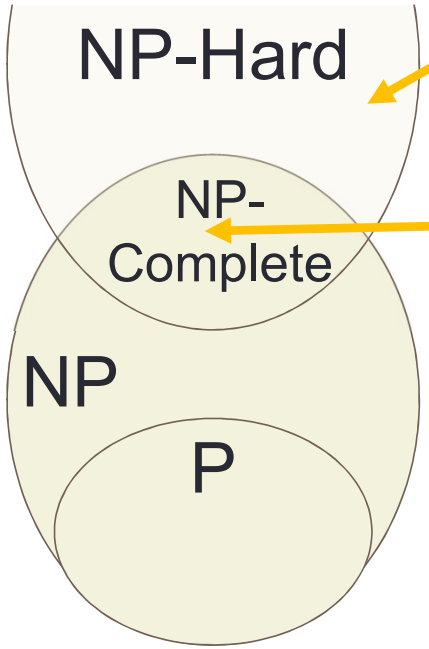
CSE 100: NP-COMPLETENESS AND CSE INTERVIEWS

Announcements

- PA3
  - Final submission deadline 11:59pm on Thursday, December 6 (slip days allowed)
- HW5
  - Due today!
- Final exam: next week
  - 12/11/2018 Tu 3:00p-5:59p WLH 2005

Complexity Theory



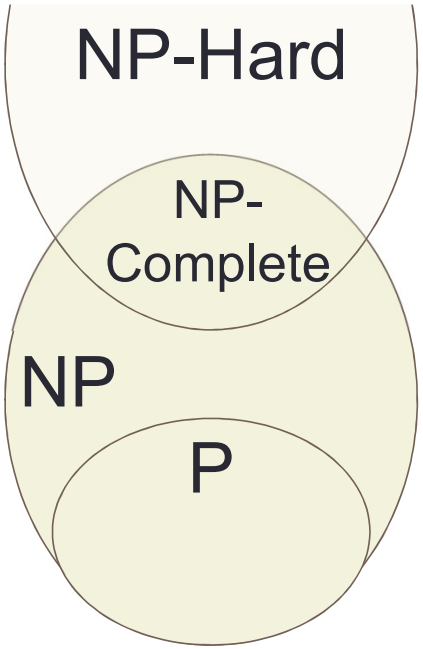
(Hierarchy if P != NP)

NP-Hard: Problems are *at least* as difficult to solve as hardest problems in NP

NP-Complete: No known polynomial time algorithm to find a solution, but can check a solution in polynomial time

A polynomial time solution for *any* NP-Complete problem would solve *all* NP-Complete problems

Complexity Theory

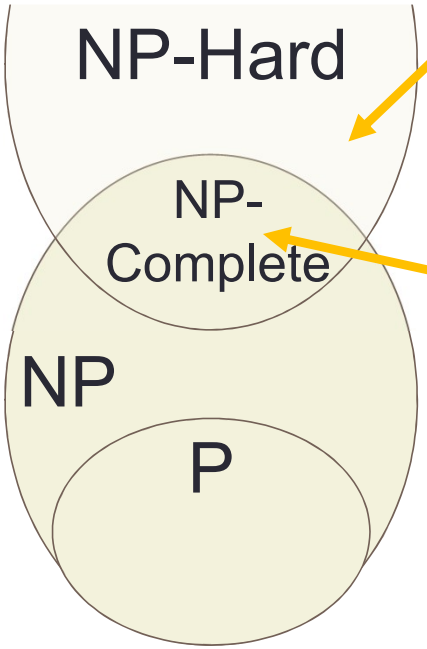


TSP "optimization": given n cities with one Hometown and all pairwise distances, plan a tour starting and ending at Hometown that visits every city exactly once and has minimum distance.

Where does (do you think) the TSP optimization problem fits into this diagram?

A. In P (there is a polynomial time way to find a solution)
B. In NP/NP-Complete (we might not know a polynomial time way to find a solution, but if someone gives us a proposed solution, we can verify whether or not it's correct)
C. NP-Hard (neither of the above is true)
D. I have no idea! I'm so confused!

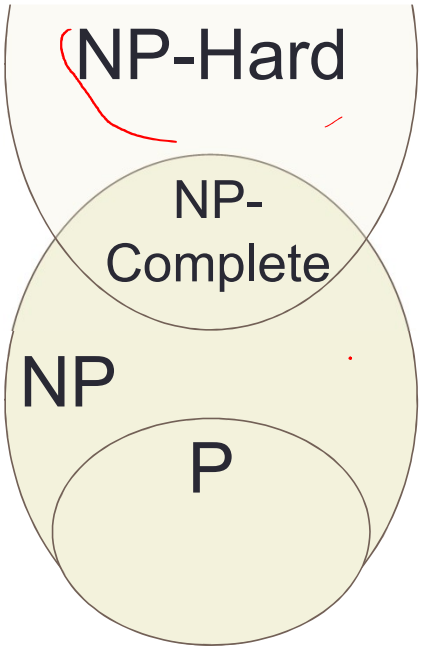
Complexity Theory



TSP "optimization": given n cities with one Hometown and all pairwise distances, plan a tour starting and ending at Hometown that visits every city exactly once and has minimum distance.

TSP "decision": given n cities with one Hometown and all pairwise distances, plan a tour starting and ending at Hometown that visits every city exactly once and has a distance less than L.

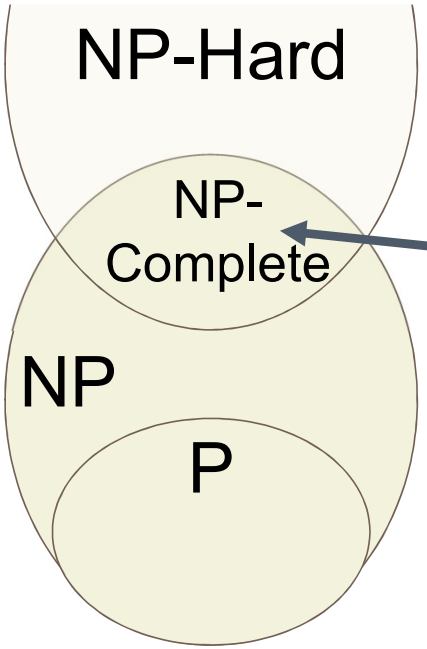
Complexity Theory



Since TSP (both versions) is NP-Hard, solving it in polynomial time may be difficult (if not impossible)

Next time… how to prove a problem is NP-Hard.

Complexity Theory



To prove a problem is NP-Complete you must do two things:

1. **Prove it is in NP**
2. **Prove it is NP-Hard**

Proving a Problem is in NP

To prove a problem is NP-Complete you must do two things:

1. **Prove it is in NP**
2. **Prove it is NP-Hard**

For all problems in NP, there exists an algorithm that can verify whether a solution to the problem is correct or not in polynomial time.

To show a problem is in NP, you must give such an algorithm.

TSP-decision is in NP

To prove a problem is NP-Complete you must do two things:

1. **Prove it is in NP**
2. **Prove it is NP-Hard**

Given a path $v_0 \ldots v_n$, you can verify whether it visits each node and has distance less than L in polynomial time:

dist = 0
For k = 0…n-1:
    If v_k previously marked as visited, fail
    Mark v_k as visited
    dist += weight (v_k, v_{k+1})
If v_0 != v_n fail
If any vertices are not marked as visited, fail
If dist >= L, fail

Else Success

Is TSP-decision NP-Hard?

To prove a problem is NP-Complete you must do two things:

1. **Prove it is in NP**
2. **Prove it is NP-Hard**

To prove a problem is NP-Hard, you must prove it is *at least* as hard as any problem in NP.

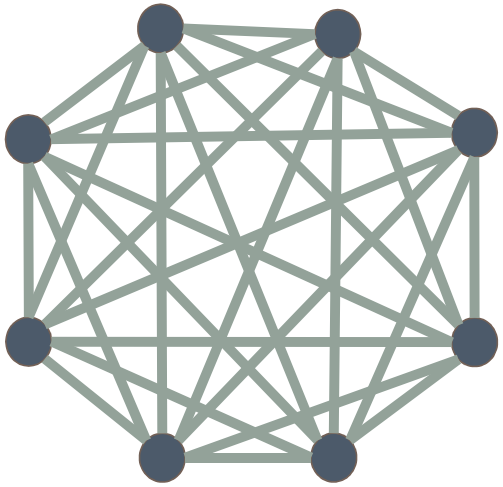
So we should choose a problem that we know is "as hard as any other problem in NP" and show that TSP-decision is at least as hard.

We'll choose a problem called "Hamiltonian Cycle"

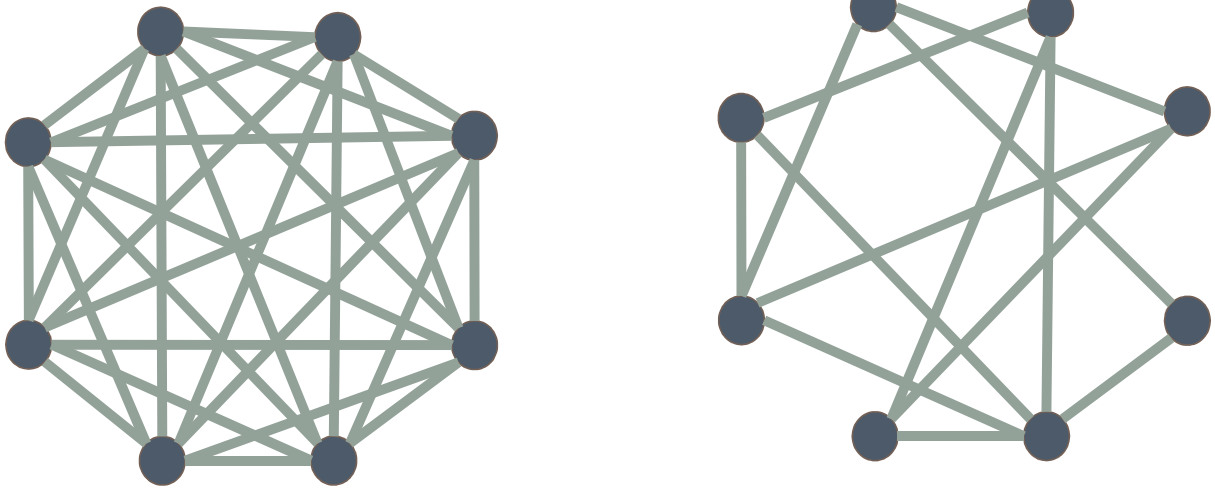
(You have to trust me that Hamiltonian Cycle is "as hard as any other problem in NP". It's somewhat involved to prove it.)

In TSP, given n cities with one Hometown and all pairwise distances, plan a tour starting and ending at Hometown that visits every city exactly once and has minimum distance.

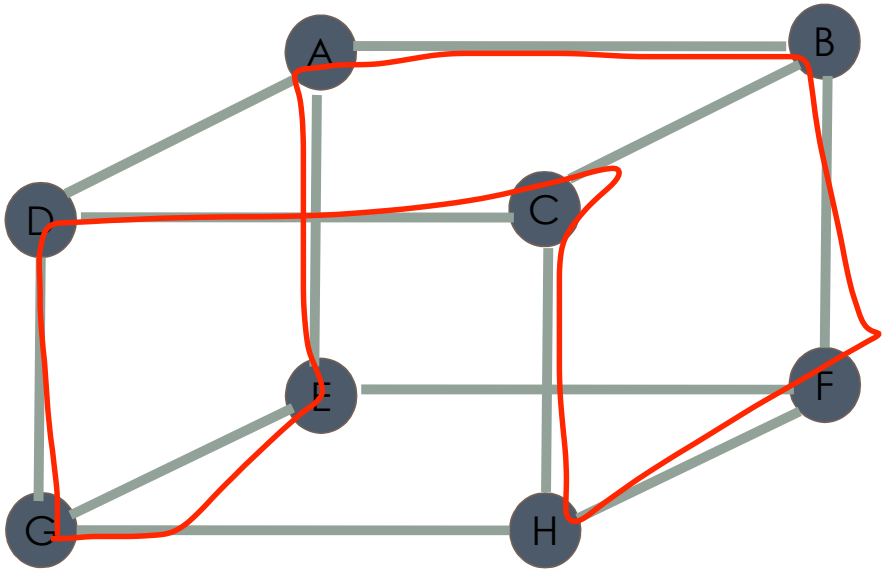
~~In TSP,~~ given n cities with ~~one Hometown and~~ all pairwise distances, plan a tour that visits every city exactly once ~~and has minimum distance~~.



~~In TSP,~~ given n cities ~~with one Hometown and all pairwise distances~~, plan a tour that visits every city exactly once ~~and has minimum distance~~.



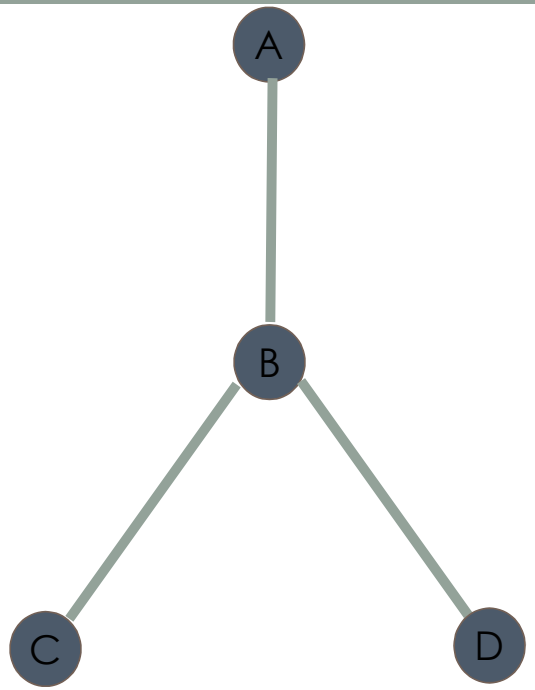
A graph is Hamiltonian if there is a path (or cycle) through the graph which visits each vertex exactly once.



Does this graph have a Hamiltonian cycle?

A. Yes (If yes, what is it?)

B. No



Does this graph have a Hamiltonian cycle?

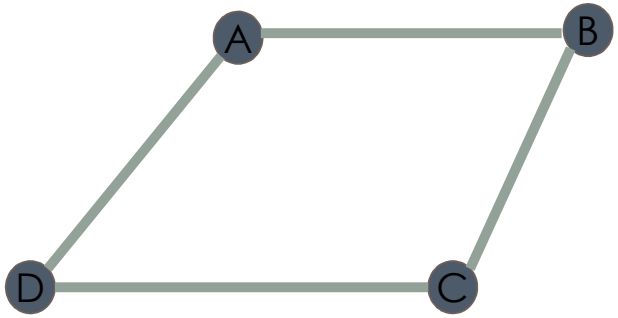
A. Yes (If yes, what is it?)

B. No

Is TSP-decision NP-Hard?

To prove a problem is NP-Complete you must do two things:

1. **Prove it is in NP**
2. **Prove it is NP-Hard**



To prove a problem is NP-Hard, you must prove it is *at least* as hard as any problem in NP.

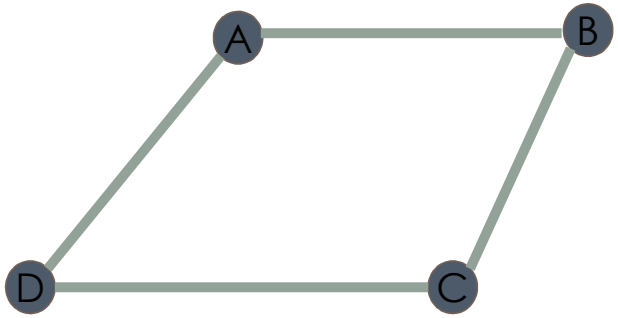
So we should choose a problem that we know is "as hard as any other problem in NP" and show that TSP-decision is at least as hard.

We'll choose a problem called "Hamiltonian Cycle"

(You have to trust me that Hamiltonian Cycle is "as hard as any other problem in NP". It's somewhat involved to prove it.)

Is TSP-decision NP-Hard?

To prove a problem is NP-Complete you must do two things:
1. **Prove it is in NP**
2. **Prove it is NP-Hard**



Goal: Show TSP-decision is at least as hard as Hamiltonian Cycle.

But how?

We will use something called a reduction where we reduce Hamiltonian Cycle to a version of TSP-decision in polynomial time.

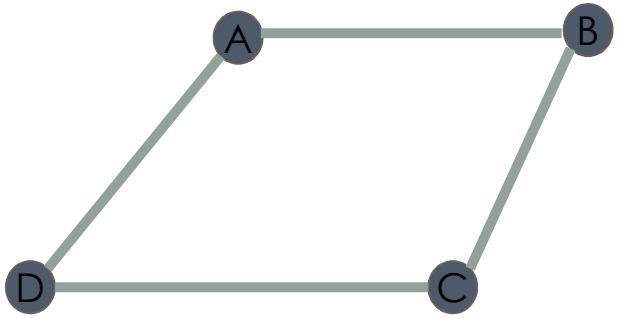
$$\text{Hamiltonian Cycle} \leq_P \text{TSP-decision}$$

"Hamiltonian Cycle is polynomial time reducible to TSP-decision"

“TSP-Decision is at least as hard as Hamiltonian Cycle”
“A solution to TSP-Decision is powerful enough to solve Hamiltonian Cycle in polynomial time.”

Is TSP-decision NP-Hard?

To prove a problem is NP-Complete you must do two things:
1. **Prove it is in NP**
2. **Prove it is NP-Hard**



Goal: Reduce Hamiltonian Cycle TSP-decision in polynomial time

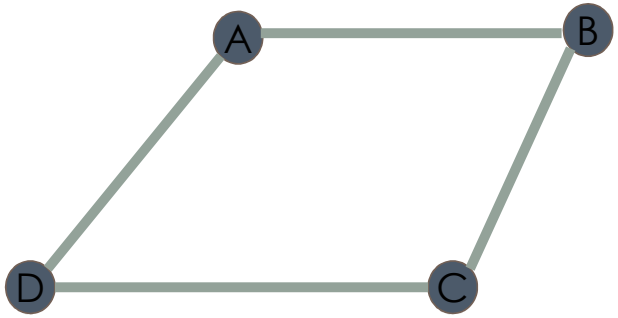
$$\text{Hamiltonian Cycle} \leq_P \text{TSP-decision}$$

Assume you have a black box for solving TSP-decision.
Use that black box to solve Hamiltonian Cycle.

Then if the black box turns out to be polynomial time,
The solution for Hamiltonian Cycle will be polynomial time too!

Is TSP-decision NP-Hard?

To prove a problem is NP-Complete you must do two things:
1. **Prove it is in NP**
2. **Prove it is NP-Hard**



Hamiltonian Cycle $\leq_P$ TSP-decision

Assume you have a black box for solving TSP-decision.
Use that black box to solve Hamiltonian Cycle.

Can you directly pass this graph into TSP-decision?

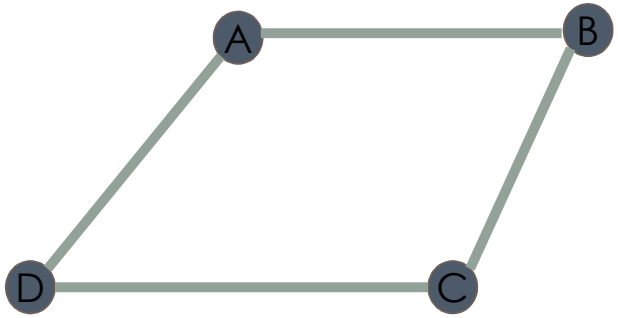
A. Yes
B. No

ask…

Is TSP-decision NP-Hard?

To prove a problem is NP-Complete you must do two things:

1. **Prove it is in NP**
2. **Prove it is NP-Hard**



Goal: Reduce Hamiltonian Cycle TSP-decision in polynomial time

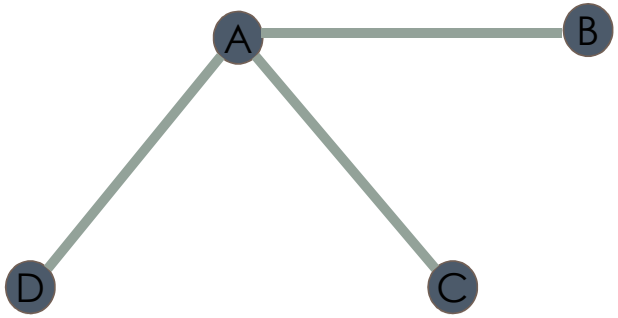
$$\text{Hamiltonian Cycle} \leq_P \text{TSP-decision}$$

Assume we have TSP-decision(Graph, Hometown, L) that returns true if there is a tour starting and ending in Hometown with total distance < L and false if not.

```
HamiltonianCycle(G):
    Let L = |V| + 1
    Construct G', a complete graph with vertices from G
    and with edge weights as follows:
        if edge exists in G, edge weight is 1
        otherwise edge weight is INF
    Choose an arbitrary vertex v in G' as the hometown
    return TSP-decision(G', v, L)
```

Is TSP-decision NP-Hard?

To prove a problem is NP-Complete you must do two things:
1. **Prove it is in NP**
2. **Prove it is NP-Hard**



Goal: Reduce Hamiltonian Cycle TSP-decision in polynomial time

$$\text{Hamiltonian Cycle} \leq_P \text{TSP-decision}$$

Assume we have TSP-decision(Graph, Hometown, L) that returns true if there is a tour starting and ending in Hometown with total distance < L and false if not.

```
HamiltonianCycle(G):
    Let L = |V| + 1
    Construct G', a complete graph with vertices from G
    and with edge weights as follows:
        if edge exists in G, edge weight is 1
        otherwise edge weight is INF
    Choose an arbitrary vertex v in G' as the hometown
    return TSP-decision(G', v, L)
```

Clicker question break

- Reducing problem X to problem Y in polynomial time means:
  A. Assume you have a black box for solving X. Devise a polynomial time algorithm that uses that black box to solve Y.
  B. Assume you have a black box for solving Y. Devise a polynomial time algorithm that uses that black box to solve X.

Clicker question break

- Assume you know that problem C is NP-Complete. You want to show that problem D is NP-Hard. Should you:

A. Assume you have a black box for problem C and use it to solve problem D

B. Assume you have a black box for problem D and use it to solve problem C

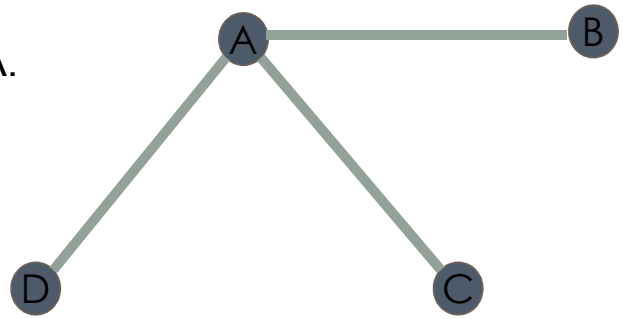
Clicker question break

- If we know problem A is "at least as hard as any problem in NP" then we can show problem B is NP hard by:
  A. Reducing problem A to problem B ($A \leq_P B$)
  B. Reducing problem B to problem A ($B \leq_P A$)

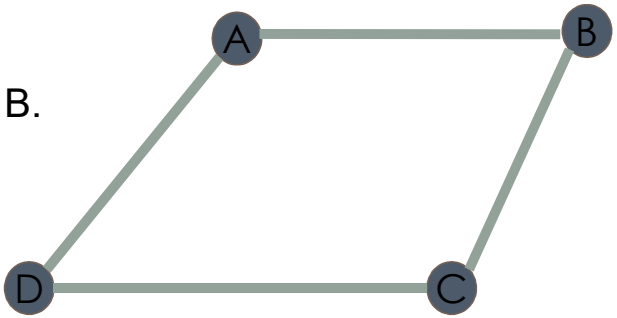
Hamiltonian Path

A Hamiltonian path is a path that visits each vertex exactly once, but doesn't return to the starting vertex
Which of the following graphs have Hamiltonian Paths?

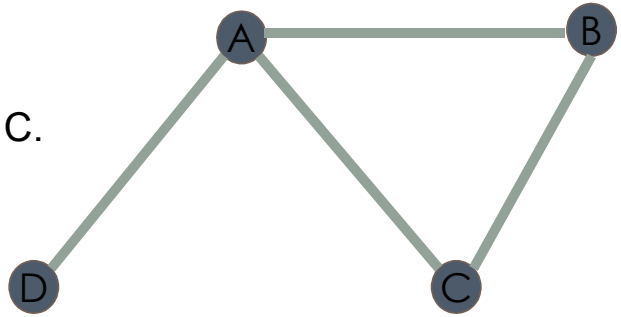
A.



B.



C.

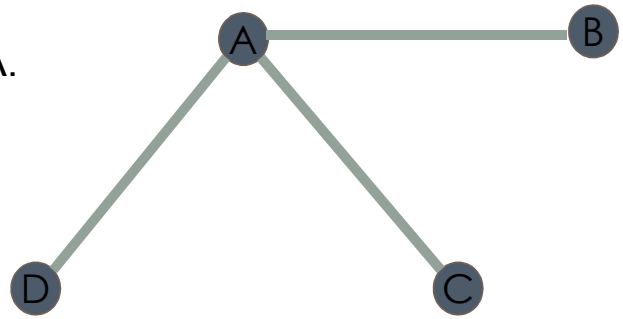


D. B&C
E. All of them

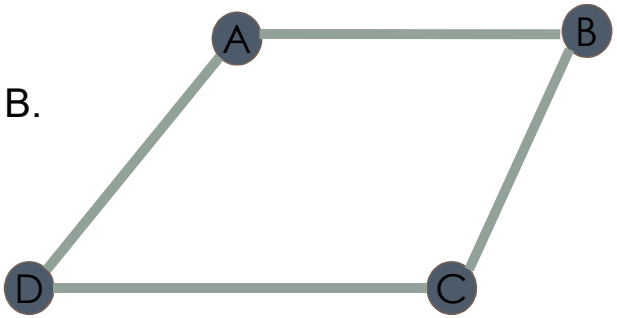
Hamiltonian Path vs. Hamiltonian Cycle

Which of the following graphs have Hamiltonian Cycles?

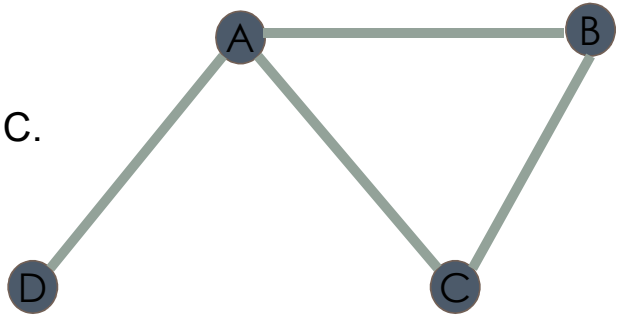
A.



B.



C.



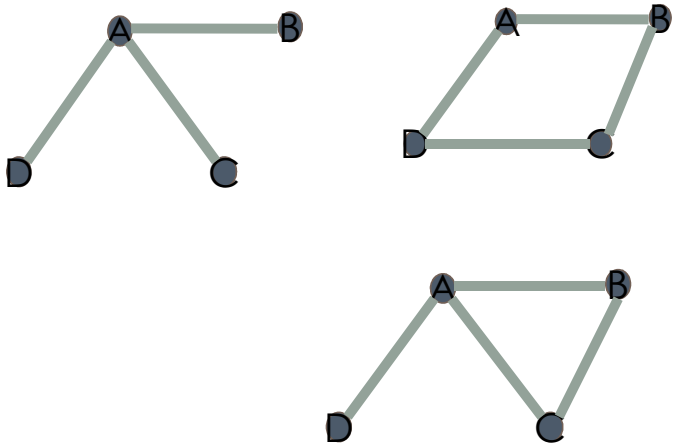
D. B&C

E. All of them

Prove Hamiltonian Path is NP-Complete

A Hamiltonian path is a path that visits each vertex exactly once, but doesn't return to the starting vertex.

Your goal, convert an instance of a Hamiltonian path problem to an instance of a Hamiltonian Cycle (in poly-time)



To prove a problem is NP-Complete you must do two things:

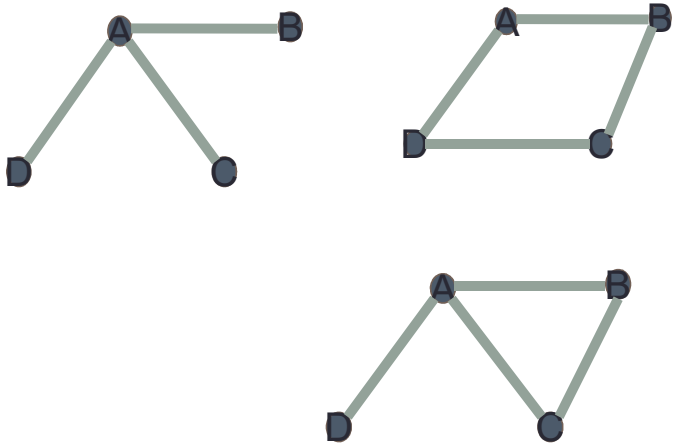
1. **Prove it is in NP**
2. **Prove it is NP-Hard**

Prove Hamiltonian Path is NP-Complete

Hamiltonian Cycle $\leq_P$ Hamiltonian Path

"Hamiltonian Cycle reduces to Hamiltonian Path"

```
HamiltonianCycle(G):
    Construct G', with vertices and edges from G
    Arbitrarily choose a vertex v and make a copy v' such that
    v' has all the same edges to all the same vertices as v.
    Then add two new vertices to G', w and w' such that there
    is a single edge between w and v and a single edge
    between w' and v'.
    return HamiltonianPath(G').
```



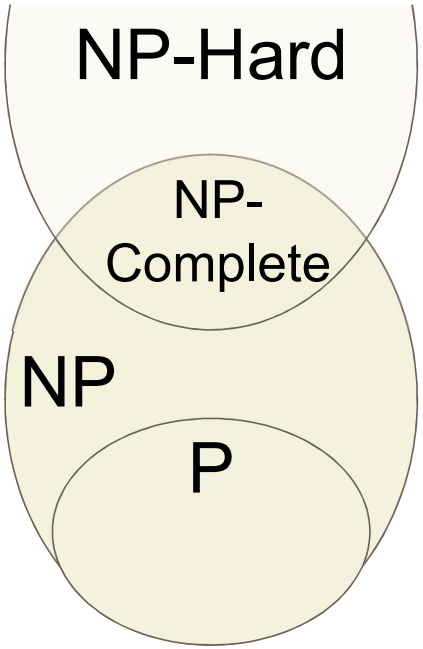
If G has a Ham cycle, then G' has a Ham Path that starts in w and ends in w'.

If G did not have a Ham cycle, then G' cannot have a Ham path. Any Ham path in G' must start in w and end in w' (or vice versa). This means it must get from v to v' via a Hamiltonian path, which is only possible if there was a Hamiltonian cycle in the original graph.

To prove a problem is NP-Complete you must do two things:

1. **Prove it is in NP**
2. **Prove it is NP-Hard**

Complexity Theory



How you might use this?

You are given a new problem.

You want to know if it's hard to solve…

Is it similar to known NP-complete problems?

Satisfiability problem (SAT)

Given a set of boolean values $x_1, \ldots, x_n$, does there exist an interpretation which satisfies a Boolean formula N?

Example formula:

$(x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_4)$

do we need to know this…?

Satisfiability problem (SAT)

Example formula:

$(x_1 \vee x_2) \wedge$
$(\neg x_1 \vee \neg x_2) \wedge$
$(x_1 \vee \neg x_2) \wedge$
$(\neg x_1 \vee x_2 \vee x_3) \wedge$
$(\neg x_2 \vee \neg x_3)$

Is there a solution to the equation above?

A. Yes
B. No

Satisfiability problem (SAT) -> 3SAT

Notice each clause has 3 literals?

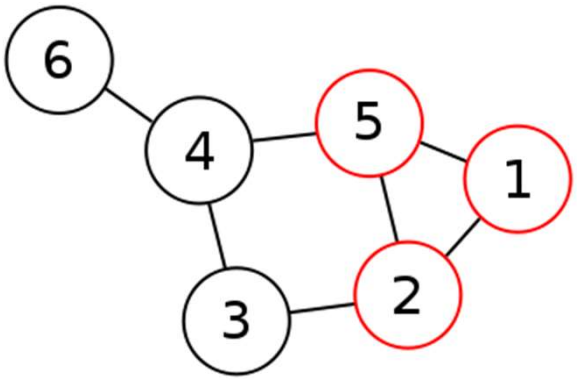
$(x_1 \vee x_2) \wedge$
$(\neg x_1 \vee \neg x_2) \wedge$
$(x_1 \vee \neg x_2) \wedge$
$(\neg x_1 \vee x_2 \vee x_3) \wedge$
$(\neg x_2 \vee \neg x_3)$

Any more than 3 literal problem can be converted to 3 literal.

Clique

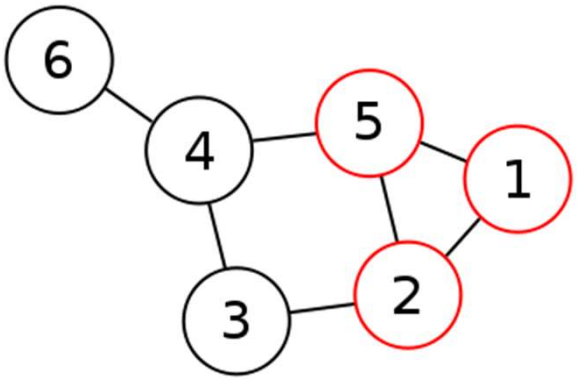
A clique in a graph G is a complete subgraph of G. It is a subset K of the vertices such that every two vertices in K are connected by an edge in G.

? ? ?



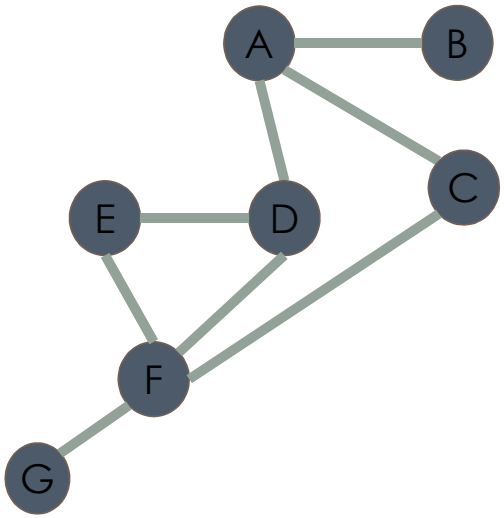
Vertex Cover

Given a graph G, what is the smallest set of vertices such that each edge in the graph is adjacent to at least one vertex in the set.



Vertex Cover

Given a graph G, what is the smallest set of vertices such that each edge in the graph is adjacent to at least one vertex in the set.



In this example above, what is the smallest vertex cover?

Set Cover

Given a set of elements {1,2,...,n} (the universe) and a collection S of m sets whose union equals the universe, the set cover problem is to identify the smallest sub-collection of S whose union equals the universe.

For example, consider the universe U={1,2,3,4,5,6} and the collection of sets

S={ {1,2,3}, {2,3,4,5}, {4,5,6}, {3,4,5} }

What is the smallest sub-collection whose union is the universe?

Knapsack

Given a set of items, each with a value and weight, determine the elements to add to the collection such that the weight is below a limit w and the value is as large as possible.

A: 1 lbs, $2
B: 12 lbs, $14
C: 4 lbs, $9
D: 2 lbs, $1
E: 1 lbs, $1
F: 3 lbs, $4

Assuming you have as many of each item in set A-F as you desire, what is the best solution to keep the weight at <= 19 lbs?

Subset Sum

Given a set of integers, is there a non-empty subset of those integers whose sum is 0?

-20, -8, -4, -2, 5, 9

Is there a solution to this problem above?

A. Yes
B. No

Sudoku

Given an n x n matrix, is there a solution?

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

3x3 puzzle

If you can reduce to 3-SAT…

There are powerful known 3SAT solvers. If they succeed, they can solve “hard” problems quickly. But if they can’t, runtimes explode.

Many, many more problems

https://en.wikipedia.org/wiki/List_of_NP-complete_problems

CSE 100: HOW TO ACE THE TECHNICAL INTERVIEW

Mastering the Technical Interview!

- We talked to Google recruiters, and what do you think they identified as the most common issue for their academically well-prepared undergraduate interviewees?

A. They don't know enough data structures
B. They don't know the running times of the data structures
C. They don't know C++ well enough
D. They can't solve algorithmic problems
E. They can't communicate well

OK, so now how do you master the interview?

First, addressing imposter syndrome

- Watch the following videos on the imposter syndrome and stereotype threat (If you only want to watch one, make it the last one).
  - https://www.coursera.org/learn/cs-tech-interview/lecture/zTZEg/imposter-syndrome-and-stereotype-threat
  - https://www.coursera.org/learn/cs-tech-interview/lecture/3ocxM/growth-mindset
  - https://www.coursera.org/learn/cs-tech-interview/lecture/T8dy6/imposter-syndrome-gallery

Technical interviews: Introductions

https://www.coursera.org/learn/cs-tech-interview/lecture/ntOUJ/good-and-bad-example-introductions

Skip to 0:44, explain Christine is acting like a CS student who helped us develop this MOOC

1. 0:44-1:02 – way too excited. Doesn't convey why she's good for the job
2. 1:29 – 1:59 – way too confident. I wouldn't say I'm an expert in 10 languages and I've been programming since 1996. Maybe solid in 2 or 3, but expert? This just seems like cognitive dissidents.
3. 2:38 – 3:12 – much better: who she is, concrete example, strong delivery/soft skills, left openings for lots of questions.

More in the video about good delivery!

Introductions!

Pair up. One is the interviewer, the other is the candidate. The interviewer will ask: "Tell me a little about yourself" Then the candidate will have 60 seconds to give a response.

Then switch and repeat

Introductions feedback

Now, give your partner feedback on the following:

- Did the candidate clearly explain their experience and background?
- Did the candidate successfully "brand" themselves and convey the unique qualities and experiences that they bring?
- Did the candidate make good eye contact?
- Did the candidate have any distractive nervous tics?
- Did the candidate speak clearly and confidently?

Technical Interview Practice!

- Try answering the question on the following slide.
- Step 1: take about 5 minutes on your own.
- Step 2: pair up and come up with a joint solution

Technical Interview Practice

- Write a data structure with the following public functions (and of course, efficiency matters):

1. Insert
2. Remove
3. Get Random (returns an element from the data structure – all elements should be equally likely)

There's another example next, if you have the time. Otherwise skip. No need to go over an answer, the next videos will get you there. A good answer is a combination of a hash_map for fast insert_remove and an ArrayList for fast getRandom. The hash_map can just store an index into the ArrayList

Technical Interview Practice (extra)!

- Given a binary tree, design an algorithm that creates a linked list of all the nodes at each depth (for example, if you have a tree at depth D, you'll have D linked lists.)

From Cracking the Coding Interview by McDowell

Technical Interview Practice (extra)!

- You need to manage events manually. Events are objects that have a tuple: {address, time}. Address is a unique id for the object and time is when the event needs to occur. But, your program also needs to know if an event is already issued, so you need to be able to do a find.
- Operations:
  - Insert ({address, time}, object*)
  - Find (address) (returns object* if present, nullptr if not)
  - Top() – return the time the next event should occur
  - Pop() – return the object that should occur next

Technical Interview Evaluation

- Good mock-interview

https://www.coursera.org/learn/cs-tech-interview/lecture/0VItH/highlights-from-a-good-mock-interview

Skip to 0:30
1. 0:30 Asks Clarifying questions
2. 2:32 – walks through a quick example
3. 3:28 – comes up with solution, then
   4:29 analyzes complexity
4. Iterate starting at 4:20
5. Better combo solution at 6:30 – can stop at the coding part 7:46

Technical Interview Evaluation

- Near-miss mock-interview

https://www.coursera.org/learn/cs-tech-interview/lecture/GW7yu/pitfalls-in-a-bad-mock-interview

Start at beginning.

1. No clarifying question. Interviewer tries to help him look at high level, but he wants to code too much too early
2. Hints to at least figure out method signatures first
3. Tries to get him to think through an example, hint missed
4. Has moved to hash maps, but is a bit unconfident. Needs help to see they can't do it.
5. The hint on expected hash map behavior is too much – they're saving the interviewee but the interview is probably lost by now.

Evaluation (in person)

- Did the candidate ask (good) clarifying questions?
- Did the candidate work through an example before starting?
- Did the candidate do a high-level performance analysis before coding?
- Did the candidate talk continuously when writing code?
- Did the candidate finish their coding task by tracing through an example to verify that their solution was correct?
- Did the candidate find and correct errors in their code?

Technical Interviews: The Phone Interview

- Watch these videos (they are in sequence):
  - https://www.coursera.org/learn/cs-tech-interview/lecture/btS4L/sample-demonstrating-key-pitfalls
  - https://www.coursera.org/learn/cs-tech-interview/lecture/MVGCw/talking-through-processes
  - https://www.coursera.org/learn/cs-tech-interview/lecture/HiReT/getting-started
  - https://www.coursera.org/learn/cs-tech-interview/lecture/yRDrf/getting-stuck-and-recovering-from-mistakes
  - https://www.coursera.org/learn/cs-tech-interview/lecture/Nr5qq/correctness-and-testing
  - https://www.coursera.org/learn/cs-tech-interview/lecture/RWKlN/a-very-good-phone-interview

There's more! Check out our full course

- https://www.coursera.org/learn/cs-tech-interview
  - Overview of the process
  - Introducing yourself
  - Phone Interviews
  - Presenting work you've done in the past
  - Live-coding interview questions
- + example problems and resources

For more practice: Phone interview question

```
class TSTNode {
public:
  TSTNode* left;
  TSTNode* right;
  TSTNode* middle;
  char letter;
  bool isWord;
  TSTNode(char letter);
};
```

Assume you have a C++ class that represents a node in a TST as shown here:

Write a method

```
bool insert(string key, TSTNode* root)
```

that inserts **`key`** into the TST rooted at **`root`** and returns **`true`** if the key was successfully inserted, and **`false`** otherwise.

Evaluation (in person/phone)

- Did the candidate ask (good) clarifying questions?
- Did the candidate work through an example before starting?
- Did the candidate do a high-level performance analysis before coding?
- Did the candidate talk continuously when writing code?
- Did the candidate finish their coding task by tracing through an example to verify that their solution was correct?
- Did the candidate find and correct errors in their code?