CSCI 335
Software Design and Analysis III

(NP-complete problems)

Easy vs. Hard

- What does it mean for a problem to have an easy solution? What about a hard solution?
  - Can a clever algorithm find the solution directly?
  - Do we have to brute-force search for a solution in a large space of possible solutions?
- Some problems have solutions but are impossible to solve, we call these undecidable problems.

Interesting Hard Problems

- Scheduling
- Map coloring
- Bin packing
- Traveling salesman
- Puzzle solving
- Protein folding
- Theorem proving

The Halting Problem

- Assume we have a program HALT(P, X) that returns true if program P halts on input X and false if P goes into an infinite loop on input X.
- Consider the program LOOPER(P)
  - LOOPER(P) halts if HALT(P, P) is false.
  - LOOPER(P) infinitely loops if HALT(P, P) is true.
- What happens when we run LOOPER(LOOPER)?

The Halting Problem

- LOOPER(LOOPER) halts iff LOOPER(LOOPER) does not halt. This is a logical contradiction.
- This means that our assumption that the program HALT(P, X) can exists was false.
- The Halting Problem is undecidable.

“Easy problems”

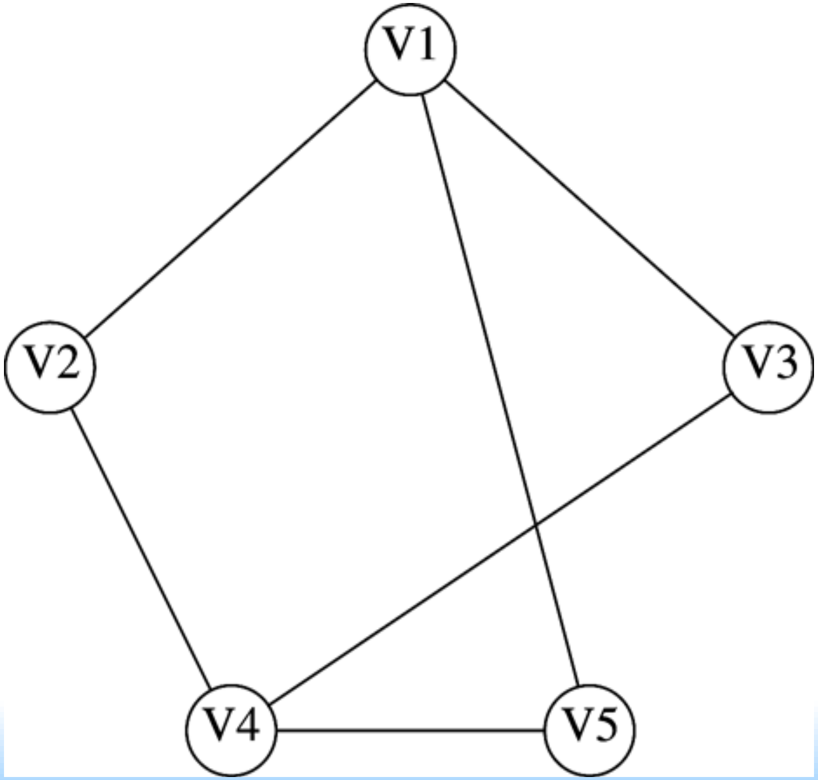
- Most of the problems we solved.
- Polynomial time complexity (but degree may be high).

The Class NP

- Nondeterministic polynomial-time
- Nondeterministic machines: a useful theoretical construct => The machine guesses the "correct" instruction to follow
- **Example problem =>**
  - **Hamiltonian cycle:** is there a simple cycle in a graph that will visit all vertices?
- A nondeterministic program will start from a vertex, and will guess the correct next vertex on the simple cycle if such cycle exists =>

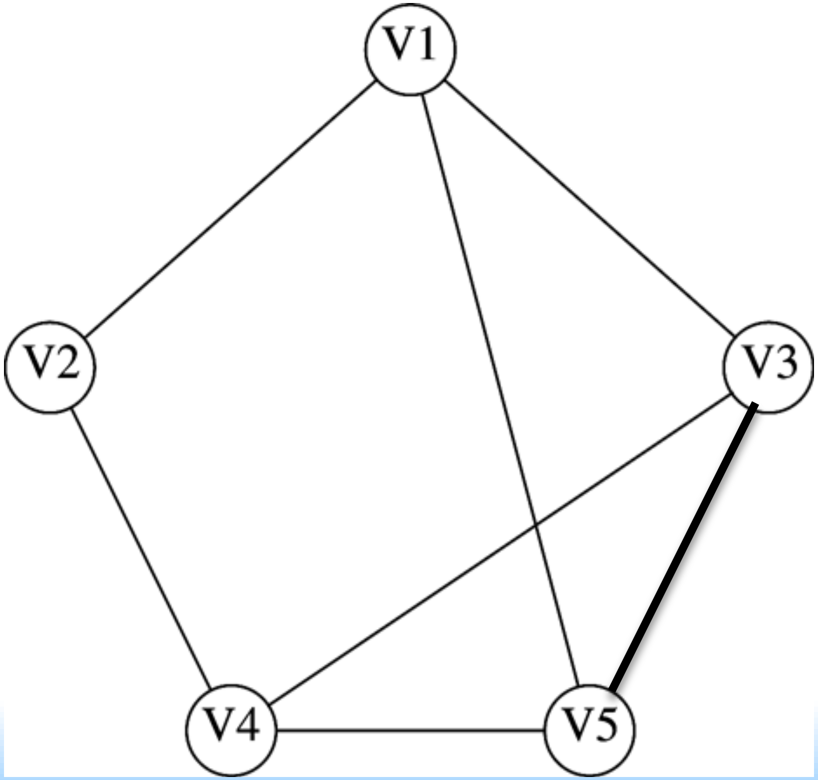
In polynomial time it will provide the result (YES answer).

Hamiltonian Cycle Example



Does this graph have a Hamiltonian cycle?

Hamiltonian Cycle Example



Does this graph have a Hamiltonian cycle?

The class NP

- If answer is known it can be <u>verified</u> in polynomial time
  - Hamiltonian cycle problem is in NP (why?)
  - Given a graph, is there a simple path of length > K (in NP?)
  - Given a graph G, say yes if G DOES NOT have a Hamiltonian cycle (in NP?)

The class P

- Deterministic polynomial time.
- P is a subset of NP.
  - That means that every problem in P is also in NP.
- Which of the problems we solved are in P?
- Is Hamiltonian cycle in P?

Reduction

- Suppose we have two problems P1 and P2
- *P1 can be polynomially* ***reduced*** to P2 as follows:
  - Provide a mapping so that an instance of P1 can be mapped to an instance of P2 (in **polynomial** time).
  - Solve P2.
  - Map the solutions back to P1 (in **polynomial** time).

  ➔ So if you know how to solve P2, you can solve P1 with an additional polynomial cost.

➔ Simple example P1: Calculator with decimal input

P2: Calculator with binary input

NP-complete problems

- A problem **p is NP-complete if**

  ALL other NP problems are polynomially reducible to it.
- If a polynomial solution is found for **p then the whole class NP** becomes polynomial (in deterministic machines).
- That result still eludes us..

NP-complete problems

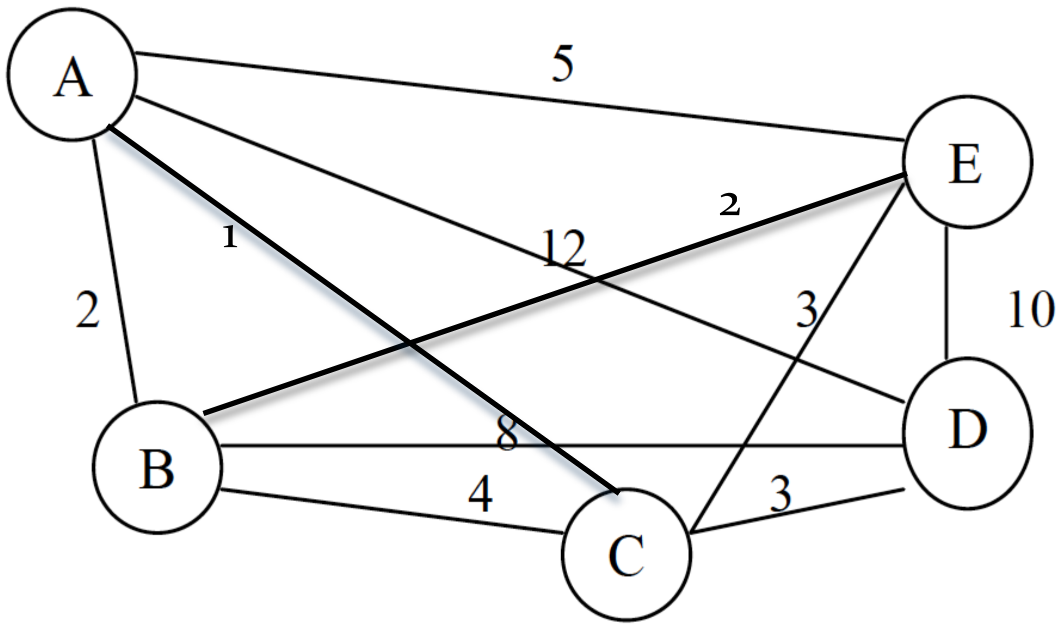
- The Hamiltonian cycle is an **NP-complete** problem

  (^^we will not prove the above)
- The **<u>Traveling Salesman Problem (TSP)</u>**:

"Given a complete graph G=(V,E) with edge costs and an integer K, is there a simple cycle that visits all vertices and has total cost <= K ?"

  - Extremely important problem.
- *We will prove that TSP is NP-complete.*

TSP Example



Is there a simple cycle that visits all vertices and has total cost <= K ?

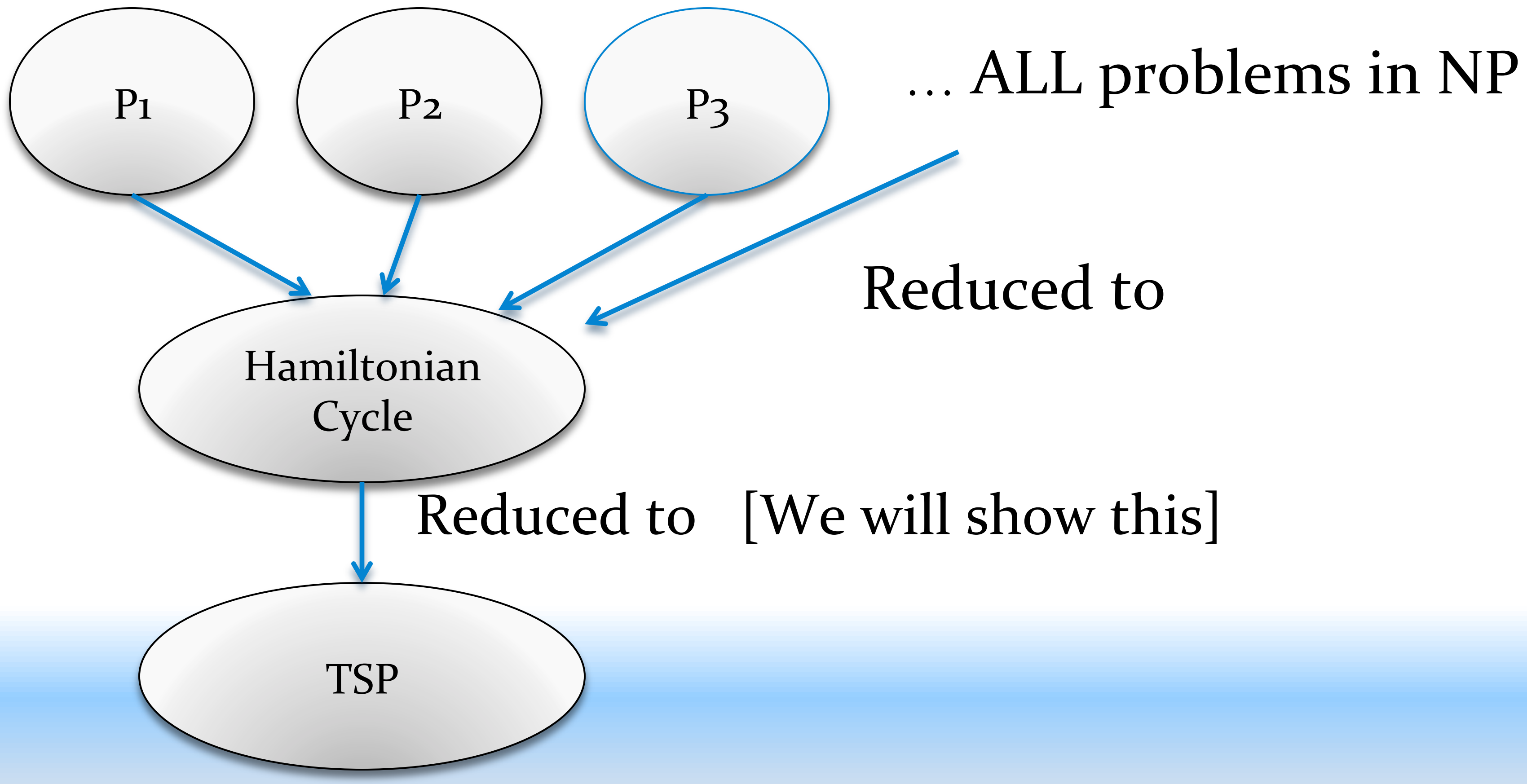
For example: K = 3 or K = 14…

NP-complete problems

- TSP is **NP-complete**
- **Proof outline:**

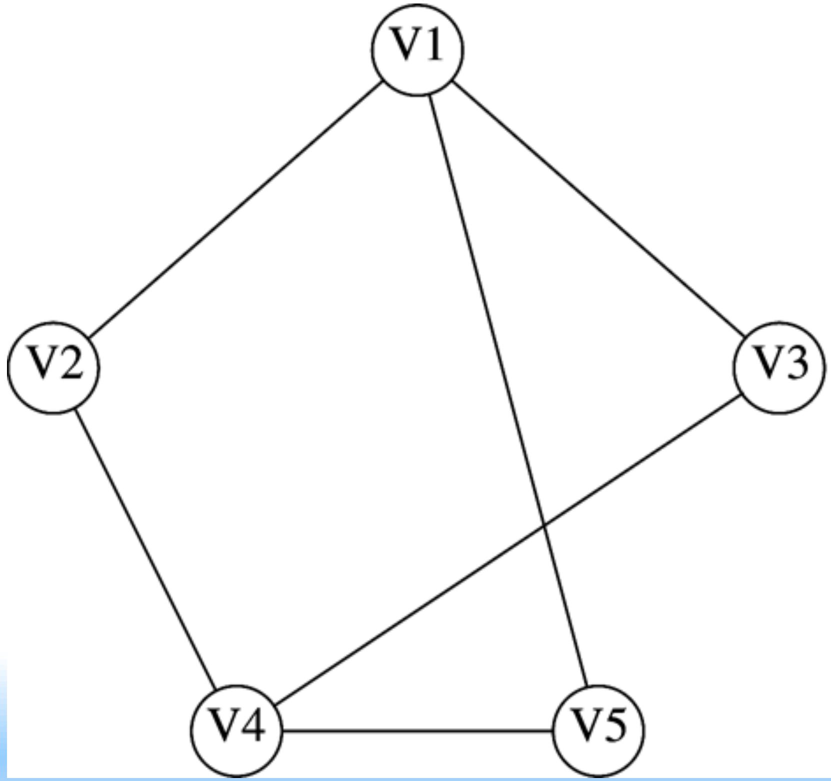
1) The Hamiltonian cycle can be polynomially reduced to TSP
2) Every NP problem can be reduced to the Hamiltonian cycle (since Hamiltonian cycle is NP-complete)
3) Therefore every NP problem can be reduced to TSP

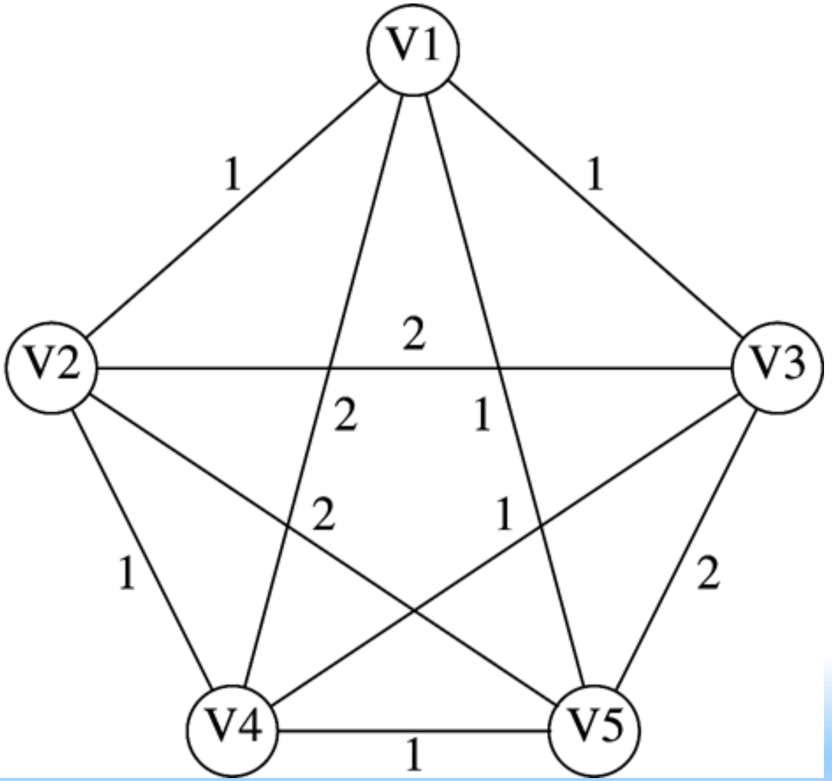
Proof diagram



Hamiltonian Cycle reduced to TSP

Reduction: INPUT for Hamiltonian Cycle => INPUT for TSP (K = |V|)



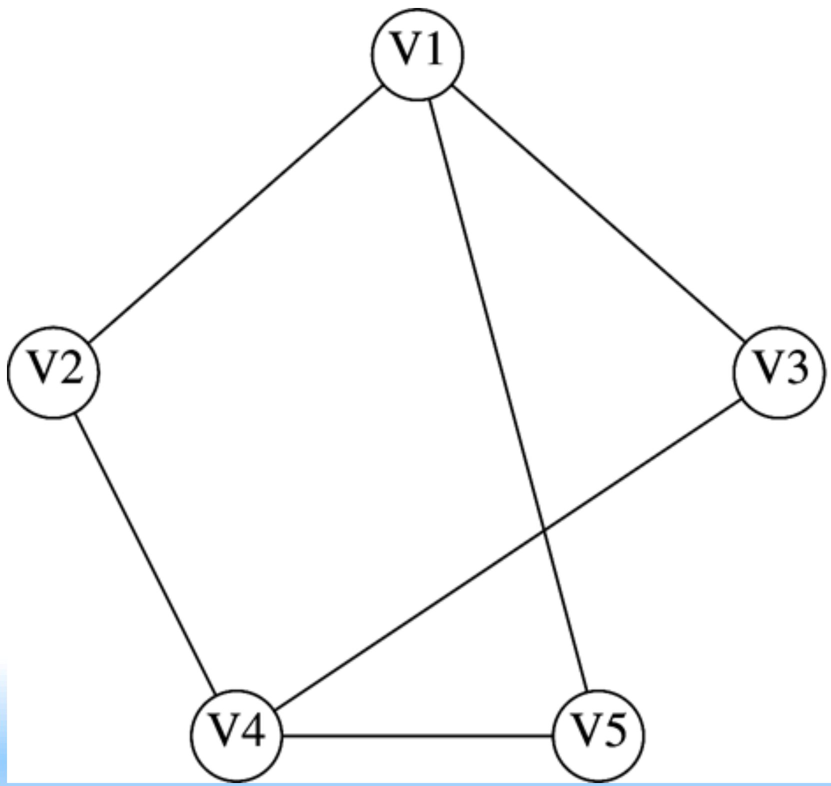


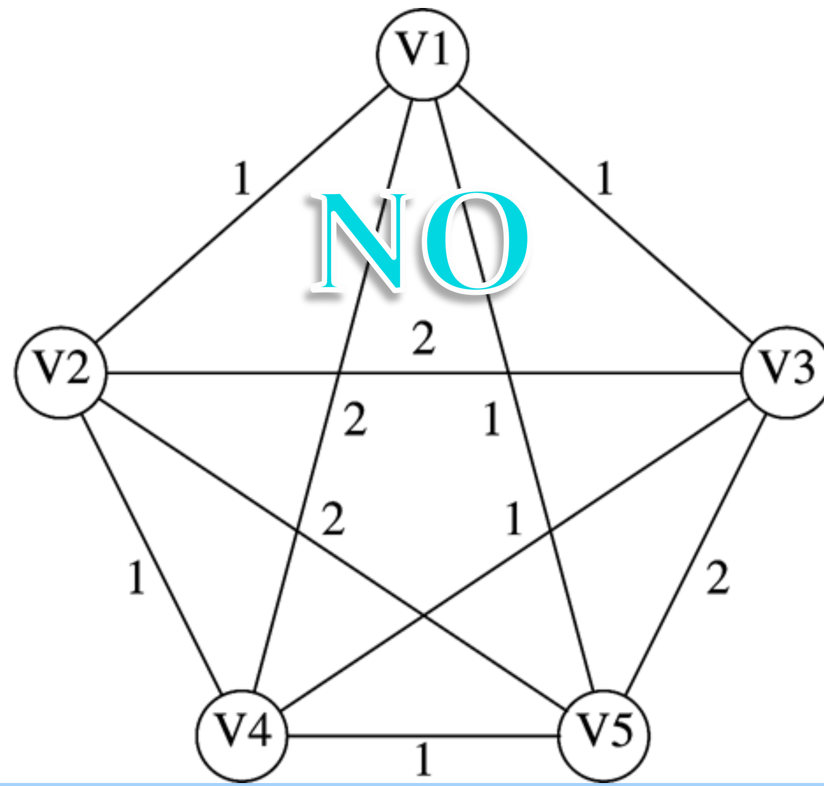
Is there a simple cycle that visit all vertices?

Is there a simple cycle that visits all vertices and has total cost <= K

Hamiltonian Cycle reduced to TSP

Reduction: INPUT for Hamiltonian Cycle => INPUT for TSP (K = |V|)



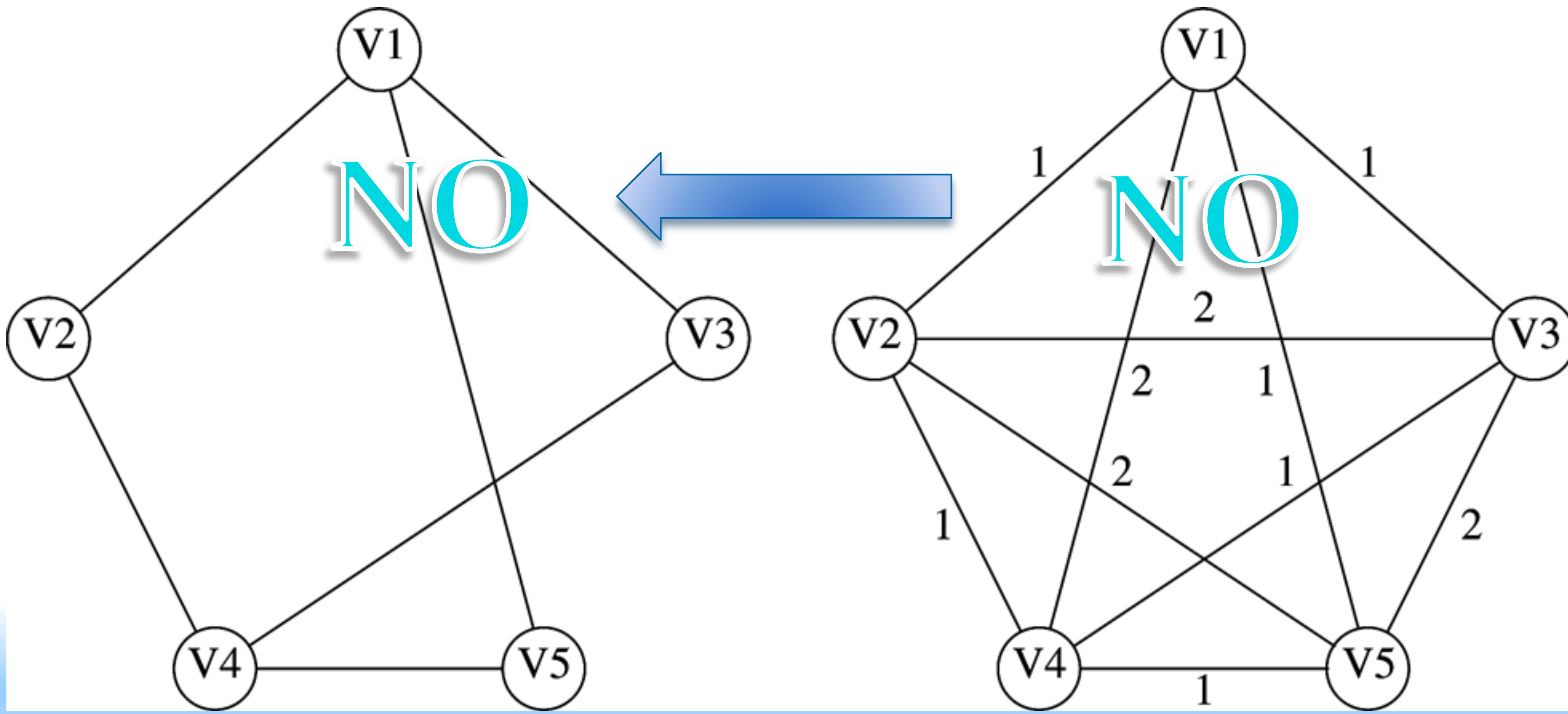


Is there a simple cycle that visit all vertices?

Is there a simple cycle that visits all vertices and has total cost <= K

Hamiltonian Cycle reduced to TSP

Reduction: INPUT for Hamiltonian Cycle => INPUT for TSP (K = |V|)

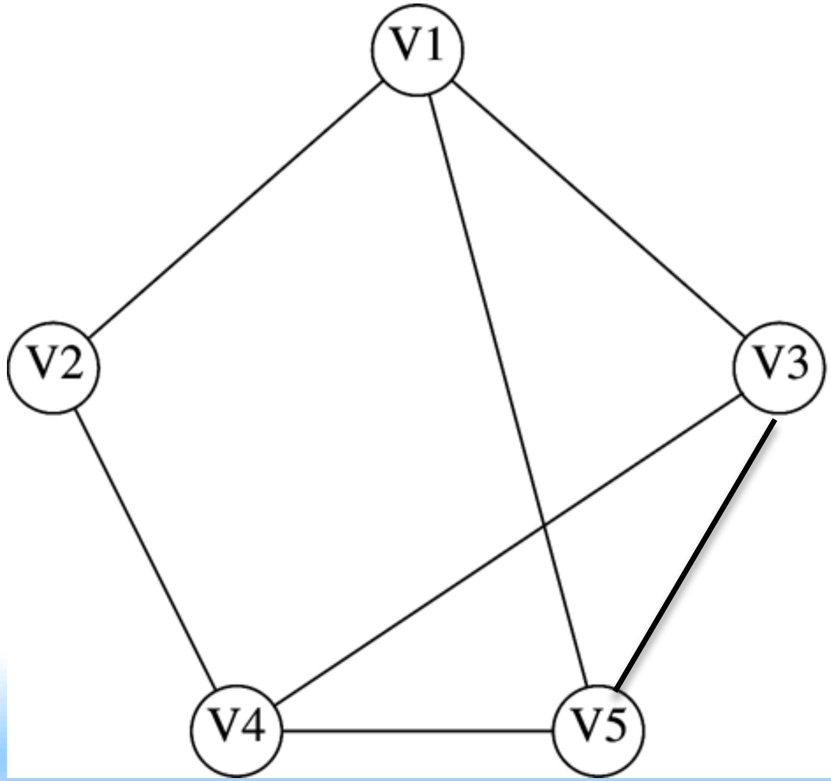


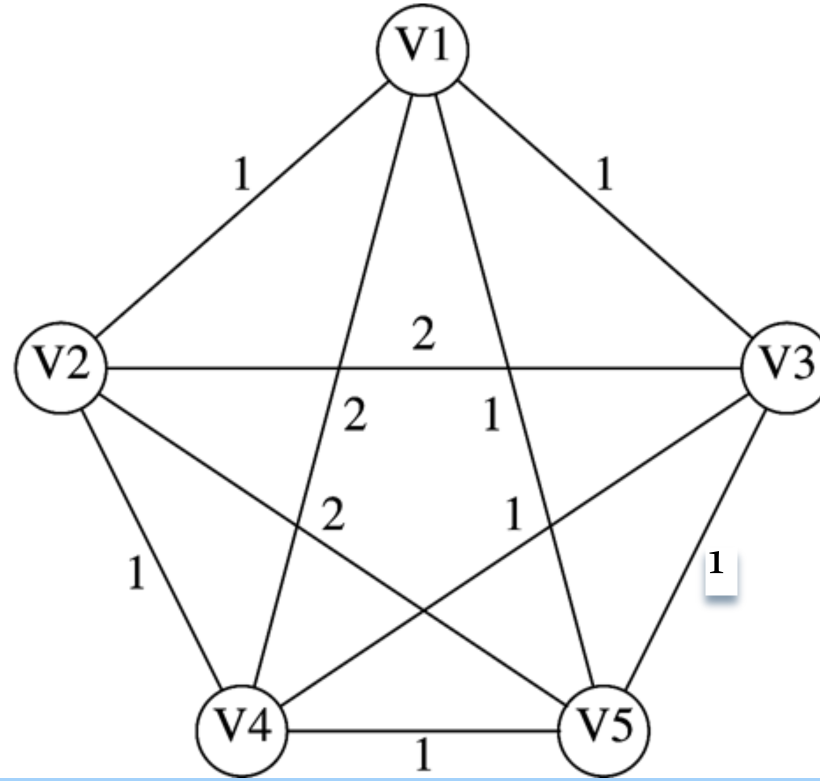
Is there a simple cycle that visit all vertices?

Is there a simple cycle that visits all vertices and has total cost <= K

Hamiltonian Cycle reduced to TSP

Reduction: INPUT for Hamiltonian Cycle => INPUT for TSP (K = |V|)



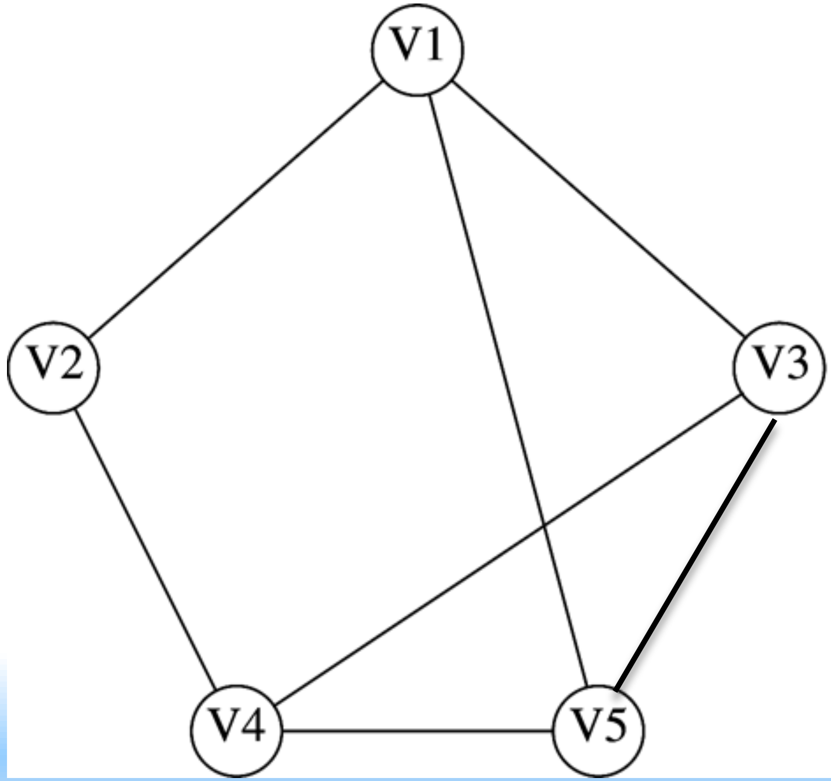


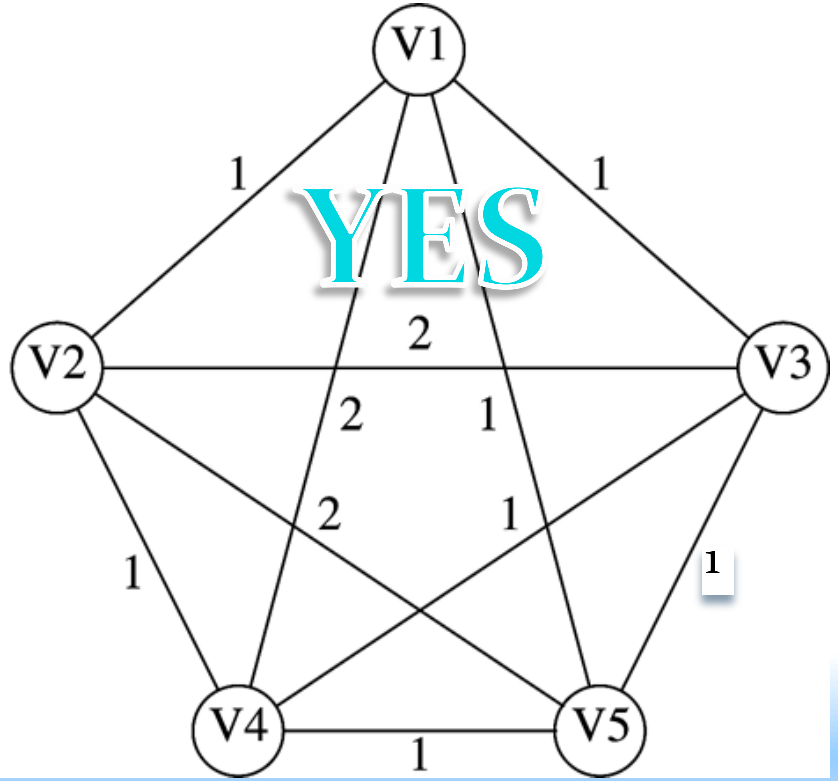
Is there a simple cycle that visit all vertices?

Is there a simple cycle that visits all vertices and has total cost <= K

Hamiltonian Cycle reduced to TSP

Reduction: INPUT for Hamiltonian Cycle => INPUT for TSP (K = |V|)



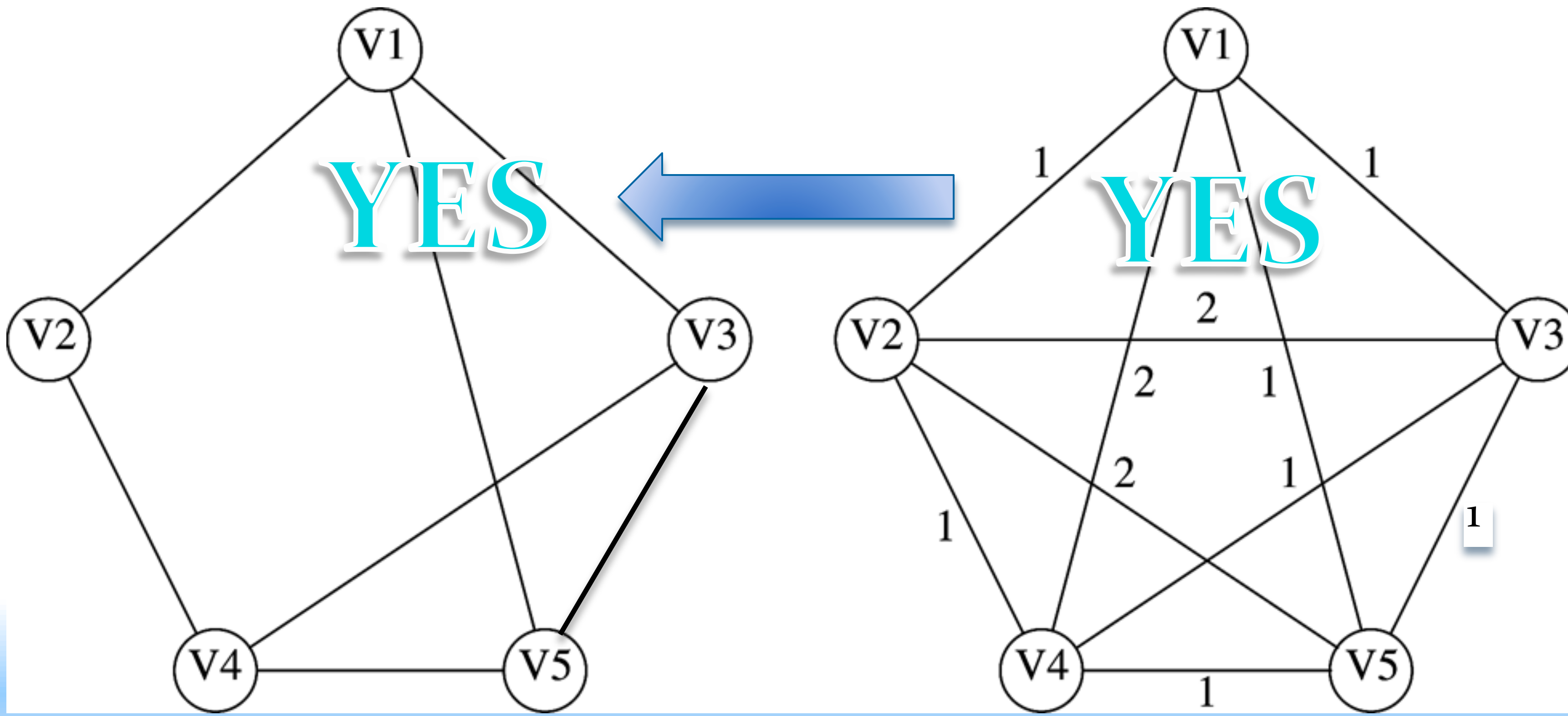


Is there a simple cycle that visit all vertices?

Is there a simple cycle that visits all vertices and has total cost <= K

Hamiltonian Cycle reduced to TSP

Reduction: INPUT for Hamiltonian Cycle => INPUT for TSP (K = |V|)



Is there a simple cycle that visit all vertices?

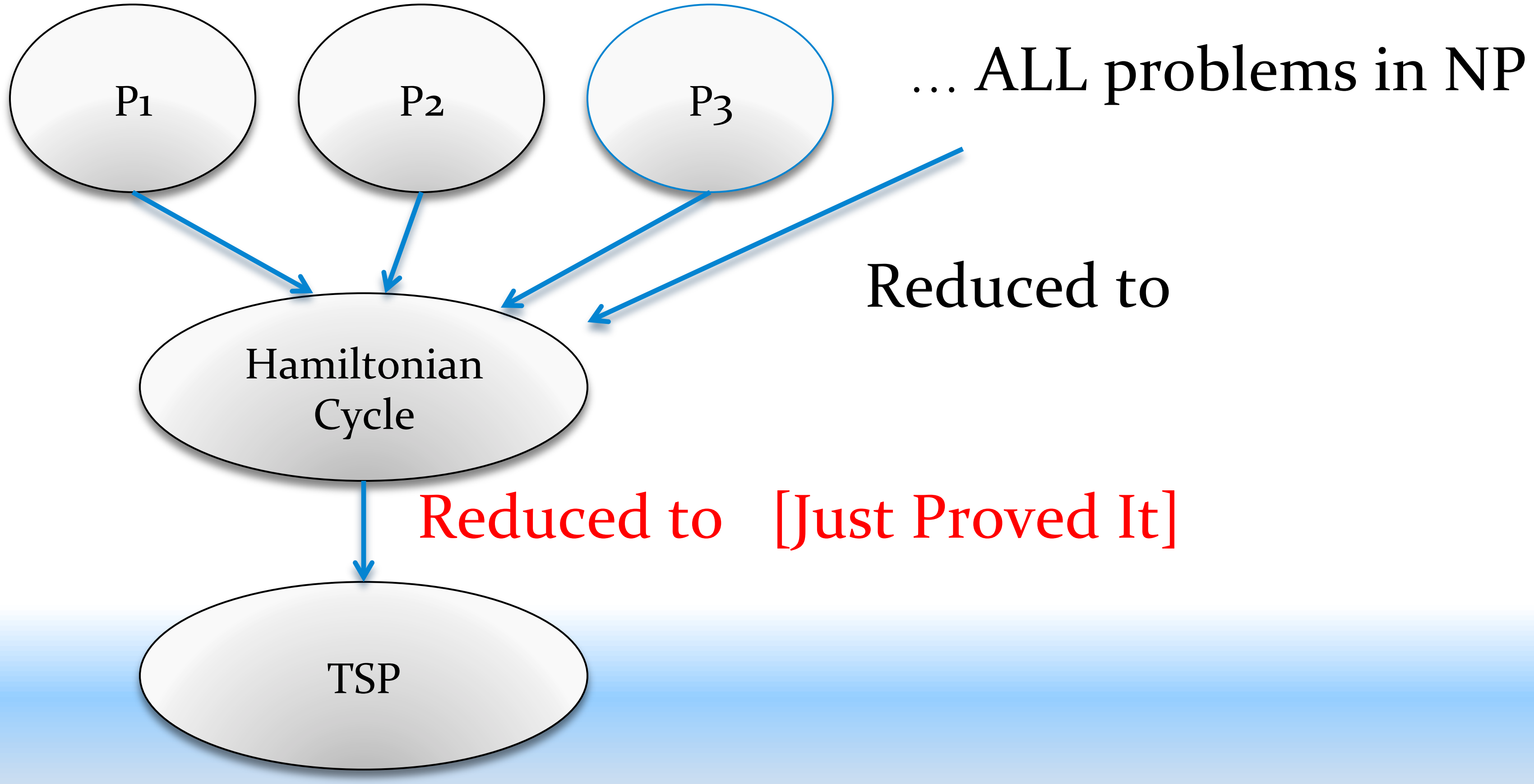
Is there a simple cycle that visits all vertices and has total cost <= K

Hamiltonian Cycle reduced to TSP

- Given a graph G (input to HC) construct (**in polynomial time**) graph G' (input to TSP).
- If you have an algorithm that solves TSP for K = |V| for graph G'

  then you have a solution for HC for graph G.

Proof diagram



NP-Complete

- Why is Hamiltonian NP-complete?
- Fundamental result:

The satisfiability problem (SAT) is NP complete

proved in 1971 by Cook

→ After that polynomial reductions have been shown from SAT to Hamiltonian, and to other NP-complete problems, such as:

→ Longest Path (in Graphs), Bin Packing, Knapsack, Graph coloring, … (long list)

Summary

- Decidable vs. Undecidable problems.
- The class NP.
- Question: Does NP contain all decidable problems?
- Question (rephrased): Are there decidable problems not in the class NP?

Problems that are not in NP?

- Answer: There exist decidable problems that are not in class NP.

Problems that are not in NP?

- Answer: There exist decidable problems that are not in class NP.
- Example: Given a graph determine whether it DOES NOT have a Hamiltonian cycle.
  - Why is it decidable?
  - Why is it not in NP?

Is P = NP?

- Note that P is a subset of NP.
  - That means that each problem in P is also in NP.
- Question: Is there a problem X that is in NP but not in P?
- Answer?