Алгоритмы и модели вычислений, группы $774 - 775$, задание 2

- Кормен–Лейзерсон–Ривест–Штайн, глава 34, параграф 1. ***ОСОБОЕ ВНИМАНИЕ*** уделите теореме 34.2 и комментариям после неё, посвящённым неконструктивному доказательству этой теоремы. На семинаре я уделил мало внимания кодировкам, на эту часть параграфа тоже обратите внимание.

- Конспекты первых двух лекций Мусатова по курсу теории сложностей у ФИВТов:
  http://ru.discrete-mathematics.org/fall2017/3/complexity/compl-book.pdf.
  Во второй лекции нужны только пункты 2.1 и 2.2. Оба файлы лежат в Телеграм-канале.

- Хорошая книжка по теории сложностей на английском - http://theory.cs.princeton.edu/complexity/book.pdf. Из неё непосредственно к курсу относится только раздел 1.5, но если будете читать его, то ***ОБЯЗАТЕЛЬНО*** убедитесь, что разделы $1.1 - 1.4$ вы понимаете после прохождения курса ТФС. Не бойтесь математического английского! Он простой, серьёзно.

- Также рекомендую книгу блестящих математиков Гача и Ловаса по вычислительной сложности:
  http://www.cs.elte.hu/∼lovasz/complexity.pdf. Здесь вам стоит разобраться с главой 4, в ней разобраны много примеров доказательства принадлежности языков классу $\mathcal{P}$.

Замечания по поводу 2-го семинара

Одна из главных целей курса научиться чётко обосновывать принадлежность языков к классу сложности. Пока нас будут больше всего интересовать классы $\mathcal{P}$ и $\mathcal{NP}$. Если Вы хотите получить оценку не меньше хора, то необходимо (но недостаточно!) разобраться в указанной литературе ***НА КОНКРЕТНЫХ ПРИМЕРАХ*** как обосновывать утверждения типа $L \in \mathcal{P}$, $L \in \mathcal{NP}$. Про $\mathcal{NP}$ речь пойдёт на следующем семинаре. Этот текст написан в надежде прояснить мой рассказ на семинаре определения класса $\mathcal{P}$, если было не очень понятно, а также сделать важное замечание про разрешимые и рекурсивно перечислимые языки (про них были вопросы).

- На семинаре при введении класса $\mathcal{P}$ мы рассматривали ***детерминированные*** машины Тьюринга (МТ) с двумя финальными состояниями *Accept* и *Reject*, то есть у которых функция переходов однозначная и всюду определённая. Это важно помнить, потому что определение класса $\mathcal{NP}$ опирается на ***недетерминированные*** машины Тьюринга и позже вы почувствуете разницу в их вычислительных способностях.

- При этом для рассмотренных МТ поведение на каждом входном слове было следующим - слово w принимается МТ, если за конечное число тактов работы МТ останавливается в состоянии *Accept*; отвергается, если за конечное число тактов работы МТ останавливается в состоянии *Reject*; возможен случай, когда МТ не останавливается на входе. После этого я сформулировал ***неканоническое*** определение разрешимого языка и должен поправить себя. А именно, я сказал, что L является разрешимым, если существует такая МТ $\mathcal{M}$, что «$\mathcal{M}$ останавливается на входе w в состоянии *Accept*» $\Leftrightarrow w \in L$ - точнее такие языки называть рекурсивно перечислимыми, как кто-то заметил на перерыве (лайк от меня). Напомню ***каноническое*** определение разрешимого языка из нашего курса ТФС, которое более требовательно к МТ, чем указанное выше, и именно им мы будем пользоваться (а прошлым не будем). Язык L называется разрешимым, если для него существует МТ (говорят, что эта МТ разрешает язык L), которая для слов $w \in L$ завершит работу в *Accept*, а для слов $w \notin L$ завершит работу в *Reject*. Таким образом, МТ, разрешающая какой-то язык, останавливается ***ВСЕГДА***. Но для определения класса $\mathcal{P}$ это ***НЕВАЖНО***, как утверждается в теореме 34.2 в книге Кормен–Лейзерсон–Ривест–Штайн. Имеется в виду факт эквивалентности следующих двух определений класса $\mathcal{P}$:

  1) $L \in \mathcal{P}$, если существуют МТ $\mathcal{M}$, константа c и полином $p(n)$ такие, что условие $\mathcal{M}$ «остановится на входе w в состоянии *Accept* за число тактов не более $c \cdot p(|w|)$» $\Leftrightarrow w \in L$. Это определение было дано на семинаре и здесь не требуется, чтобы $\mathcal{M}$ останавливалась на всех входах.

  2) Альтернативное ***естественное*** определение: $L \in \mathcal{P}$, если существует МТ $\mathcal{M}$, которая должна разрешать язык L и быть полиномиальной по входу. Определение понятия «Разрешать язык» написано выше. «Полиномиальная по входу» значит, что существуют константа c и полином $p(n)$ такие, что на ***любом*** входе w машина остановится за число тактов не более $c \cdot p(|w|)$.

- Идея доказательства эквивалентности двух определений абсолютно аналогична той, которая использовалась при доказательстве импликации $L \in \mathcal{P} \to \overline{L} \in \mathcal{P}$ - из 2) очевидно следует 1), а из 1) следует 2), если рассмотреть вспомогательную

машину $\widetilde{\mathcal{M}}$, которая на входе w будет симулировать работу $\mathcal{M}$ и «следить» за числом выполненных тактов машины $\mathcal{M}$ на входе w. Как только это число превышает $c \cdot p(|w|)$, машина $\widetilde{\mathcal{M}}$ останавливается в состоянии *Reject*. Очевидно, что новая машина $\widetilde{\mathcal{M}}$ будет разрешать язык L, если вспомнить определение машины $\mathcal{M}$. Остаётся только явно предъявить новую константу $\widetilde{c}$ и полином $\widetilde{p}(x)$, ограничивающие время работы $\widetilde{\mathcal{M}}$ на словах w, в зависимости от c и $p(n)$. Формально в исходном определении надо работать с одноленточной $\widetilde{\mathcal{M}}$, в то время как здесь, аналогично семинарской МТ, указанная МТ $\widetilde{\mathcal{M}}$ проще всего моделируется с помощью двух лент. Корректность этого перехода объясняется теоремой 4.41 в книге Вялого по ТФС - она как раз о моделировании многоленточных МТ на одноленточной, важно только понять как зависит сложность этого моделирования. Утверждение из первой лекции Мусатова в разделе «Распознавание языков» показывает, что оно возрастает в O-асимптотике не более, чем в квадрат раз. А поскольку полиномы замкнуты относительно композиции, суммирования и умножения, то это окончательно обосновывает полиномиальность числа тактов для ***одноленточной*** машины $\widetilde{\mathcal{M}}$ на словах $w \in L$ после её «переделывания» из двухленточной. ***Контрольный вопрос***: попробуйте явно предъявить константу $\widetilde{c}$ и полином $\widetilde{p}(n)$, чтобы завершить доказательство. *Hint*: напомню, что на семинаре рассматривалась двухленточная МТ, конструкция приведена в следующем пункте.

- Ещё раз, чтобы расставить всё по полочкам, в письменной форме опишу доказательство импликации $L \in \mathcal{P} \to \overline{L} \in \mathcal{P}$, которое приводил на семинаре, где класс $\mathcal{P}$ задавался определением 1). Для одноленточной машины $\mathcal{M}$ из определения $L \in \mathcal{P}$ строится двухленточная машина $\widetilde{\mathcal{M}}$, которая делает следующее: ***1)*** на входе w на первой ленте $\widetilde{\mathcal{M}}$ записывает на второй ленте $c \cdot p(|w|)$ новых символов A и возвращает головку в начало второй ленты; ***2)*** головка на первой ленте симулирует работу машины $\mathcal{M}$ на входе w, а головка на второй ленте синхронно (одновременно) вместе с каждым тактом первой головки передвигается направо на 1 написанный символ A; ***3)*** остановка $\widetilde{\mathcal{M}}$ определяется так - если головка $\widetilde{\mathcal{M}}$ на второй ленте дошла до пустого символа, то $\widetilde{\mathcal{M}}$ останавливается в состоянии *Accept*, также остановка $\widetilde{\mathcal{M}}$ определяется поведением остановки $\mathcal{M}$, работу на входе w которой симулирует головка на первой ленте $\widetilde{\mathcal{M}}$, только состояние остановки $\widetilde{\mathcal{M}}$ будет противоположном тому, в котором $\mathcal{M}$ останавливается на входе w. Очевидно, что время работы машины $\widetilde{\mathcal{M}}$ на произвольном входе w ограничивается временем записи головкой на второй ленте $c \cdot p(|w|)$ символов, её возврата в начало, а также ещё одного прохода второй головки на $c \cdot p(|w|)$ символов. Остаётся понять, что это число не больше $T(|w|) + c \cdot p(|w|) = O(p(|w|))$ тактов, если выполнено $T(|w|) = O\big(p(|w|)\big)$, где $T(|w|)$ - время на запись $c \cdot p(|w|)$ символов на вторую ленту по входу w на первой ленте. ***Контрольный вопрос***: понимаете ли Вы как реализовать такую двухленточную МТ и явно оценить её число тактов каким-то полиномом от $|w|$ в зависимости от константы c и полинома $p(|w|)$? Таким образом, полиномиальность по входу построенной машины доказана. Теперь поймём, почему она разрешает язык $\overline{L}$. В случае, если первая машина $\mathcal{M}$ в какой-то момент останавливается на входе w в состоянии *Accept* не более, чем за $c \cdot p(|w|)$ тактов, то это равносильно $w \in L$, так что головка машины $\widetilde{\mathcal{M}}$ на входе w на второй ленте ***по построению*** не дойдёт до пустого символа в пункте ***3)***. Поэтому в этом случае для машины $\widetilde{\mathcal{M}}$ вместо *Accept* она остановится во время выполнения пункта ***2)*** в состоянии *Reject* как только машина $\widetilde{\mathcal{M}}$ на первой ленте просимулировала работу $\mathcal{M}$ на входе w (в результате которой $\mathcal{M}$ закончила работу в состоянии *Accept* как раз не более чем за $c \cdot p(|w|)$ тактов). В случае, если машина $\mathcal{M}$ не останавливается на входе w или останавливается в состоянии *Accept* более, чем за $c \cdot p(|w|)$ тактов работы, то это означает, что $w \notin L$, поэтому головка машины $\widetilde{\mathcal{M}}$ через $c \cdot p(|w|)$ тактов дойдёт до пустого символа и пункта ***3*** и для машины $\widetilde{\mathcal{M}}$ мы перейдём в состояние *Accept*. Если же машина $\mathcal{M}$ останавливается на входе w в состоянии *Reject*, то это последний нерассмотренный случай $w \notin L$. Если число тактов работы для машины $\mathcal{M}$ на таком входе больше $c \cdot p(|w|)$, то головка на второй ленте машины $\widetilde{\mathcal{M}}$ дойдёт до пустого символа и построенная МТ остановится в состоянии *Accept* ***по построению*** пункта ***3***. Иначе машина $\widetilde{\mathcal{M}}$ также остановится в состоянии *Accept* ***по построению*** в пункте ***3*** из-за симуляции на первой ленте работы машины $\mathcal{M}$ на слове w, а здесь мы рассматриваем как раз случай, когда $\mathcal{M}$ останавливается на входе w в состоянии *Reject*. На этом все возможные варианты поведения МТ $\mathcal{M}$ на входе w рассмотрены и во всех случаях мы проверили, что двухленточная МТ $\widetilde{\mathcal{M}}$ полиномиальна и разрешает язык $\overline{L}$. Завершающее доказательство действие состоит в «переделывании» двухленточной МТ $\widetilde{\mathcal{M}}$ в одноленточную, которая также будет полиномиальной по входе согласно утверждению в первой лекции Мусатова. ***А ТЕПЕРЬ ВНИМАНИЕ***: если пользоваться определением 2), то всё доказательство сводится к фразе в одну строчку «для построения машины $\widetilde{\mathcal{M}}$ поменяем местами правила перехода в *Accept* и *Reject* в машине $\mathcal{M}$»! ***Логичный вопрос***: а зачем тогда вообще надо вводить определение 1), если определение 2) даёт элементарное доказательство этого факта? Цель этого примера показать, что стоит понимать как работать в разных определениях одних и тех же вещей. Возможно, вы узнали что-то новое в этом тексте после курса ТФС и он написан с целью заполнить возможные пробелы в знаниях по этому курсу. Помимо этого, очень важно, чтобы мы с вами говорили на одном языке. Например, ваш лектор АМВ, по всей видимости, в каноническом задании планирует использовать определение 1) и полезно понимать как оно связано с естественным определением 2).

- На семинаре возникал вопрос про корректность рассмотренного доказательства импликации $L \in \mathcal{P} \to \overline{L} \in \mathcal{P}$, связанный с априорным знанием полинома и константы в определении языка из $\mathcal{M}$ для построенного алгоритма. В приведенном наброске доказательства эквивалентности двух определений возникает аналогичный естественный вопрос: как машина $\widetilde{\mathcal{M}}$ ***«знает»*** полином $p(n)$ и константу c? Ведь она ***должна*** по входному слову w записать на вторую ленту величину $c \cdot p(|w|)$. И вообще, построили ли мы алгоритм $\widetilde{\mathcal{M}}$, который корректно работает, ведь он кажется неконструктивным?

Ответ: да, построили. Посмотрите ещё раз на определения класса $\mathcal{P}$. Ключевое слово здесь «существуют». МТ определяется главным образом своей функцией переходов, то есть своей программой. В силу существования полинома $p(n)$ и константы c с нужными свойствами существует МТ, у которой в программе явно «прописан» алгоритм вычисления по входу w величины $c \cdot p(|w|)$, так как этот полином и константа определяются исключительно языком $L \in \mathcal{P}$. Да, доказательство действительно неконструктивное, об этом также написано в теореме 34.2 Кормен–Лейзерсон–Ривест–Штайн, но это не должно вас отталкивать. В математике полно неконструктивных вещей, особенно в матане.

- Сложность можно измерять в разных моделях вычислений, а время работы – в разных единицах. В модели, где алгоритмом считается машина Тьюринга, время работы измерятся в тактах работы МТ. В модели, где алгоритмом считается машина RAM (с равнодоступной памятью), время работы измеряется в тактах работы машины RAM. Есть принципиальная разница между двумя этими моделями. МТ работает с битовыми (или в другой системе счисления) записями чисел, поэтому во всех оценках сложности надо помнить, что запись числа n требует логарифмического места на ленте, и все операции типа сложения происходят не за один такт. Машина RAM умеет оперировать с числами произвольной точности, при этом любое число занимает лишь одну ячейку машины, а стандартные арифметические операции выполняются за один такт. В реальных компьютерах процессоры адаптированы под работу с числами ограниченной точности, но всё-таки достаточно большой, поэтому в пределах этой точности обычно операции считаются выполняющимися за $O(1)$, а когда обрабатываемые числа превосходят стандартные ограничения, приходится работать с так называемой длинной арифметикой, где нужно вести учёт сложности арифметических операций. Обычно, если есть неоднозначность, в тексте задачи будет оговариваться, можно ли считать, что арифметические операции выполняются за $O(1)$, или нельзя.
- ***РЕЗЮМЕ***: в домашнем задании/на тестах/на контрольных вы можете пользоваться любым из двух определений класса $\mathcal{P}$, описанных выше. Более того, ***мы советуем*** пользоваться определением 2), оно более простое. Целью курса не является умение разобраться с тем, как строить полиномиальные машины Тьюринга, это формальное определение нужно лишь для аккуратной аксиоматики всей этой науки. Обычно в доказательстве полиномиальности какого-то языка мы будем пользоваться привычной RAM-моделью, но в некоторых задачах вам нужно будет уметь применять этот формализм на языке МТ и в этих случаях будет специально сказано об этом.
- Спасибо, если прочитали до конца, я старался для вас :)

Задание 2

Сначала начнём с совета для написания тестов: народ, лучше не пытаться зарешать всё и сразу на изях (как некоторые делали при написании первого теста) так бывает только если вы очень прошарены в алгоритмах, как, например, мой коллега Саша, который занимается этой наукой очень давно. Лучше попробовать потратить больше времени на меньшее число задач, но добить их, чтобы получить полный балл. Но на потоковых контрольных всё наоборот: не стоит зацикливаться на отдельной задаче, их много, если не получается решить какую-то, то забивайте и смотрите дальше.

1. Докажите следующие свойства класса $\mathcal{P}$:

   (i) Класс $\mathcal{P}$ замкнут относительно конкатенации. То есть если $A \in \mathcal{P}$ и $B \in \mathcal{P}$, то $L = \{ab \mid a \in A,\ b \in B\} \in \mathcal{P}$

   (ii) Класс $\mathcal{P}$ замкнут относительно итерации. ($A \in \mathcal{P} \ \Rightarrow \ A^* \in \mathcal{P}$)

   (iii) Класс $\mathcal{P}$ замкнут относительно четной итерации. ($A \in \mathcal{P} \ \Rightarrow \ (AA)^* \in \mathcal{P}$)

   В этих задачах мы хотим от вас увидеть решение в одном из двух формализмов: на языке МТ (это предпочтительней) или на неформальном языке алгоритмов в духе доказательства теоремы 34.2 в книге Кормен–Лейзерсон–Ривест–Штайн. Если Вы выберете первый путь, то Вам нужно будет описать как работает новая МТ по той, которая взята из определения $L \in \mathcal{P}$, и доказывать её полиномиальность по входу, а также её возможность разрешать рассматриваемый язык. Если второй, то Вам всё равно нужно очень аккуратно обосновать полиномиальность и корректность нового алгоритма с чёткими доказанными утверждениями. Вообще, поменьше лозунгов и больше сути!

2. Приведите пример языка L, не лежащего в классе $\mathcal{P}$ такого, что язык L^* в классе $\mathcal{P}$ лежит. Важной частью этой задачи является доказательство того, что L действительно не лежит в классе $\mathcal{P}$, что значит, что он не принимается ***НИКАКОЙ*** полиномиальной по входу машиной Тьюринга, разрешающей язык L.

3. Существует ли язык $L \notin \mathcal{P}$, такой, что язык множества его подслов $A(L) \in \mathcal{P}$? По определению, считаем $A(L) = \{b \mid \exists a, c \in \Sigma^* \ : \ |ac| > 0,\ abc \in L\}$.

4. *(Задача 20 из канонического задания)* Регулярный язык L задан регулярным выражением. Постройте полиномиальный алгоритм проверки непринадлежности $\omega \notin L$. Вы должны самостоятельно определить, что понимаете под длиной входа и выписать явную оценку трудоемкости предложенного алгоритма.

5. *(Задача 21 из канонического задания)* Здесь стоит неоднократно прочитать условие! Языки $L_1, L_2, \ldots, L_{2019}$ заданы своими регулярными выражениями. Постройте полиномиальный алгоритм, проверяющий, что пересечение этих языков не пусто ($\cap_{i=1}^{2019} L_i \neq \emptyset$). На семинаре была разобрана аналогичная задача для ДКА. На самом деле, задача с ДКА при нефиксированном входе - числу автоматов - является $PSPACE$-полной, об этом можно почитать как-нибудь здесь на 8-й странице http://www.cs.cornell.edu/ kozen/Papers/LowerBounds.pdf. Возможно, мы расскажем о классе $PSPACE$, но это неточно. Подумайте перед полусеместровой контрольной: связана ли задача о непустоте пересечения конечного семейства ДКА в фиксированном алфавите с задачей о разрешимости СЛАУ над конечным полем?

6. Вычислите $2566 \cdot 9601$ с помощью алгоритма Карацубы. Предполагается, что вы можете производить операции умножения только над числами, не превосходящими 10.

7. Дан массив из n элементов, на которых определено лишь отношение равенства (коммутативное, рефлексивное, транзитивное). Постройте как можно более быстрый алгоритм, проверяющий, есть ли в нем элемент, повторяющийся не менее $n/2$ раз. Используйте не более $O(\log N)$ дополнительной памяти.

8. **В прошлый раз вы расстроили семинаристов своими действиями в подобной задаче**.
ДОКАЖИТЕ, что для нахождения двух минимальных элементов массива необходимо не менее $n + \log_2 n + c$ сравнений.

Практическая часть

(Дедлайн совместно с теоретической)

1. Запрограммируйте алгоритм Штрассена, сравните реальное время его работы со временем работы умножения матриц по определению. Найдите порядок размера матриц, начиная с которого алгоритм Штрассена дает выигрыш.

2. Рассмотрите странный с некоторой точки зрения алгоритм умножения матриц, попробуйте объяснить, почему он дает выигрыш в скорости в сравнении с обоими предыдущими алгоритмами. Укажите его асимптотику.

```
const int N;
std::vector<std::vector<double>> A;
std::vector<std::vector<double>> B;
// filling A, B via some values
// ...
// Now A and B is N * N matrices

std::vector<std::vector<double>> C(N, std::vector<double>(N, 0));

for (size_t i = 0; i < N; ++i)
for (size_t j = i + 1; j < N; ++j)
std::swap(A[i][j], A[j][i]);

for (size_t i = 0; i < N; ++i)
for (size_t j = 0; j < N; ++j) {
register double result = 0;
for (size_t k = 0; k < N; ++k)
result += A[i][k] * B[j][k];
C[i][j] = result;
}

// C is result
```

Необязательные задачи

(можно сдавать до конца семестра)

1. Придумайте способ вычисления периода последовательности $x_n = (b + ba + \ldots + ba^n) \mod M$ с асимптотикой $O(\sqrt{M})$.