COL351: Analysis and Design of Algorithms

Ragesh Jaiswal, CSE, IITD

Course Overview

- Basic graph algorithms
- Algorithm Design Techniques:
  - Greedy Algorithms
  - Divide and Conquer
  - Dynamic Programming
  - Network Flow
- Computational Intractability

Computational Intractability

Definition (Efficient Algorithms)

An algorithm is said to be *efficient* iff it runs in time polynomial in the input size. Such algorithms are also called *polynomial-time* algorithms.

Definition (Efficient Algorithms)

An algorithm is said to be *efficient* iff it runs in time polynomial in the input size. Such algorithms are also called *polynomial-time* algorithms.

- Question 1: Given a problem, does there exist an efficient algorithm to solve the problem?

Definition (Efficient Algorithms)

An algorithm is said to be *efficient* iff it runs in time polynomial in the input size. Such algorithms are also called *polynomial-time* algorithms.

- Question 1: Given a problem, does there exist an efficient algorithm to solve the problem?
- There are lots of problems arising in various fields for which this question is unresolved.
- Question 2: Are these problems related in some manner?

Definition (Efficient Algorithms)

An algorithm is said to be *efficient* iff it runs in time polynomial in the input size. Such algorithms are also called *polynomial-time* algorithms.

- Question 1: Given a problem, does there exist an efficient algorithm to solve the problem?
- There are lots of problems arising in various fields for which this question is unresolved.
- Question 2: Are these problems related in some manner?
- Question 3: If someone discovers an efficient algorithm to one of these difficult problems, then does that mean that there are efficient algorithms for other problems? If so, how do we obtain such an algorithm.

- NP-complete problems: This is a large class of problems such that all problems in this class are equivalent in the following sense:

  The existence of a polynomial-time algorithm for any one problem in this class implies the existence of polynomial-time algorithm for all of them.

- NP-complete problems: This is a large class of problems such that all problems in this class are equivalent in the following sense:

  The existence of a polynomial-time algorithm for any one problem in this class implies the existence of polynomial-time algorithm for all of them.

- Polynomial-time reduction:
  - Consider two problems X and Y.
  - Suppose there is a *black box* that solves arbitrary instances of problem X.
  - Suppose any arbitrary instance of problem Y can be solved using a polynomial number of standard computational steps and a polynomial number of calls to the black box that solves instance of problem X.
  - If the previous statement is true, then we say that Y is polynomial-time reducible to X. A short notation for this is $Y \leq_p X$.

- Polynomial-time reduction:
  - Consider two problems X and Y.
  - Suppose there is a *black box* that solves arbitrary instances of problem X.
  - Suppose any arbitrary instance of problem Y can be solved using a polynomial number of standard computational steps and a polynomial number of calls to the black box that solves instance of problem X.
  - If the previous statement is true, then we say that Y is polynomial-time reducible to X. A short notation for this is $Y \leq_p X$.
- Claim 1: BIPARTITE-MATCHING $\leq_p$ MAX-FLOW.

- Polynomial-time reduction:
  - Consider two problems X and Y.
  - Suppose there is a *black box* that solves arbitrary instances of problem X.
  - Suppose any arbitrary instance of problem Y can be solved using a polynomial number of standard computational steps and a polynomial number of calls to the black box that solves instance of problem X.
  - If the previous statement is true, then we say that Y is polynomial-time reducible to X. A short notation for this is $Y \leq_p X$.
- Claim 2: Suppose $Y \leq_p X$. If X can be solved in polynomial time, then Y can be solved in polynomial time.

- Polynomial-time reduction:
  - Consider two problems X and Y.
  - Suppose there is a *black box* that solves arbitrary instances of problem X.
  - Suppose any arbitrary instance of problem Y can be solved using a polynomial number of standard computational steps and a polynomial number of calls to the black box that solves instance of problem X.
  - If the previous statement is true, then we say that Y is polynomial-time reducible to X. A short notation for this is $Y \leq_p X$.
- Claim 2: Suppose $Y \leq_p X$. If X can be solved in polynomial time, then Y can be solved in polynomial time.
- Claim 3: Suppose $Y \leq_p X$. If Y cannot be solved in polynomial time, then X cannot be solved in polynomial time.

Definition (Independent Set)

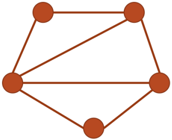
Given a graph $G = (V, E)$, a subset $I \subseteq V$ of vertices is called an independent set of G iff there are no edges between any pair of vertices in I.

Problem

INDEPENDENT-SET: Given a graph $G = (V, E)$ and an integer k, check if there is an independent set of size at least k in G.

Problem

MAXIMUM-INDEPENDENT-SET: Given a graph $G = (V, E)$, output the size of independent set of G of maximum cardinality.



Definition (Independent Set)

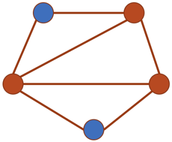
Given a graph $G = (V, E)$, a subset $I \subseteq V$ of vertices is called an independent set of G iff there are no edges between any pair of vertices in I.

Problem

INDEPENDENT-SET: Given a graph $G = (V, E)$ and an integer k, check if there is an independent set of size at least k in G.

Problem

MAXIMUM-INDEPENDENT-SET: Given a graph $G = (V, E)$, output the size of independent set of G of maximum cardinality.



Definition (Independent Set)

Given a graph $G = (V, E)$, a subset $I \subseteq V$ of vertices is called an independent set of G iff there are no edges between any pair of vertices in I.

Problem

INDEPENDENT-SET: Given a graph $G = (V, E)$ and an integer k, check if there is an independent set of size at least k in G.

Problem

MAXIMUM-INDEPENDENT-SET: Given a graph $G = (V, E)$, output the size of independent set of G of maximum cardinality.

- Claim 1: MAXIMUM-INDEPENDENT-SET $\leq_p$ INDEPENDENT-SET.

Definition (Independent Set)

Given a graph $G = (V, E)$, a subset $I \subseteq V$ of vertices is called an independent set of G iff there are no edges between any pair of vertices in I.

Problem

INDEPENDENT-SET: Given a graph $G = (V, E)$ and an integer k, check if there is an independent set of size at least k in G.

Problem

MAXIMUM-INDEPENDENT-SET: Given a graph $G = (V, E)$, output the size of independent set of G of maximum cardinality.

- Claim 1: MAXIMUM-INDEPENDENT-SET $\leq_p$ INDEPENDENT-SET.
- Claim 2: INDEPENDENT-SET $\leq_p$ MAXIMUM-INDEPENDENT-SET.

Definition (Vertex Cover)

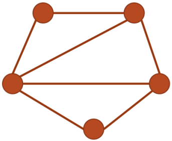
Given a graph $G = (V, E)$, a subset $S \subseteq V$ of vertices is called a vertex cover of G iff for any edge (u, v) in the graph at least one of u, v is in S.

Problem

VERTEX-COVER: Given a graph $G = (V, E)$ and an integer k, check if there is a vertex cover of size at most k in G.

Problem

MINIMUM-VERTEX-COVER: Given a graph $G = (V, E)$, output the size of vertex cover of G of minimum cardinality.



Definition (Vertex Cover)

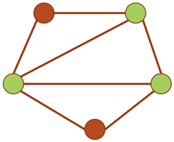
Given a graph $G = (V, E)$, a subset $S \subseteq V$ of vertices is called a vertex cover of G iff for any edge (u, v) in the graph at least one of u, v is in S.

Problem

VERTEX-COVER: Given a graph $G = (V, E)$ and an integer k, check if there is a vertex cover of size at most k in G.

Problem

MINIMUM-VERTEX-COVER: Given a graph $G = (V, E)$, output the size of vertex cover of G of minimum cardinality.



Definition (Vertex Cover)

Given a graph $G = (V, E)$, a subset $S \subseteq V$ of vertices is called a vertex cover of G iff for any edge (u, v) in the graph at least one of u, v is in S.

Problem

VERTEX-COVER: Given a graph $G = (V, E)$ and an integer k, check if there is a vertex cover of size at most k in G.

Problem

MINIMUM-VERTEX-COVER: Given a graph $G = (V, E)$, output the size of vertex cover of G of minimum cardinality.

- Claim 3: MINIMUM-VERTEX-COVER $\leq_p$ VERTEX-COVER.

Definition (Vertex Cover)

Given a graph $G = (V, E)$, a subset $S \subseteq V$ of vertices is called a vertex cover of G iff for any edge (u, v) in the graph at least one of u, v is in S.

Problem

VERTEX-COVER: Given a graph $G = (V, E)$ and an integer k, check if there is a vertex cover of size at most k in G.

Problem

MINIMUM-VERTEX-COVER: Given a graph $G = (V, E)$, output the size of vertex cover of G of minimum cardinality.

- Claim 3: MINIMUM-VERTEX-COVER $\leq_p$ VERTEX-COVER.
- Claim 4: VERTEX-COVER $\leq_p$ MINIMUM-VERTEX-COVER.

- Claim 5: INDEPENDENT-SET $\leq_p$ VERTEX-COVER.

Proof of Claim 5

- Claim 5.1: Let I be an independent set of G, then $V - I$ is a vertex cover of G.

- Claim 5: INDEPENDENT-SET $\leq_p$ VERTEX-COVER.

Proof of Claim 5

- Claim 5.1: Let I be an independent set of G, then $V - I$ is a vertex cover of G.
- Claim 5.2: Let S be a vertex cover of G, then $V - S$ is an independent set of G.

- Claim 5: INDEPENDENT-SET $\leq_p$ VERTEX-COVER.

Proof of Claim 5

- Claim 5.1: Let I be an independent set of G, then $V - I$ is a vertex cover of G.
- Claim 5.2: Let S be a vertex cover of G, then $V - S$ is an independent set of G.
- Claim 5.3: G has an independent set of size at least k if and only if G has a vertex cover of size at most $n - k$.

- Claim 5: INDEPENDENT-SET $\leq_p$ VERTEX-COVER.

Proof of Claim 5

- Claim 5.1: Let I be an independent set of G, then $V - I$ is a vertex cover of G.
- Claim 5.2: Let S be a vertex cover of G, then $V - S$ is an independent set of G.
- Claim 5.3: G has an independent set of size at least k if and only if G has a vertex cover of size at most $n - k$.
- Given an instance (G, k) of the independent set problem, create an instance $(G, n - k)$ of the vertex cover problem, make a single query to the block box for solving the vertex cover problem and return the answer that is returned by the black box. $\square$

- Claim 6: MINIMUM-VERTEX-COVER $\leq_p$ MAXIMUM-INDEPENDENT-SET.

- Claim 6: MINIMUM-VERTEX-COVER $\leq_p$ MAXIMUM-INDEPENDENT-SET.

Proof of Claim 6

- Claim 6.1: G has an independent set of size k if and only if G has a vertex cover of size $n - k$.
- Make a single call to the black box for the maximum independent problem with input G. If the black box returns k, then return $n - k$. $\square$

- Claim 6: MINIMUM-VERTEX-COVER $\leq_p$ MAXIMUM-INDEPENDENT-SET.

Proof of Claim 6

- Claim 6.1: G has an independent set of size k if and only if G has a vertex cover of size $n - k$.
- Make a single call to the black box for the maximum independent problem with input G. If the black box returns k, then return $n - k$. $\square$

Another proof of Claim 6

- MINIMUM-VERTEX-COVER $\leq_p$ VERTEX-COVER
- VERTEX-COVER $\leq_p$ INDEPENDENT-SET
- INDEPENDENT-SET $\leq_p$ MAXIMUM-INDEPENDENT-SET $\square$

Theorem

If $X \leq_p Y$ and $Y \leq_p Z$, then $X \leq_p Z$.

Problem

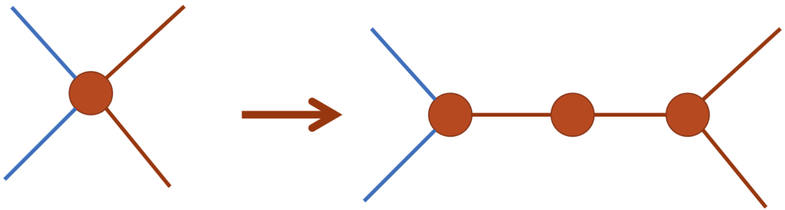
DEG-3-INDEPENDENT-SET: Given a graph $G = (V, E)$ of bounded degree 3 *(i.e., all vertices have degree* ≤ 3*)* and an integer k, check if there is an independent set of size at least k in G.

- Claim 1: INDEPENDENT-SET $\leq_p$ DEG-3-INDEPENDENT-SET

Problem

DEG-3-INDEPENDENT-SET: Given a graph $G = (V, E)$ of bounded degree 3 *(i.e., all vertices have degree* ≤ 3*)* and an integer k, check if there is an independent set of size at least k in G.

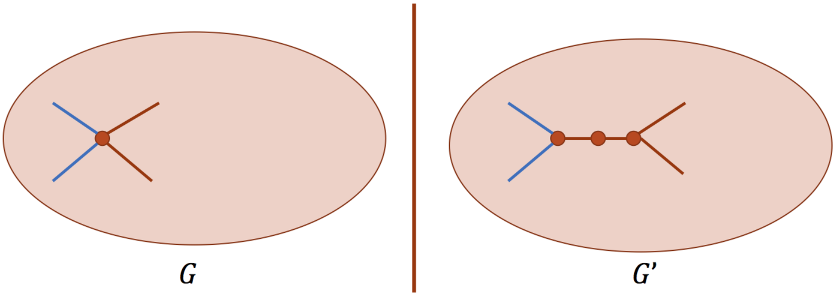
- Claim 1: INDEPENDENT-SET $\leq_p$ DEG-3-INDEPENDENT-SET
  - Idea: "Split" all vertices.



- Claim 1: INDEPENDENT-SET $\leq_p$ DEG-3-INDEPENDENT-SET

Proof of Claim 1

- Consider graph G' constructed by "splitting" a vertex of G.
- Claim 1.1: G has an independent set of size at least k if and only if G' has an independent set of size at least $(k+1)$.



End