Машина Тьюринга. 22.12.25

План

1. Введение и мотивация
    1. Попытки формализовать понятие "алгоритм"
    2. Зачем нужны формальные модели вычислений?
    3. Историческая справка
    4. Цель модели: максимально общая, но точно определённая модель вычислений
2. Машина Тьюринга: формальное определение
    1. Компоненты
    2. Пример простой машины
    3. Диаграмма переходов
3. Вычислимость по Тьюрингу
    1. Что означает?
    2. Тезис Тьюринга-Чёрча
    3. Эквивалентность, рекурсия и т.д.
4. Проблема остановки (Halting Problem)
    1. Формулировка
    2. Интуитивно
    3. Формальное доказательство
    4. Идея редукции
5. Разрешимые и неразрешимые проблемы
6. Зачем? в чём связь?

1. Введение и мотивация

В течение всего курса возникало слово "алгоритм". Алгоритм поиска кратчайших путей, хеш-таблица реализует алгоритм поиска и вставки и т.д. Но что строго считать алгоритмом, а что нет? Это уже вопрос сложный

В начале 20 века математики столкнулись с необходимостью формализовать понятие алгоритма. Почему? потому что появились задачи, для которых не было ясно: можно ли их решить с помощью какого-либо метода или нет?

Пример: *«Существует ли общий метод, который по произвольному уравнению в целых числах определяет, имеет ли оно решение?»*
Это знаменитая **десятая проблема Гильберта (1900)**. Она не просто просит решить уравнение — она спрашивает, существует ли *универсальный алгоритм* для всех таких уравнений.

Для ответа на подобные вопросы нужно сначала чётко определить, что такое "алгоритм". И в 1930-х годах появились несколько независимых, но эквивалентных попыток:

- рекурсивные функци (Гёдель, Клини)
- λ-исчисление (Чёрч)
- самая наглядная и влиятельная - машина Тьюринга (Алан Тьюринг, 1936)

Алан Тьюринг - британский математик, предложил мысленную вычислительную машину. которая работает с лентой, головкой и конечным управлением. Это была попытка смоделировать действия человека, выполняющего вычисления по строгим правилам, с карандашом и листком бумаги.

Почему модель оказалась удачной?

- проста и интуитивна
- всеобъемлюща. позже оказалось, что всё, что можно вычислить на любом другом разумном устройстве (включая современные компьютеры), можно вычислить и на машине Тьюринга
- позволяет говорить о вычислениях как о математических объектах - например, кодировать одну машину Тьюринга как вход другой

Главная цель модели Тьюринга - дать математически строгое определние того, что вообще можно вычислить
Именно с этой моделью связан один из ключевых результатов всей теории вычислений: существуют задачи, которые принципиально не могут быть решены никаким алгоритмом.

---

2. Машина Тьюринга

Машина Тьюринга - математическая модель вычислений, которая состоит из нескольких простых, но мощных компонентов. Несмотря на свою простоту, она может имитировать любую вычислимую функцию.

2.1 Интуитивное представление

Представьте человека, сидящего за столов с бесконечной лентой, разбитой на клетки. В каждой клетке - символ из конечного алфавита (к примеру 0, 1, пробел). У человека есть:

- головка, чтобы читать и писать в текущую клетку
- инструкция (таблица): "Если ты видишь символ *a* и находишься в состоянии *q*, то запиши символ *b*, перейди в состояние *q* и сдвинь головку влево/вправо"
  Этот человек не думает, он просто механически следует инструкциям.

2.2 Формальное определение

Машина Тьюринга (одноклеточная, детерминированная) - это семёрка:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$$

где:

- Q - конечное множество состояний;
- Σ - входной алфавит (не содержит пустой символ ' ')
- Γ - ленточный алфавит, такой что $\Sigma \subseteq \Gamma$ и $\_ \in \Gamma$ (пустой символ часто обозначают как _)
- $\delta : Q \ X \, \Gamma \rightarrow \ Q \ X \, \Gamma \ X \ \{L, R\}$ - функция перехода (для детерминированной машины)
- $q_0 \in Q$ - начальное состояние
- $q_{accept} \in Q$ - состояние принятия
- $q_{reject} \in Q, q_{reject} \neq q_{accept}$ - состояние отклонения

Машина останавливается, когда переходит в q_{accept} или q_{reject}, если этого не происходит - она работает бесконечно

2.3 Как работает машина?

1. Входная строка записана в начале ленты, всё остальное - пустые символы.
2. Головка указывает на первый символ.
3. Машина начинает с состояния q_0.
4. На каждом шаге
   - читает текущий символ
   - по функции δ определяет: что записать, Куда перейти, в какое состояние переключиться
5. Если достигнуто q_{accept} - вход принят; если q_{reject} - отклонён.

Машина Тьюринга не обязана останавливаться на всех входах

2.4 Пример

Задача

Дана строка из 0 и 1, определить есть ли хотя бы одна 1

- Состояния: q_0 (старт), q_{accept}, q_{reject}
- Алфавит: $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \_\}$
- Переходы
  - $\delta(q_0, 0) = (q_0, 0, R)$ - идём вправо пока видим 0
  - $\delta(q_0, 1) = (q_{accept}, 1, R)$ - нашли 1 $\rightarrow$ принимаем
  - $\delta(q_0, \_) = (q_{reject}, \_, R)$ - дошли до конца не найдя 1 $\rightarrow$ отклоняем

2.5 Уточнения

- Рассматривают так же многоклеточные машины - они эквивалентны одноклеточным по вычислительной сложности
- Можно разрешить недетерминизм - тоже не даёт новых вычислимых функций (но влияет на сложность).
- Можно моделировать любую программу как машину Тьюринга (и наоборот).

Машины Тьюринга не используют на практике, только как теоретический инструмент доказательств

3. Вычислимость по Тьюрингу

Что значит, что задача "вычислима"?

3.1 Формальное понятие вычислимости

Говорят, что частичная функция $f : \Sigma^* \to \Sigma^*$ вычислима по Тьюрингу, если существует машина Тьюринга M такая, что:

- для любого входа $w \in dom(f)$, машина M, получив w на ленте, останавливается на оставляет на ленте $f(w)$;
- для любого $w \notin dom(f)$, машина либо не останавливается, либо ведёт себя произвольно (в случае частичной функции это допустимо)

Если функция всюду определена (т.е. $dom(f) = \Sigma^*$) и вычислима, её называют *всюду вычислимой*

аналогично: язык $L \subseteq \Sigma^*$ разрешим, если существует машина Тьюринга, которая всегда останавливается и принимает ровно те строки, что принадлежат L

3.2 Тезис Тьюринга - Чёрча

Теперь самый важный философско - математический момент

Тезис Тьюринга - Чёрча:
Любая функция, которая может быть вычислена каким-либо "эффективным методом" (т.е. алгоритмически), вычислима на машине Тьюринга.

Это не теорема, а тезис - гипотеза, подтверждённая всей практикой вычислений. Почему в него можно верить?

- Все альтернативные формализации алгоритма (λ-исчисление, рекурсивные функции, нормальные алгоритмы Маркова и др.) оказались эквивалентны по вычислительной мощности машине Тьюринга.
- Все реальные языки программирования (C, Python, Java и ассемблер) могут быть смоделированы машиной Тьюринга - и наоборот
- Никто за 90+ лет не предложил разумной модели вычислений, которая строго сильнее машины Тьюринга и при этом реализуема.

3.3 Почему это важно?

- Можно доказать невозможность решения задачи, показывая, что даже машина Тьюринга не может справиться с ней
- Даёт абсолютную границу: не "мы пока не знаем алгоритма", а "алгоритма не существует в принципе"

---

4. Проблема остановки

4.1 Формальное определение

Рассмотрим язык
$HALT = \{(M, w) \mid M - \textit{машина Тьюринга}, w \in \Sigma^*, \ \textit{и } M \textit{ останавливается на входе } w\}$
(M, w) - закодированная пара (машина, вход) в виде сторки, например через двоичное представление

вопрос: разрешим ли язык HALT?
существует ли машина W, которая на любом входе останавливается и выдаёт:

- "да", если M принимает w
- "нет", если M не останавливается w

4.2 Мотивация

Допустим, мы пишем отладчик или антивирус. Хотелось бы иметь функцию, которая скажет, а зациклится ли программа?

Тьюринг доказал в 1936 году что такая функция невозможна

4.3 Доказательство неразрешимости (от противного)

Шаг 1. Предположим, такая машина H существует

- $H((M, w))$ останавливается и выводит:
  - "accept", если $M(w)$ останавливается
  - "reject", если $M(w)$ не останавливается

Шаг 2.
Построим инверсированную машину

Машина D получает на вход описание машины M (то есть ⟨M⟩) и делает следующее:

1. Запускает H(⟨M,⟨M⟩⟩) — то есть спрашивает: «остановится ли M, если ей дать на вход *саму себя*?»
2. Если H отвечает **«да»** (остановится), то D **зацикливается**.
3. Если H отвечает **«нет»** (не остановится), то D **останавливается**.

$D((M))$:

- зацикливается, если M((M)) останавливается
- останавливается, если M((M)) зацикливается

Шаг 3. Применим D к самой себе: D((D))

Остановится ли D на входе D?
Допустим да, тогда по определению, это возможно **только если** D(⟨D⟩) **не останавливается** → противоречие.
Допустим, **нет**. Тогда по определению D, это возможно **только если** D(⟨D⟩) **останавливается** → снова противоречие.

Интересные моменты:

- **Самоприменимость** (машина, которая получает на вход описание самой себя) — мощный приём в теории вычислений.
- **Диагональный аргумент** (в духе Кантора): мы «оборачиваем» предполагаемый решатель против него самого.

4.5 Следствие: границы автоматизации

- Нельзя автоматически проверить, зациклится ли программа.
- Нельзя автоматически доказать корректность произвольной программы.
- Нельзя построить универсальный «анти-баг» детектор.

НО проблема решаема в частных случаях (допустим программа без циклов), однако в общем случае - нет

---

5. Разрешимые и неразрешимые проблемы

1. **Разрешимая (decidable) задача**
— это задача распознавания (язык $L \subseteq \Sigma^*$), для которой существует **машина Тьюринга, которая всегда останавливается** и:

   - принимает вход, если он принадлежит L;
   - отклоняет вход, если он **не** принадлежит L.

   Такая машина называется **решателем (decider)**.
2. **Распознаваемая (recognizable, или semi-decidable) задача**
— это язык L, для которого существует машина Тьюринга, которая:

   - **останавливается и принимает**, если вход $\in L$;
   - **либо отклоняет, либо работает вечно**, если вход $\notin L$.

   Другими словами: вы всегда получите «да», если ответ действительно «да» — но «нет» может никогда не появиться.

🔶 **Важно**:

- Всякая **разрешимая** задача — **распознаваема**.
- Но **не всякая распознаваемая задача — разрешима**.
- Проблема остановки **распознаваема**, но **не разрешима**.

Можно построить машину R, которая на входе $\langle M, w \rangle$:

- **симулирует** работу M на w шаг за шагом;
- если M когда-нибудь остановится — R тоже останавливается и принимает.

Если же $M(w)$ зацикливается — R тоже зациклится.
Но это **достаточно** для распознавания: мы всегда скажем «да» в правильных случаях.

Задача	Разрешима?	Распознаваема?	Комментарий
Проверка, является ли строка палиндромом	+	+	Простой линейный алгоритм
Принадлежность строки	+	+	ДКА всегда останавливается

Задача	Разрешима?	Распознаваема?	Комментарий
регулярному языку			
Принадлежность строке контекстно-свободному языку	+	+	Алгоритм CYK или МП-автомат
Проблема остановки	-	+	Классический пример
Эквивалентность двух машин Тьюринга	-	-	Даже распознать нельзя
Существует ли вход, на котором M останавливается?	-	+	Это язык $L_{halt} = \langle M \rangle \mid \exists w : M(w) \downarrow$ — распознаваем, но не разрешим

Заключение

Одна из главных задач теоретической информатики - понимание границ возможности.

1. Не все задачи можно решить - и это нормально.
2. Разумный дизайн начинается с границ. Знание о неразрешимости позволяет сужать задачу до разрешимого подмножества.
    1. вместо произвольных программ - только программы без циклов
    2. вместо полной эквивалентности - эквивалентность по тестам
    3. вместо общего анализа - статистический анализ с ограничениями
3. Машина Тьюринга - фундамент всей теории