Reductions

https://cs.pomona.edu/classes/cs140/
is \( \log(n) \) polynomial? \( \rightarrow \) yes \( \land \)

\( \log(n) = n \)

\( o(1) \) \( \rightarrow \) \( \Omega(1) \) \( \rightarrow \) \( \Omega(n) \)

Receivants (time memory)

Problem input size: \( n \)
Outline

**Topics and Learning Objectives**
- Discuss the process of reducing one problem to another

**Exercise**
- None
P, NP, NP-Complete, NP-Hard

**P**
- Problems that can be solved in polynomial time

**NP**
- Decision problems (output is yes or no) with polynomial size answers that can be verified in polynomial time

**NP-Complete**
- Problems that are in NP and are also NP-Hard

**NP-Hard**
- A known NP-Complete (or NP-Hard) problem can be reduced to the given problem
Reduction

• Instead of taking the time to mathematically prove that some algorithm/problem belongs to a certain class, we can take a shortcut.

$$\in \mathcal{P} \cap \mathcal{NP}$$

• We can put a problem in a specific class by looking at its relative difficulty.

• [Some Problem] is as hard as [Some Other Problem].

• “The Traveling Salesperson Problem (decision variant) is as hard as the Hamiltonian Cycle Problem, and Hamiltonian Cycle is NP-Complete.”
Reduction Preview

A reduction involves two different problems

We can reduce problem A to problem B if
• We have a polynomial time algorithm for converting an input to problem A into an equivalent input for problem B

If we can perform a reduction, then we can say things like
• If B is in P then A is in P
• If B is in NP then A is in NP
• B is at least as hard as A (though B might be much harder—you can always convert a problem into something that takes way more work)
Reduction Preview

• We can do better than the Floyd-Warshall algorithm for graphs that are not very dense.

  • For example, a clever trick reduces the all-pairs shortest path problem (with negative edge lengths) to one invocation of the Bellman-Ford algorithm followed by \(n - 1\) invocations of Dijkstra’s algorithm.

  • This reduction, which is called Johnson’s algorithm, runs in \(O(mn) + (n - 1) \cdot O(m \log n) = O(mn \log n)\) time.

  • This is subcubic in \(n\) except when \(m\) is very close to quadratic in \(n\).
Reduction for NP-Complete

• Given a new problem (and algorithm) called $P_{new}$

• Let’s say we have an algorithm to solve it, but we don’t know to what class it belongs

  Problem $P_{new}$ is at least as hard as problem $P_{known}$

• Reduce $P_{known}$ to $P_{new}$
  • Solve $P_{known}$ using a polynomial number of calls to the algorithm for $P_{new}$
  • Reduce to the **harder** or **known** problem
Example Reduction (Reduce $P_{\text{known}}$ to $P_{\text{new}}$)

1. Instance of $P_{\text{known}}$
2. Transform to $P_{\text{new}}$
3. Instance of $P_{\text{new}}$
4. Solve Instance of $P_{\text{new}}$
5. Solution to $P_{\text{new}}$
6. Transform to $P_{\text{known}}$
7. Solution to $P_{\text{known}}$

New Algorithm for $P_{\text{new}}$

Must take only a polynomial amount of time

WE ALREADY PROVED THE RUNNING TIME OF $P_{\text{known}}$ SO, WE MUST HAVE FOUND A NEW WAY TO IMPLEMENT THE SAME THING USING $P_{\text{new}}$
Prove two algorithms belong to the same class

\( P_{\text{known}} \) is the all-pairs shortest path problem

\( P_{\text{new}} \) is the single-source shortest path problem

Reduce \( P_{\text{known}} \) to \( P_{\text{new}} \)

Both in \( P \)

BF, Dijkstra, Johnson

Must take only a polynomial amount of time
Examples of Reductions

Reduce median selection to sorting.
• Finding the median value of an array of numbers is as hard as sorting the number and sorting the number can be solved in polynomial-time.
• Note: finding the median is easier than comparison-based sorting.

Reduce cycle detection to DFS
• Detecting a cycle in a graph is as hard as performing a depth first search and DFS can be done in polynomial-time.
• This is related to Kruskal’s minimum spanning tree algorithm

Reduce all pairs shortest path to single source shortest path
• Computing all pairs shortest paths is as hard as computing the shortest path from one node to every other node n times, which can be done in polynomial time.
• Invoke polynomial time algorithm “n times” is still polynomial time (just increase exponent by 1).
Reductions (Reducibility)

The *S*-Independent Set Problem

• Given a graph $G$ and a number $S$, is there a set of nodes of size $S$ in $G$ such that no two nodes in the set are directly connected in $G$ (they are independent of each other)?
Reductions (Reducibility)

The $S$-Independent Set Problem

- Given a graph $G$ and a number $S$, is there a set of nodes of size $S$ in $G$ such that no two nodes in the set are directly connected in $G$ (they are independent of each other)?

![Diagram of a graph $G$ with nodes and edges, and a set of $S = 3$ nodes highlighted.](image)
The $k$-Clique Problem

• Given a graph $G$ and a number $k$, is there a set of nodes of size $k$ in $G$ such that all nodes are directly connected with one another?

![Graph $G$ with $k = 4$]
The \textit{k-Clique Problem}

- Given a graph $G$ and a number $k$, is there a set of nodes of size $k$ in $G$ such that all nodes are directly connected with one another?
Reductions (Reducibility)

The $k$-Clique Problem

• Given a graph $G$ and a number $k$, is there a set of nodes of size $k$ in $G$ such that all nodes are directly connected with one another?
Reductions (Reducibility)

The \( k \)-Clique Problem

- Given a graph \( G \) and a number \( k \), is there a set of nodes of size \( k \) in \( G \) such that all nodes are directly connected with one another?
Reductions (Reducibility)

The S-Independent Set Problem
• Given a graph $G$ and a number $S$, is there a set of nodes of size $S$ in $G$ such that no two nodes in the set are directly connected in $G$ (they are independent of each other)?

The $k$-Clique Problem
• Given a graph $G$ and a number $k$, is there a set of nodes of size $k$ in $G$ such that all nodes are directly connected with one another?
Reduce \text{S-Independent Set} to \text{k-Clique}

\textbf{The S-Independent Set Problem}

Given a graph \(G\) and a number \(S\), is there a set of nodes of size \(S\) in \(G\) such that no two nodes in the set are directly connected in \(G\)?

\textbf{The k-Clique Problem}

Given a graph \(G\) and a number \(k\), is there a set of nodes of size \(k\) in \(G\) such that all nodes are directly connected with one another?

We don’t know the computational classification of \text{k-Clique}.

We do know the computational classification of \text{S-Independent Set}.

How do we use \text{S-Independent Set} to find the computational classification of \text{k-Clique}?

Reduce \text{S-Independent Set} to \text{k-Clique}.

If we can perform the reduction, then \text{k-Clique} must be as hard as \text{S-Independent Set}.
Reduce $S$-Independent Set to $k$-Clique

Must take only a polynomial amount of time
We want to find the $S$-Independent set of $G$.

Let's instead find the $k$-Clique of $H$. ($k = S$)

Where $H$ is the complement of $G$. 
We want to find the $S$-Independent set of $G$.

$G$ has an $S$-Independent set if and only if $H$ has a $k$-Clique (we’re not going to prove this).

Let’s instead find the $k$-Clique of $H$. ($k = S$)

Where $H$ is the complement of $G$. 
Let $S = 4$, and thus $k = 4$

We want to find the $S$-Independent set of $G$

$G$ has an $S$-Independent set if and only if $H$ has a $k$-Clique (we’re not going to prove this)

Let’s instead find the $k$-Clique of $H$. ($k = S$)

Where $H$ is the complement of $G$. 
We want to find the $S$-Independent set of $G$

Let $S = 4$, and thus $k = 4$

$G$ has an $S$-Independent set if and only if $H$ has a $k$-Clique (we’re not going to prove this)

Let’s instead find the $k$-Clique of $H$. ($k = S$)

Where $H$ is the complement of $G$. 
We want to find the $S$-Independent set of $G$.

$G$ has an $S$-Independent set if and only if $H$ has a $k$-Clique (we're not going to prove this).

Let's instead find the $k$-Clique of $H$. ($k = S$)

Where $H$ is the complement of $G$.

Let $S = 4$, and thus $k = 4$.
Let $S = 4$, and thus $k = 4$

We want to find the $S$-Independent set of $G$

$G$ has an $S$-Independent set if and only if $H$ has a $k$-Clique
(we’re not going to prove this)

Let’s instead find the $k$-Clique of $H$. ($k = S$)

Where $H$ is the complement of $G$. 
We want to find the $S$-Independent set of $G$.

Let $S = 4$, and thus $k = 4$.

$G$ has an $S$-Independent set if and only if $H$ has a $k$-Clique (we’re not going to prove this).

Let’s instead find the $k$-Clique of $H$. ($k = S$).

Where $H$ is the complement of $G$. 
We want to find the $S$-Independent set of $G$

Let $S = 4$, and thus $k = 4$

These 4 nodes comprise a size 4 clique of $H$; return true

$G$ has an $S$-Independent set if and only if $H$ has a $k$-Clique (we’re not going to prove this)

Let’s instead find the $k$-Clique of $H$. ($k = S$)

Where $H$ is the complement of $G$. 
We want to find the $S$-Independent set of $G$; return true

Let $S = 4$, and thus $k = 4$

These 4 nodes comprise a size 4 independent set of $G$; return true

$G$ has an $S$-Independent set if and only if $H$ has a $k$-Clique (we’re not going to prove this)

Let’s instead find the $k$-Clique of $H$. ($k = S$)

These 4 nodes comprise a size 4 clique of $H$; return true

Where $H$ is the complement of $G$. 
Reduce $S$-Independent Set to $k$-Clique

Since the $S$-Independent Set Problem can be reduced to the $k$-Clique Problem, and the $S$-Independent Set Problem can be solved in polynomial time, then the $k$-Clique Problem can also be solved in polynomial time (or faster).
**The S-Independent Set Problem**

Given a graph $G$ and a number $S$, is there a set of nodes of size $S$ in $G$ such that no two nodes in the set are directly connected in $G$?

**The k-Clique Problem**

Given a graph $G$ and a number $k$, is there a set of nodes of size $k$ in $G$ such that all nodes are directly connected with one another?

We don’t know the computational classification of $k$-Clique.

We do know the computational classification of $S$-Independent Set.

How do we use $S$-Independent Set to find the computational classification of $k$-Clique?

Reduce $S$-Independent Set to $k$-Clique.

If we can perform the reduction, then $k$-Clique must be as hard as $S$-Independent Set.
Proving a Problem $X$ is NP-Complete

(Effectively we are trying to say that $X$ cannot be solved in $O(n^k)$)

1. First prove that $X$ is in NP (it can be verified in polynomial time)

2. Next prove that $X$ is NP-Hard
   1. Reduce some known NP-Complete or NP-Hard problem $Y$ to $X$
   2. This implies that any and all NP-Complete problems can be reduced to $X$
   3. All NP-Complete problems have been reduced to another in an interconnected web (the original problem is known as 3SAT)
3 SAT

satisfiability

OR

Not

\[(X_0 \lor X_6 \lor X_{10}) \land (X_1 \lor X_6 \lor X_{11}) \land \ldots \land (X_6 \lor X_{24} \lor X_{25})\]

AND

Can you come up with some values for \( X = (X_0, \ldots, X_n) \), such that the formula is True?