42. Polynomial Hierarchy

1 Not All the Problems Are from the Class NP

When we introduced the class NP, we argued that most practical problems are from the class NP: namely, in these problems:

- if someone proposes a candidate for a solution,
- then we can check, in feasible time, whether this candidate is indeed a solution.

However, we also mentioned that not all problems are like this. For example, optimization problems are, in general, *not* from the class NP. For example:

- it is feasible to check that a given bridge design satisfies all the specifications, but
- how can we check that a proposed design is optimal – e.g., the cheapest? the only way is to check this optimality is to compare this design with all other possible designs, and there are exponentially many of them.

In this lecture, we will see other examples of problems beyond the class NP.

2 Towards a General Description of Problems

Let us start again with the class P. The simplest class is the class of all the problems that can be solved in feasible (polynomial) time, i.e., the class P.

Each instance of this class can be described by a feasibly checkable formula $C(x)$.

Revisiting the class NP. In the class NP, we need to find a solution y that satisfies a feasible condition $C(x, y)$ for the given input. In mathematics, existence is described by the quantifier symbol $\exists$. Thus, such problems have the form $\exists y\, C(x, y)$.

Alternative notations. That there exists a string y satisfying a property $C(x, y)$ means that:

- either the string $00\ldots00$ has this property,
- or the string $00\ldots01$ has this property, etc.

Thus, "there exists" is nothing else but a long "or". As we recall, in computer engineering, "or" is described as summation symbol $+$. Thus, "there exists" is a sum. The usual symbol for the sum is Σ – a Greek version of the letter S, the first letter of the word "sum". So the class NP is also denoted Σ_1P; here, a subscript 1 means that we have only one quantifier.

As we will see, there are classes described by several quantifiers, they will denoted by Σ_2P, etc.

Similarly, formula with a "universal" quantifier "for all" ($\forall y\, C(x, y)$) would mean that:

- the string $00\ldots00$ has this property,
- and the string $00\ldots01$ has this property, etc.

Thus, "for all" is nothing else but a long "and". As we recall, in computer engineering, "and" is described as a product. Thus, "for all" is a product. The usual symbol for the product is Π – a Greek version of the letter P, the first letter of the word "product". So, the corresponding class will denoted by Π_1P. We can also consider Π_2P, etc.

Such classes form what is known as *polynomial hierarchy.*

Where is optimization in polynomial hierarchy? Finding an optimal solution means finding a solution y such that:

- for every other solution z,
- the solution y is better than the solution z.

This can be described as $\exists y\, \forall z\, C(x, y, z)$, where $C(x, y, z)$ is a feasible formula that describes that y is a solution and that y is better than z.

This formula starts with the existential quantifier and it has two quantifiers, so it belongs to the class Σ_2P.

Classes corresponding to games and conflict situations. Suppose that we want to find a strategy that will enable us to win in 2 of our moves. This means that we want to find a move y so that:

- for every possible opponent's move z,
- we will be able to find our second move t for which we win.

In terms of quantifiers, we have $\exists y\, \forall z\, \exists t\, C(x, y, z, t)$ for some feasible property $C(x, y, z, t)$.

This formula starts with the existential quantifier and has 3 quantifiers, so it belongs to the class Σ_3P.

If our goal is to win in 3 of our moves, we similarly get Σ_5P.

Class PSPACE: what is it and why this notation. What if we do not limit the number of moves – only require that this number is feasible? In this case, we have formulas with feasible number of quantifiers. This class is called PSPACE.

This name comes for historical reasons:

- In the past, computer memory was very limited, so we wanted to check that we can perform computations within a limited memory.
- It turns out that the class of all problem that can be solved within a feasible memory is exactly PSPACE.

General description of a polynomial hierarchy. For every n, we have two classes:

- the class Σ_nP of problems described by formulas

  $$\exists x_1\, \forall x_2\, \ldots C(x, x_1, x_2, \ldots)$$

  with n quantifiers that starts with the existential quantifier $\exists$, and
- the class Π_nP of problems described by formulas

  $$\forall x_1\, \exists x_2\, \ldots C(x, x_1, x_2, \ldots)$$

  with n quantifiers that start with the universal quantifier $\forall$.

Important comment. What if we have two existential quantifiers $\exists x_1\, \exists x_2$ one after another? In this case:

- the existence $\exists x_1\, \exists x_2$ is equivalent to the existence of a pair (x_1, x_2), and
- we have already learned – when we showed that the halting problem is not decidable – that we can feasibly convert pairs to numbers and vice versa.

Thus, if we have two existential quantifiers one after another, we can replace them with a single existential quantifier. Similarly, if we have universal quantifiers one after another, then:

- the formula $\forall x_1\, \forall x_2\, F$ means that the property F holds for all x_1 and for all x_2, and
- this is equivalent to saying that the formula F holds for all possible pairs (x_1, x_2).

We know that we can feasibly convert pairs to numbers and vice versa. Thus, if we have universal quantifiers one after another, we can replace them with a single universal quantifier.

So, it makes sense to only consider cases when after an existential quantifier, there is a universal quantifier, and after the universal quantifier, there is an existential quantifier. This means that, e.g.:

- the class Σ_3P contains formulas of the type $\exists x_1\, \forall x_2\, \exists x_3\, C(x, x_1, x_2, x_3)$, and
- the class Π_3P contains formulas of the type $\forall x_1\, \exists x_2\, \forall x_3\, C(x, x_1, x_2, x_3)$.

Additional notations. There is also a special notation $\Delta_n\mathrm{P}$ for the class of all problems that belong to both these classes, i.e., $\Delta_n\mathrm{P} = \Sigma_n\mathrm{P} \cap \Pi_n\mathrm{P}$.

Relation between newly defined classes. For each formula F, we can always add a "fake" quantifier corresponding to a new variable z which is not occurring in the formula at all: we can have $\exists z\, F$, and we can have $\forall z\, F$. For example, "there exists z for which $2 + 2 = 4$" means exactly the same as "$2 + 2 = 4$". So:

- When we add an existential quantifier to a formula with n quantifiers, we get a formula with $n+1$ quantifiers starting with $\exists$, i.e., we get a formula from the class $\Sigma_{n+1}\mathrm{P}$; thus, $\Sigma_n\mathrm{P} \subseteq \Sigma_{n+1}\mathrm{P}$ and $\Pi_n\mathrm{P} \subseteq \Sigma_{n+1}\mathrm{P}$.
- When we add a universal quantifier to a formula with n quantifiers, we get a formula with $n+1$ quantifiers starting with $\forall$, i.e., we get a formula from the class $\Pi_{n+1}\mathrm{P}$; thus, $\Sigma_n\mathrm{P} \subseteq \Pi_{n+1}\mathrm{P}$ and $\Pi_n\mathrm{P} \subseteq \Pi_{n+1}\mathrm{P}$.

In particular, $\mathrm{NP} = \Sigma_1\mathrm{P} \subseteq \Sigma_2\mathrm{P}$ and $\mathrm{NP} = \Sigma_1\mathrm{P} \subseteq \Pi_2\mathrm{P}$.

For the same reason, for each n, we have $\Sigma_n\mathrm{P} \subseteq \mathrm{PSPACE}$ and $\Pi_n\mathrm{P} \subseteq \mathrm{PSPACE}$.

Comment. These inclusions is why this is called *hierarchy*: some classes are subsets of others.

3 Computing with an Oracle

What is an oracle. In the usual estimation of computational steps, we counted every elementary step, i.e., we went down to the level of elementary computer operations. This makes perfect sense if we write a code "from scratch".

In practice, however, people rarely write a code from scratch, they usually use others' methods and libraries. Even in Java, we do not write our own code for functions like sine or cosine, we use the code contained in the compiler. As a result, it is difficult to estimate the number of computational steps:

- Some of these compilers are proprietary, we do not know how many elementary steps are contained in each such operation.
- Moreover, implementations change, so even if we find the number of operations now, it may change in the next release.

It is therefore desirable to have a measure of a program's computation time that does not depend on how exactly these special methods are implemented. The simplest idea is just to count each call to an external method as 1 step.

In theory of computation, external methods are called *oracles*, and computations in which we use this oracle are called *computations with an oracle*.

Historical comment: where did the word "oracle" come from? In Ancient Greece, there were people who claimed to be able to predict the future

– usually, after taking some hallucinogenic drugs. They were called oracles. People would come to ask their advice and get answers.

No one cares how the oracles come up with an answer: all the users needed was the answer. Similarly, when, in a Java program, we call a method for computing the sine, most of us do not care how this method works, all we need is the answer. That is why this is called oracles.

By the way, in the ancient times, did it work? As expected, sometimes it did, sometimes it didn't – and sometimes, the oracle cheated. Once, when a king asked the oracle to predict the outcome of a decisive battle with his neighboring king, the oracle replied that a great army will be defeated – and the oracle did not say which of the two armies will be defeated :-(.

Relativized classes. For each set A, instead of counting all the steps, we can count checking, for a given x, where $x \in A$, as 1 step. We can then have similarly defined classes P^A – problems that can be solved in polynomial number of steps (but where each step may be solving a problem from the class A).

For example, the fact that every problem from the class NP can be reduced to SAT means that if we count each call to SAT as one step, we can solve all NP problems in polynomial time: $\mathrm{NP} \subseteq \mathrm{P}^{\mathrm{SAT}}$.

We can similarly define classes $\mathrm{NP}^A = \Sigma_1\mathrm{P}^A$, etc.

Relativized classes and polynomial hierarchy: first example. What would, e.g., $\Sigma_3\mathrm{P}^{\Pi_2\mathrm{P}}$ mean?

- By definition of the polynomial hierarchy, Σ_3 means that we have three quantifiers starting with $\exists$, i.e., that we have a formula $\exists x_1\, \forall x_2\, \exists x_3\, C(x, x_1, x_2, x_3)$, where $C(x, x_1, x_2, x_3)$ is feasible with respect to the oracle $\Pi_2\mathrm{P}$.
- By definition of Π_2, this means, in turn, that $C(x, x_1, x_2, x_3)$ has the form $\forall x_4\, \exists x_5\, F(x, x_1, x_2, x_3, x_4, x_5)$, where $F(x, x_1, x_2, x_3, x_4, x_5)$ is really feasible.

Substituting this expression for $C(x, x_1, x_2, x_3)$ into the above formula, we get

$$\exists x_1\, \forall x_2\, \exists x_3\, \forall x_4\, \exists x_5\, F(x, x_1, x_2, x_3, x_4, x_5).$$

This formula starts with $\exists$ and has 5 quantifiers, so it belongs to $\Sigma_5\mathrm{P}$. In other words, $\Sigma_3\mathrm{P}^{\Pi_2\mathrm{P}} \subseteq \Sigma_5\mathrm{P}$.

Relativized classes and polynomial hierarchy: second example. What is $\Sigma_3\mathrm{P}^{\Sigma_2\mathrm{P}}$?

- Similarly to the previous case, we have a formula

  $$\exists x_1\, \forall x_2\, \exists x_3\, C(x, x_1, x_2, x_3),$$

  where $C(x, x_1, x_2, x_3)$ is feasible with respect to the oracle $\Sigma_2\mathrm{P}$.

- By definition of Σ_2, this means, in turn, that $C(x, x_1, x_2, x_3)$ has the form

$$\exists x_4\, \forall x_5\, F(x, x_1, x_2, x_3, x_4, x_5),$$

  where $F(x, x_1, x_2, x_3, x_4, x_5)$ is really feasible.

Substituting this expression for $C(x, x_1, x_2, x_3)$ into the above formula, we get

$$\exists x_1\, \forall x_2\, \exists x_3\, \exists x_4\, \forall x_5\, F(x, x_1, x_2, x_3, x_4, x_5).$$

This formula also starts with $\exists$ and has 5 quantifiers, so it does belong to the class Σ_5P. However, in this case, we can have a better answer. Indeed, as we have mentioned earlier:

- if we have two existential quantifiers one after another, we can replace them with a single existential quantifier, and
- if we have universal quantifiers one after another, we can replace them with a single universal quantifier.

So, we can replace $\exists x_3\, \exists x_4$ with a single quantifier $\exists x_3'$, and get the formula

$$\exists x_1\, \forall x_2\, \exists x_3'\, \forall x_5\, F(x, x_1, x_2, x_3', x_5),$$

i.e., $\Sigma_3 \mathrm{P}^{\Sigma_2 \mathrm{P}} \subseteq \Sigma_4$P.

Try yourself on any example of two other classes of polynomial hierarchy.

4 Relativized P = NP Question

A natural question is: if we consider relativized classes, what can be say about the classes P^A and NP^A? It turns out that:

- there exists an oracle A for which $\mathrm{P}^A \neq \mathrm{NP}^A$, and
- there exists an oracle A for which $\mathrm{P}^A = \mathrm{NP}^A$.

We will not prove the first result, but the second result is easy to prove: we can simply take $A = \mathrm{PSPACE}$. Indeed, in this case:

- $\mathrm{P}^{\mathrm{PSPACE}}$ is simply the class of formulas with a feasible number of quantifiers, and
- $\mathrm{NP}^{\mathrm{PSPACE}} = \Sigma_1 \mathrm{P}^{\mathrm{PSPACE}}$ is formulas for the type $\exists x_1\, C^{\mathrm{PSPACE}}$, i.e., of the type $\exists x_1\, \forall x_2\, \ldots F$, where the number of quantifiers is 1 + feasible.

If we add 1 to feasible, we still get feasible, so indeed $\mathrm{P}^{\mathrm{PSPACE}} = \mathrm{NP}^{\mathrm{PSPACE}}$.