

# Лекция 7

Подготовил Миронов Данил

2.11.15

Теория алгоритмов 2015

# Проблем 1

## Problem (1)

По данным МТ:  $M$  и  $X$  определить  $\exists ? c : |c| < p(|x|) :$

$M(x, c) = 1$ , где  $p$  - известный полином

При условии, что  $M$  - ДМТ, работающая за полином

В этом случае  $P_1$  будет  $NP$  полной задачей

## Вопрос

Почему эта задача в классе  $NP$  ?

## Вопрос

Что такое класс  $NP$  ?

## Ответ

Класс задач, которые решаются на недетерминированной МТ за полиномиальное время

Определение плохое, так как НДМТ это не очень понятное свойство.

Используем второе определение

## Определение

$L$  -  $NP$  язык, если  $\exists M_L$  - ДМТ, работающая за полином такая, что если некое слово  $w \in L$ , то  $\exists c$ , что  $|c| \leq p(|w|)$  и  $M_L(w, c) = 1$ , где  $c$  - сертификат

Это означает, что очень легко проверить, что некое слово принадлежит языку  $L$ . То есть если нам предъявят сертификат, то мы за полиномиальное время убедимся, что условие задачи лежит в  $L$

## Пример

Язык всех графов, содержащих гамильтонов цикл.

Тогда  $w$  - какой-то граф

Говорим, что существует такая процедура сертификации, что граф будет принадлежать множеству всех графов, содержащих гамильтонов цикл, если существует некий сертификат полиномиальной длины такой, что мой сертификат скажет - ОК

Таким сертификатом является собственно гамильтонов цикл

## Пример (Продолжение)

Как мы понимаем, что гамильтонов цикл - это  $NP$  язык ?  
Мы знаем, что есть сертифицирующая машина, которая принимает граф и цикл. Затем убеждается, что этот цикл лежит в графе.

Дальше для каждого графа, который содержит гамильтонов цикл мы можем найти такой сертификат, что эта МТ скажет - ОК.

Если в графе есть гамильтонов цикл, то это будет сам гамильтонов цикл

Если нет гамильтонового цикла, то мы никак не сможем заставить эту машину сказать - ДА.

То есть такая сертифицирующая машина существует и поэтому утверждаем, что  $L$  -  $NP$  язык

### Пример (Итог)

Для  $NP$  языков существует простая (полиномиальная) система проверки вхождения в этот язык на основании каких-то дополнительных данных.

## Вопрос

Почему  $P_1$  это  $NP$  язык ?



Разбираемся по порядку

### Вопрос

Что здесь является словом языка ?

Что является сертификатом ?

### Ответ

На входе есть  $(M, x)$ , нужно уметь проверять, что  $(M, x) \in L_{P1}$

Язык  $L_{P1}$  - язык пар  $(M, x)$ , такой, что  $\exists c$ , такое, что

$M(x, c) = 1$ , где  $M$  - МТ,  $c$  - некий вход

Со всеми сопутствующими ограничениями на полиномиальность.

Сертификатом здесь будет  $c$

Так как в  $L_{P1}$  входит пара  $(M, x)$  для которой  $\exists c$

Если нам предъявят  $c$  то мы сможем убедиться, что пара в языке

## Вопрос

Какая у нас машина для процедуры сертификации ?

## Ответ

Машина, которая производит сертификацию вхождения слова в язык проблемы  $P_1$

У этой машины два аргумента:

- кандидат на слово языка  $M_{L_p}$
- сертификат  $c$

Она должна отвечать ДА или НЕТ

## Определение

Язык называется  $NP$  полным, если

- лежит в классе  $NP$
- $NP$  трудный

## Определение

Язык  $NP$  трудный, когда любая проблема из класса  $NP$  сводится к нему по Карпу

## Определение

Сводимость по Карпу

$L_1 \rightsquigarrow_c L_2$ , если  $\exists f$  такая, что  $f(w) \in L_2 \Leftrightarrow w \in L_1$

$f$  - полиномиальная функция, которая переводит слова, которые мы проверяем на принадлежность к  $L_1$  в другие слова, которые мы проверяем на принадлежность к  $L_2$  и образ слова принадлежит  $L_2$  тогда и только тогда, когда само слово принадлежит  $L_1$

## Вопрос

Почему  $L_{P_1}$   $NP$  трудный ?

## Ответ

Пусть  $L$  - какой-то  $NP$  язык

Как  $L$  свести по Карпу к языку  $L_{P_1}$

Нужно построить  $f$  - полиномиальную функцию

$L$  -  $NP$  язык  $\overset{\text{Опр}}{\Rightarrow} \exists M_L$  такая, что

- работает полиномиальное время

-  $w \in L \Leftrightarrow \exists c : M_L(w, c) = 1$

$w$  превращаем в пару  $(M_L, w)$

Получаем формулировку  $P_1$  задачи

## Вопрос

Как проверить, что  $w \in L \Leftrightarrow (M_L, w) \in P_1$  ?

## Ответ

$\Rightarrow$

Известно, что

$$w \in L \Leftrightarrow \exists c : M_L(w, c) = 1 \Rightarrow (M_L, w) \in P_1 \Rightarrow (M_L, w) \in L_{P_1}$$

$\Leftarrow$

Все переходы обратимы

$$w \in L \underset{\text{Опр } NP}{\Leftrightarrow} \exists c : M_L(w, c) = 1 \underset{\text{Опр } P_1}{\Leftrightarrow} (M_L, w) \in P_1 \Leftrightarrow (M_L, w) \in L_{P_1}$$

### Замечание

Есть задачи, которые являются  $NP$  полными и имеют естественную формулировку.

## Задача (SAT (выполнимости))

Язык булевых формул, которые удовлетворимы

Пусть есть  $x_1, \dots, x_n$  - булевы переменные

И набор дизъюнкций вида

$$\begin{aligned} a_{11}(x_1) \vee a_{12}(x_2) \vee \dots \vee a_{1n}(x_n) \\ \vdots \\ a_{m1}(x_1) \vee a_{m2}(x_2) \vee \dots \vee a_{mn}(x_n) \end{aligned}$$

Где

$$a_{ij} = \begin{cases} \neg \\ \epsilon \\ 0 \end{cases}$$

Таким образом порождаем  $m$  элементарных дизъюнктов

## Замечание

Дизъюнкты выполнимы  $\Leftrightarrow \exists x_1 \dots x_n$ , такие, что все они обращаются в истину

## Пример

$$x_1 \vee \neg x_2 \vee x_3$$

$$\neg x_1 \vee \neg x_3$$

$$x_1 \vee \neg x_2$$

При  $x_1 = 1$   $x_2 = 1$   $x_3 = 0$  получаем тожд. истинное высказывание, значит система выполнима



### Замечание

Дизъюнкты выполнимы  $\Leftrightarrow$  конъюнкция дизъюнктов равна 1 при некоторых  $x_1 \dots x_n$

### Замечание

SAT - язык выполнимых КНФ

### Пример (Невыполнимая КНФ)

$$\begin{array}{l} x_1 \\ \& \\ \neg x_1 \vee \neg x_2 \\ \& \\ \neg x_2 \end{array}$$

## Задача

Докажем, что SAT -  $NP$  полный язык (Теорема Кука)

### Доказательство.

1) Покажем, что SAT лежит в  $NP$

Нужно убедиться, что можно полиномиально быстро проверять решение. Здесь сертификат - это набор  $x_1 \dots x_n$ . Когда мы видим набор и видим формулу, мы можем сразу понять превращает ли он ее в истину. □

## Доказательство.

2) Покажем, что каждая задача из  $NP$  сводится к нашей  
 Уже показали, что каждая задача из  $NP$  сводится к  $P_1$

То есть  $\forall L \in NP \ L \rightsquigarrow_c P_1$

Сводимость по Карпу это транзитивное отношение

(У нас будет композиция полиномиально вычислимых функций, которая полиномиально вычислима)

Нужно доказать, что  $P_1 \rightsquigarrow_c SAT$

Нужно для  $(M, x)$  построить некую булеву формулу  $B$  КНФ такую, что  $B$  выполнима  $\Leftrightarrow \exists c : M(x, c) = 1$



## Продолжение.

Для доказательства применим технику, которую мы использовали в доказательстве теоремы Геделя  
 Сделаем булеву формулу, которая эмулирует работу МТ, поэтому она будет выполнима тогда, когда остановится исходная МТ.

У нас будет тотальная, бинарная МТ

1. Бинарная: алфавит состоит из 1 и 0  $\Rightarrow x, c$  - битовые строки
2. Тотальная: для  $\forall q \in Q$

$$\begin{aligned} q, 1 &\Rightarrow q^1, s^1, d^1 \\ q, 0 &\Rightarrow q^0, s^0, d^0 \end{aligned} \tag{1}$$



## Замечание

\* Для понятного объяснения далее считаем  
Чтобы записать конкретные формулы в дальнейшем

$$\begin{aligned}q, 1 &\Rightarrow q^1, 0, \leftarrow \\ q, 0 &\Rightarrow q^0, s^0, .\end{aligned}$$

## Продолжение.

МТ бинарная, тотальная и полиномиально ограничена.

Ее память тоже ограничена полиномом

Если захотим хранить состояния машины за все время работы, то это все равно будет полином

Чтобы построить булеву функцию, определим, какие переменные будут в нее входить:

$M_{ij}$  - состояние  $i$ -ой ячейки памяти на такте  $j$

$Q_{ij} = 1 \Leftrightarrow$  МТ находилась в состоянии  $i$  на такте  $j$

$P_{ij} = 1 \Leftrightarrow$  МТ на такте  $j$  смотрела на позицию  $i$



## Продолжение.

Теперь хотим переписать (1) как булеву функцию.

Берем состояние МТ  $q$  и две инструкции, которые она выполняет, находясь в этом состоянии

Для каждого  $j, i$  запишем формулы, где  $i$  - индекс пробегающий ленту,  $j$  - индекс, пробегающий время

$$Q_{q,j} \& P_{i,j} \& M_{i,j} \rightarrow Q_{q^{1,j+1}} \& P_{i-1,j} \& \neg M_{i,j+1}$$

Время ограничено полиномом, память ограничена полиномом, количество состояний тоже ограничено полиномом, поэтому это все еще будет полином

Нужно еще добавить условия, чтобы это было осмысленно с точки зрения МТ:

- В каждый момент времени МТ может находиться только в одном состоянии
- В каждый момент времени МТ может смотреть только в одну ячейку памяти





## Продолжение.

Сложная часть еще состоит в том, чтобы показать, что эти КНФ имеют полиномиальный размер. (Иногда, когда мы приводим к КНФ формула может приобрести полиномиальный размер)

"В данном случае верьте мне, они все будут полиномиальны"



## Продолжение.

Как же с помощью построенного решить исходную задачу ?

3. Если  $M(x, c) = 1$ , то  $\exists t$ , что с момента времени  $t$   $M$  находится в состоянии  $\bar{q}$

4. Вход  $M$  - это просто строка  $xs$ . Предполагаем, что есть какой-то внутренний формат, чтобы их разделить

Добавляем дополнительные формулы, так как хотим, чтобы полученная формула была не просто эмуляцией МТ, но и удовлетворяла начальному условию:

- $M_{i,0} = x_i, i < |x|$

- $Q_{\bar{q},D}$ , где  $D$  - общее количество тактов работы машины, которую мы эмулируем

Это все, так как свободными переменными останутся только переменные  $M_{i,|x+1|}, \dots$



### Следствие

*Ограниченную по времени  $MT$  можно смоделировать одной булевой формулой*

### Следствие

*Задача  $SAT$  -  $NP$  - полна*

### Замечание

Многие задачи теории сложности формулируются на языке булевых формул