPP в случае, когда граф, образованный рёбрами R, имеет константное число компонент связности c (степень полинома может зависеть от c).

(в) Имплементируйте этот алгоритм.

91. **Задача о деревенском почтальоне-2**

Имплементируйте любой аппроксимационный алгоритм для задачи RPP с точностью $\frac{3}{2}$.

92. **Задача о рюкзаке**

Имеется набор из n предметов. У каждого предмета есть положительный вес w и стоимость c. Также дано неотрицательное число W — вместимость рюкзака. Требуется найти такое множество предметов M, чтобы оно помещалось в рюкзак, и суммарная стоимость предметов была максимальна. То есть:

$$\sum_{x \in M} c(x) \to \max$$
$$s.t. \sum_{x \in M} w(x) \leqslant W$$

Задание

Постройте полиномиальную схему приближения для данной задачи. То есть необходимо придумать и имплементировать алгоритм, который получает на вход экземпляр задачи о рюкзаке, а также произвольное (рациональное) $\varepsilon > 0$, и находит $(1 + \varepsilon)$-приближенное решение. Алгоритм должен работать за полиномиальное время относительно размера исходной задачи и $1/\varepsilon$.

93. **Задача об оптимальном расписании**

Имеется множество работ J и множество машин M. Также задана функция $p : J \times M \to \mathbb{R}_+$. Значение p_{ij} означает время выполнение i-й работы на j-й машине.

Требуется построить распределение работ по машинам так, чтобы все работы были выполнены и чтобы конечное время выполнения всех работ было минимально. Иначе говоря, требуется найти функцию $x : J \times M \to \{0, 1\}$ такую, что:

$$\max_{j \in M} \sum_{i} x_{ij} p_{ij} \to \min$$
$$s.t. \ \forall i \sum_{j \in M} x_{ij} = 1$$

Рассмотрим частный случай задачи, когда производительности всех машин одинаковы, то есть $p_{ij} = p_i$.

Задание

(а) Докажите, что задача об оптимальном расписании является **NP**-полной (подразумевается задача поиска расписания длительности не более k).

(б) Постройте алгоритм, дающий $\frac{4}{3}$-приближение для случая машин одинаковой производительности, а также имплементируйте его.

94. **Задача о наименьшей надстроке**

Пусть задано множество строк $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ над конечным алфавитом. Требуется найти такую строку ω, что все строки из множества A являются подстроками ω и длина ω минимальна.

Задание

(а) Докажите, что задача является **NP**-полной (подразумевается задача поиска надстроки длины не больше k).

(б) Постройте алгоритм, дающий 4-приближение.

(в) Известно (но не доказано), что жадный алгоритм (который на каждом шаге находит строки с наибольшим перекрытием и склеивает их) даёт 2-приближение. Имплементируйте его и убедитесь, что на коротких строках алгоритм действительно даёт 2-приближение.

95. **Вершинное покрытие**

Известно, что для задачи вершинного покрытия существует простой алгоритм, дающий 2-приближение. Он состоит в том, чтобы найти максимальное паросочетание в графе и взять по вершине из каждого ребра.

Рассмотрим взвешенный вариант задачи. То есть дан граф $G = (V, E)$, а также функция $w : V \to \mathbb{R}_+$. Требуется найти множество минимального веса, покрывающее все рёбра графа, т. е. такое $U \subset V$, что для любого $(u, v) \in E$ верно $u \in U$ или $v \in U$ и сумма $\sum_{u \in U} w(u)$ минимальна.

Задание

Требуется построить алгоритм дающий 2-приближение для взвешенной задачи вершинного покрытия, а также имплементировать его.

Подсказка: рассмотрите частный случай графа, в котором $w(u) = c \cdot \deg(u)$. Ответ для таких графов строится просто, остается декомпозировать исходную задачу на задачи такого вида.

96. **3-раскрашиваемость**

Требуется имлементировать алгоритм проверки, можно ли покрасить граф из n вершин в 3 цвета, за время $O(c^n)$ для $c \leqslant 2$. Реализация алгоритма для $c > 1.8$ принесёт не больше 7 баллов, для меньших c — вплоть до 12 при выборе наилучшего известного алгоритма.

97. **Раскраска 3-раскрашиваемого графа**

Известно, что граф можно раскрасить в 3 цвета. Имплементируйте полиномиальный алгоритм, красящий граф в $O(\sqrt{n})$ цветов. (Известно много алгоритмов разной степени сложности, в том числе с гораздо лучшими оценками. Чем более простой алгоритм вы используете, тем больше нужно рассказать о том, как вы ищете и анализируете тестовые примеры).

98. **Поиск максимальной клики**

Требуется найти клику максимального размера в некотором графе. Можно имплементировать любую эвристику.

99. **Поиск максимального разреза**

Требуется разбить вершины графа на 2 группы, между которыми проведено максимально возможное число рёбер. Имплементируйте алгоритм Гёманса–Уильямсона[26] поиска приближённо максимального разреза с точностью 0.878.

100. **Поиск наименее плотного разреза**

Требуется разбить вершины графа на 2 группы, так чтобы отношения числа рёбер между ними к размеру меньшей группы было как можно меньше. Имплементируйте алгоритм Лейтона–Рао [41] приближённого решения с точностью $O(\log n)$. (Можно и алгоритм Ароры–Рао–Вазирани [9] с точностью $O(\sqrt{\log n})$, но это значительно сложнее).

101. **Поиск наименьшего доминирующего множества**

Доминирующим множеством в графе $G = (V, E)$ называется множество $M \subset V$, такое что любая вершина v либо лежит в M, либо соединена ребром с одной из вершин, лежащих в M.

Задание

(а) Докажите, что задача поиска наименьшего доминирующего множества **NP**-трудная;

(б) Имплементируйте какой-нибудь алгоритм поиска наименьшего доминирующего множества, работающий за $O(c^n)$ для $c < 1.9$.

102. **Поиск наименьшего связного доминирующего множества**

Доминирующим множеством в графе $G = (V, E)$ называется множество $M \subset V$, такое что любая вершина v либо лежит в M, либо соединена ребром с одной из вершин, лежащих в M.

Задание

(а) Докажите, что задача поиска наименьшего связного доминирующего множества **NP**-трудная;

(б) Имплементируйте какой-нибудь алгоритм поиска наименьшего связного доминирующего множества, работающий за $O(c^n)$ для $c < 1.95$.

103. **Поиск наименьшего разрезающего циклы набора рёбер**

Разрезающим циклы набором рёбер (feedback arc set) в ориентированном графе $G = (V, E)$ называется множество $F \subset E$, такое что любой ориентированный цикл в G содержит хотя бы одно ребро из F.

Задание

(а) Докажите, что задача поиска разрезающего циклы набора рёбер **NP**-трудная;

(б) Имплементируйте какой-нибудь полиномиальный алгоритм приближённого решения этой задачи со степенью приближения, равной $o(n^{\varepsilon})$ для любого $\varepsilon > 0$, где n — число вершин в графе.

104. **Поиск наибольшего ациклического подграфа**

Пусть задан ориентированный граф $G = (V, E)$. Требуется найти наибольший по числу рёбер подграф (не обязательно индуцированный), в котором нет ориентированных циклов (maximum acyclic graph).

Задание

(а) Докажите, что задача поиска наибольшего ациклического подграфа является **NP**-трудной;

(б) Докажите, что задача поиска ациклического подграфа, имеющего размер не меньше половины от наибольшего, решается за полиномиальное время;

(в) Имплементируйте получившийся алгоритм и проанализируйте его работу.

105. **Задача о наименьшем мультиразрезе**

Пусть задан взвешенный неориентированный граф $(G = (V, E), w\colon E \to \mathbb{R}_+)$, в котором отмечены вершины $s_1, \ldots, s_k$. Требуется найти наименьший мультиразрез (minimum multicut), т. е. множество рёбер E' наименьшего суммарного веса, такое что отмеченные вершины находятся в разных компонентах $(V, E \setminus E')$.

Задание

(а) Докажите, что задача поиска наименьшего мультиразреза является **NP**-трудной при $k \geqslant 3$;

(б) Докажите, что задача поиска мультиразреза, имеющего вес не больше удвоенного наименьшего, решается за полиномиальное время (есть и более точные алгоритмы);

(в) Имплементируйте получившийся алгоритм и проанализируйте его работу.

106. **Задача о наименьшем k-разрезе**

Пусть задан взвешенный неориентированный граф $(G = (V, E), w\colon E \to \mathbb{R}_+)$. Требуется найти наименьший k-разрез (minimum k-cut), т. е. множество рёбер E' наименьшего суммарного веса, такое что граф $(V, E \setminus E')$ имеет не менее k компонент связности.

Задание

(а) Докажите, что задача поиска наименьшего k-разреза является **NP**-трудной, если k задано как часть условия;

(б) Докажите, что задача поиска k-разреза, имеющего вес не больше удвоенного наименьшего, решается за полиномиальное время (есть и более точные алгоритмы);

(в) Имплементируйте получившийся алгоритм и проанализируйте его работу.

107. **Задача о размещении ненасыщаемых мощностей (UFLP)**

Пусь заданы два конечных множества F (мощности) и (потребители). Стоимость подключения потребителя i к мощности j равна c_{ij}, стоимость включения мощности j равна f_j. Требуется выбрать множество включённых мощностей $F' \subset F$ и подключить каждого потребителя к ближайшей мощности, так чтобы суммарные затраты были минимальны.

Задание

(а) Напишите явно задачу оптимизации.

(б) Докажите, что эта задача является **NP**-полной.

(в) Придумайте любой алгоритм аппроксимации этой задачи с константным множителем.

(г) Имплементируйте этот алгоритм

108. **Максимизация 2-КНФ**

Требуется найти набор значений, при котором выполнено как можно больше дизъюнктов в 2-КНФ. Имплементируйте алгоритм Гёманса–Уильямсона [26] приближённого решения с точностью 0.878 или алгоритм Левина–Ливната–Цвика [42] с точностью 0.940.

109. **Выполнимость 3-КНФ**

Постройте и имплементируйте алгоритм, про который можно доказать следующее:

(а) Он распознаёт выполнимость 3-КНФ;

(б) В случае $\mathbf{P} = \mathbf{NP}$ на выполнимых формулах он заканчивает работу за полиномиальное время.

Указание: нужно прежде всего придумать, как выполнить второе условие — это нужно воспринимать как теоретическую задачу, — а потом полученный алгоритм можно скрестить с любым другим, распознающим выполнимость.

110. **Генетический алгоритм**

Примените генетический алгоритм для решения какой-нибудь из описанных выше задач оптимизации. В этом проекте предполагается сравнительно небольшая теоретическая часть, поскольку не требуется доказывать корректность алгоритма (однако всё ещё стоит доказать **NP**-трудность решаемой задачи), поэтому практической части должно быть уделено повышенное внимание. Нужно опробовать несколько различных эвристик (функций скрещивания, мутации, способов инициализировать популяцию и т. п.), сравнить их и попытаться объяснить, почему одни работают лучше, чем другие. Не забывайте, что оценивать нужно не только качество получаемого решения, но и время работы алгоритма. В идеале работа над проектом должна строиться инкрементальным образом: вы пишете алгоритм, тестируете его, находите у него слабые места, пытаетесь исправить их с помощью новых эвристик и так далее.

Хорошим бонусом к вашей работе будет визуализация «эволюции решения». Также вы можете сравнить ваши результаты с результатами классических алгоритмов для той же задачи.

По согласованию с семинаристом вы можете решать другие трудные задачи или же использовать другие эвристические методы решения. Также можно делать отдельные проекты в паре, реализуя и один из указанных выше алгоритмов, и генетический. В этом случае снимается ограничение на различие проектов внутри одной группы.

111. **Задача о расписаниях, промышленная версия**

Выберите одну из статей [59], [16], [39] и разберитесь в постановке задачи о составлении расписания в виде целочисленной линейной программы. Решите составленную программу любым пакетом для решений линейных программ и получите готовое расписание. При необходимости можно упростить задачу, убрав некоторые производственные узлы, но это рекомендуется делать только в крайнем случае. В исследовании должно быть описано, как вы настраивали решатель, чтобы он справился с задачей. Также рекомендуется подумать о том, как переформулировать задачу для решателя (возможно с потерей оптимальности), чтобы ускорить решение.

112. **Задача об оптимальной факторизации разреженной матрицы**

Требуется самостоятельно изучить, что такое LU-разложение матрицы (это можно сделать, например, в [54, гл. 3]), и изучить статью Тарьяна [50], а также [54, гл. 6, 8], и сравнить предложенные эвристики выбора перестановок строк для повышения разреженности матриц L и U. В коде необходимо явно строить граф матрицы и ещё до выполнения факторизации на его основе предсказывать количество ненулевых элементов L и/или U. Необходимо на разных разреженных матрицах в виде графика показать, как много ненулевых элементов остаётся после разных эвристик выбора порядка строк (вершин графа). Также необходимо написать LU-разложение и проверить его корректность. Проект можно делать в

команде до 3 человек, тогда к LU-разложению добавляется задача исследовать эвристику выбора порядка строк для разложения Холецкого.

113. **Программирование в ограничениях (constraint programming)**

Изучите материал в [51, гл. 1–4, 10]. Поулчите у В. Кондратюка датасет для покраски графа и реализуйте 5–6 эвристик, предложенных в книге, а также сравните их качество для задачи покраски (график времени работы/числа цветов для разных графов и эвристик). Среди комбинаций эвристик обязательно рассмотреть базовые: Forward-check, MAC и полная поддержка согласованности после шагов backtracking, backjumping и backlabeling как альтернатива поддержке согласованности, работа с симметриями: SBDS. SBDD. Бейзлайновую покраску, которую предложено обогнать, вам предоставят.

114. **Генетические и меметические алгоритмы для задачи TSP**

Изучите материал в [25, гл. 7–8] и реализуйте выбранную схему метаэвристик для задачи TSP (датасет и бейзлайн выдаст В. Кондратюк). На хор(7) локальный поиск можно писать простым способом (брать всякий раз жадно лучшего «соседа»). На отл рекомендуется использовать подходы из [56, гл. 5]. В генетике изучите 3–4 разных подхода к мутированию, скрещиванию и отбору (можно посмотреть [56, гл. 10]). Рекомендуется сходить на семинары по дискретной оптимизации по темам «Генетика/меметика». Сравните подходы по скорости получения решения и качеству решения (графиками).

115. **Роевой алгоритм для задачи TSP**

Изучите материал в [25, гл. 10–11] и реализуйте выбранную схему метаэвристик для задачи TSP (датасет и бейзлайн выдаст В. Кондратюк). Задача несёт творческий характер: предложите, каким способом частица в PSO «хранит» свою стратегию выбора пути и как она её согласует с остальными частицами. Реализуйте муравьиный алгоритм с разными стратегиями распыления феромонов на графе. Сравните предложенные эвристики. Рекомендуется сходить на семинары по дискретной оптимизации по темам «Роевые алгоритмы».

Рекомендации по оформлению текстов в LaTeX

Часто в текстах встречаются одни и те же стандартные ошибки, связанные с оформлением. Вот некоторые рекомендации, как их избежать.

1. Обязательно проверяйте наличие ошибок и опечаток в spell checker'е (встроенный есть в онлайн-редакторе Overleaf, нужно выбрать русский язык в настройках).

2. Названия классов пишутся с начертанием `\bf` (можно также использовать команду `\mathbf{}`). Напр., **P**, **NP**. Можно подключить пакет `complexity` с опцией `classfont=bold`.

3. Выразительные названия языков пишутся с начертанием `\sf` (можно также использовать команду `\mathsf{}`). Напр., SAT, PATH, но A и S. Букву L для языков нежелательно использовать, чтобы не путать с классом **L**. Можно подключить пакет `complexity` с опцией `langfont=sanserif`.

4. Базовые математические множества пишутся с начертанием `\mathbb`. Напр., $\mathbb{N}$, $\mathbb{Z}$. При использовании $\mathbb{N}$ уточняйте, лежит ли в нём 0.

5. Одиночные переменные внутри текста нужно всё равно включать в математическую формулу. Напр., «в формуле φ содержится n переменных», а не «в формуле φ содержится n переменных».

6. После закрывающихся кавычек (не в конце предложения) должен идти пробел. Подробнее на `https://tex.stackexchange.com/questions/10670`. Можно использовать пакет `babel` с кавычками-«ёлочками», заданными командами `<<` и `>>`.

7. Все используемые определения (объектов, не изучаемых в рамках курса) должны быть сформулированы максимально строго и недвусмысленно. Если работа занимается изучением свойств каких-то объектов (или их использует), она сначала должна чётко их описать.

8. В доказательствах и формулировках не должно быть свободных переменных. По каждой должен быть написан (словесный) квантор. Напр., «для любого языка A существует слово x такое, что...».

9. Длинные математические формулы оформляйте как выключные с использованием `$$...$$`, или `\[...\]`, или окружения `equation` и ему подобных. Если нумеруете формулы, используйте `equation`.

10. Не бойтесь использовать большие открывающие и закрывающие скобки при помощи команд `\left` и `\right`. Напр., $\left(\frac{1}{2} + \frac{p}{q}\right)^3$ вместо $(\frac{1}{2} + \frac{p}{q})^3$.

11. Предпочтительные написания: `\varphi` для φ; `\varepsilon` для ε; `\varnothing` для $\varnothing$; `\rightarrow` для $\rightarrow$; `\Rightarrow` для $\Rightarrow$; `\Leftrightarrow` для $\Leftrightarrow$; `\log` для $\log$; `\mid` для $\mid$; `\colon` для $\colon$; `---` для —.

12. Как писать сравнения по модулю: `https://tex.stackexchange.com/questions/137073`.

13. Не используйте * для умножения, опускайте знак или пишите $\cdot$ через `\cdot`.

14. Не начинайте предложения с переменной. Напр., «Граф G называется d-регулярным, если...» вместо «d-регулярный граф – такой граф, что...».

15. Не пишите две переменные или более сложные формулы подряд без словесных разделителей. Напр., «пусть в φ содержится k кванторов» вместо «пусть в φ k кванторов».

16. При оформлении списка использованной литературы пользуйтесь BibTeX: `https://www.overleaf.com/learn/latex/bibliography_management_with_bibtex`, особенно если источников много и они понадобятся для других работ. На многих сайтах вместе со статьями можно найти кнопку “Export citation”, благодаря которой можно получить информацию о статье для BibTeX.

Список литературы

[1] S. Aaronson. “Is P Versus NP Formally Independent?” In: *Bulletin of the EATCS* 81 (2003), pp. 109–136. URL: http://www.scottaaronson.com/papers/indep.pdf (cit. on p. 4).

[2] S. Aaronson and A. Wigderson. “Algebrization: A New Barrier in Complexity Theory”. In: *TOCT* 1.1 (2009). URL: http://www.scottaaronson.com/papers/alg.pdf (cit. on p. 5).

[3] L. M. Adleman and M. A. Huang. “Recognizing Primes in Random Polynomial Time”. In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. 1987, pp. 462–469. URL: http://doi.acm.org/10.1145/28395.28445 (cit. on p. 8).

[4] W. Aiello, S. Goldwasser, and J. Håstad. “On the power of interaction”. In: *Combinatorica* 10.1 (1990), pp. 3–25. URL: http://dx.doi.org/10.1007/BF02122692 (cit. on p. 5).

[5] M. Ajtai. “Σ^1_1-formulae on finite structures”. In: *Annals of pure and applied logic* 24.1 (1983), pp. 1–48 (cit. on p. 4).

[6] N. Alon and R. B. Boppana. “The monotone circuit complexity of Boolean functions”. In: *Combinatorica* 7.1 (1987), pp. 1–22 (cit. on p. 4).

[7] S. Arora. “Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems”. In: *Journal of the ACM (JACM)* 45.5 (1998), pp. 753–782 (cit. on p. 11).

[8] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009 (cit. on p. 3).

[9] S. Arora, S. Rao, and U. Vazirani. “Expander flows, geometric embeddings and graph partitioning”. In: *Journal of the ACM (JACM)* 56.2 (2009), pp. 1–37 (cit. on p. 14).

[10] V. Arvind and J. Köbler. “Graph isomorphism is low for ZPP(NP) and other lowness results”. In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 2000, pp. 431–442 (cit. on p. 3).

[11] V. Arvind and P. P. Kurur. “Graph isomorphism is in SPP”. In: *Information and Computation* 204.5 (2006), pp. 835–852 (cit. on p. 3).

[12] A. Banerjee, C. Peikert, and A. Rosen. “Pseudorandom functions and lattices”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2012, pp. 719–737. URL: https://link.springer.com/content/pdf/10.1007/978-3-642-29011-4_42.pdf (cit. on p. 3).

[13] R. Beigel, N. Reingold, and D. A. Spielman. “PP Is Closed under Intersection”. In: *J. Comput. Syst. Sci.* 50.2 (1995), pp. 191–202. URL: http://dx.doi.org/10.1006/jcss.1995.1017 (cit. on p. 5).

[14] M. Bellare, O. Goldreich, and M. Sudan. “Free bits, PCPs, and nonapproximability—towards tight results”. In: *SIAM Journal on Computing* 27.3 (1998), pp. 804–915 (cit. on p. 5).

[15] P. Berman and M. Karpinski. "8/7-approximation algorithm for (1, 2)-TSP". In: *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. 2006, pp. 641–648 (cit. on p. 11).

[16] F. Blomer and H.-O. Gunther. "LP-based heuristics for scheduling chemical batch processes". In: *International Journal of Production Research* 38.5 (2000), pp. 1029–1051 (cit. on p. 17).

[17] M. Blum. "A Machine-Independent Theory of the Complexity of Recursive Functions". In: *J. ACM* 14.2 (Apr. 1967), pp. 322–336. URL: `http://port70.net/~nsz/articles/classic/blum_complexity_1976.pdf` (cit. on p. 4).

[18] I. Dinur and S. Safra. "The importance of being biased". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002, pp. 33–42 (cit. on p. 5).

[19] R. G. Downey and M. R. Fellows. *Fundamentals of parameterized complexity*. Vol. 4. Springer, 2013 (cit. on p. 4).

[20] L. Engebretsen and J. Holmerin. "Clique is hard to approximate within $n^{1-o(1)}$". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2000, pp. 2–12 (cit. on p. 5).

[21] K. Etessami and M. Yannakakis. "On the complexity of Nash equilibria and other fixed points". In: *SIAM Journal on Computing* 39.6 (2010), pp. 2531–2597 (cit. on p. 3).

[22] L. Fortnow and M. Sipser. "Are There Interactive Protocols for CO-NP Languages?" In: *Inf. Process. Lett.* 28.5 (1988), pp. 249–251. URL: `http://dx.doi.org/10.1016/0020-0190(88)90199-8` (cit. on p. 5).

[23] M. Furst, J. B. Saxe, and M. Sipser. "Parity, circuits, and the polynomial-time hierarchy". In: *Mathematical systems theory* 17.1 (1984), pp. 13–27 (cit. on p. 4). Orig. conf. vers.: "Parity, circuits, and the polynomial-time hierarchy". In: *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*. 1981, pp. 260–270.

[24] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979 (cit. on p. 6). Пер.: М. Гэри и Д. Джонсон. *Вычислительные машины и труднорешаемые задачи*. Москва: Мир, 1982.

[25] M. Gendreau, J.-Y. Potvin, et al. *Handbook of metaheuristics*. Vol. 2. Springer, 2010 (cit. on p. 18).

[26] M. X. Goemans and D. P. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* 42.6 (1995), pp. 1115–1145 (cit. on pp. 14, 16).

[27] O. Goldreich. "On Promise Problems (a survey in memory of Shimon Even [1935-2004])". In: *Electronic Colloquium on Computational Complexity (ECCC)* 018 (2005). URL: `http://www.wisdom.weizmann.ac.il/~oded/prpr.html` (cit. on p. 4).

[28] J. Hartmanis, V. Sewelson, and N. Immerman. "Sparse sets in $\mathbf{NP} - \mathbf{P}$: **EXPTIME** versus **NEXPTIME**". In: *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. 1983, pp. 382–391 (cit. on p. 5).

[29] J. Hastad. "Almost optimal lower bounds for small depth circuits". In: *Proceedings of the eighteenth annual ACM symposium on Theory of computing*. 1986, pp. 6–20 (cit. on p. 4).

[30] J. Håstad. "Clique is hard to approximate within $n^{1-\varepsilon}$". In: *Electronic Colloquium on Computational Complexity (ECCC)* 4.38 (1997). URL: `http://eccc.hpi-web.de/eccc-reports/1997/TR97-038/index.html` (cit. on p. 5).

[31] J. Håstad. "Some optimal inapproximability results". In: *Journal of the ACM (JACM)* 48.4 (2001), pp. 798–859 (cit. on p. 5).

[32] L. A. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2002. URL: `http://dx.doi.org/10.1007/978-3-662-04880-1` (cit. on p. 2).

[33] D. S. Hochbaum, ed. *Approximation algoritmas for* **NP***-hard problems*. PWS, 1997 (cit. on p. 7).

[34] R. Impagliazzo and A. Wigderson. "$\mathbf{P} = \mathbf{BPP}$ if $\mathbf{E}$ requires exponential circuits: Derandomizing the XOR lemma". In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. 1997, pp. 220–229 (cit. on p. 5).

[35] V. Kabanets and R. Impagliazzo. "Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds". In: *Computational Complexity* 13.1-2 (2004), pp. 1–46. URL: `http://dx.doi.org/10.1007/s00037-004-0182-6` (cit. on p. 5).

[36] M. Karpinski, M. Lampis, and R. Schmied. "New inapproximability bounds for TSP". In: *Journal of Computer and System Sciences* 81.8 (2015), pp. 1665–1677 (cit. on p. 5).

[37] S. Khot. "Improved inapproximability results for maxclique, chromatic number and approximate graph coloring". In: *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE. 2001, pp. 600–609 (cit. on p. 5).

[38] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Progress in Theoretical Computer Science. Springer, 1993 (cit. on p. 3).

[39] G. M. Kopanos, L. Puigjaner, G. M. Kopanos, and L. Puigjaner. "Production Scheduling in Large-Scale Multistage Batch Process Industries". In: *Solving Large-Scale Production Scheduling and Planning in the Process Industries* (2019), pp. 169–188 (cit. on p. 17).

[40] M. Lampis. "Improved inapproximability for TSP". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2012, pp. 243–253 (cit. on p. 5).

[41] T. Leighton and S. Rao. "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms". In: *Journal of the ACM (JACM)* 46.6 (1999), pp. 787–832 (cit. on p. 14).

[42] M. Lewin, D. Livnat, and U. Zwick. "Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems". In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2002, pp. 67–82 (cit. on p. 16).

[43] N. Linial, Y. Mansour, and N. Nisan. "Constant depth circuits, Fourier transform, and learnability". In: *Journal of the ACM* 40.3 (1993), pp. 607–620 (cit. on p. 5). Orig. conf. vers.: N. Linial, Y. Mansour, and N. Nisan. "Constant depth circuits, Fourier transform, and learnability". In: *30th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society. 1989, pp. 574–579.

[44] M. Naor and O. Reingold. "Number-theoretic constructions of efficient pseudo-random functions". In: *Journal of the ACM (JACM)* 51.2 (2004), pp. 231–262. URL: `https://dl.acm.org/doi/pdf/10.1145/972639.972643` (cit. on p. 3).

[45] C. H. Papadimitriou and S. Vempala. "On the approximability of the traveling salesman problem". In: *Combinatorica* 26.1 (2006), pp. 101–120 (cit. on p. 5).

[46] C. H. Papadimitriou and M. Yannakakis. "The traveling salesman problem with distances one and two". In: *Mathematics of Operations Research* 18.1 (1993), pp. 1–11 (cit. on p. 11).

[47] R. Raz and A. Wigderson. "Monotone circuits for matching require linear depth". In: *Journal of the ACM (JACM)* 39.3 (1992), pp. 736–744 (cit. on p. 4).

[48] A. A. Razborov and S. Rudich. "Natural Proofs". In: *J. Comput. Syst. Sci.* 55.1 (1997), pp. 24–35. URL: `http://dx.doi.org/10.1006/jcss.1997.1494` (cit. on p. 5). Orig. conf. vers.: "Natural proofs". In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. 1994, pp. 204–213.

[49] O. Reingold. "Undirected connectivity in log-space". In: *J. ACM* 55.4 (2008). URL: `http://doi.acm.org/10.1145/1391289.1391291` (cit. on p. 4).

[50] D. J. Rose and R. E. Tarjan. "Algorithmic aspects of vertex elimination on directed graphs". In: *SIAM Journal on Applied Mathematics* 34.1 (1978), pp. 176–197 (cit. on p. 17).

[51] F. Rossi, P. Van Beek, and T. Walsh. *Handbook of constraint programming*. Elsevier, 2006 (cit. on p. 18).

[52] M. Schaefer and C. Umans. "Completeness in the Polynomial-Time Hierarchy: A Compendium". In: *ACM SIGACT News* 33.3 (), pp. 32–49. URL: `https://www.researchgate.net/publication/245726393` (cit. on pp. 6, 7).

[53] S. Schmitz. "Complexity hierarchies beyond elementary". In: *ACM Transactions on Computation Theory (TOCT)* 8.1 (2016), pp. 1–36 (cit. on p. 3).

[54] J. Scott and M. Tuma. *Algorithms for sparse linear systems*. Springer Nature, 2023 (cit. on p. 17).

[55] R. Smolensky. "Algebraic methods in the theory of lower bounds for Boolean circuit complexity". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 1987, pp. 77–82 (cit. on p. 5).

[56] É. D. Taillard. *Design of heuristic algorithms for hard optimization: with python codes for the travelling salesman problem*. Springer Nature, 2023 (cit. on p. 18).

[57] É. Tardos. “The gap between monotone and non-monotone circuit complexity is exponential”. In: *Combinatorica* 8.1 (1988), pp. 141–142 (cit. on p. 4).

[58] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013 (cit. on p. 7).

[59] S. Wang and M. Guignard. “Hybridizing discrete-and continuous-time models for batch sizing and scheduling problems”. In: *Computers & operations research* 33.4 (2006), pp. 971–993 (cit. on p. 17).

[60] R. R. Williams. “Simulating Time With Square-Root Space”. In: *arXiv preprint arXiv:2502.17779* (2025). URL: https://arxiv.org/abs/2502.17779 (cit. on p. 5).

[61] R. Williams. “Nonuniform ACC Circuit Lower Bounds”. In: *J. ACM* 61.1 (2014), 2:1–2:32. URL: https://doi.org/10.1145/2559903 (cit. on p. 5). Orig. conf. vers.: “Non-uniform ACC Circuit Lower Bounds”. In: *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*. IEEE Computer Society, 2011, pp. 115–125. URL: https://doi.org/10.1109/CCC.2011.36.

[62] D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011 (cit. on p. 7).

[63] M. Yannakakis. “Computing the minimum fill-in is NP-complete”. In: *SIAM Journal on Algebraic Discrete Methods* 2.1 (1981), pp. 77–79 (cit. on p. 5).

[64] A. C.-C. Yao. “Separating the polynomial-time hierarchy by oracles”. In: *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*. IEEE. 1985, pp. 1–10 (cit. on p. 4).

[65] D. Zuckerman. “Linear degree extractors and the inapproximability of max clique and chromatic number”. In: *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. 2006, pp. 681–690 (cit. on p. 5).

[66] А. А. Разборов. *Алгебраическая сложность*. Москва: МЦНМО, 2016 (цит. на с. 3).

[67] А. А. Разборов. “Нижние оценки монотонной сложности некоторых булевых функций”. В: *Доклады Академии наук* 281.4 (1985), с. 798—801 (цит. на с. 4).

[68] А. А. Разборов. “Нижние оценки размера схем ограниченной глубины в полном базисе, содержащем функцию логического сложения”. В: *Математические заметки* 41.4 (1987), с. 598—607 (цит. на с. 5).