CS 341: Algorithms
Module 8: Intractability and Undecidability

Armin Jamshidpey, Eugene Zima

Based on lecture notes by many previous CS 341 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Spring 2019

Certificates

Certificate: Informally, a certificate for a yes-instance I is some "extra information" C which makes it easy to verify that I is a yes-instance.

Certificate Verification Algorithm: Suppose that Ver is an algorithm that verifies certificates for yes-instances. Then Ver(I, C) outputs "yes" if I is a yes-Instance and C is a valid certificate for I. If Ver(I, C) outputs "no", then either I is a no-instance, or I is a yes-instance and C is an invalid certificate.

Polynomial-time Certificate Verification Algorithm: A certificate verification algorithm Ver is a polynomial-time certificate verification algorithm if the complexity of Ver is $O(n^k)$, where k is a positive integer and $n = \mathit{Size}(\mathrm{I})$.

The Complexity Class NP

Certificate Verification Algorithm: A certificate verification algorithm Ver is said to solve a decision problem Π provided that

- for every yes-instance I, there exists a certificate C such that Ver(I, C) outputs "yes".
- for every no-instance I and for every certificate C, Ver(I, C) outputs "no".

The Complexity Class NP denotes the set of all decision problems that have polynomial-time certificate verification algorithms solving them. We write $\Pi \in NP$ if the decision problem Π is in the complexity class NP.

Finding Certificates vs Verifying Certificates: It is not required to be able to find a certificate C for a yes-instance in polynomial time in order to say that a decision problem $\Pi \in NP$.

Important Fact: $P \subseteq NP$.

Certificate verification algorithm for subset sum.
Recall subset sum problem has as input an array of integers $a_1, a_2, \ldots, a_n$ and an integer S (target value).
The question is whether or not there is subset of integers $a_1, a_2, \ldots, a_n$ that sums to S.
A certificate consists of an array $B = [b_1, \ldots, b_n]$, $b_i \in \{0, 1\}$.

```
SubS-D_verifier(a,S,B) {
   for i  = 1 to n do
     S = S - a[i]*b[i]
   od
  return (S==0)
}
```

The verification algorithm takes time $\Theta(n)$, so it is polynomial time in size of the instance. Thus, **SubS-D** $\in NP$.

Recall **CLIQUE-D** decision problem: given a graph $G = (V, E)$, and an integer k, decide if graph G has a cliques of size $\geq k$.
A certificate is subset $C \subseteq V$ that might be a clique of size $\geq k$.

```
CLIQUE-D_verifier(V,E,k,C) {
  m = |C|;
  if m<k return FALSE;
  for i = 1 to |C|-1 do
    for j = i+1 to |C| do
      if E[C[i],C[j]]==0
\\      edge (C[i],C[j]) is not in E
        then return FALSE;
    od
  od
  return TRUE;
}
```

The verification algorithm takes time $O(|V|^2)$, so it is polynomial time in size of the instance. Thus **CLIQUE-D** $\in NP$.

Certificate Verification Algorithm for Hamiltonian Cycle

A certificate consists of an n-tuple, $X = [x_1, \ldots, x_n]$, that might be a hamiltonian cycle for a given graph $G = (V, E)$ (where $n = |V|$).

Algorithm 1: Hamiltonian Cycle Certificate Verification(G, X)

$flag \leftarrow$ **true**
$Used \leftarrow \{x_1\}$
$j \leftarrow 2$
while $(j \leq n)$ **and** $flag$ **do**
    $flag \leftarrow (x_j \notin Used)$ **and** $(\{x_{j-1}, x_j\} \in E)$
    If $(j = n)$ **then** $flag \leftarrow flag$ **and** $(\{x_n, x_1\} \in E)$
    $Used \leftarrow Used \cup \{x_j\}$
    $j \leftarrow j + 1$
return $(flag)$

Polynomial Transformations

For a decision problem Π, let $\mathcal{I}(\Pi)$ denote the set of all instances of Π. Let $\mathcal{I}_{yes}(\Pi)$ and $\mathcal{I}_{no}(\Pi)$ denote the set of all yes-instances and no-instances (respectively) of Π.
Suppose that $\Pi_1$and Π_2 are decision problems. We say that there is a polynomial transformation from Π_1 to Π_2 (denoted $\Pi_1 \leq_P \Pi_2$) if there exists a function $f : \mathcal{I}(\Pi_1) \rightarrow \mathcal{I}(\Pi_2)$ such that the following properties are satisfied:

- $f(\mathrm{I})$ is computable in polynomial time (as a function of size(I), where $\mathrm{I} \in \mathcal{I}(\Pi_1)$)
- if $\mathrm{I} \in \mathcal{I}_{yes}(\Pi_1)$, then $f(\mathrm{I}) \in \mathcal{I}_{yes}(\Pi_2)$
- if $\mathrm{I} \in \mathcal{I}_{no}(\Pi_1)$, then $f(\mathrm{I}) \in \mathcal{I}_{no}(\Pi_2)$

Polynomial Transformations (cont.)

Polynomial transformations are also known as Karp reductions or many-one reductions.

A polynomial transformation can be thought of as a (simple) special case of a polynomial-time Turing reduction, i.e., if $\Pi_1 \leq_P \Pi_2$, then $\Pi_1 \leq_P^T \Pi_2$. Given a polynomial transformation f from Π_1 to Π_2 , the corresponding Turing reduction is as follows:

- Given $\mathrm{I} \in \mathcal{I}(\Pi_1)$, construct $f(\mathrm{I}) \in \mathcal{I}(\Pi_2)$.
- Given an oracle for Π_2 , say A, run $A(f(\mathrm{I}))$.

We transform the instance, and then make a single call to the oracle.

Very important point: We do not know whether I is a yes-instance or a no-instance of Π_1 when we transform it to an instance $f(\mathrm{I})$ of Π_2 .

To prove the implication "if $\mathrm{I} \in \mathcal{I}_{no}(\Pi_1)$, then $f(\mathrm{I}) \in \mathcal{I}_{no}(\Pi_2)$", we usually prove the contrapositive statement "if $f(\mathrm{I}) \in \mathcal{I}_{yes}(\Pi_2)$, then $\mathrm{I} \in \mathcal{I}_{yes}(\Pi_1)$.

Two Graph Theory Decision Problems

Clique Problem

Instance: An undirected graph $G = (V, E)$ and an integer k, where $1 \leq k \leq |V|$.
Question: Does G contain a clique of size $\geq k$? (A clique is a subset of vertices $W \subseteq V$ such that $uv \in E$ for all $u, v \in W, u \neq v$.)

Vertex Cover Problem

Instance: An undirected graph $G = (V, E)$ and an integer k, where $1 \leq k \leq |V|$.
Question: Does G contain a vertex cover of size $\leq k$? (A vertex cover is a subset of vertices $W \subseteq V$ such that $\{u, v\} \cap W \neq \emptyset$ for all edges $uv \in E$.)

Clique $\leq_P$ Vertex-Cover

Suppose that $\mathrm{I} = (G, k)$ is an instance of Clique, where $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$ and $1 \leq k \leq n$.
Construct an instance $f(\mathrm{I}) = (H, \ell)$ of Vertex Cover, where $H = (V, F)$, $\ell = nk$ and

$$v_i v_j \in F \Leftrightarrow v_i v_j \notin E\ .$$

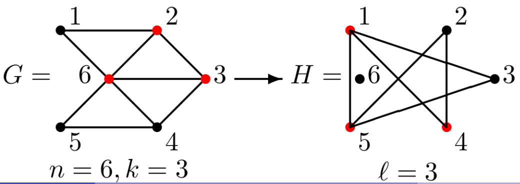
H is called the complement of G, because every edge of G is a non-edge of H and every non-edge of G is an edge of H.
We have *Size*$(\mathrm{I}) = n^2 + \log_2 k \in \Theta(n^2)$. Computing H takes time $\Theta(n^2)$ and computing ℓ takes time $\Theta(\log n)$, so $f(\mathrm{I})$ can be computed in time $\Theta($*Size*$(\mathrm{I}))$, which is polynomial time.

Clique $\leq_P$ Vertex-Cover (cont.)

Suppose I is a yes-instance of Clique. Therefore there exists a set of k vertices W such that $uv \in E$ for all $u, v \in W$. Define $W' = V \setminus W$.

Clearly $|W'| = n - k = \ell$. We claim that W' is a vertex cover of H. Suppose $uv \in F$ (so $uv \notin E$). If $\{u, v\} \cap W' \neq \emptyset$, we are done, so assume $u, v \notin W'$. Therefore $u, v \in W$. But $uv \notin E$, so W is not a clique. This is a contradiction and hence $f(\mathrm{I})$ is a yes-instance of Vertex Cover.

Suppose $f(\mathrm{I})$ is a yes-instance of Vertex Cover. Therefore there exists a set of $\ell = n - k$ vertices W' that is a vertex cover of H. Define $W = V \setminus W'$. Clearly $|W| = k$. We claim that W is a clique in G . . . .



Properties of Polynomial-time Transformations

Theorem 1

If Π_1 and Π_2 are decision problems, $\Pi_1 \leq_P \Pi_2$ and $\Pi_2 \in P$, then $\Pi_1 \in P$.

Proof.

Suppose A is a poly-time algorithm for Π_2 , having complexity $O(m^\ell)$ on an instance of size m. Suppose f is a transformation from Π_1 to Π_2 having complexity $O(n^k)$ on an instance of size n. We solve Π_1 as follows:

1. Given $\mathrm{I} \in \mathcal{I}(\Pi_1)$, construct $f(\mathrm{I}) \in \mathcal{I}(\Pi_2)$.
2. Run $A(f(\mathrm{I}))$.

It is clear that this yields the correct answer. We need to show that these two steps can be carried out in polynomial time as a function of $n = \mathit{Size}(\mathrm{I})$. Step (1) can be executed in time $O(n^k)$ and it yields an instance $f(\mathrm{I})$ having size $m \in O(n^k)$. Step (2) takes time $O(m^\ell)$. Since $m \in O(n^k)$, the time for step (2) is $O(n^{k\ell})$, as is the total time to execute both steps.

Properties of Polynomial-time Transformations (cont.)

Theorem 2

Suppose that Π_1, Π_2 and Π_3 are decision problems. If $\Pi_1 \leq_P \Pi_2$ and $\Pi_2 \leq_P \Pi_3$, then $\Pi_1 \leq_P \Pi_3$.

Proof

We have a polynomial transformation f from Π_1 to Π_2 , and another polynomial transformation$g$ from Π_2 to Π_3 . We define $h = f \circ g$, i.e., $h(\mathrm{I}) = g(f(\mathrm{I}))$ for all instances I of Π_1 . (Exercise: fill in the details.)

The Complexity Class NPC

The complexity class NPC denotes the set of all decision problems Π that satisfy the following two properties:

- $\Pi \in NP$
- For all $\Pi' \in NP, \Pi' \leq_P \Pi$.

NPC is an abbreviation for NP-complete.

Note that the definition does not imply that NP-complete problems exist!