Chapter 7: Inputs encoding

ENSIIE - Computational complexity theory

Dimitri Watel (dimitri.watel@ensiie.fr)

2022

First remark

Compress an instance

If we can compress an instance, we artificially increase the complexity of the problem, which depends on the size of that instance. We can see that increase as the time we need to extract the instance. Conversely, we can try to extract an instance in order to make it easier. Some problem stay hard even when we extract the instance: those problems are called *strong*. Others are called *weak*.

Usual size of the input

Definition

The size of the input is the number of bits used to encode the input.

Example

- for a graph: number of nodes and number of edges
- for a list: number of elements
- for an integer k: $\log_2(k)$
- for a list of integers: $\sum_{k \in L} \log_2(k)$
- ...

Usually, this size is denoted by $|x|$ or n.

Binary encoding VS Unary encoding

Binary encoding

Encode an integer k with *binary encoding* on a Turing machine consists in writing using the base-2 system with $\log_2(k)$ bits surrounded by two white symbols. The memory space used by k is $log_2(k) + 2$.

Unary encoding

Encode an integer k with *unary encoding* on a Turing machine consists in writing with k cells filled with a 1 surrounded by two white symbols. The memory space used by k is $k + 2$.

Complexity and encoding

Observation

Let Π be a decision problem, x be an instance of Π containing an integer W and $\mathcal{A}$ be an algorithm with complexity $O(W)$ solving Π.

- If W is unary encoded, $\mathcal{A}$ is polynomial.
- If W is binary encoded, $\mathcal{A}$ is exponential.

Strong NP-Completeness

Definition

Let Π be a decision problem, we say that Π is *strongly NP-Complete* if it is NP-Complete when we encode the integers of the instance of Π with unary encoding.

Weak NP-Completeness

Definition

Let Π be a decision problem, we say that Π is *weakly NP-Complete* if it is NP-Complete and if there exists a polynomial algorithm to solve it when we encode the integers of the instance of Π with unary encoding.

Usual size of the input: $2nd$ try.

Definition

Let x be an input containing a set x_0 of non numerical objects and a set of k integers $(x_1, x_2, \dots, x_k)$, we define the size of x in multiple ways :

- the total usual size $|x|$: the number of bits we need to encode it;
- the size $l(x)$ that does not depend on the values of the integers : $|x_0| + k$;
- the size $\max(x)$ related to the integers : $\max\limits_{[\![1;k]\!]} x_i$.

Strong NP-Completeness: alternative definition

Definition

Let Π be a decision problem, we say that Π is *strongly NP-Complete* if it is NP-Complete when the instances are restricted to the ones where $\max(x)$ is polynomially bounded by $l(x)$.

Theorem

The two definitions of strong NP-Completeness are equivalent.

Weak NP-Completeness

Definition: pseudo-polynomial complexity

Let Π be a decision problem, an algorithm solving Π is *pseudo-polynomial* if its time complexity is polynomial in $l(x)$ and $\max(x)$.

Definition

Let Π be a decision problem, we say that Π is *weakly NP-Complete* if it is NP-Complete and if there exists a pseudo-polynomial algorithm to solve it.

Theorem

The two definitions of weak NP-Completeness are equivalent.

Strong $\neq$ Weak

Theorem

If P $\neq$ NP then a strongly NP-Complete problem is not weakly NP-Complete and conversely.

Show that a problem is strongly NP-Complete.

Theorem

Let Π be an NP-Complete problem containing no integer, then Π is strongly NP-Complete.

Theorem

Let Π_1 be a strongly NP-Complete problem and Π_2 be a decision problem. If there exists a polynomial Karp reduction such that, for every instance x of Π_1, x is transformed into an instance y of Π_2 such that $\max(y)$ is polynomially bounded in $l(x)$ and/or $\max(x)$, then Π_2 is strongly NP-Complete.

Last remark

Compress an instance

We explained the compression through the most common example: the encoding of the integers. It is possible to reproduce this theory with any other kind of objects such as the succinct graphs or the succinct boolean formulas.

Arithmetical Turing machine

Definition

A arithmetical Turing machine is a Turing machine that can manipulate integers in constant time, whatever their size is, using for example using the following operations:

- moving on the tape from one integer to another
- copying an integer on the tape
- sum
- product
- division
- comparison
- ...

Strongly polynomial problems

Definition

A decision problem Π is strongly polynomial if there exists an arithmetical Turing machine that solve every instance x of Π with a polynomial time complexity in $I(x)$ and a polynomial space complexity in $|x|$.

Theorem

Every strongly polynomial decision problem Π is in P.

Weakly polynomial problems

Definition

A decision problem Π is strongly polynomial if is belongs to P and if it is not strongly polynomial.

The strong/weak polynomiality has no link with the compression or the decompression of the integers: we only search here for problems for which the complexity is freed from the size of the integers.