**Reading:** 8.4-8.5

#### Last time:

- tractability and intractability
- decision problems

### Today:

- $\mathcal{NP}$ -completeness
- "notorious problem" NP.
- redutions from 3-SAT.

# Problem 1: Independent Set (INDEP-SET)

input: G = (V, E)

output:  $S \subset V$ 

- satisfying  $\forall v \in S, (u, v) \notin E$
- maximizing |S|

Problem 2: SAT

# Problem 3: Traveling Salesman (TSP)

input:

• G = (V, E), complete graph.

•  $c(\cdot) = \text{costs on edges.}$ 

output: cycle  ${\cal C}$  that

- passes through all vertices exactly once.
- minimizes total cost  $\sum_{e \in C} c(e)$ .

### Problem 4: 3-SAT

input: boolean formula  $f(\mathbf{z})$ 

- in conjunctive normal form (CNF)
- three literals per or-clause
- or-clauses anded together.

#### output:

- "Yes" if assignment  $\mathbf{z}$  with  $f(\mathbf{z}) = T$  exists
- "No" otherwise.

# Problem 5: Hamiltonian Cycle (HC)

input: G = (V, E) (directed)

output: cycle C to visit each vertex exactly once.

**Note:** since  $X_d =_{\mathcal{P}} X$ , we write "X" but we mean " $X_d$ "

# A notoriously hard problem

"one problem to solve them all"

Note: all example problem have short certificates that could easily verify "yes" instance.

**Def:**  $\mathcal{NP}$  is the class of problems that have short (polynomial sized) certificates that can easily (in polynomial time) verify "yes" instances.

**Historical Note:**  $\mathcal{NP}$  = non-deterministic **Fact:** NP is  $\mathcal{NP}$ -complete. polynomial time

"a nondeterministic algorithm could guess the certificate and then verify it in polynomial time"

**Note:** Not all problems are in  $\mathcal{NP}$ .

E.g., unsatisfiability.

Def:

- Problem X is in  $\mathcal{NP}$  if exists short easily-verifiable certificate.
- Problem X is  $\mathcal{NP}$ -hard if  $\forall Y \in$  $\mathcal{NP}, Y \leq_{\mathcal{P}} X.$
- Problem X is  $\mathcal{NP}$ -complete if  $X \in \mathcal{NP}$ and X is  $\overline{\mathcal{NP}}$ -hard.

**Lemma:** INDEP-SET  $\in \mathcal{NP}$ .

Lemma: SAT  $\in \mathcal{NP}$ .

Lemma:  $TSP \in \mathcal{NP}$ .

Goal: show INDEP-SET, SAT, TSP are  $\mathcal{NP}$ -complete.

# Notorious Problem: NP

input:

- decision problem verifier program VP.
- polynomial  $p(\cdot)$ .
- decision problem instance: x

output:

- "Yes" if exists certificate c such that VP(x,c) has "verified = true" at computational step p(|x|).
- "No" otherwise.

Note: Unknown whether  $\mathcal{P} = \mathcal{NP}$ .

**Note:**  $\leq_{\mathcal{P}}$  is transitive: if  $Y \leq_{\mathcal{P}} X$  and  $X \leq_{\mathcal{P}} Z$  then  $Y \leq_{\mathcal{P}} Z$ .

### Plan:

- 1. NP  $\leq_{\mathcal{P}} \cdots \leq_{\mathcal{P}} 3$ -SAT
- 2. 3-SAT  $\leq_{\mathcal{P}}$  INDEP-SET
- 3. 3-SAT  $\leq_{\mathcal{P}}$  HC  $\leq_{\mathcal{P}}$  TSP

# Problem: Hamiltonian Cycle

input: G = (V, E) (directed)

output: cycle C to visit each vertex exactly once.

**Lemma:** hamiltonian cycle is  $\mathcal{NP}$ -hard

**Proof:** (reduction from 3-SAT)

#### Step 1: construction

- turn 3-SAT formula f in to graph G with hamiltonian cycle iff f is satisfiable.
- idea: variable = isolated path, right-toleft = true, left-to-right = false.
- idea: clause is node, which needs to be hit by at most one literal being true.
- construction:
  - left-right path per variable.
  - splice in clause nodes.

#### Step 2: runtime.

Step 3: correctness.

# $\mathbf{TSP}$

# **Lemma 0.1** *TSP is* $\mathcal{NP}$ *-hard.*

**Proof:** reduction from Hamiltonian Cycle

- encode edges with cost 1
- encode non-edges with cost n.

 $\Rightarrow$  exists HC iff TSP cost  $\leq n$