Содержание

1 Адреса

Проверяющие — Степан Гатилов, Александр Стененко и Александр Малых.

Основной адрес проверяющих: `algorithms.shad.nsk@gmail.com`

Страница курса: `https://compscicenter.ru/courses/algorithms-2/nsk/2018-spring`

Курс в системе anytask: `http://anytask.org/course/283`

2 Критерии оценки

Оценка за семестр определяется по успешности решения домашних заданий. Всего в осеннем семестре выдаётся 6 заданий, а в весеннем — 5 заданий.

Оценка в семестре выставляется по двум параметрам. Пусть S — суммарное количество набранных баллов за сдачу задач по *упрощённой схеме*, и F — количество задач, успешно сданных по *полной схеме*. Тогда оценка выводится следующим образом:

- "отлично": ($F \geqslant 2$ и $S \geqslant 1000$) или ($F \geqslant 3$ и $S \geqslant 700$).
- "хорошо": ($F \geqslant 2$ и $S \geqslant 700$) или ($F \geqslant 3$ и $S \geqslant 500$).
- "зачет": ($F \geqslant 2$ и $S \geqslant 500$) или ($F \geqslant 3$ и $S \geqslant 350$).

Критерии выставления оценок могут измениться в ходе семестра. В таком случае актуальная информация будет написана на странице курса.

3 Схема сдачи домашних заданий

Для каждого домашнего задания на странице курса появляются: разбалловка по задачам, дедлайны по сдаче, дополнительные ограничения (если есть). Условия задач можно посмотреть в системе тестирования или в системе anytask. Все программы должны быть написаны на языке C++.

Бóльшую часть задач нужно сдавать по упрощённой схеме. При этом вам достаточно написать программу, которая решает задачу, и успешно сдать её в автоматическую систему тестирования (см. раздел 4.2). Обычно в каждой домашке четыре задачи, и вы можете сдавать любые из них на ваш выбор по упрощённой схеме. За каждую успешно сданную в систему задачу вам начисляется некоторое количество баллов, которое зависит от сложности задачи и от количества неудачных попыток при сдаче. В конце семестра все баллы за сданные задачи суммируются, и от суммы S зависит оценка за семестр.

Кроме того, от вас требуется сдать несколько задач по полной схеме. Задачи, которые можно сдавать по полной схеме, выбирают преподаватели. Когда появляется задача, которую нужно сдавать таким образом, об этом пишется на странице курса в описании домашки. Всего в семестре будет три такие задачи, для получения оценки за семестр вам нужно сдать две-три из них (подробности выше).

Чтобы сдать задачу по полной схеме, необходимо сдать три фазы:

- **Теория:** Сдать описание алгоритма решения и его теоретическое обоснование (см. раздел 4.1).
- **Практика:** Сдать код программы решения в автоматическую систему тестирования (см. раздел 4.2).
- **Ревью:** Пройти с решением Code Review (см. раздел 4.3).

Задание считается сданным по полной схеме только тогда, когда по нему зачтены все три этапа (теория, решение, ревью).

4 Сдача по полной схеме

4.1 Теоретическое обоснование решения

Теоретические решения следует посылать в систему anytask по курсу. Для отправки выберите задачу с названием “ДЗ N теория” (вместо N должен стоять номер домашки). Напишите в задачу комментарий любого содержания, возможно пустой, прикрепив к нему PDF-файл с описанием вашего решения.

PDF-файл должен содержать:

- алгоритм решения;
- доказательство правильности;
- временная сложность — асимптотика;
- затраты памяти — асимптотика;

Все пункты обязательны!

Всё содержимое теоретического решения должно быть написано лично вами. Списывание из публичных источников и у “товарища” строго запрещается. Если какой-то факт был доказан на лекциях или семинарах, то можно просто сослаться на него (не приводя доказательство).

Если алгоритм некорректен, или в обоснованиях содержатся серьезные изъяны, то вы получите ответ с комментариями. В таком случае нужно исправить отмеченные проблемы и выслать новую версию.

Теоретическое решение можно присылать сколько угодно раз до дедлайна по теории, ошибки в нем никак не влияют на итоговую оценку по задаче. После дедлайна по теории никакие теоретические решения не принимаются. Если вы не успели сдать нам правильную теорию до дедлайна по теории, задачу вам уже не зачтут.

Теоретическое решение считается засчитанным только тогда, когда в одном из ответов будет написана фраза “Теорию засчитываю” (или что-то очень похожее).

4.2 Автоматическая проверка программы

Практические решения можно посылать в yandex.contest через систему anytask. Для этого надо зайти в anytask, выбрать задачу и отправить по ней комментарий, прикрепив к нему один исходный файл с расширением `cpp`. Чтобы отправка решений работала, вам нужно предварительно привязать ваш логин в yandex.context к профилю системы anytask.

Также вы можете также заходить в систему yandex.contest напрямую, используя ссылку в описании домашки на странице курса.

Используйте ваш логин yandex-а для системы yandex.contest. **Не** используйте логины вида `ShadXXXX-YYY`.

Инструкция, как пользоваться системой и как потренироваться сдавать задачи на примере “сложение двух чисел”, лежит рядом в этой инструкцией. Попробуйте получить все варианты описанных ниже результатов, сдавая пробную задачу. Если не получается сдать задачу, пишите вопрос семинаристу на почту.

Для каждого домашнего задания будет поднят отдельный контест. В системе по каждой задаче хранится конечное число тестов, каждый из которых ваша программа должна пройти. В систему посылается ОДИН файл с кодом программы, которая решает задачу. Да, к сожалению, даже если у вас несколько классов, придется весь код перед отправкой скопировать в один файл, никуда не денешься.

Программа должна читать из стандартного входа и писать в стандартный вывод. Входной формат гарантируется: будет так, как описано в условии. Строго придерживайтесь выходного формата. Лишние пробелы и лишние переводы строк — не страшно, а вот пропустить пробел, вывести не в том порядке, пропустить перевод строки — может быть фатально для вас.

Система выдает один из следующих результатов на каждый запуск:

- `OK` — задача прошла успешно. Полученные при этом баллы вы можете посмотреть на вкладке “монитор”.

- `PCF - Precompile check failed` — code style violation, вы нарушили какие-то из требований по оформлению кода. Чтобы узнать, какие именно, нажмите на ссылку report и почитайте. Также вы можете получить такую ошибку, если использовали при реализации какую-нибудь стандартную структуру данных или какой-нибудь стандартный алгоритм из библиотеки STL, который в данной задаче было использовать запрещено в условии задачи. Штраф: отсутствует.

- `CE - Compilation error` — не компилируется на сервере. Чтобы узнать, почему, нажмите на ссылку report и посмотрите, какие ошибки нашел компилятор. Штраф: отсутствует.

- `WA - Wrong answer` — на некотором тесте программа выдала неверный ответ. Тест не выдается, пишите стресс-тесты, ищите баги. Штраф: **-1 балл**.

- `TL - Time limit exceeded` — слишком долго работает. Скорее всего, у вас асимптотически неправильное время работы решения. Ограничения по времени будут

выставляться в 2–3 раза больше, чем время работы авторского решения на максимальном тесте. Этого должно хватать любому правильному решению. Штраф: **-2 балла**.

- `ML - Memory limit exceeded` — используете слишком много памяти. Ограничение памяти почти во всех задачах сейчас 64 мегабайта. Штраф: **-2 балла**.
- `RE - Run-time error` — произошла ошибка выполнения. Штраф: **-2 балла**. Неочевидные возможные причины: чтение из файла вместо стандартного ввода; запись в файл вместо стандартного вывода; переполнение стека. Очевидные причины: выход за границы массива, деление на ноль, удаление несуществующего указателя, etc. Подумайте, не выходите ли вы где-то за пределы вектора, не переполняется ли стек рекурсии, не удаляете ли что-то из пустой структуры данных, не обращаетесь ли не к своей памяти, не используете ли невалидные итераторы и не пользуетесь ли библиотечными функциями, не выполнив требования, которые предъявляются к их параметрам.
- `PE - Presentation error` — вывод не соответствует формату. Например, просили вывести число, а выведена строка. В этом случае вы можете получить `Wrong answer` либо `Presentation error`, это не гарантируется заранее. Штраф: **-1 балл**.

Обратите внимание на указанный в конце каждого результата штраф. Как только вы успешно сдаёте задачу (получаете `OK` в первый раз), вам начисляются баллы за эту задачу за вычетом штрафов за все неудачные попытки до этого. Заметьте, что штраф не добавляется, если ваше решение не скомпилировалось. Если вы не смогли успешно сдать задачу, то штраф вам не начисляется. При вычислении штрафа считаются только попытки по этой задачи. Если вы уже успешно сдали задачу, штрафы по ней более не начисляются.

Рассмотрим пример. Допустим, вы сдаёте задачу с номинальной стоимостью в 30 баллов, и у вас получается следующая последовательность результатов в системе: PCF, PCF, PCF, PCF, PCF, CE, WA, RE, RE, TL, TL, WA, OK, WA, OK, OK, TL, OK. В таком случае получается суммарный штраф -10 баллов, а значит вы получаете 20 баллов за задачу.

Весь код сдаваемой программы должен быть написан лично вами. Списывание из публичных источников и у “товарища” строго запрещается. В некоторых задачах нельзя пользоваться определёнными частями стандартной библиотеки. Если такое ограничение есть, то оно указывается в описании домашки.

4.3 Code Review

При сдаче задачи по полной схеме нужно отправить код решения на Code Review после того, как оно получило `OK` в автоматической системе,

Для Code Review используется система ReviewBoard, интегрированная с anytask. Для отправки кода на ревью выберите задачу “ДЗ Z ревью”, где Z — номер домашки. Прикрепите к пустому комментарию один файл с исходным кодом вашего решения. После отправки появится ссылка на этот код в ReviewBoard. Каждый раз, когда проверяющий отправляет новые комментарии по коду, в соответствующей задаче anytask автоматически появляется сообщение об этом. Исправьте проблемы и отправьте новую версию кода аналогичным образом. *Не изменяйте название файла с исходным кодом при повторных отправках кода на review!*

На этой стадии фактических ошибок в вашей программе мы, скорее всего, уже не найдем, но будем делать замечания по стилю кода, а также по поводу разных опасных вещей, которых в коде делать не следует, и наоборот, про best practices, как лучше всего делать некоторые вещи. По этим замечаниям код надо будет исправлять и присылать заново до тех пор, пока у нас не останется замечаний и мы не зачтем вам задачу. Можно пытаться спорить, но только если конструктивно, то есть если у вас есть аргументы в пользу вашего текущего решения и против наших замечаний, либо если вам просто непонятно, по какой причине мы просим вас что-то исправить. В этом случае спор даже поощряется: это может быть полезно обеим сторонам.

Code review считается засчитанным только тогда, когда в одном из ответов будет написана фраза "Засчитываю задачу" (или что-то очень похожее).

5 Дедлайны

Точные дедлайны для каждого домашнего задания написаны на странице курса в разделе с домашними заданиями.

Теоретическое решение (теорию) можно сдавать в течение примерно 2 недель с момента его получения. Решение в автоматическую систему (практику) принимается в течение примерно 4 недель. Дедлайна на ревью нет (кроме конца семестра, см. ниже). При сдаче задания по упрощённой схеме дедлайн совпадает с дедлайном по практике.

Просьба укладываться в дедлайны, лучше с запасом. При посылке чего-либо после дедлайна нет никакой гарантии, что это что-то будет проверено. Дедлайн по практике соблюдается автоматически: если задача сдана после дедлайна, баллы за неё точно не будут начислены. По уважительной причине мы можем разрешить прислать что-нибудь позже, но уважительными причинами считаются только вещи вроде болезни, свадьбы, рождения ребенка, несчастных случаев и т.д., то есть варианты "аврал на работе", "проблемы с учебой" и "прозевал срок" нас не убеждают.

Самый главный дедлайн — это конец семестра. Осенний семестр заканчивается **20 декабря**, весенний — **31 мая**.

6 Стиль кода

Часть вашего обучения состоит в том, чтобы научиться писать код в хорошем стиле. Код, легко понятный другим людям. Код, в котором мала вероятность допустить дурацкую ошибку. Прежде, чем писать какой-либо код, рекомендуется почитать какой-нибудь c++ style guide (наберите эту строку в вашем любимом поисковике и читайте первую попавшуюся ссылку). Например, вот этот

`http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml`

Соблюдение какого-либо конкретного code style guide в рамках нашего курса **не** требуется, однако указываемые в таких документах правила, а особенно пояснения к ним, могут быть вам полезны.

Некоторые требования к стилю исходного кода проверяются автоматически. При посылке решения в проверяющую систему (см. раздел 4.2) запускается автоматическая проверка стиля. Если код ее не проходит, то будет выдано соответствующее сообщение от системы, и программа не будет допущена до тестов. Кроме того, большое внимание стилю оформления кода будет уделено на фазе Code Review (см. раздел 4.3).

7 Советы: как решать задачи, писать решения и тестировать их

О примерной допустимой сложности решения можно догадаться по ограничениям. Порядок количества операций не должен превышать нескольких сотен миллионов. Таким образом, если ограничение на число n в задаче равно 1 000, то это, скорее всего, $O(n^2)$ или $O(n^2 \log n)$ (в редких случаях $O(n^3)$), если 10 000, то $O(n \log n)$ или $O(n^2)$, если 100 000, то $O(n)$ или $O(n \log n)$ или $O(n\sqrt{n})$. Допустимую сложность по памяти оценивайте исходя из того, что у вас есть 64 Мб оперативной памяти, одна переменная типа `int` занимает 4 байта, `char` — 1 байт, `double` — 8 байт. В большинстве случаев сложнее будет уложиться в ограничение по времени, чем по памяти.

Тестируйте свои программы сами. Тестируйте на крайних случаях: если дано ограничение $1 \le n \le 1000$, то проверьте $n = 1, 2, 3, 1000$. Придумывайте маленькие тесты. Тестируйте также другие ограничения, не только на количество данных, но и на сами данные. Например, если дано, что стоимости чего-нибудь не превосходят 1 000 000 000 по модулю, то попробуйте тесты, где все числа равны миллиарду; где все числа равны минус миллиарду; где есть и миллиард, и минус миллиард. Ограничения по времени будут даны в тексте условия каждой задачи в автоматической системе тестирования. Обязательно проверяйте время работы программы на заранее сгенерированных файлах с тестами максимального размера.

Если у вас не получается сдать задачу в систему, никак не можете пройти какой-то тест, знайте, что сам тест мы вам все равно никогда не выдадим. Все тесты, которые есть к задачам, были сделаны теми же способами, которые описаны в предыдущем абзаце.

8 Часто задаваемые вопросы

1. **Q:** Написал решение, работающее за (почти) линейное время от размера входных данных. Однако оно получает вердикт `TL` в системе тестирования, что делать?

   A: Если в задаче возможен большой объём входных/выходных данных (порядка мегабайта), то может тормозить сам ввод/вывод. В таких задачах не рекомендуется использовать стандартные потоки ввода/вывода из C++ (т.е. что-то вроде `iostream`, `fstream`, `std::cin`, `std::cout`). Вместо этого лучше использовать стандартный ввод/вывод языка C (т.е. `cstdio`, `printf`, `scanf`), он работает намного быстрее.

   Это не касается задач с малым объёмом ввода/вывода: в них лучше использовать потоки языка C++.

2. **Q:** Могу ли я сдавать программу в автоматическую систему проверки до того, как мне проверили теорию?

   A: Разумеется. Необязательно ждать ответа от нас, что теоретическое решение правильное, чтобы пытаться сдавать код в систему.

3. **Q:** У меня никак не получается сдать задачу в систему, можете помочь?

   A: Вы можете написать письмо семинаристу на почту и задать свои вопросы. Срочные вопросы отправляйте с пометкой «URGENT!!!» в теме письма. Но кон-

кретные тесты, на которых валится ваша программа, мы вам не выдадим, а скорее всего отправим придумывать свои.

4. **Q:** Я заболел, пролежал неделю в больнице и из-за этого не смог вовремя прислать решение задачи. Могу я сдать домашку сейчас?

   A: Да, болезнь — это уважительная причина для продления срока сдачи.

5. **Q:** У меня был аврал на работе, а еще меня бросила девушка, я очень расстроился и не успел вовремя сдать домашку. Можно мне сделать это сейчас?

   A: Нет, это неуважительная причина. Учитесь планировать свое время и не откладывать все на последний момент.