Day 4: Reducibility, NP-Completeness, Key Results

Zach Neveu

May 9, 2019

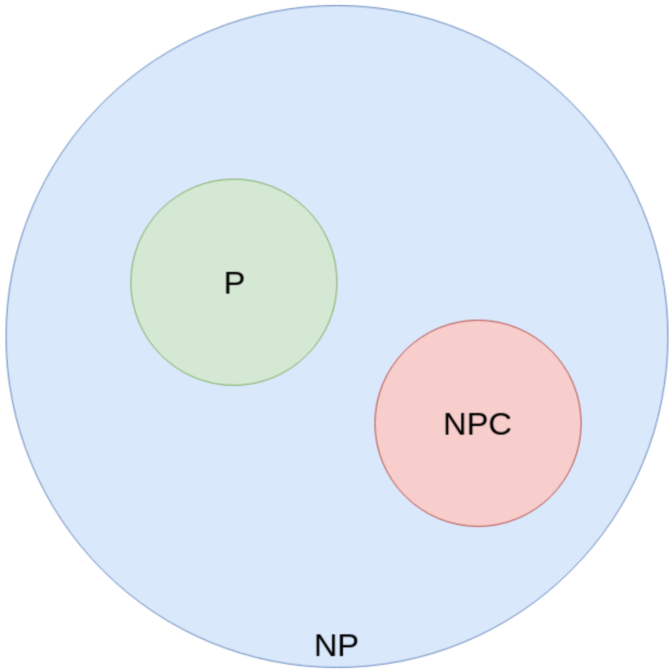

Figure 1: Venn Diagram of P, NP-Complete, and NP

1 Reducibility

Example:
Subset Sum: Given a set of integers and a target, t, is there a subset, S for which $\sum S = t$.
Subset Partition: given a set of integers, can they be partitioned into 2 sets with equal sums?

- If Subset Sum is solved, is it possible to solve subset partition?
- YES! Solve subset sum with $t = \frac{1}{2}\sum S$ where S is all items
- We've just used an SS solver to solve SP! This means that SP reduces to SS.
- If Instance is "no" in SS, it is also "no" in SP

Reducibility: Given problems L_1 and L_2, we say that L_1 is reducible to L_2 in polynomial time if we can rewrite any instance of L_1 as an instance of L_2 such that both instances have the same answer.

Notation: $L_1 \leq L_2$ means that L_1 is reducible to L_2. Starting point, L_1, is on the left. $SP \leq SS$

Example: $HC \leq TSP$
Must be able to rewrite HC as TSP such that they have the same answer.

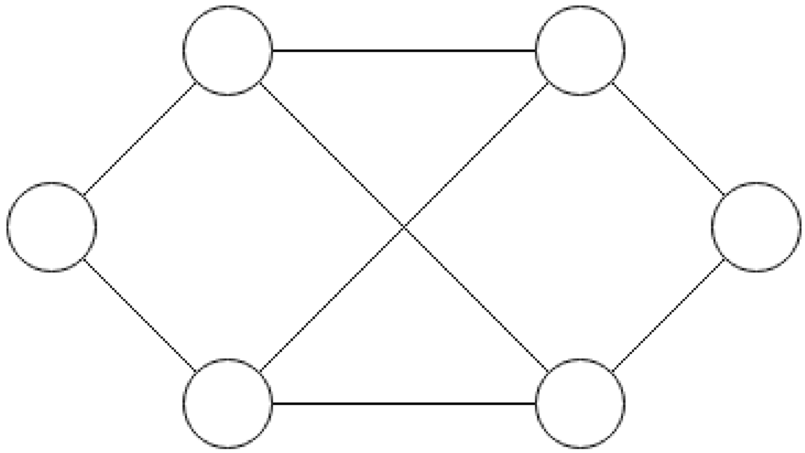

Figure 2: Graph for $HC \leq TSP$ Proof

For proof, must be able to show either:

- A: $yes \rightarrow yes$ and $yes \leftarrow yes$
- B: $yes \rightarrow yes$ and $no \rightarrow no$
- Either A or B requires two steps
- Sometimes one path is much easier
- Option B for $HC \leq TSP$
- If HC is yes instance (HC exists), then the found HC makes TSP a yes instance for weights=1 and bound=num_nodes
- If HC is no instance (no HC exists), then TSP is also no instance because no HCs exist for any cost.

Why is Reduction Useful?

- What if SP is intractable, and SS is in P?
- This is impossible! Reducibility allows you to solve SP in polynomial-time by transforming into SS and solving.

2 NP-completeness

A problem, L, is NP-Complete if:

- $L \in NP$

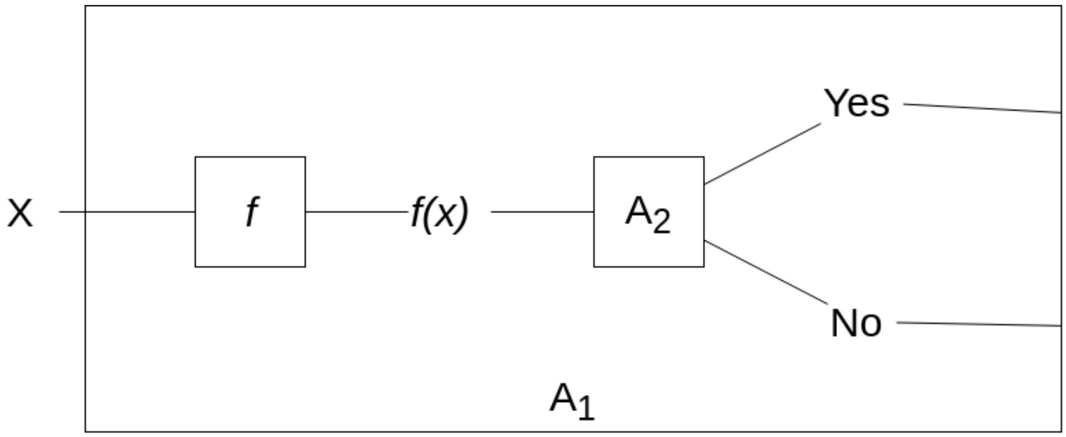

Figure 3: Solving A_1 using A_2 Solver and Reducibility

- For every $L' \in NP$, $L' \leq L$

In words, Every problem in NP should be reducible to L in polynomial time. This essentially means that all NP complete problems are harder than or equal to any other problem in NP. How do we show this?

3 Key Results

1. If $L_1 \leq L_2$, and $L_2 \in P$, then $L_1 \in P$
2. If $L_1 \leq L_2$ and $L_1 \notin P$, then $L_2 \notin P$
3. If L is NPC and $L \in P$, then $NP \in P$
4. If $L' \in NP$ such that $L' \notin P$, then all $NPC \notin P$

4 NPC Examples

Satisfiability (SAT)

- 1971 - Cook found first NPC problem!
- Satisfiability Problem (first one!)
- Consider boolean expression $\overline{x}_3(x_1 + \overline{x}_2 + x_3)$
- Expression is satisfiable if a set of inputs exists which can produce a true output from the expression.
- Given a POS form of an expression, is it satisfiable?
- Ex: $(x_1 + x_2 + x_3)(x_1 + \overline{x}_2)(x_2 + \overline{x}_3)(x_3 + \overline{x}_1)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3)$

  - Each clause must be satisfiable
  - Going by hand from left to right, we can find that this isn't satisfiable.
- How can every problem be reduced to this?
- All problems in NP have a verification algorithm
- Verification algorithm can be expressed as a satisfiability instance, this is the reduction.
- This shows that $SAT \in NPC$! First problem ever done.
- This result can be leveraged to prove that other problems are NPC
- $NP \leq SAT$

Evolution of Problems

- Year after SAT, first 10 problems shown to be NPC
- After 50 years there are TONS of problems in the list of NPC
- Problems from every field on here.
- When you have a new problem, look for a similar problem that is proved to be NPC and reduce it to your problem.

Arbitrary Problem L_2

- If $L_1 \in NPC$ and $L_1 \leq L_2$ than $L_2 \in NPC$