Практика №7 по курсу «Теория алгоритмов»
«Комплементарные классы сложности»

Группы ФТ-301, ФТ-302

Комплементарный класс множества языков X определяется как $\texttt{co-X} = \{\bar{\mathcal{L}} \mid \mathcal{L} \in X\}$.

Рассмотрим определение класса NP через понятие сертификатов: $\mathcal{L} \in \texttt{NP} \Leftrightarrow$ существует машина Тьюринга M такая, что для любого $x \in \mathcal{L}$ существует *сертификат* y полиномиального размера от длины x, то есть $|y| \leq p(|x|)$ такой, что $M(x, y) = \texttt{accept}$. Если же $x \notin \mathcal{L}$, то для любого полиномиального сертификата верно, что $M(x, y) = \texttt{reject}$.

В рамках такого определения комплементарным классом co-NP будет являться множество языков $\mathcal{L}$ таких, что существует машина Тьюринга M такая, что для любого $x \notin \mathcal{L}$ существует сертификат y полиномиальной длины такой, что $M(x, y) = \texttt{accept}$. Если же $x \in \mathcal{L}$, то для всех полиномиальных сертификатов $M(x, y) = \texttt{reject}$.

1 Упражнение. P = co-P

Докажем, что классы P и co-P совпадают. Для этого возмьём произвольный язык $\mathcal{L} \in \texttt{P}$. Для данного языка существует детерминированная машина Тьюринга M, распознающая его за полиномиальное время. Построим машину Тьюринга M', в которой просто поменяем местами состояния q_{accept} и q_{reject}. Несложно заметить, что M' распознает язык $\bar{\mathcal{L}}$, а значит $\bar{\mathcal{L}} \in \texttt{P}$. Отсюда следует, что $\texttt{co-P} \subset \texttt{P}$. Аналогичным образом можно доказать обратное включение, откуда получаем, что $\texttt{P} = \texttt{co-P}$.

1.1 NP и co-NP

Как соотносятся между собой классы NP и co-NP? Можно ли с ними провернуть такой же трюк для доказательства равенства, а если нет - то в чём будет проблема?

Для начала вспомним структуру недетерминированной машины Тьюринга, после чего рассмотрим определение класса NP через НМТ:

- Недетерминированная машина Тьюринга также задается шестёркой $\langle \Gamma, Q, \delta, q_{start}, q_{accept}, q_{reject} \rangle$, где самое главное отличие состоит в том, что δ является многозначным отображением ($\delta \subset (\Gamma \times Q) \times (\Gamma \times Q \times \{\leftarrow, \uparrow, \rightarrow\})$
- В отличие от детерминированной машины Тьюринга, НМТ имеет несколько путей вычисления, которые в совокупности образуют дерево вычислений
- НМТ принимает вход только в том случае, если существует **хотя бы один** путь вычисления, заканчивающийся в состоянии q_{accept}
- Время работы НМТ вычисляется как длина самого глубокого пути вычисления

Тогда класс NP через НМТ определяется следующим образом: $\mathcal{L} \in \texttt{NP} \Leftrightarrow$ существует **недетерминированная** машина Тьюринга, распознающая язык $\mathcal{L}$ за полиномиальное время.

Проблема при попытке адаптировать НМТ для языка $\mathcal{L}$ к распознаванию языка $\bar{\mathcal{L}}$ возникает из-за того, что простая смена состояний q_{accept} и q_{reject} не приводит к какому-либо предсказемому результату, потому что если среди всех веток вычислений для слова x присутствовала хотя бы одна ветка, заначивающаяся состоянием q_{reject}, то после смены состояний новая НМТ будет всё ещё распознавать слово x исходного языка.

Вопрос о соотношении классов NP и co-NP до сих пор остается открытым.

1.2 Упражнение

Решим следующую задачу: докажите, что если $\texttt{NP} \neq \texttt{co-NP}$, то $\texttt{NP} \neq \texttt{P}$.

Проведём доказательство от противного — пусть $\texttt{NP} \neq \texttt{co-NP}$ и $\texttt{NP} = \texttt{P}$. Но мы знаем, что $\texttt{P} = \texttt{co-P}$, а значит $\texttt{NP} = \texttt{co-NP}$ — противоречие.

2 Задачи связанные с проверкой на простоту

Рассмотрим пару задач, связанных с проверкой числа на простоту:

- $\texttt{IsPrime}(x)$ — является ли число x простым
- $\texttt{IsComposite}(x)$ — является ли число x составным

Будем считать, что исходное число x записано на ленте в двоичном виде, так как в противном случае задача представляется достаточно тривиальной — если x записано в унарной системе счисления, то можно за $O(x)$ проверить число на простоту, однако если же x записано на ленте в системе счисления с основанием больше 1, то длина входа в таком случае равна $O(\log x)$ и тривиальные алгоритмы будут работать за экспоненциалное время относительно длины входа $O(x) = O(2^n),$ где $n = \log x$.

Легко доказать, что `IsComposite` $\in$ `NP`, а `IsPrime` $\in$ `co-NP`, так как для первой задачи существует сертификат $x = a \cdot b, a > 1, b > 1$ разложения на множители, а язык `IsPrime` является комплементарным к языку `IsComposite`.

Более сложной задачей является доказательство соотношения `IsPrime` $\in$ `NP`. Для этого нужно придумать достаточно компактный сертификат простоты числа x (см. `http://bit.ly/prime-cert`).

Ещё более важный результат был получен относительно недавно — в 2008 году было доказано, что `IsPrime` $\in$ `P`, а значит и `IsComposite` $\in$ `P`.

Рассмотрим теперь задачу распознавания `HasBigFactor`(n, d) — имеет ли число n в своём разложении простой множитель $p > d$? Несложно заметить, что `HasBigFactor` $\in$ `NP`, так как существует сертификат принадлежности числа языку - достаточно предъявить разложение $n = a \cdot p$, где p — простое и $p > d$. **Важно** понимать, что либо сертифицирующий алгоритм должен проверять p на простоту, либо мы должны предоставить полиномиальный сертификат простоты числа p (оба варианты будут правильными с учётом результата 2008 года). Также ясно, что `HasBigFactor` $\in$ `co-NP`, так как в качестве отрицательного сертификата достаточно предоставить полное разложение числа на простые множители (и опять же сертифицировать каждый множитель либо проверять множители на простоту в рамках алгоритма MT).

Таким образом `HasBigFactor` $\in$ `NP` $\cap$ `co-NP`. Данный пример интересен тем, что неизвестно, принадлежит ли данная задача классу `P` (ясно, что `P` $\subset$ `NP` $\cap$ `co-NP`).