Computational Model

Meng Ziyu

January 16, 2022

1 Class $\mathcal{PP}$

$\mathcal{PP}$, whose full name is **P**robabilistic **P**olynomial Time, is a class of computational model. What makes it different from classes $\mathcal{P}, \mathcal{NP}, \mathcal{NPC}$ is that it uses Randomized Algorithm.

What's Randomized Algorithm? Take an example, given an binary tree, from the root node to the leaf nodes, for each step, we toss a coin to choose one of the two directions. During this process, since that for each time we toss a coin, it's probabilistic and independent from the prior choices, we say that this algorithm is randomized.

So, let's talk back to $\mathcal{PP}$, the definition of it is described below.

> **Class $\mathcal{PP}$:** *$\mathcal{PP}$ is a class of decision problems solvable by a Probabilistic Turning machine in polynomial time, with an error probability of less than 1/2 for all instances.*

A Probabilistic Turning machine is a kind of non-deterministic Turning machine that chooses between the available transitions at each point according to some probability distribution. Thus, for a given input, the Probabilistic Turning machine has stochastic result. More detailed speaking, for a deterministic Turning machine, it consists 7 elements: $M = (Q, \Sigma, \Gamma, q_0, A, R, \delta)$, but for a Probabilistic Turning machine, it has 8 elements: $M = (Q, \Sigma, \Gamma, q_0, A, R, \delta_1, \delta_2)$. In each step, the Probabilistic Turning machine probabilistically and independently choose the transition function δ_1 or δ_2.

Because of this feature, a certain input may be accepted this time and be rejected in the next time. To accommodate this, a language L is said to be recognized with error probability ϵ by a Probabilistic Turning machine M that: 1. $\forall w \in L$, Pr[M accepts w] $\geq$ 1-ϵ; 2. $\forall w \notin L$, Pr[M rejects w] $\geq$ 1-ϵ. And we call this kind of Turing machines that are polynomially-bound and probabilistic as **Probabilistic Polynomial-Time machines** ($\mathcal{PPT}$). Hence, $\mathcal{PP}$ is the complexity class containing all problems solvable by a PPT machine with an error probability of less than 1/2 ($\epsilon \leq 1/2$).

Difference between non-deterministic and probabilistic. The non-deterministic Turning machine is an unreal model that is useful for solving $\mathcal{NP}$ problems. But for probabilistic model, it's a real model.

2 Class $\mathcal{BPP}$

$\mathcal{BPP}$, whose full name is **B**ounded-error **P**robabilistic **P**olynomial time. $\mathcal{BPP}$ is one of the largest practical classes of problems, meaning most problems of interest in $\mathcal{BPP}$ have efficient probabilistic algorithms that can be run quickly on real modern machines

We give the definition of it.

> **Class $\mathcal{BPP}$:** *$\mathcal{BPP}$ is the class of decision problems solvable by a probabilistic Turing machine in polynomial time with an error probability bounded away from 1/3 for all instances.*

The phrase "bounded-probability" means the success probability is bounded away from the number 1/2. In face, replacing the 2/3 by any other constant greater than 1/2 will not change the class defined.

Theorem: *The following are both equivalent definitions of $\mathcal{BPP}$:*

1. $L \in \mathcal{BPP}$ if there exists a $\mathcal{PPT}$ machine M and a positive polynomial that
$$\Pr[M(x) = \chi_L(x)] \geq \frac{1}{2} + \frac{1}{|p(x)|}$$
2. $L \in \mathcal{BPP}$ if there exists a $\mathcal{PPT}$ machine M and a positive polynomial that
$$\Pr[M(x) = \chi_L(x)] \geq 1 - \frac{1}{2^{|p(x)|}}$$

*$M(x) = 1$ *iff* M *accepts* x, $M(x) = 0$ *iff* M *rejects* x

*$\chi_L(x) = 1$ *iff* $x \in L$, $\chi_L(x) = 0$ *iff* $x \notin L$

We could see that the main difference between of $\mathcal{PP}$ and $\mathcal{BPP}$ is that they bounded differently. And thus we have $\mathcal{BPP} \subset \mathcal{PP}$. And also, $\mathcal{BPP}$ contains $\mathcal{P}$, since a deterministic machine is a special case of a probabilistic machine. So we have $\mathcal{P} \subset \mathcal{BPP} \subset \mathcal{PP}$.

3 Class $\mathcal{P}/poly$

In computational complexity theory, $P/poly$ is the complexity class of languages recognized by a polynomial-time Turing machine with a polynomial-bounded advice function.

What is Advise Function? An advice is an extra input to a Turing machine that is allowed to depend on the length n of the input, but not on the input itself. A decision problem is in the complexity class $P/f(n)$ if there is a polynomial time Turing machine M with the following property: *For any n, there is an advice string A of length $f(n)$ such that, for any input x of length n, the machine M correctly decides the problem on the input x, given x and A.*

The formal Definition of $\mathcal{P}/poly$ gives below:

Non-uniform Polynomial Time: *The complexity class non-uniform polynomial time (denoted $\mathcal{P}/poly$) is the class of languages L that can be recognized by a non-uniform sequence of polynomial time "machines". Namely, $L \in \mathcal{P}/poly$ if there exists an infinite sequence of machines $M_1, M_2, \ldots$ that satisfying the following:*

1. *There exists a polynomial $p()$ such that for every n, the description of M_n has a length bounded above $p(n)$;*
2. *There exists a polynomial $q()$ such that for every n, the running time of M_n on each input of length n is bounded above $q(n)$;*
3. *For every n and $x \in \{0,1\}^n$, the machine M_n accepts x iff $x \in L$.*

The most common complexity class involving advice is $P/poly$ where advice length $f(n)$ can be any polynomial in n. $P/poly$ is equal to the class of decision problems such that, for every n, there exists a polynomial size Boolean circuit correctly deciding the problem on all inputs of length n. It's easy to prove it from a high level. The size of a Boolean Circuit is number of its edges and it has size $O(| < M_n > | + n + t^2(n))$ where M_n is the machine that accepts input with length n and $t(n)$ is a bound on the running time of M_n, and since the size of circuit is polynomial, the running time of it is also polynomial.