3.10. Комбинаторная теория сложности. Временная и емкостная сложность. Сложностные классы P, NP, PS. Сведения, NP-полные задачи

Теория сложности вычислений является разделом теории вычислений, изучающим стоимость работы, требуемой для решения вычислительной проблемы. Стоимость обычно измеряется абстрактными понятиями времени и пространства, называемыми вычислительными ресурсами.

Алгоритм отождествляется с детерминированной машиной Тьюринга, которая вычисляет ответ по данному на входную ленту слову из входного алфавита ∑ . Временем работы алгоритма TM(x) при фиксированном входном слове x называется количество рабочих тактов машины Тьюринга от начала до остановки машины. Сложностью функции F: ∑* -> {0, 1}, вычисляемой некоторой машиной Тьюринга, называется функция C: N -> N, зависящая от длины входного слова и равная максимуму времени работы машины по всем входным словам фиксированной длины: $C_M(n) = \max_{\{x:\ |x| = n\}} T_M(x)$.

Если для функции F существует детерминированная машина Тьюринга M такая, что $C_M(n) < P(n)$, то говорят, что она принадлежит классу P, или полиномиальна по времени.

В теории выч сложности обычно играет роль полиномиальность/не полиномиальность. Переход между вычислителями (МТ, язык программирования) можно осуществить за полиномиальную прибавку к времени.

Так же играет роль: используем мы ДМТ или НДМТ.

Если для функции F существует недетерминированная машина Тьюринга M такая, что она может допустить каждый вход языка L за время <= P(n), то говорят, что она принадлежит классу NP. **NB:** может допустить означает, что существует такая последовательность недетерминированных переходов, что w из L будет допущено машиной Тьюринга и время за которое она асилит это сделать <= P(|w|).

Класс NP включает в себя класс P.

Язык L_1 называется **сводимым (по Карпу)** к языку L_2, если существует функция, F: ∑* -> ∑*, вычислимая за полиномиальное время, обладающая следующим свойством: F(x) принадлежит L_2 тогда и только тогда, когда x принадлежит L_1. Язык L_2 называется **NP-трудным**, если любой язык из класса NP сводится к нему. Язык называют **NP-полным**, если он NP-труден и при этом сам лежит в классе NP. Таким образом, если будет найден алгоритм, решающий хоть одну NP-полную задачу за полиномиальное время, все NP-задачи будут лежать в классе P.

Класс сложности co-NP – множество языков дополнение которых лежит в NP.

Класс языков PSPACE - множество языков, допустимых детерминированной машиной Тьюринга с полиномиальным ограничением пространства.

Класс языков NPSPACE - множество языков, допустимых недетерминированной машиной Тьюринга с полиномиальным ограничением пространства.

P входит в (или равно) **NP** входит в (или равно) **PSPACE == NPSPACE.** (доказательство PSPACE == NPSPACE смотрим в ХМУ, я возьму экземпляр на экзамен …). Более того: **P** строго входит в **PSPACE**.

Пример сведения по карпу любой задачи из NP к SAT или 3-SAT с доказательством опять смотрим в ХМУ.

Общая схема сведения такова: за полиномиальное время МТ не сможет побывать более чем в P(|w|) ячейках следовательно её состояние можно описать строкой вида $X_1 \dots X_L$ L ~~ P(|w|). Значит её мгновенные состояния будут: $(X_1^{(1)} \dots X_L^{(1)}) \dots (X_1^{(T)} \dots X_L^{(T)})$, T <= P(|w|). Научимся формулировать утверждения вида: если в момент времени t в i-й позиции стояла X_i^t то в i + 1 – й момент будет X_i^{t+1}. При помощи этих утверждений можно сформулировать все правила переходов МТ. Запишем полученный набор правил через оператор && и получим булеву формулу, которая выполнима <=> МТ допускает вход w. Это и есть сведение по Карпу.

NB: помимо сведения по Карпу есть ещё и сведение по Куку, которое опирается на наличие ОРАКУЛА, которой умеет “быстро” решать задачу T_2 и к этой задаче мы хотим свести T_1. Эти сведения не эквивалентны в сведении по Карпу мы “очень верим” в то, что NP != co-NP.

Есть теорема:

NP == co-NP <=> существует NP-C проблема, дополнение до которой лежит в NP. Доказательство – в ХМУ.

3.10

Задача лежит в PSPACE-C, если она из PSPACE и к ней можно полиномиально свести любую задачу из PSPACE. Примером PSPACE-C является проблема формулы с кванторами (имеет ли данная КБФ без свободных переменных значение 1). Подробности в ХМУ.