Tandy Warnow

December 26, 2016

Introduction to Computational Complexity

Tandy Warnow

Today

Today we will discuss:

- $\mathcal{NP}$
- co-$\mathcal{NP}$
- $\mathcal{P}$
- $\mathcal{NP}$-hard and $\mathcal{NP}$-complete

Decision Problems

A **decision problem** returns YES or NO on every input. Hence, a decision problem is usually described by a YES/NO question.

- 2-colorability (Can the input graph be vertex colored with red and blue, so that no edge connects vertices of the same color?)
- 3-colorability (Can the input graph be vertex colored with red, blue, and green, so that no edge connects vertices of the same color?)
- 2-SAT (Is the input 2CNF formula satisfiable?)
- 3-SAT (Is the input 3CNF formula satisfiable?)
- Hamiltonian Graph (Does the input graph have a cycle that visits every vertex exactly once?)
- Eulerian Graph (Does the input graph have a cycle that visits every edge exactly once?)

3-colorability

Suppose I want to figure out if a graph $G = (V, E)$ has a proper 3-coloring of the vertices.
My friend says it does but I'm not sure, and my graph is too big for me to look at all the possible colorings.
How can my friend convince me?

3-colorability: proving YES

The proof that $G = (V, E)$ has a 3-coloring is simple:

- show the assignment of colors to the vertices
- verify no two adjacent vertices have the same color

Note:

- The time to prove YES is polynomial *once* the coloring is found
- This is not relevant to the case where the graph cannot be 3-colored. Thus the 4-clique cannot be 3-colored, so we wouldn't try to prove it can be.

3-colorability: proving NO

Now suppose I have another large graph $G = (V, E)$, and my friend says it doesn't have a proper 3-coloring of its vertices. How can my friend convince me that she is right?
Certainly she can list all the 3^n colorings (where $n = |V|$) and we can check them all (exhaustive search) to see if any of them are proper. This would work, but it is really expensive.

Decision Problems

Remember that a decision problem returns YES or NO for every input.

- The YES-instances are the inputs for which the answers are YES.
- The NO-instances are the inputs for which the answers are NO.

$\mathcal{NP}$

$\mathcal{NP}$ = "Non-deterministic Polynomial Time"

Definition: $\mathcal{NP}$ is the **set** of **decision problems** for which you can prove YES-instances are YES-instances in polynomial time.

$\mathcal{NP}$

Which of these decision problems are in $\mathcal{NP}$?

- 2-colorability (Can the input graph be vertex colored with red and blue, so that no edge connects vertices of the same color?)
- 3-colorability (Can the input graph be vertex colored with red, blue, and green, so that no edge connects vertices of the same color?)
- 2-SAT (Is the input 2CNF formula satisfiable?)
- 3-SAT (Is the input 3CNF formula satisfiable?)
- Hamiltonian Graph (Does the input graph have a cycle that visits every vertex exactly once?)
- Eulerian Graph (Does the input graph have a cycle that visits every edge exactly once?)

co-$\mathcal{NP}$

co-$\mathcal{NP}$ is the set of decision problems where there are polynomial time proofs that NO-INSTANCES are NO-INSTANCES.
We won't discuss co-$\mathcal{NP}$ very much.

$\mathcal{P}$

$\mathcal{P}$ is the set of decision problems that can be solved in polynomial time.
Which of these decision problems are in $\mathcal{P}$?

- 2-colorability (Can the input graph be vertex colored with red and blue, so that no edge connects vertices of the same color?)
- 3-colorability (Can the input graph be vertex colored with red, blue, and green, so that no edge connects vertices of the same color?)
- 2-SAT (Is the input 2CNF formula satisfiable?)
- 3-SAT (Is the input 3CNF formula satisfiable?)
- Hamiltonian Graph (Does the input graph have a cycle that visits every vertex exactly once?)
- Eulerian Graph (Does the input graph have a cycle that visits every edge exactly once?)

$\mathcal{P}$ and $\mathcal{NP}$

- Note that $\mathcal{P} \subseteq \mathcal{NP}$.
- Hence $\mathcal{NP}$ contains easy problems and perhaps not so easy problems.
- Does $\mathcal{NP}$ contain problems that *cannot* be solved in polynomial time? If so then $\mathcal{P} \neq \mathcal{NP}$. Otherwise, $\mathcal{P} = \mathcal{NP}$.

Reducing between problems

Suppose you have two decision problems, π and π'.
Suppose $\pi' \in \mathcal{P}$, and that $\mathcal{A}$ is an algorithm to solve π'.
We would like to use $\mathcal{A}$ to help solve π *in polynomial time*.

To do this, we try to make the inputs to π look like inputs to π'.
What else do we need?

Karp Reductions

Suppose you have two decision problems, π and π'.
Suppose $\pi' \in \mathcal{P}$, and that $\mathcal{A}$ is a polynomial time algorithm to solve π'.
Suppose we have a function F that maps inputs to π to inputs to π', so that:

- YES-instances of π map to YES-instances of π'
- NO-instances of π map to NO-instances of π'
- The function F takes polynomial time to compute
- The size of $F(I)$ is polynomial in the size of I for any input I

Such a function F is called a *Karp* reduction, after Dick Karp (Berkeley) who came up with them.

Karp reduction

If π and π' are two decision problems in NP and F is a Karp reduction from π to π', we write this as

$$\pi \propto \pi',$$

and say that "π reduces to π'".

Note that if $\pi_1 \propto \pi_2$ and $\pi_2 \propto \pi_3$, then $\pi_1 \propto \pi_3$. That is, Karp Reductions are transitive.

Karp Reductions

Suppose $\pi' \in \mathcal{P}$, and that $\mathcal{A}$ is a polynomial time algorithm to solve π'.
Suppose $\pi \propto \pi'$, and that F is the Karp Reduction.

Given instance I to π,

- Compute $F(I)$, which is an instance of π'.
- Run algorithm $\mathcal{A}$ on $F(I)$
- If $\mathcal{A}$ says YES, then return YES; otherwise return NO.

So: if you can find a Karp Reduction from π to π', then if π' can be solved in polynomial time then so can π!

Karp reduction

We can summarize the previous discussion as follows:

Theorem: If $\pi' \in P$ and $\pi \propto \pi'$ then $\pi \in P$.

Karp reduction

Question: If $\pi \in P$ and $\pi \propto \pi'$, do we learn anything?

Karp reduction

Remember the 2-colorability problem:

- Input: Graph $G = (V, E)$
- Question: Can we 2-color the vertices of G so that no two adjacent vertices get the same color?

and consider the 3-colorability problem:

- Input: Graph $G = (V, E)$
- Question: Can we 3-color the vertices of G so that no two adjacent vertices get the same color?

We will show that 2-colorability $\propto$ 3-colorability.

2-colorability $\propto$ 3-colorability

Let F map instances of 2-colorability to instances of 3-colorability as follows.
Given graph $G = (V, E)$, let $F(G)$ be the graph $G' = (V', E')$ defined by

- $V' = V \cup \{v^*\}$, where v^* is a new vertex
- $E' = E \cup \{(v^*, v) : v \in V\}$

It is not hard to see that F is a Karp reduction.
(In particular, G can be 2-colored if and only if $F(G)$ can be 3-colored.)

Yet, 2-colorability is in $\mathcal{P}$.

Do we learn anything about 3-colorability?

Karp reduction

When you show that a problem π reduces to π', then you learn that π is no harder than π'.

Therefore, knowing that π' can be solved in polynomial time really tells you something - that π can be solved in polynomial time.

However, if $\pi \in \mathcal{P}$ and $\pi \propto \pi'$, you don't learn anything about π'. It could be that π' is solvable in polynomial time, or maybe not!

Karp reduction

In 1972, Dick Karp proved that every problem in $\mathcal{NP}$ reduces to 3-SAT.

Written differently: $\forall \pi \in \mathcal{NP}, \pi \propto$ 3-SAT

What does this mean?

(See `https://en.wikipedia.org/wiki/Karp%27s_21_NP-complete_problems`)

Karp reduction

Suppose $\forall \pi' \in \mathcal{NP}$, $\pi' \propto \pi$.
Further suppose that $\pi \in \mathcal{P}$.
What can you deduce from this?

Karp reduction

Theorem: Suppose $\forall \pi' \in \mathcal{NP}$, $\pi' \propto \pi$, and $\pi \in \mathcal{P}$. Then $\mathcal{P} = \mathcal{NP}$.

$\mathcal{NP}$-hard

A problem π is said to be $\mathcal{NP}$-hard if a polynomial time algorithm to solve π could be used to solve *every* problem in $\mathcal{NP}$ in polynomial time.
Notes:

- If π satisfies $\forall \pi' \in \mathcal{NP}, \pi' \propto \pi$, then π is $\mathcal{NP}$-hard.
- If any $\mathcal{NP}$-hard problem can be solved in polynomial time, then so can all problems in $\mathcal{NP}$!
- Karp already proved that 3-SAT is $\mathcal{NP}$-hard (as well as 20 other problems). Many other problems have been proven to be $\mathcal{NP}$-hard.
- To prove that a new decision problem π is $\mathcal{NP}$-hard, you just have to find a known $\mathcal{NP}$-hard problem π' and show that $\pi' \propto \pi$.

$\mathcal{NP}$-complete

A problem π is said to be $\mathcal{NP}$-complete if and only if

- $\pi \in \mathcal{NP}$
- π is $\mathcal{NP}$-hard

$\mathcal{NP}$-hard

Use the fact that 3-colorability is $\mathcal{NP}$-hard to prove that 4-colorability is $\mathcal{NP}$-hard.