Содержание

1 Общая информация

1.1 Преподаватели

Лектор — Максим Александрович Бабенко.

Семинаристы — Михаил Левин, Вячеслав Алипов, Илья Шишков, Антон Полднев, Егор Хайруллин.

Ассистенты — Родион Пермин, Максим Иванов, Павел Мельничук, Александр Тиунов, Алиса Смирнова.

1.2 Контакты

Общий адрес курса `ys.algorithms@gmail.com`

В случае срочных вопросов пишите письмо с пометкой в теме письма "URGENT!!!". Если вам не отвечают с общего адреса курса алгоритмов *более недели* или вы остались не удовлетворены ответом, пишите Мише на `mikhail.levin+ys@gmail.com`.

Вики-страница курса:

`https://wiki.school.yandex.ru/shad/groups/2015/semester2/Algorithms2`

На вики публикуются все домашние задания и ссылки на видео лекций, удобней всего подписаться на изменения страницы, чтобы вовремя узнавать о новостях. О новых домашних заданиях мы будем дополнительно уведомлять вас по email.

2 Семинары

В то время как лекции посвящены теоретическим аспектам, семинары будут больше направлены на практику. В частности, мы подробно остановимся на styleguide, который вы обязаны соблюдать при сдаче домашних заданий, а также будем давать и другие советы по написанию кода. И конечно же, мы будем отвечать на ваши вопросы.

Помимо этого на семинарах будут разбираться задачи повышенной сложности. Мы будем вместе придумывать для них теоретическое решение, возможно, частично его реализовывать. После этого вы сможете реализовать эту задачу до конца дома, сдать ее в систему и получить дополнительные баллы. Таким образом, семинары дают возможность повысить вашу оценку за семестр.

3 Как сдавать домашние задания

За семестр будет выдано несколько домашних заданий, 6 в осеннем семестре и 5 в весеннем. В каждом домашнем задании, за редким исключением, 4 задачи. Каждая задача стоит определенное количество баллов, в зависимости от её сложности. *Домашние задания необходимо выполнять самостоятельно. Замеченные за списыванием будут считаться должниками по курсу к концу семестра, смогут зачесть курс только после конца семестра и не смогут получить более "зачета" по курсу.*

Все задачи сдаются в автоматическую систему проверки — Яндекс.Контест, — которая будет проверять ваши решения-программы.

3.1 Задачи на ревью

За семестр будет выдано 2 задачи, которые надо не только сдать в систему, но и пройти Code Review у семинаристов. Ревью состоит из трех этапов:

1. проверка теоретического решения задачи
2. проверка интерфейса программы
3. ревью кода программы целиком

Проверка происходит в системе Anytask, страница нашего курса расположена здесь:

`http://anytask.org/course/44`

Важно: Для регистрации в системе вам понадобится инвайт, который будет выслан вам в течение первых двух недель семестра. Если вы видите, что выложено домашнее задание, в котором нужно отправлять код на ревью, а инвайта у вас нет, напишите нам на email.

Для каждой задачи на codereview в Anytask будет две сущности: одна для проверки теории, другая для проверки кода. Так проверяющим будет проще отслеживать очередь на проверку и быстрее проверять теоретические решения. Для некоторых задач в ответ на зачтенную теорию мы будем присылать уже готовый интерфейс программы – реализовать надо будет в точности его.

3.1.1 Проверка теоретического решения

Присылайте pdf файл с описанием вашего решения:

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

Все пункты обязательны! Если можете доказать, что быстрее невозможно или меньше памяти использовать невозможно, — тоже пишите, это полезно.

3.1.2 Code Review

После того как сдадите код в систему (для этого пользуйтесь вашим логином вида SHAD2015-*), нужно будет отправить его на проверку нам. Однако сначала мы будем проверять не весь код, а только его “каркас” — интерфейс, а именно: все классы с методами, все функции, а так же комментарии к классам и функциям.

Коротко, что имеется в виду под словом интерфейс:

- подключение заголовочных файлов, т.е. все include’ы
- объявление пользовательских типов
- объявление пользовательских классов и структур, всех их полей и методов *без реализации*
- комментарии ко всем структурам и классам, объясняющие, для чего этот класс/структура используется
- заголовки всех функций, как они есть в работающей программе

- комментарии ко всем функциям, объясняющие, что каждая функция делает
- main как он есть в работающей программе

Таким образом, в своем решении вы сначала размещаете интерфейс — все глобальные константы, все сигнатуры функций (объявления), интерфейсы классов с сигнатурами методов и объявлениями переменных-членов и констант, main вместе с реализацией, а потом, после main — полный код с реализациями функций, методов, классов. При отправке кода в Anytask программа автоматически отправляется на проверку в контест, и только в случае успешного прохождения всех тестов в системе ваш код попадет к нам на ревью.

Мы, скорее всего, уже фактических ошибок в вашей программе после этого не найдем (хотя по некоторым задачам это не так: скорее всего, найдем :)), но будем делать замечания по стилю кода, а также по поводу разных опасных вещей, которых в коде делать никогда не надо, 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 это конструктивно, 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. При этом спорить в духе “я считаю, что это все равно, и мне лень исправлять”, — бесполезно, все равно придется исправить.

Важно: если код требует значительной доработки, проверяющий оставляет за собой право ограничиться неполным списком замечаний.

Важно: система проверки Anytask используется нами в этом учебном году впервые, и в настоящее время она активно дорабатывается. Если у вас возникают вопросы или пожелания по работе с этой системой, вы всегда можете задать их нам по email.

4 Дедлайны

И о сроках. Задачи по каждой домашке можно сдавать в автоматическую систему проверки в течение примерно 4 недель, точный срок указывается на вики-странице курса одновременно с выдачей домашнего задания. После дедлайна получить баллы за сданные задачи будет невозможно, однако это не мешает продолжать сдавать задачи, если вам интересно, а времени не хватило. В день дедлайна полученные в системе баллы будут выставляться в табличку с результатами. Зависимость оценки от набранных баллов написана на вики-странице курса.

Что касается сдачи задач семинаристам (тех самых двух задач, по которым нужно пройти ревью), то тут сроки такие: на сдачу теории дается 1 неделя с момента выдачи задачи, и еще 3 недели на то, чтобы пройти все тесты в системе и прислать интерфейс работающей программы на ревью. Затем Code Review идет

до тех пор, пока все замечания не будут исправлены, либо не закончится семестр. Все дедлайны будут прописаны на вики-странице курса.

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. Дедлайн по практике означает, что нужно успеть сдать код в систему до этого момента, а также **успеть сделать ревью в Anytask**. Необязательно успеть полностью зачесть задачу до дедлайна по практике, это вообще редко происходит, так как code review с нами проходит довольно долго, особенно поначалу, пока вы еще не привыкли соблюдать все правила.

Все сроки жестко соблюдаются. Присылать что-либо позже объявленного срока бесполезно. 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**.

4.1 Мидтерм

В середине семестра есть промежуточный дедлайн — мидтерм. В первом семестре это **1 ноября**, во втором — **1 апреля**. К этой дате надо сдать в систему тестирования **не менее двух задач**.

5 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, обязательно прочитайте 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`

Обязательно прочитайте code style guide к нашему курсу (выложен на странице курса вместе с остальными инструкциями). Все их следует исполнять сразу, еще при сдаче программ в систему. Соблюдая эти правила, вы не только делаете код более читаемым, вследствие чего лучше ориентируетесь в нем сами, но и снижаете вероятность совершить ошибку за счет некоторых мнемонических приемов и обхода некоторых частых проблем. Поэтому вы повышаете свои шансы сдать задачу с первого раза и получить по ней больше баллов. Кроме того, вы ускоряете процесс code review, так как, если вы нарушите какие-то из стандартных замечаний, вам все равно придется исправлять это в процессе code review. Проще

подготовиться и прислать сразу качественный код, и не тратить наше время на те ошибки или недочеты, которые настолько часто делают новички, что мы их предусмотрели заранее.

Некоторые пункты из требований к стилю проверяются автоматически. При посылке решения в проверяющую систему на коде запускается автоматическая проверка стиля. Если код ее не проходит, то будет выдано соответствующее сообщение от системы и программа не будет допущена до тестов. Таким образом, ваш код должен выглядеть правильно, работать правильно, а также не вызывать дополнительных замечаний у нас.

6 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 скорости, чем по памяти.

Тестируйте свои программы сами, так как в первую очередь мы будем оценивать правильность работы вашей программы. Ограничения по времени будут даны для каждой задачи на вики-странице курса. Тестируйте на крайних случаях: если дано ограничение $1 \le n \le 1000$, то проверьте $n = 1, 2, 3, 1000$. Придумывайте маленькие тесты, вбивайте их руками. 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.

Придумывайте тесты на крайние случаи заранее, исходя из условия задачи, а не из вашего решения. Лучше всего заранее придумывайте набор тестов, просчитывайте на бумажке ответы руками, а потом уже пишите программу и следите, чтобы ответы совпали. Генерируйте себе файлы с максимальными тестами. Для проверки времени работы — именно файлы, а не тесты в памяти, так как часть времени в любой задаче занимает ввод и вывод, а иногда они занимают большую часть времени, и вы можете не пройти ограничения по времени, если будете делать ввод-вывод неаккуратно. И, наконец, самая мощная техника тестирования — стресс-тестирование (о ней будет рассказано на одном из семинаров).

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 либо сделаны руками, либо сгенерированы случайным образом по некоторому шаблону. Вы можете сделать то же самое: как протестировать крайние случаи ручными тестами, так и сгенерировать много случайных тестов, реализовать стресс-тест — и вы наверняка сможете найти свою ошибку, потому что тесты к задаче были получены таким же способом. Думайте, какие бывают крайние случаи в задаче, помимо тех, которые видны из ограничений задачи, и тестируйте эти случаи.

7 Часто задаваемые вопросы

1. **Q:** Я отправил теорию вчера, а мне до сих пор никто не ответил. Почему?

   A: Мы стремимся к тому, чтобы среднее время ответа было порядка трех дней, быстрее невозможно из-за большого количества студентов, а также того факта, что почти все вы будете присылать свои решения одновременно, в последний день. Лучше присылать раньше, тогда вас и проверят раньше, и есть шанс что-то успеть исправить в теоретическом решении до дедлайна, если оно было изначально неправильным.

2. **Q:** Могу ли я сдавать программу в 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. Вы должны полностью доказать свое решение, и этого должно быть достаточно для вас, чтобы быть в нем уверенным и сдавать в систему. Конечно, иногда будет оказываться, что решение неправильное, если вы его неправильно доказали, и мы-то это проверим, зададим вопросы, вы сможете прислать новое решение, если срок по сдаче теории еще не закончился. В этом случае вам было бы «выгоднее» сначала убедиться, что мы тоже думаем, что ваше решение правильное, и только после этого сдавать в систему код. Однако, в настоящей жизни не будет никаких людей, проверяющих ваши решения, поэтому нужно научиться строго их доказывать, чтобы быть уверенными в решении без посторонней помощи.

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

   A: Вы можете написать письмо на адрес курса `ys.algorithms@gmail.com` и задать свои вопросы. Срочные вопросы отправляйте с пометкой «URGENT!!!» в теме письма. Но конкретные тесты, на которых валится ваша программа, мы вам не выдадим, а скорее всего отправим придумывать свои.

Внимание: по одной задаче не создавайте несколько разных тредов, всегда отвечайте на последнее свое или наше письмо, а не пишите новое.

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.