Cook-Levin

HAM

NP-completeness

CLIQUE

3SAT
Reductions

Or

• How to link between problems' complexity, while not knowing what they are
Goal:

- Formalize the notion of "reductions"

Plan:

- Define Karp reductions
- Example: show HAMPATH ≤p HAMCYCLE
- Closeness under reductions
- Define Cook reductions
- Discuss Completeness
an efficient procedure for $A$ using

$B$

$A$ cannot be radically harder than $B$

- In other words:

$B$ is at least as hard as $A$
Karp reductions – Definition

A is polynomial-time reducible to B
(denote $A \leq_p B$)

If there exists a poly-time-computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that for every $w$

$w \in A \iff f(w) \in B$

i.e., $\exists$ poly-time TM that outputs $f(w)$ on input $w$

$f$ is a poly-time reduction of $A$ to $B$
Karp Reductions - Illustrated

\[ \Sigma^* - A \rightarrow f \rightarrow \Sigma^* - B \]
To Do:

1. Come up with a reduction-function $f$
2. Show $f$ is polynomial time computable
3. Prove $f$ is a reduction, i.e., show:
   - $w \in A \rightarrow f(w) \in B$
   - $w \in A \leftarrow f(w) \in B$

- We'll use reductions that, by default, would be of this type, which is called:
  - Polynomial-time mapping reduction
  - Polynomial-time many-one reduction
  - Polynomial-time Karp reduction
Reductions and Efficiency

Polynomial-time algorithm for $A$

Polynomial-time algorithm for $B$
Hamiltonian Path Instance:

- A directed graph $G=(V,E)$

Decision Problem:

- Is there a path in $G$, which goes through every vertex exactly once?
Hamiltonian Cycle Instance:

- a directed graph $G=(V,E)$.

Decision Problem:

- Is there a simple cycle in $G$ that paths through each vertex exactly once?
\[ f((V, E)) = (V \cup \{u\}, E \cup V \times \{u\}) \]

**Completeness:**

- Given a Hamiltonian path \((v_0, \ldots, v_n)\) in \(G\), \((v_0, \ldots, v_n, u)\) is a Hamiltonian cycle in \(G'\).

**Soundness:**

- Given a Hamiltonian cycle \((v_0, \ldots, v_n, u)\) in \(G'\), removing \(u\) yields a Hamiltonian path.
To Do:

- Come up with a reduction-function $f$
- Show $f$ is polynomial time computable
- Prove $f$ is a reduction, i.e., show:
  - $w \in \text{HAMPATH} \quad \rightarrow \quad f(w) \in \text{HAMCYCLE}$
  - $w \in \text{HAMPATH} \quad \leftarrow \quad f(w) \in \text{HAMCYCLE}$
Closeness Under Reductions: Definition

A complexity class $C$ is **closed under poly-time reductions** if:

- $L$ is reducible to $L'$ and $L' \in C$ \(\Rightarrow\)

  $L$ is also in $C$. 

![Diagram showing the relationship between $L$, $L'$, and $C$]

---

13
**Observation**

**Theorem:**

- $\text{P, NP, PSPACE}$ and $\text{EXPTIME}$ are closed under polynomial-time Karp reductions.

**Proof:**

- Do it yourself!!
Log-space Reductions

$A$ is log-space reducible to $B$ (denote $A \leq_L B$)

If there exists a log-space-computable function $f : \Sigma^* \rightarrow \Sigma^*$

such that for every $w$

$w \in A \iff f(w) \in B$

i.e., $\exists$ log-space TM that outputs $f(w)$ on input $w$

$f$ is a log-space reduction of $A$ to $B$

Theorem:

- $L$, $NL$, $P$, $NP$, $PSPACE$ and $EXPTIME$ are closed under log-space reductions.
Cook Reduction:

- Assuming an efficient procedure that decides B, construct one for A.

Karp is a special case of Cook reduction:

It allows only 1 call to B, whose outcome must be outputted as is.
Cook red. : HAMCYCLE → HAMPATH.

1. Let $E' = E$

2. If $E' = \emptyset$ reject

3. Choose (any) $<u, v>$ in $E'$

4. If HAMPATH ($<V+\{w, z\}, E'+\{<w, u>, <v, z>\}>$) accept

5. $E' = E' - \{<u, v>\}$

6. Go to step 2
**Definition: C-complete**

For a class $C$ of decision problems and a language $L \in C$, $L$ is **C-complete** if:

$L' \in C \Rightarrow L'$ is reducible to $L$.

**Theorem:**

- $L$ is complete for classes $C$, $C' \Rightarrow C = C'$

**Proof:**

- All languages in $C$ and in $C'$ are reducible to $L$, which is in both. Since both are closed under reductions, they're the same.

**Theorem:**

- Any $L \in NPC$, $L \in P \Rightarrow P = NP$
Summary

Discussed types of reductions:
- **Cook** vs. **Karp** reductions
- Poly-time vs. log-space

Defined:
- "completeness"

Find: **L** and **C** s.t. **L** is **C**-complete

Discussed a way to show:
**equality** between **complexity classes**
The Cook/Levin theorem:

SAT is NP-Complete:
Goal:

- In the beginning... of NP-Completeness

Plan:

- SAT - definition and examples
- The Cook-Levin Theorem
- Look ahead
SAT Instance:

- A Boolean formula.

Decision Problem:

- Is the formula **satisfiable**?

SAT or unSAT?

\[ x_1 \land \neg x_1 \]

\[ ((x_1 \lor x_2 \lor \neg x_3) \land \neg x_1) \lor \neg (x_3 \land x_2) \]

Theorem:

- SAT is in NP

Proof:

- Can verify an ass. efficiently
Theorem:
- SAT is NP-Complete

Proof Outline:
- Given an NP machine $M$ and an input $w$, construct a Boolean formula $\varphi_{M,w}$ such that $\varphi_{M,w}$ satisfiable $\iff M$ accepts $w$. 
Represent a computation as a configurations` table

Input

Tape ends

cn

cne

machine` s state

Tableaux
\[ Q = \{ q_0, q_1, q_{\text{accept}}, q_{\text{reject}} \} \]

\[ \Sigma = \{ 0, 1 \} \]

\[ \Gamma = \{ 0, 1, \_ \} \]

\[ \delta(q_0, 1) = \{(q_1, \_, R)\} \]
\[ \delta(q_1, 1) = \{(q_0, \_, R)\} \]
\[ \delta(q_0, 0) = \{(q_0, \_, R)\} \]
\[ \delta(q_1, 0) = \{(q_1, \_, R)\} \]
\[ \delta(q_0, \_) = \{(q_{\text{acc}}, \_, L)\} \]
\[ \delta(q_1, \_) = \{(q_{\text{rej}}, \_, L)\} \]
\( Q = \{ q_0, q_1, q_{\text{accept}}, q_{\text{reject}} \} \)

\( \Sigma = \{ 0, 1 \} \)

\( \Gamma = \{ 0, 1, \_ \} \)

\[ \delta(q_0, 1) = \{ (q_1, \_), R \} \]
\[ \delta(q_1, 1) = \{ (q_0, \_), R \} \]
\[ \delta(q_0, 0) = \{ (q_0, \_), R \} \]
\[ \delta(q_1, 0) = \{ (q_1, \_), R \} \]
\[ \delta(q_0, \_) = \{ (q_{\text{accr}}, \_), L \} \]
\[ \delta(q_1, \_) = \{ (q_{\text{rel}}, \_), L \} \]

\( \Delta_M \subseteq (\Sigma \cup \Gamma \{ \# \})^6 \)

\[
\begin{array}{cccc}
\# & q_0 & 0 & \\
\# & q_0 & 0 & \\
\# & q_0 & 0 & \\
\# & q_0 & 1 & \\
0 & q_0 & 0 & \\
0 & q_0 & 0 & \\
1 & 1 & 1 & 0 \\
0 & q_0 & 1 & 1 \\
1 & q_0 & 0 & 1 \\
1 & q_0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
1 & 0 & q_0 & 1 \\
1 & 0 & q_0 & 1 \\
1 & 0 & q_0 & 1 \\
\_ & 0 & 0 & 1 \\
\_ & 0 & 0 & 1 \\
\_ & 0 & 0 & 1 \\
\_ & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
1 & 0 & q_0 & 1 \\
1 & 0 & q_0 & 1 \\
1 & 0 & q_0 & 1 \\
\end{array}
\]
Boolean $X_{i,j,s}$ standing for: does cell $i,j$ has value $s$?
\( \varphi_{M,w} = \begin{cases} \geq 1 \text{ value} & \begin{cases} \bigwedge_{1 \leq i, j \leq c \epsilon} \left( \bigvee_{s \in (\Gamma \cup \{\#\})} x_{i,j,s} \right) \bigwedge_{s \not\in (\Gamma \cup \{\#\})} \left( x_{i,j,s} \bigvee x_{i,j,t} \right) \end{cases} \\ \text{<2 values} & \text{for each entry} \end{cases} \)

\begin{align*}
\& X_{0,0,\#} \land X_{0,1,q_0} \land X_{0,2,w_1} \land \cdots \land X_{0,n+1,w_n} \land X_{0,n+2,\_} \land \cdots \land X_{0,c \epsilon,\_} \land X_{0,c \epsilon+1,\#} \\
\& \land \left( \bigvee \left( s_{0,0,0}, s_{0,1,0}, s_{1,0,1}, s_{2,0,1} \right) \in \Delta_M \right) \land \left( i = 0, 1, 2, j = 0, 1 \right) \left[ x_{i+i', j+j', s_i, j'} \right] \\
\& \land \exists \text{ accepting configuration} \land x_{i,j,q_{acc}}
\end{align*}
We have just shown SAT is NP-hard, as any NP language can be reduced to SAT.
Henceforth, to show a problem $A$ is NP-hard, it suffices to reduce SAT to $A$. 
Furthermore, once we've shown $A$ is NP-hard, we can reduce from it to show other problems NP-hard.
proven SAT is NP-Complete

Consider SAT the Genesis problem, and explored how to proceed and show other problems are NP-hard
Goal:

- introduce some additional NP-Complete problems.

Plan:

- 3SAT
- CLIQUE & INDEPENDENT-SET
Recall: $L$ is NPC if
- $L \in NP$
- $L$ NP-hard - via Karp-reduction

So far we only showed one such problem: **SAT**

- which, however, is not up for the tasks ahead

Next we show a special case of **SAT** is NPC:
- **3SAT**

3SAT Instance:
- 3CNF formula

Decision Problem:
- Is it satisfiable?

 Conjunctive Normal Form - 3 literals in each clause

3CNF:
- $(x\lor y\lor \neg z) \land (x\lor \neg y\lor z)$
- $(x\lor \neg x\lor \neg x) \land (\neg x\lor \neg \neg x\lor \neg x)$
Claim:
- \( 3\text{SAT} \in \text{NP} \)

Claim:
- \( 3\text{SAT} \in \text{NP-hard} \)

Proof:
- amend our \( \text{SAT} \) formula, so it becomes \( 3\text{CNF} \)
- First make it a \( \text{CNF} \): use \( \text{DNF} \rightarrow \text{CNF} \) on 3rd line

\[
\phi_{M,w} = \bigwedge_{1 \leq i,j \leq cn^e} \left( \bigvee_{s \in (\Gamma \cup \Omega \cup \{\#\})} x_{i,j,s} \right) \bigwedge_{s \notin \Gamma} \left( x_{i,j,s} \lor x_{i,j,t} \right)
\]

\[
\bigwedge_{0 \leq i,j \leq cn^e} \left( \bigvee_{s_0,0, s_0,1, s_1,0, s_1,1, s_2,0, s_2,1} \in \Delta_M \left( \bigwedge_{i'=0,1,2, j'=0,1} \left[ x_{i+i', j+j', s_i, j'} \right] \right) \right)
\]

\[
\bigwedge_{0 \leq i,j \leq cn^e} \left( \bigvee_{s_{i,j,q_{acc}}} x_{i,j,q_{acc}} \right)
\]
(x ∨ y) ∧ (x_1 ∨ x_2 ∨ ... ∨ x_t) ∧ ...

clauses with 1 or 2 literals

replication

clauses with more than 3 literals

split

(x_1 ∨ x_2 ∨ c_{11}) ∧ (¬c_{11} ∨ x_3 ∨ c_{12}) ∧ ... ∧ (¬c_{1t-3} ∨ x_{t-1} ∨ x_t)

QED

3SAT is NP-Complete
CLIQUE instance:
- A graph $G=(V,E)$ and a threshold $k$

Decision problem:
- Is there a set of nodes $C=\{v_1, \ldots, v_k\} \subseteq V$, such that $\forall u, v \in C$: $(u,v) \in E$?

Observation:
- CLIQUE $\in$ NP

Proof:
- Given $C$, verify all inner edges are in $G$
1 vertex for 1 occurrence
inconsistency ⇔ non-edge

$\alpha \neq \delta$

$K = \text{number of clauses}$

$\Sigma_{3SAT} \leq_p \text{CLIQUE}$
Completeness:

Let $A$ be a satisfying assignment to $\phi$, $C(A)$ contains 1 $v_\alpha$ s.t. $A(v_\alpha)$ for every clause.

Soundness:

- In a clique $C$ in $G$ of size $k$, each variable has $\leq 1$ of its literals - vertex in $C$.
- Extend to a satisfying assignment to $\phi$. 

Each triplet disconnected $k$-clique has 1 vertex in each.
INDEPENDENT-SET is NPC

**IS instance:**

- A graph $G=(V,E)$ and a threshold $k$

**Decision problem:**

- Is there a set of nodes $I=\{v_1, \ldots, v_k\} \subseteq V$, s.t. $\forall u, v \in I$: $(u,v) \notin E$

**Observation:**

- $IS \in \text{NP}$

**Proof:**

- Given $I$, verify all inner edges not in $G$

**Observation:**

- $IS$ is NP-hard

*Clique=IS on complement graph*
Reductions

Polynomial Time Reductions

Completeness

Hamiltonian Path

Log Space Reductions

Completeness

Complexity Classes

NP

co-NP

P

L

NL

EXPTIME

PSPACE

Hamilton, William Rowan

Karp, Richard

Cook, Stephen Arthur

Levin, Leonid