Recall

Summary of Lecture 19

How to prove a problem $Z$ is NP-complete

1. $Z$ is in NP
2. $X \leq_p Z$, for some known NP-complete problem $X$. Use a \textit{many-one} reduction.

Next: more NP-completeness proofs
Why use a many-one reductions?

- a many-one reduction is a special case of Turing reduction, so it is a stronger result to prove that there is a many-one reduction

- it gives more structure and will make your NP-completeness proofs easier to find and to prove correct

- convention

Is there always a many-one reduction to prove that a problem is NP-complete? i.e., if $X, Z$ are in NP and $X \leq_P Z$ with a Turing reduction, then is there a many-one reduction $X \leq_P Z$?

This is an open question, but it holds in every known case.
Clique.

**Input:** Graph $G = (V, E)$, number $k$.

**Question:** Does $G$ have a clique of size $\geq k$?

Recall: a clique is a set of vertices, every two joined by an edge.

**Observe:** $C \subseteq V$ is a clique in $G$ iff $C$ is an independent set in $G^c$

Recall: $G^c$, the complement of $G$, has vertices $V$, edge $(u,v)$ iff $(u,v) \notin E(G)$

**Theorem.** Clique is NP-complete.

**Proof.**

1. Clique is in NP.

   **Certificate:**
   
   **Verification:**

2. [a known NP-complete problem] $\leq_p$ Clique
2. Independent Set $\leq_P$ Clique

Assume we have a polynomial time algorithm for Clique. Make a polynomial time algorithm for Independent Set — use a many-one reduction.

Input for Independent Set: Graph $G = (V,E)$, number $k$.
Output: Does $G$ have an independent set of size $k$?

- construct a graph $G'$ and choose a number $k'$ such that $G$ has an independent set of size $\geq k$ iff $G'$ has a clique of size $\geq k'$
- run the Clique algorithm on $G', k'$
- return its answer

Construction:

Runtime:

Correctness:
Vertex Cover.

**Input:** Graph $G = (V, E)$, number $k$.

**Question:** Does $G$ have a vertex cover of size $\leq k$?

A *vertex cover* is a set $S \subseteq V$ such that every edge $(u, v) \in E$ has $u$ or $v$ (or both) in $S$.

**Observe:** $S \subseteq V$ is a vertex cover in $G$ iff $V - S$ is an independent set in $G$.

**Theorem.** Vertex Cover is NP-complete.

**Proof.**

1. Vertex Cover is in NP.

**Exercise.**

2.
2. Independent Set $\leq_P$ Vertex Cover

Assume we have a polynomial time algorithm for Vertex Cover. Make a polynomial time algorithm for Independent Set — use a many-one reduction.

Input for Independent Set: Graph $G = (V,E)$, number $k$.
Output: Does $G$ have an independent set of size $k$?

- construct a graph $G'$ and choose a number $k'$ such that
  $G$ has an independent set of size $\geq k$ iff $G'$ has a vertex cover of size $\leq k'$
- run the Vertex Cover algorithm on $G'$, $k'$
- return its answer

Construction:

Runtime:

Correctness:
Road map of NP-completeness

Circuit SAT \leq_P 3-SAT \leq_P Ham.cycle \leq_P TSP

\leq_P Ind. Set \leq_P Vertex Cover \leq_P Set Cover

\leq_P Subset Sum
History of NP-completeness

Proof that 3-SAT is NP-complete due to Stephen Cook, U. Toronto, 1971, and independently to Leonid Levin.

The other “first” NP-completeness proofs we cover are due to Richard Karp, UC Berkeley.

Richard Karp


Stephen Cook, 1968


https://ocul-wtl.orimo.exlibrisgroup.com/permalink/01OCUL_WTL/156lh75/cdi_crossref_primary_10_2307_2273574

https://ocul-wtl.orimo.exlibrisgroup.com/permalink/01OCUL_WTL/156lh75/cdi_crossref_primary_10_2307_2273574
Summary of Lecture 20, Part 1

Clique and Vertex Cover are NP-complete

What you should know from Lecture 20, Part 1:

- how to prove a problem is NP-complete using a polynomial time many-one reduction

Next:

\[
\text{Ind. Set} \leq_p \text{Vertex Cover} \leq_p \text{Set Cover}
\]

\[
\text{Circuit SAT} \leq_p 3\text{-SAT} \leq_p \text{Ham.cycle} \leq_p \text{TSP}
\]

\[
\text{Subset Sum}
\]
**Directed Hamiltonian cycle.**

**Input:** Directed graph $G = (V,E)$.

**Question:** Does $G$ have a directed Hamiltonian cycle?

**Theorem.** Directed Hamiltonian cycle is NP-complete.

**Proof.**

1. Directed Hamiltonian cycle is in NP.

2. $3$-SAT $\leq_p$ Directed Hamiltonian cycle

   Assume we have a polynomial time algorithm for Directed Ham. cycle. Make a polynomial time algorithm for $3$-SAT — use a many-one reduction.

   Input: A $3$-SAT formula $F$ with clauses $C_1 \ldots C_m$ on variables $x_1 \ldots x_n$

   Output: Is $F$ satisfiable?

   - construct a directed graph $G$ such that $G$ has a directed Ham. cycle iff $F$ is satisfiable
   - run the Directed Ham. cycle algorithm on $G$
   - return its answer

This seems tricky! The problems seem so different!
Input: A 3-SAT formula $F$ with clauses $C_1 \ldots C_m$ on variables $x_1 \ldots x_n$

Construct a directed graph $G$ such that

$G$ has a directed Ham. cycle iff $F$ is satisfiable

Idea: for each variable $x_i$, there is a part of $G$ (a “variable gadget”) that chooses whether $x_i$ is True or False
Input: A 3-SAT formula $F$ with clauses $C_1 \ldots C_m$ on variables $x_1 \ldots x_n$

Construct a directed graph $G$ such that

$G$ has a directed Ham. cycle iff $F$ is satisfiable

All the variable gadgets together:
Input: A 3-SAT formula $F$ with clauses $C_1 \ldots C_m$ on variables $x_1 \ldots x_n$

Construct a directed graph $G$ such that

$G$ has a directed Ham. cycle iff $F$ is satisfiable

For each clause $C_j$ we must make a “clause gadget” such that the cycle can go through the clause gadget iff one of the literals is True.

Clause gadget for clause $C = ( x_1 \vee \neg x_2 \vee x_3 )$

Idea: visit vertex $C$ by detouring off the $x_1$ True path OR the $x_2$ False path OR the $x_3$ True path
Input: A 3-SAT formula $F$ with clauses $C_1 \ldots C_m$ on variables $x_1 \ldots x_n$

Construct a directed graph $G$ such that

$G$ has a directed Ham. cycle iff $F$ is satisfiable

For each clause $C_j$ we must make a “clause gadget” such that the cycle can go through the clause gadget iff one of the literals is True.

Note: make sure to leave a spare vertex between two clause detours
Claim. $G$ has a directed Ham. cycle iff $F$ is satisfiable

Proof.

$\Leftarrow$ Suppose $F$ is satisfiable. Traverse the variable paths in the True/False directions. For each clause $C$, at least one literal is True — take the detour from that path to vertex $C$. This gives a directed Ham. cycle.

$\Rightarrow$ Suppose $G$ has a directed Hamiltonian path.

Claim. The only way to visit $C$ is by detouring off a variable path.

Suppose we use $(a,C)$. Show must use $(C,b)$. (e.g. can’t have $e,b,a,C$ then to different chain). Can’t use $(a,d)$ so must enter $d$ from left. Must use $(d,a)$. Can’t use $(b,a)$. Must use $(b,e)$. Must use $(C,b)$.

Thus the Hamiltonian cycle must traverse a True or False path for each variable, and must visit each clause vertex off such a path. So this corresponds to a satisfying truth-value assignment.

Claim. This construction takes polynomial time.
Theorem. [undirected] Hamiltonian cycle is NP-complete.

Proof.
1. Hamiltonian cycle is in NP.

2. Directed Hamiltonian cycle $\leq_p$ Hamiltonian cycle
   Assume we have a polynomial time algorithm for Ham. cycle. Make a polynomial time algorithm for Directed Ham. cycle — use a many-one reduction.

   Input: A directed graph $G$.
   Output: Does $G$ have a directed Ham. cycle?
   - construct an undirected graph $G'$ such that 
     $G$ has a directed Ham. cycle iff $G'$ has a Ham. cycle
   - run the Ham. cycle algorithm on $G'$
   - return its answer
Exercises.

**Theorem.** Travelling Salesman Problem (directed or undirected) is NP-complete.

**Theorem.** Hamiltonian path (directed or undirected) is NP-complete.
Summary of Lecture 20

NP-completeness of Independent Set, Vertex Cover, Hamiltonian cycle, TSP

What you should know from Lecture 20:

- how to prove a problem is NP-complete using a polynomial time many-one reduction

Next:

- Circuit SAT $\leq_P$ 3-SAT
- Ind. Set $\leq_P$ Vertex Cover $\leq_P$ Set Cover
- Ham. cycle $\leq_P$ TSP
- Subset Sum