COMP 3011
DESIGN AND ANALYSIS OF ALGORITHMS
FALL 2024

NP-completeness

LI Bo
Department of Computing
The Hong Kong Polytechnic University

TONIGHT

Y301

Familiarize yourself with exam question formats and address any possible concerns.

✓ 6:35 pm – 8:05 pm Practice Midterm

✓ 8:20 pm – 9:00 pm Review of Sample Solutions and Overview

ESTABLISHING NP-COMPLETENESS

NP-complete
A problem $Y \in NP$ with the property that for every problem $X \in NP$, $X \leq_P Y$.

Example: SAT $\leq_P$ 3-SAT $\leq_P$ INDEPENDENT-SET $\leq_P$ VERTEX-COVER $\leq_P$ SET-COVER.

3-SAT$\leq_P$SUBSET-SUM $\equiv_P$ PARTITION$\leq_P$KNAPSACK

Recipe.

To prove that $Y \in$ NP-complete:

- Step 1. Show that $Y \in$ NP.
- Step 2. Choose an NP-complete problem X.
- Step 3. Prove that $X \leq_P Y$.

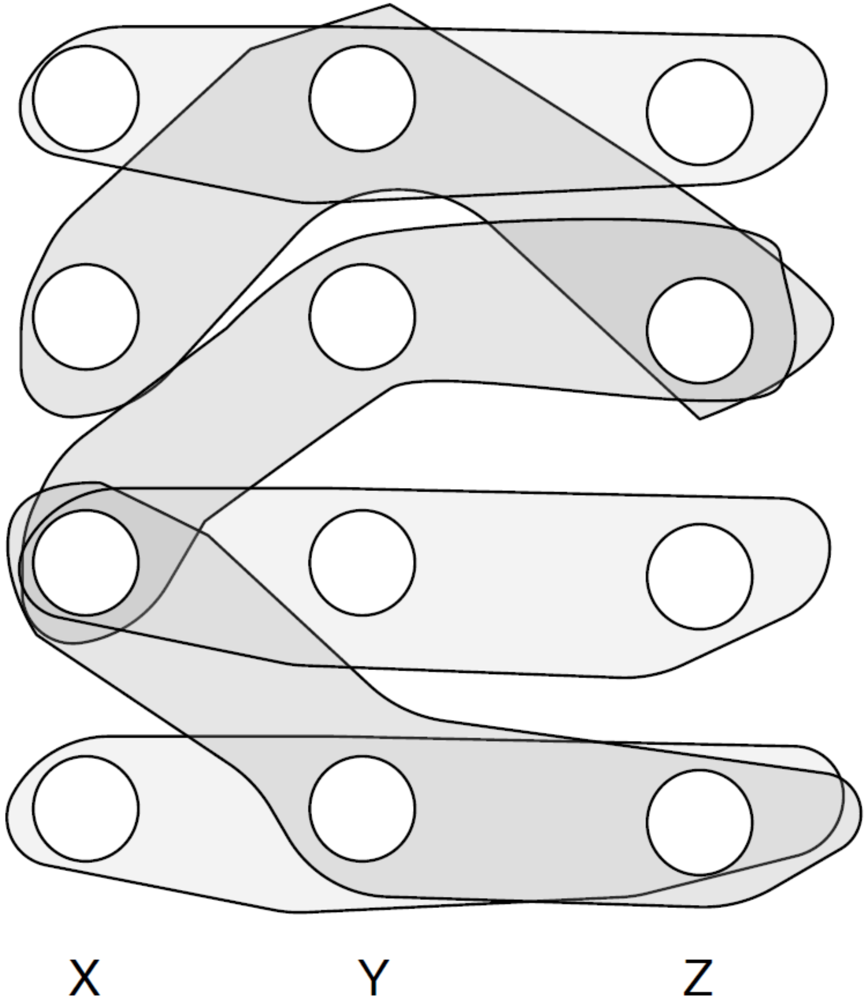
SIX BASIC NP-COMPLETE PROBLEMS

Six Basic NP-Complete problems

- 3-Satisfiability (3-SAT)
- ***3-Dimensional Matching (3DM)*** $3SAT \leq_p 3DM$

  Exact Cover by 3-Sets (X3C) $3DM \leq_p X3C$
- Vertex Cover (VC)
- Independent Set (IS)
- ***Hamiltonian Cycle (HC)*** $3SAT \leq_p HC$, $VC \leq_p HC$
- Partition

3-Dimensional Matching (3DM)



Given: Sets X, Y, Z, each of size n, and a set $T \subset X \times Y \times Z$ of order triplets.

Question: is there a set of n triplets in T such that each element is contained in exactly one triplet?

Exact Cover by 3-Sets (X3C)

Given: a set U with $|U| = 3n$ and a collection C of 3-element subsets of U.

Question: Does C contain an exact cover for U, that is, a subcollection $C' \subseteq C$ such that every element of U occurs in exactly one member of C'?

Theorem. 3DM$\leq_p$X3C.

3DM: $T \subseteq X \times Y \times Z$

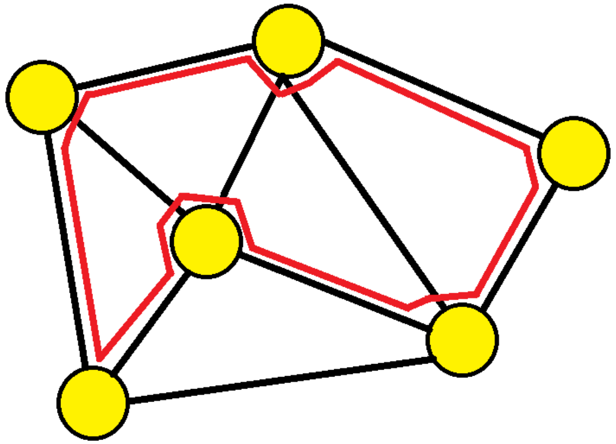
$\downarrow$

X3C: $\mathrm{U} = X \cup Y \cup Z$ (unordered)

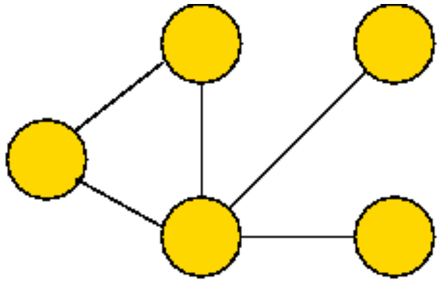
$M \subseteq T$ is a matching with size n for 3DM iff
$M \subseteq U$ is a 3-exact-cover for U

Hamiltonian Cycle Problem

We are given a unweighted and undirected graph $G = (V, E)$. Is there a cycle in G that visits each node exactly once?



Yes

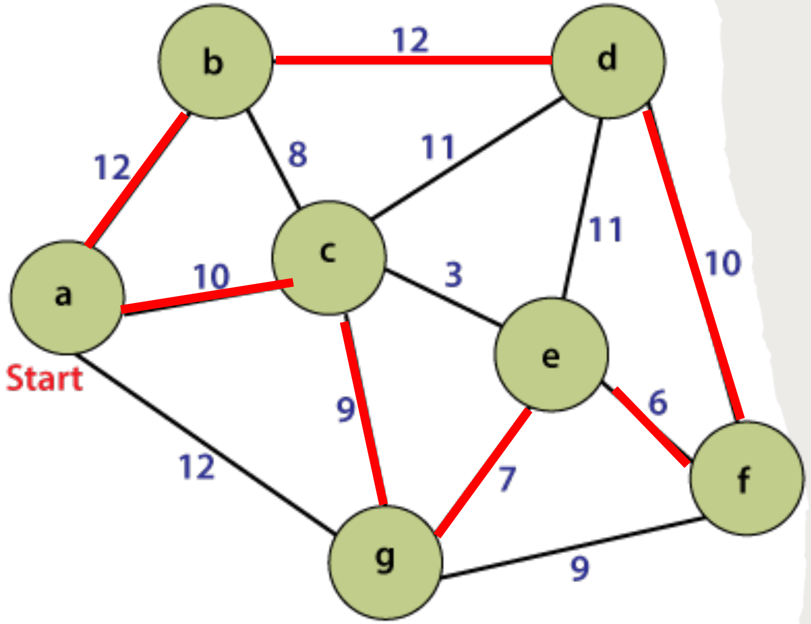


No

Travelling Salesman Problem (TSP)

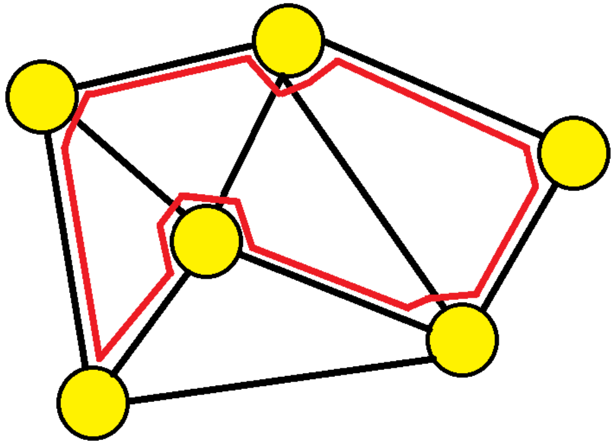
We are given n cities $1, \cdots, n$, and a nonnegative integer distance $l(i,j)$ between any two cities i and j (assume that the distances are symmetric, that is, $l(i,j) = l(j,i)$ for all i and j). We are also given a parameter K.

We are asked to determine if there is a tour of all cities with total distance no more than K.

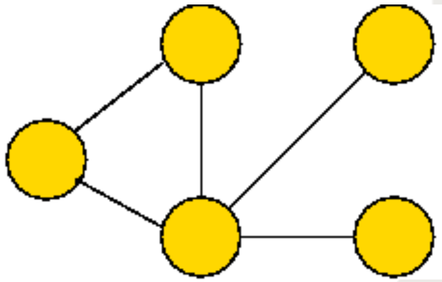


Hamiltonian Cycle Problem (undirected graph)

We are given a unweighted and undirected graph $G = (V, E)$. Is there a cycle in G that visits each node exactly once?



Yes



No

Travelling Salesman Problem (TSP)

Theorem. TSP is NP-complete.

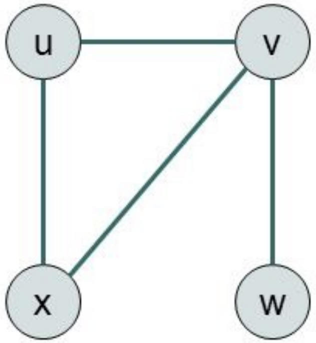
Proof:

- TSP (decision version) is in NP.
- Reduction from Hamiltonian Cycle.
- Let G be an arbitrary undirected graph with n vertices.
- Construct a length function for K_n (complete graph) as follows

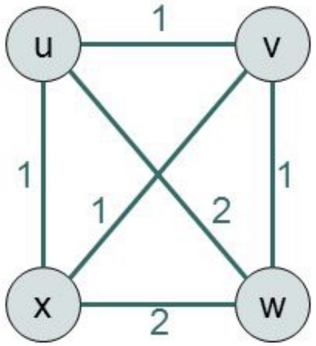
$$\ell(e) = \begin{cases} 1 & \text{if } e \text{ is an edge in } G, \\ 2 & \text{otherwise.} \end{cases}$$

- If G has a Hamiltonian cycle, then there is a TSP cycle in K_n whose length is exactly n;
- Otherwise, every TSP cycle in K_n has length at least $n+1$.

G



K_n



- If G has a Hamiltonian cycle if and only if there is a TSP cycle in K_n whose length is exactly n.

A Few More Definitions

Strong NP-completeness

- The pseudo-polynomial algorithm (for SUBSET-SUM and KNAPSACK, $O(nW)$) does not establish that P = NP!

- The NP-completeness proof for SUBSET-SUM, KNAPSACK and PARTITION uses exponentially large integers.

- Problems including VERTEX-COVER, SET-COVER, INDEPENDET-SET, were shown NP-complete via reductions that constructed only polynomially small integers.

Strongly NP-complete

If a problem remains NP-complete even if any instance of length n is restricted to contain integers of size at most $p(n)$, a polynomial, then we say that the problem is ***strongly NP-complete***.

NP-hard

NP-complete

A problem $Y \in NP$ with the property that for every problem $X \in NP$, $X \leq_P Y$.

Recipe.

To prove that $Y \in$ NP-complete:

- Step 1. Show that $Y \in$ NP.
- Step 2. Choose an NP-complete problem X.
- Step 3. Prove that $X \leq_P Y$.

- Sometimes we may be able to show that all problems in NP polynomially reduce to some problem A, but we are unable to argue that A ∈ NP.
- So, A does not qualify to be called NP-complete.
- Yes, undoubtedly A is as hard as any problem in NP, and hence most probably intractable.

NP-hard

- The term NP-hard is also used in the literature to describe optimization problems (which are not decision problems and thus not in NP).

Approximation Algorithms

Definition

For a maximization problem, an algorithm is called α-approximation if for any input, the algorithm returns a feasible solution S such that

$$\frac{f(S)}{f(O)} \geq \alpha,$$

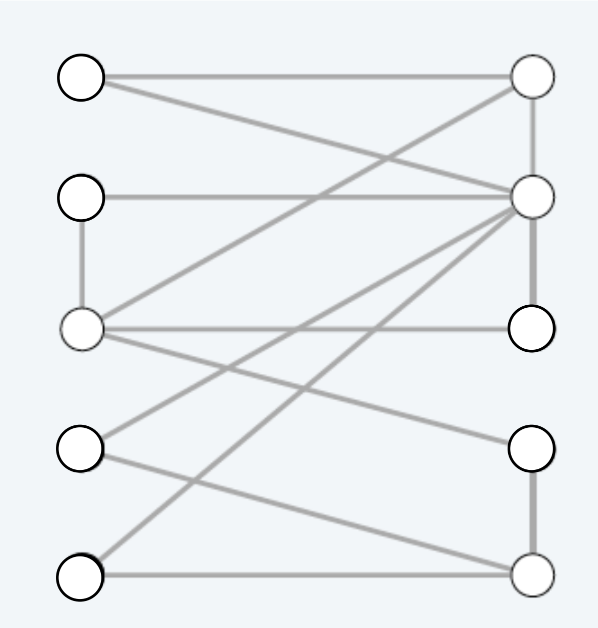
where O is an optimal solution, and f evaluates the quality of the solution.

INDEPENDENT SET

15

Independent Set Problem

Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. Find a set of maximum number of vertices such that no two are adjacent?



Intuition

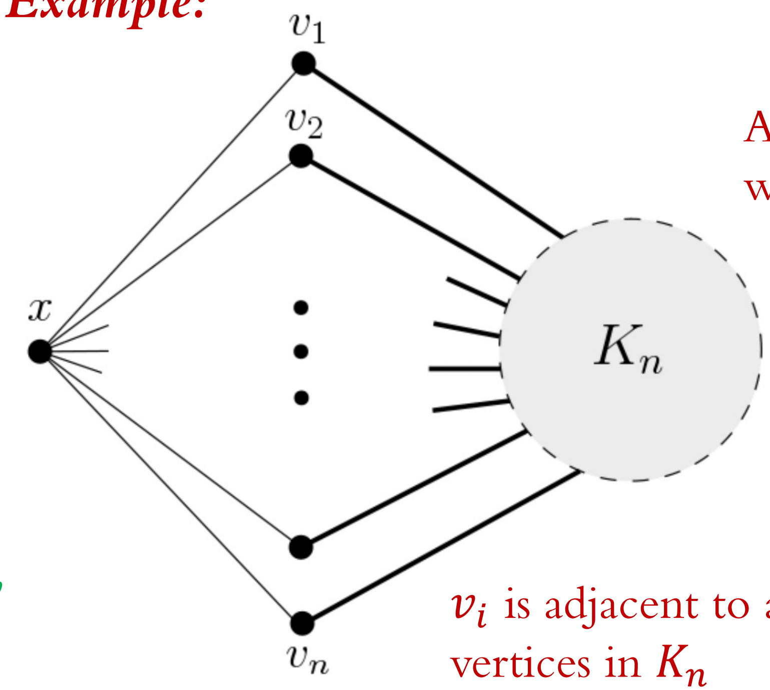
Always select the node with minimum degree.

Greedy Algorithm

Require: a graph $G = (V, E)$
  $W \leftarrow V$
  $S \leftarrow \emptyset$
  while $W \neq \emptyset$ **do**
    Find a vertex $v \in W$ with minimum degree in $G[W]$
    $W \leftarrow W \setminus N_G[v]$
    $S \leftarrow S \cup \{v\}$
  end while
  return S

the subset of vertices adjacent to v and v

A Bad Example:



A complete graph with n vertices

$OPT = n$

$ALG = 2$

v_i is adjacent to all vertices in K_n

Independent Set Problem

Greedy Algorithm

Require: a graph $G = (V, E)$

$W \leftarrow V$

$S \leftarrow \emptyset$

while $W \neq \emptyset$ **do**

Find a vertex $v \in W$ with minimum degree in $G[W]$

$W \leftarrow W \setminus N_G[v]$

$S \leftarrow S \cup \{v\}$

end while

return S

Theorem.

The Greedy Algorithm is $(1/(\Delta + 1))$-approximation for graphs with degree at most Δ.

Proof

- Every time a vertex v is picked by Greedy, at most Δ vertices are removed.
- So at the end at most $|S| \cdot (\Delta + 1)$ vertices have been removed.
- All nodes have been removed:

$$n \leq (\Delta + 1) \cdot |S|$$

That is

$$|S| \geq \frac{n}{\Delta + 1} \geq \frac{OPT}{\Delta + 1}$$