ЛЕКЦИЯ 1

ВЫЧИСЛИТЕЛЬНЫЕ РЕСУРСЫ И МОДЕЛИ

В курсе «сложность вычислений» рассматривается вопрос, с помощью каких ресурсов можно вычислить поставленную задачу при условии, что задача имеет решение. Рассмотрим основные виды вычислительных ресурсов и способы их измерения.

Вычислительные ресурсы:

1. Время;
2. Память;
3. Случайность;
4. «Подсказка»;
5. Оракул;
6. Длина программы.

Как правило, ограничение по памяти важно для узкого круга задач. Например, для вычислений в космическом пространстве. Однако в прошлом память была довольно дорогим ресурсом, и было необходимо использовать как можно меньше памяти. Первые два вида ресурсов довольно очевидны.

Рассмотрим **случайность**. Вспомним метод Монте – Карло. Изначально он использовался для вычисления интегралов в большом пространстве. Обычно при численном подсчете интеграл разбивают на маленькие элементы и считают значение каждого из них. Однако, если размерность пространства довольно большая, то посчитать значение интеграла становится сложно. Метод Монте – Карло заключается в том, что в область случайным образом бросают точки и смотрят, какой процент точек попал внутрь фигуры, объем которой требуется посчитать. Такая задача решается довольно быстро.

Есть и другие алгоритмы. Например, **задача с проверкой на простоту**. Пусть задано число длиной в 1000 знаков. Необходимо проверить, является ли это число простым. Долгое время не было известно быстрых детерминированных алгоритмов. Приходилось

проверять, делится ли заданное число на все числа от единицы до корня из этого числа. Такой алгоритм очень медленный. Также были известны некоторые вероятностные алгоритмы (например, алгоритм Миллера – Рабина). Такие алгоритмы допускают ошибку с малой вероятностью. Для того чтобы повысить точность, необходимо использовать большее число случайных битов.

В 2002 году был придуман детерминированный алгоритм, который решает задачу за полиномиальное время. Алгоритм использует полином шестой степени. Однако получение случайных битов — довольно затратная операция. В этом случае можно использовать какие-либо физические приборы (например, счетчик Гейгера). Но и в этом случае один случайный бит может «стоить» гораздо больше, чем просто ячейка памяти.

Рассмотрим четвертый ресурс — **«подсказка»**. Наличие «подсказки» означает, что известна какая-то внешняя информация, которая будет одинаковой для всех слов данной длины. На практике это означает, что не нужно решать задачу для всех возможных входов, а достаточно решить задачу для входов конкретной длины. Иногда такие задачи с «подсказкой» можно решать не с помощью программы, а с помощью схемы из функциональных элементов (т. е. не с помощью алгоритмов, а с помощью микросхем).

Пятый ресурс — **оракул**. В первую очередь этот ресурс важен в теории. На практике он используется достаточно редко. Машиной с оракулом называют абстрактную машину, предназначенную для решения какой-либо проблемы разрешимости.

Шестой ресурс — **длина программы**. Нужно учитывать длину программы, если, например, компилятор не может обработать слишком длинный код. Такая задача не очень часто встречаются на практике, но она имеет отношение к теории колмогоровской сложности.

1. Вычислительная модель

Основной вычислительной моделью является **многоленточная машина Тьюринга**. Она позволяет смоделировать такие ресурсы как случайность, «подсказку» и оракул. В некоторых более прикладных задачах используются и другие модели (например, RAM модель). Для машины Тьюринга базовой операцией является один такт машины, а базовой ячейкой памяти — одна ячейка на ленте.

В данном курсе в основном будут рассматриваться задачи распознавания языка, где нужно по заданному слову понять, лежит ли оно в языке или нет. Также будут рассмотрены задачи вычисления функции.

Сложностными классами будут классы языков. Язык — это множество слов, класс — множество языков.

Рассмотрим множество DTIME.

Определение 1: *$DTIME(t(n)) = \{L | \exists$ машина Тьюринга $M, \exists C$ такое, что:*

1. *$x \in L \Leftrightarrow M(x) = 1$,*
2. *$\forall x \in \Sigma^*, M(x)$ работает не больше, чем $C \cdot t(|x|)$ шагов$\}$.*

Σ^* — алфавит $\{0, 1\}$. ♣

Т. е. существует машина, которая работает правильно и на любом входе работает не слишком долго.

Определение 2: *Класс задач, которые разрешимы за полиномиальное время:*

$$P = \bigcup_{C=1}^{\infty} DTIME(n^C).$$

Также будут использоваться следующие классы:

$$\mathrm{E} = \bigcup_{C=1}^{\infty} \mathrm{DTIME}(2^{Cn}),$$

$$\mathrm{EXP} = \bigcup_{C=1}^{\infty} \mathrm{DTIME}(2^{n^C}).$$

Считается, что если язык лежит в классе P, то задача решается на практике. Задачи из класса E и EXP решаются очень долго. Одной из самых известных и до сих пор нерешенных задач является вопрос о равенстве классов сложности P и NP, где NP — nondeterministic polynomial. NP — это то, что можно вычислить за полиномиальное время на недетерминированной машине Тьюринга.

Определение 3: *Недетерминированная машина Тьюринга* $(\Sigma, \Gamma, Q, q_0, q_a, q_r, \delta)$, *где* Σ — *входной алфавит,*

Γ — *ленточный алфавит.*

Главное отличие заключается в том, что δ *будет многозначной функцией.*

$$\delta : \Gamma \times Q \rightrightarrows \Gamma \times Q \times \{L, N, R\}.$$

Т. е. машина вместо одного варианта имеет несколько. При этом нужный вариант машина выбирает не случайным образом.

Переход возможен, если он соответствует хотя бы одному из значений функции δ.
Корректное вычисление — последовательность конфигураций, где все переходы возможны.

Рассмотрим класс NTIME.

Определение 4: $NTIME(t(n)) = \{L | \exists$ *недетерминированная машина Тьюринга* $M, \exists C$ *такое, что:*

1. $x \in L \Leftrightarrow$ *существует корректное вычисление, начинающееся с* $q_0 x$, *и оканчивающееся в состоянии* q_a.

2. $\forall x \in \Sigma^*$ *все корректные вычисления, начинающееся с* $q_0 x$, *имеют длину не больше, чем* $C \cdot t(|x|)\}$. ♣

Аналогично введем следующие классы:

$$\mathrm{NP} = \bigcup_{C=1}^{\infty} \mathrm{NTIME}(n^C),$$

$$\mathrm{NEXP} = \bigcup_{C=1}^{\infty} \mathrm{NTIME}(2^{n^C}).$$

Рассмотрим альтернативное определение NP-класса. Идеология заключается в том, что P — класс задач, которые можно решить быстро, а NP — класс задач, у которых можно быстро проверить решение.

Рассмотрим задачу покраски графа. Требуется раскрасить вершины графа так, чтобы все соседние вершины были различных цветов. Вопрос о том, можно ли покрасить граф в два цвета, довольно простой. Задача покраски в три цвета гораздо сложнее, потому что при покраске каждый раз возникает два возможных варианта. Таким образом, получается экспоненциальный перебор. Однако если есть инструкция, в какие цвета красить вершины, то можно очень легко проверить, правильная ли это раскраска или нет. Задача раскраски графа — одна из самых известных NP-задач.

Определение 5: *$L \in NP$, если $\exists$ полиномиальный (от $(|x|)$) детерминированный алгоритм V, такой, что*

$$x \in L \Leftrightarrow \exists s \in \Sigma^* : V(x, s) = 1.$$

s доказывает, что $x \in V$, а V проверяет правильное ли s, или нет. ♣

Докажем эквивалентность двух определений.

Док-во: $\boxed{1 \Rightarrow 2}$

Пусть s описывает, какие шаги нужно делать недетерминированной машине Тьюринга. Машина одна и таже для всех x. Для конкретного символа и состояния различных вариантов очень много. В s будет записано, какой вариант нужно выбрать на первом шаге, какой вариант на втором шаге и т. д. V — симулятор недетерминированной машины Тьюринга, который моделирует ее работу, выбирая из s информацию о том, какой шаг нужно выбрать. Тогда эквивалентность следует из условия 2 первого определения.

$\boxed{2 \Rightarrow 1}$

Машина M «угадывает» s и запускает $V(x, s)$. s имеет некоторую длину, которая задана алгоритмом V. Получается двоичное дерево ветвей по всем s, из которого следует вычисление из каждого листа для конкретного s.

На ленте должно получиться следующее:

x	#	s

■

Сначала на ленте записано только x. Машина вычисляет значение $l_{\max}$ и ставит метку. После чего машина переходит в состояние написания s. Она записывает 0 или 1 и сдвигается вправо до тех пор, пока не дойдет до метки $l_{\max}$. После чего переходит в начальное состояние и делает тоже самое, что и машина V.