Classes of Problems

- Polynomial Time Verification
- The Classes P and NP
- The Classes EXP and coNP
- NP-HARD and NP-COMPLETE Problems
- Proving NP-HARDNESS
- A first NP-COMPLETE Problem

IMDAD ULLAH KHAN

Polynomial Time Verification

Computing solution to a problem vs checking a proposed solution

- Sometimes computing and verifying a solution are both "easy"
  - e.g. we can compute a MST of a graph and verify whether a claimed solution is indeed a MST in polynomial time

- Sometimes computing is not easy (yet) but verifying is easy
  - e.g. $3\text{-}\textsc{sat}(f)$ we don't know how to find a satisfying solution (or decide if one exists)
  - But verifying a claimed solution can be done in one scan of f

- Sometimes both computing and verifying a "claim" are not easy
  - e.g. not even clear how to "make" the claim that "G has no Hamiltonian cycle"?

Polynomial Time Verification

Need to formalize "checking a solution easily" independent of computation

A decision problem X is efficiently verifiable if

1. The claim: "$\mathcal{I}$ is a **Yes** instance of X" can be made in polynomial bits
   - There exists a polynomial sized certificate for **Yes** instances of X
2. A certificate can be verified in polynomial time
   - There exists a polynomial time algorithm $\mathcal{V}$ that takes the instance $\mathcal{I}$ and the certificate $\mathcal{C}$ such that $\mathcal{V}(\mathcal{I}, \mathcal{C}) =$ **Yes** iff $X(\mathcal{I}) =$ **Yes**

It takes some time to comprehend this, examples should make it clear

Polynomial Time Verification

The $\textsc{mst}(G, k)$ problem: Is there a spanning tree of G of weight $\leq k$?

$\textsc{mst}(G, k)$ is polynomial time verifiable

- A certificate could be the "*claimed spanning tree*" T for G
  - T can be written by writing vertices ids in some order ▷ $O(n \log n)$ bits
  - Adjacency matrix of edges in T ▷ $O(n^2)$ bits
- A verifier can check
  - if vertices of T are in G
  - If all edges in T are actually from G
  - If sum of weights of edges is k
- Alternatively, a certificate could be an empty string ▷ 0 bits
- A verifier can run Kruskals's algorithm to find a MST T of G
- If $w(T) \leq k$, it verifies the claim otherwise rejects the claim

Polynomial Time Verification

3-SAT(f) is polynomial time verifiable

- A certificate would be the assignment of 0 and 1's to all variables
- A verifier can evaluate f with the assignment and if the value of f is 1 it outputs **Yes** (=*verified*) otherwise **No** (=*not verified*)

Note that we do not have to design a verifier or a technique for certifying, we only need to prove their existence

- Verifier does not have to be unique
- There can be many ways to certify

  ▷ e.g. an independent set can be certified as the set of vertices, set of edges, complements thereof

- Verifier does not have to read the certificate, recall the requirement $\mathcal{V}(\mathcal{I}, \mathcal{C}) =$ **Yes** iff $X(\mathcal{I}) =$ **Yes**

Polynomial Time Verification

CLIQUE(G, k) is polynomial time verifiable

Given an instance $[G, k]$ of CLIQUE(G, k)

What could be a certificate of claim "$[G, k]$ is **Yes** instance of CLIQUE$(\cdot, \cdot)$"?

▷ What evidence prove that G has a clique of size k?

Is the certificate of polynomial length?

How can we verify that indeed $[G, k]$ is a **Yes** instance of CLIQUE(G, k)

▷ Does the verifier need to read the certificate?

Is the verifier a polynomial time algorithm?

Polynomial Time Verification

PRIME(n) and COMPOSITE(n) are polynomial time verifiable

▷ Note that they are complement of each other

- A certificate for the COMPOSITE(n) problem can be a factor d
- A verifier can just confirm that $1 < d < n$ and $d|n$

Theorem (AKS(2004))

There exists a polynomial time algorithm to check whether an integer is prime

- A certificate for PRIME(n) can be an empty string
- A verifier exists by the above theorem, using that if n is prime we verify the claim if n is not a prime we reject the claim

Polynomial Time Verification

VERTEX-COVER(G, k) is polynomial time verifiable

- What could be a certificate of claim "G has a vertex cover of size k"?
- How can we verify that indeed "G has a vertex cover of size k?

HAMILTONIAN(G) is polynomial time verifiable

- What could be a certificate of claim "G has a Hamiltonian cycle?"
- How can we verify that indeed G has a Hamiltonian cycle?

Polynomial Time Verification

Are all problems "efficiently" verifiable?

$\overline{\text{3-SAT}}(f)$ It decides whether the given formula f is not satisfiable

▷ sometime referred to as $\text{UNSAT}(f)$

Suppose one wants to claim that the formula f is not satisfiable

▷ Meaning this f is a **Yes** instance of $\overline{\text{3-SAT}}(f)$

How can one make a polynomial sized certificate to make the claim?

▷ "$[0, 1, 1, 0, \ldots 1]$ *does not satisfy* f", *does not mean* f *is not satisfiable*

Polynomial Time Verification

Are all problems "efficiently" verifiable?

Are the following problems polynomial time verifiable?

- $\overline{\text{HAMILTONIAN}}(G)$:

  ▷ It requires **Yes** output, if G does not have a Hamiltonian cycle

- NO-INDEPENEDENT-SET(G, k):

  ▷ It requires **Yes** output, if G does not have an independent set of size k

- MOSTLY-LONG-PATHS(G, s, t, k):

  ▷ Are majority of paths from s to t in G have length at least k