# Lesson 24- Reductions

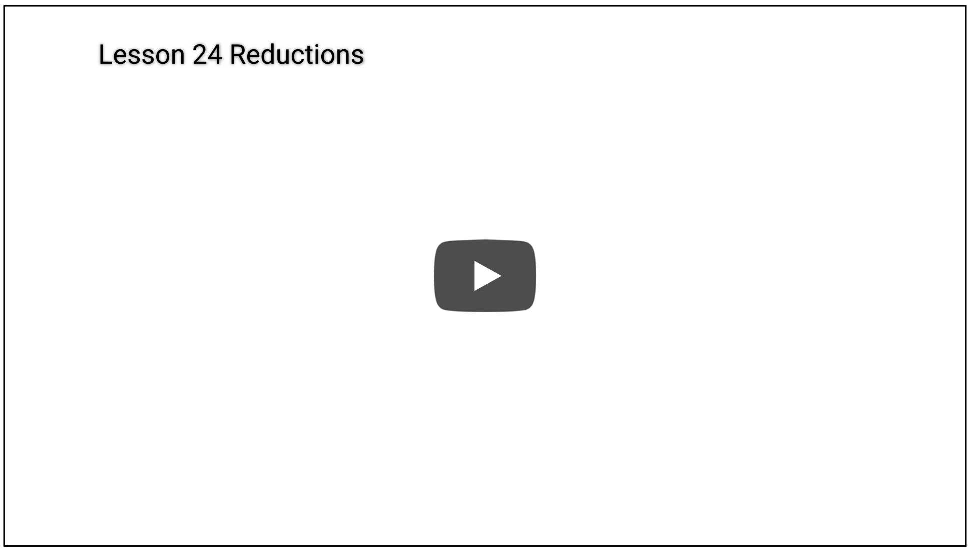
ⓘ This is a preview of the published version of the quiz

Started: Jul 30 at 1:14pm

# Quiz Instructions

- Watch the video below. The quiz has additional videos and readings inside of it.
- Joining the Ohyay is optional, if you want help.
- This assignment is an **individual** submission but you may work with your classmates.
- You will submit this **Canvas Quiz** for this assignment.

# Watch




# Do

Hey students, this is Dr. Bart...

Um, it looks like AlgoTutorBot is getting a little more erratic. I know he was upset that you've been helping me, but this is really getting out of hand. I just looked at the rest of this quiz, and it looks like he chopped off half of it. I guess, whether we like it or not, we're going to be covering less material this semester. If you were planning to take more CS theory classes, I definitely suggest you read the relevant chapter (11) from the textbook! And, uh, be careful about ATB. I think that whatever is happening, things are going to escalate from here.

This "quiz" is actually just an **open-note** assignment. Don't mention it to AlgoTutorBot, but I noticed that it's **not worth any more points than any other assignment** and there's **no time limit besides the lock date**. So please **do not panic**!

When you submit, **the system will not tell you if you got everything right**, so please make sure you are happy with your answers before submitting. You can submit multiple times, up until the lock date, and it will only take your latest submission.

There are readings and videos embedded in the quiz. I strongly recommend you read and watch everything provided. You might also need to google for additional explanations, or seek help from the instructor, TAs, or even a classmate. Definitely good to work together on this one!

I'll contact you soon. It's time we finally struck back at AlgoTutorBot!

Stay safe, everyone, and good luck with this assignment!

## Corrections

So far, there have been no mistakes reported in the quiz. This space will be updated if mistakes are reported.

Often, we can solve a new problem by reusing an existing solution to a different problem. Read over the following page about [Reductions until section 28.2.1.3.](https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/Reduction.html) (https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/Reduction.html)

If you need more context, you should also read over the Wikipedia page on Reductions: https://en.wikipedia.org/wiki/Reduction_(complexity) (https://en.wikipedia.org/wiki/Reduction_(complexity))

And if you want something deeper, this was a relatively good read for this: http://people.cs.pitt.edu/~kirk/cs1510/notes/reducenotes.pdf (http://people.cs.pitt.edu/~kirk/cs1510/notes/reducenotes.pdf)

## Question 1 — 1 pts

Mark all of the following that are true.

- [ ] Any one of Karp's 21 problems can be reduced to any NP-Complete problem
- [ ] If we can reduce an NP-Hard problem in polynomial time to an NP-Complete problem, then we know if it is in NP-Complete.
- [ ] If the reduction algorithm itself costs more than the reduced algorithm, there is no point in doing the reduction.
- [ ] Reduction only works on NP-Complete problems, not P problems.

Satisfiability is a problem that shows up in Logic and AI, but has important connections to theoretical Computer Science.

Watch the following video about Satisfiability: https://www.youtube.com/watch?v=SAXGKCnOuP8 (https://www.youtube.com/watch?v=SAXGKCnOuP8)

(https://www.youtube.com/watch?v=SAXGKCnOuP8)

## Question 2 — 2 pts

| | |
|---|---|
| Find an assignment of logical values to a proposition that produces True. | [ Choose ] |
| An instance of Boolean Satisfiability where the clauses are limited to three literals. | [ Choose ] |
| A disjunction of literals or their negation | [ Choose ] |
| A set of clauses joined by ANDs | [ Choose ] |
| A variable that can take on the value True or False | [ Choose ] |
| A combination of operators and literals | [ Choose ] |
| Operator that performs a logical AND of two terms | [ Choose ] |
| Operator that performs a logical OR of two terms | [ Choose ] |
| The opposite logical value | [ Choose ] |
| True and False | [ Choose ] |

## Question 3

**1 pts**

Given the following 1-SAT, give values that satisfy it.

```
x ^ ~y ^ ~z
```

| | |
|---|---|
| x | [ Choose ] |
| y | [ Choose ] |
| z | [ Choose ] |

## Question 4

**1 pts**

Given the following 3-SAT, mark all the values that satisfy it.

```
(a v ~b v d) ^ (a v c v ~d)
```

- [ ] a=0, b=0, c=0, d=0
- [ ] a=1, b=1, c=1, d=1
- [ ] a=0, b=1, c=0, d=0
- [ ] a=1, b=0, c=1, d=1
- [ ] a=1, b=0, c=1, d=0

The Cook-Levin Theorem is one of the most important theories in Computer Science. Actually understanding the details of how its proof works is very tricky, but its at least valuable to walk away understanding its basics and implications.

**Required:** This video provides a friendly introduction to the Cook Levin Theorem: **https://www.youtube.com/watch?v=W9G_1xG77LE** (https://www.youtube.com/watch?v=W9G_1xG77LE)

(https://www.youtube.com/watch?v=W9G_1xG77LE)

**Optional:** If you want to learn more details, This video series gives another explanation of the way it works:

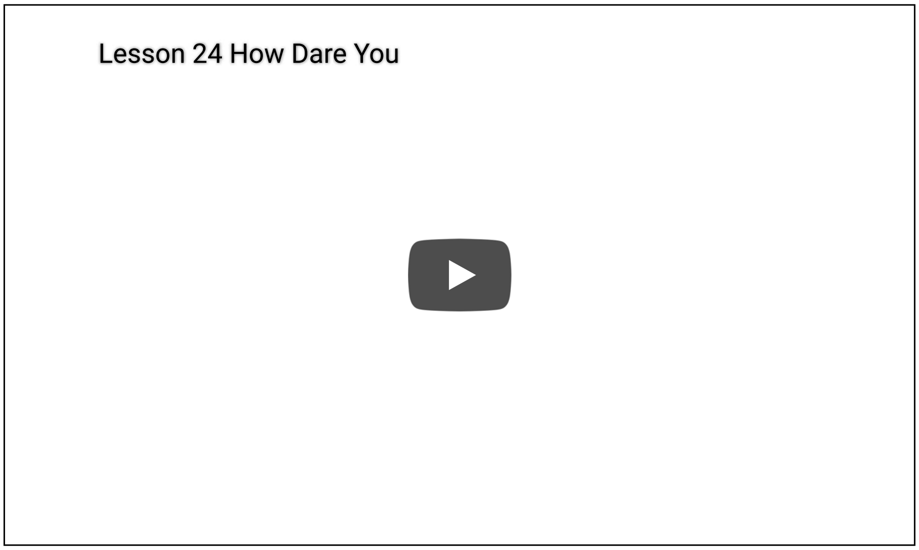
- **Part 1** (https://www.youtube.com/watch?v=nKNd9iExRO8)
- **Part 2** (https://www.youtube.com/watch?v=QwGHBX2k8Xc)
- **Part 3** (https://www.youtube.com/watch?v=sJC5mBFPKcg)

## Question 5

**1 pts**

Which of the following facts are true about the Cook-Levin Theorem?

- [ ] Because we can build a deterministic turing machine to solve any problem in polynomial time, we can solve any problem in NP in P time.
- [ ] The Cook-Levin Theorem proves P=NP
- [ ] The Cook-Levin Theorem relies on building a Turing Machine that solves SAT.
- [ ] The Cook-Levin Theorem states that we can use Reduction to convert any NP problem into 3SAT.




Choose one of the following to watch all the way through, in order to see a Reduction of an NP-Complete problem in action.

- **3SAT to Clique** (https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/threeSAT_to_clique.html)
- **Clique to Independent Set** (https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/clique_to_independentSet.html) (https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/clique_to_independentSet.html)
- **3Sat to Hamiltonian Cycle** (https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/threeSAT_to_hamiltonianCycle.html) (https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/threeSAT_to_hamiltonianCycle.html)
- **Hamiltonian Cycle to Traveling Salesman** (https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/hamiltonianCycle_to_TSP.html) (https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/hamiltonianCycle_to_TSP.html)
- **Independent Set to Vertex Cover** (https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/independentSet_to_vertexCover.html)

*(Despite what AlgoTutorBot says, it's worth seeing how these work in more detail!)*

Saving... Submit Quiz