Slido: **#ADA2022**

CSIE 2136 Algorithm Design and Analysis, Fall 2022

National Taiwan University
國立臺灣大學

Approximation Algorithms

Hsu-Chun Hsiao

Announcement

- HW4 deadlines
  - Problem 1-2: 2022/12/20 14:20
  - Problem 3-5: 2022/12/30 23:59
  - Problem 6: 2023/01/07 23:59
- Final exam: 2022/12/22
  - Closed book
  - Scope: all topics taught in ADA, focusing on those after the midterm
  - Details will be released on COOL later
- If you need to know your grade early, please notify us by email.
- No TA hours on and after 2022/12/22
- 優良助教問卷、期末教學意見調查

Agenda

- What is approximation? (same as last week's slides)
- Vertex cover
  - Approximate vertex cover (same as last week's slides)
  - Approximate weighted vertex cover
- Traveling salesman problem
  - Proving NP-completeness
  - Approximation for metric TSP & inapproximability
- Randomized approximation algorithms
  - 3-CNF-SAT
  - MAX-CUT

What is approximation?

What is approximation?

- “A value or quantity that is nearly but not exactly correct”
- Approximation algorithms
  - applied to optimization problem, not decision problems
  - should guaranteed to find solutions close to optimality, returning near-optimal answers
    - Cf. heuristics search: no guarantee
    - How “near” is near-optimal?

Approximation algorithms

- $\rho(n)$ -approximation algorithm
  - Efficient: guaranteed to run in polynomial time
  - General: guaranteed to solve every instance of the problem
  - Near-optimal: guaranteed to find solution within a factor of $\rho(n)$ of the cost of an optimal solution
- Approximation ratio $\rho(n)$
  - n: input size
  - C^*: cost of an optimal solution
  - C: cost of the solution produced by the approximation algorithm

$$\boxed{\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \le \rho(n)}$$

Maximization problem: $\frac{C^*}{C} \le \rho(n)$

Minimization problem: $\frac{C}{C^*} \le \rho(n)$

Approximate ratio ρ(n)

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \le \rho(n)$$

n: input size
C^*: cost of optimal solution
C: cost of approximate solution

- $\rho(n) \geq 1$
- $\rho(n)$ 越小越好！
- An exact algorithm has $\rho(n) = 1$

- Challenge: prove that C is close to C^* without knowing C^*!

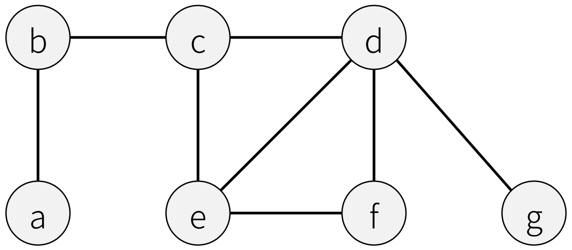
Approximate Vertex-Cover

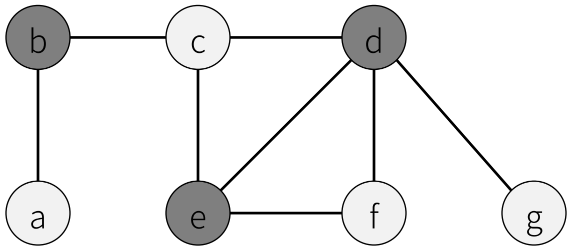
Vertex Cover

A vertex cover of $G = (V, E)$ is a subset $V' \subseteq V$ such that if $(w, v) \in E$, then $w \in V'$ or $v \in V'$ or both

The Vertex-Cover Problem (Optimization)

Find a vertex cover of minimum size in G

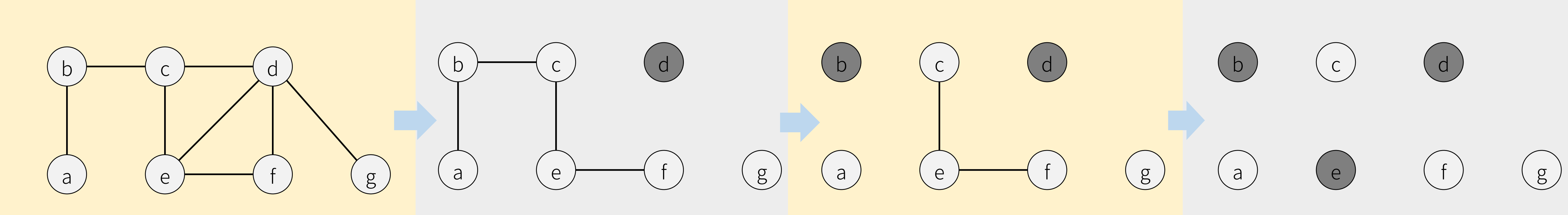




b, d, e is a minimum vertex cover
(size = 3)

Q: Consider a greedy heuristic: In each iteration, cover as many edges as possible (vertex with the maximum degree) and then delete the covered edges. Does it always find an optimal solution?

- No. Otherwise, we would have proven P=NP.



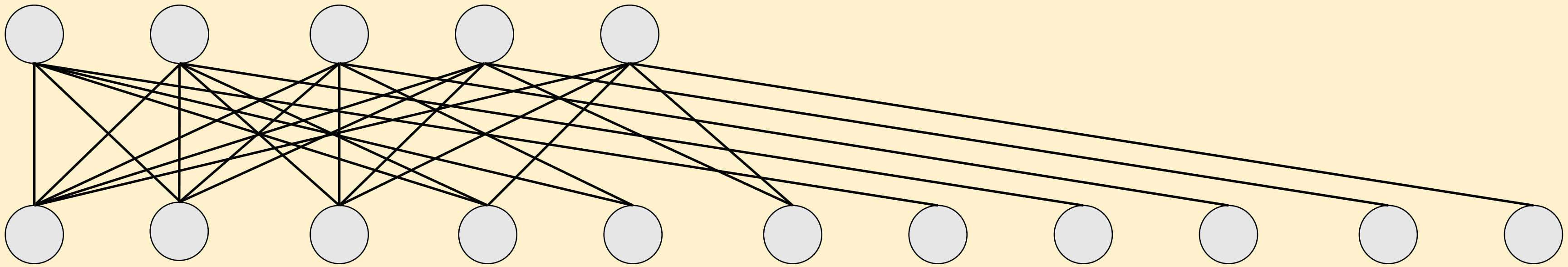
b, d, e is a vertex cover of size 3 found by the greedy algorithm (and it happens to be optimal!)

Q: Construct a graph on which the above greedy heuristic does not yield an optimal solution

Select d, c, a, e sequentially on the previous slide

Exercise 35.1-3 Construct a graph on which the above greedy heuristic does not have an approximation ratio of 2.

Hint: Try a bipartite graph with vertices of uniform degree on the left and vertices of varying degree on the right.



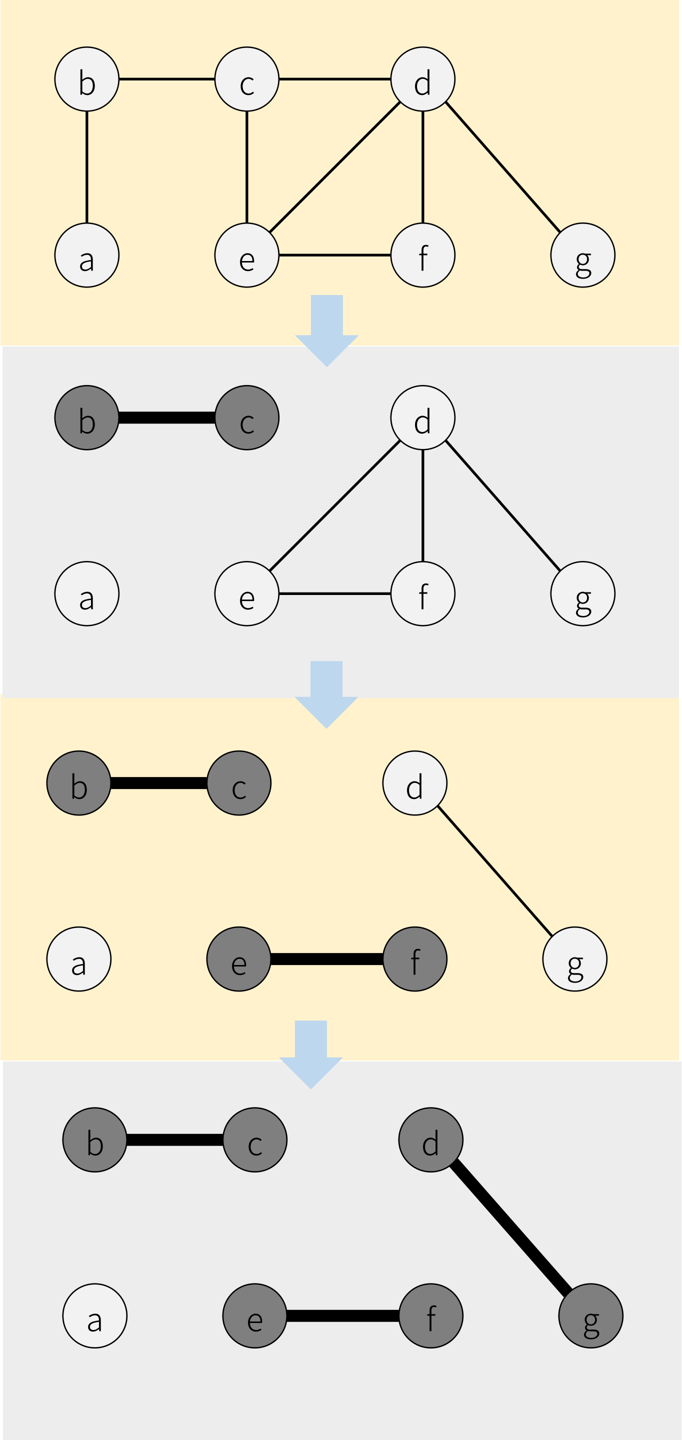
An approximation algorithm to VERTEX-COVER

- APPROX-VERTEX-COVER
  - Randomly select one edge at a time
  - Add both vertices to the cover
  - Remove all incident edges
- Running time = $O(V + E)$
- Claim: Approximation ratio $\rho(n) = 2$

APPROX-VERTEX-COVER(G)

```
C = Ø
E' = G.E
while E' ≠ Ø
    let (u, v) be an arbitrary edge of E'
    C = C ∪ {u, v}
    remove from E' every edge incident on either u or v
return C
```

An approximation algorithm to VERTEX-COVER: example



b, c, d, e, f, g is a vertex cover of size 6 found by the approximation algorithm (not optimal!)

APPROX-VERTEX-COVER is 2-approximation

Let A denote the set of edges picked in line 4.
$\Rightarrow$ size of the vertex cover $|S| = 2|A|$

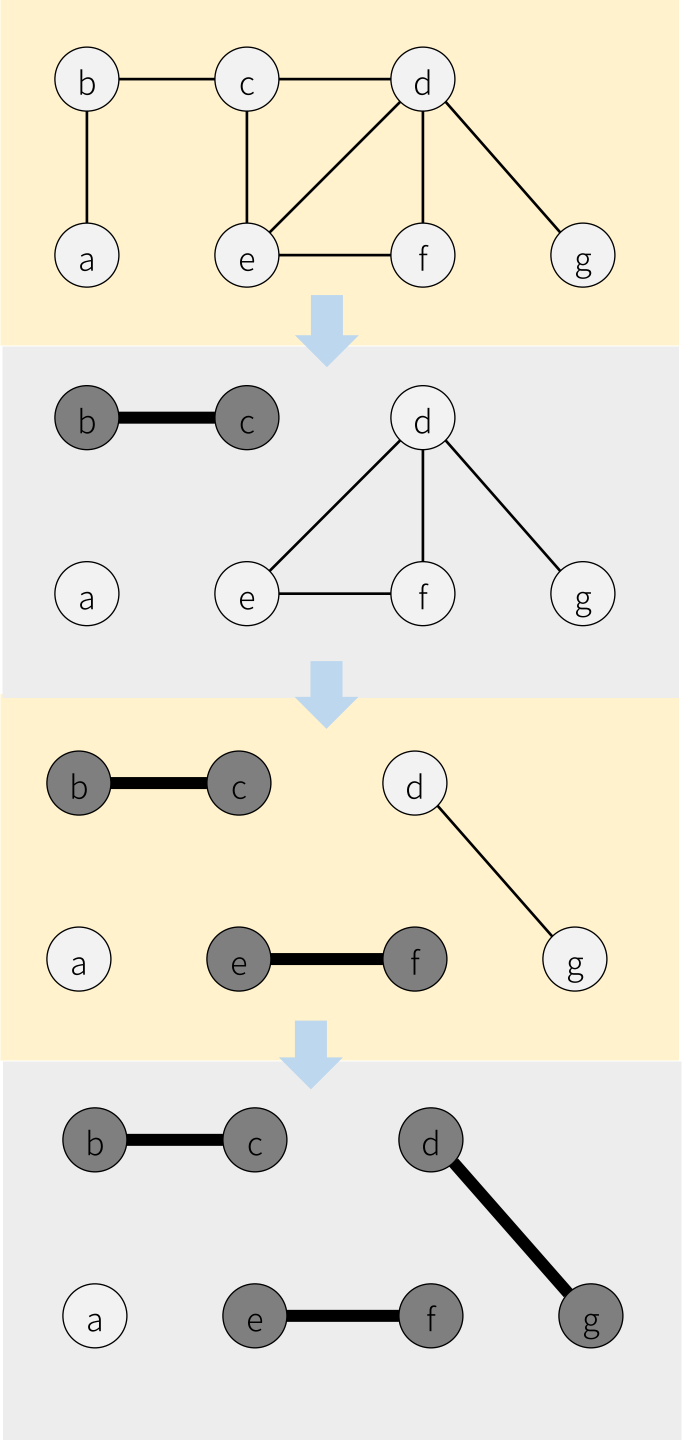
In any vertex cover S' (including S^*), every vertex v in S' covers at most one edge in A because no two edges in A share a vertex.

$\Rightarrow |A| \le |S^*|$

Combing the two equations, we have
$\frac{1}{2}|S| = |A| \le |S^*|$

$\Rightarrow \rho(n) = 2$

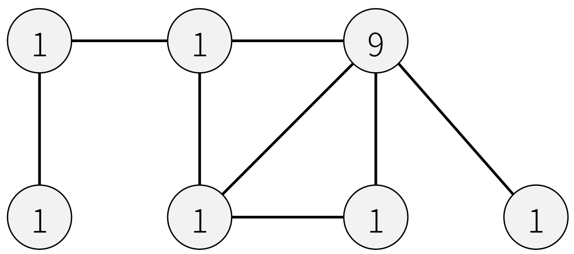
Note: the proof doesn't require knowing the actual value of $|S^*|$

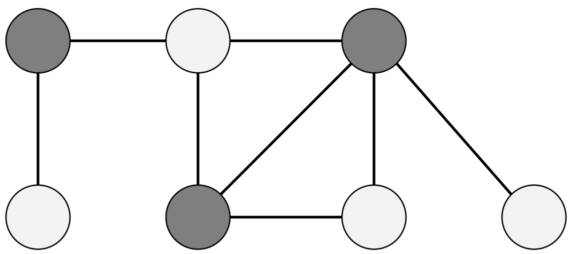


Approximate Weighted-Vertex-Cover

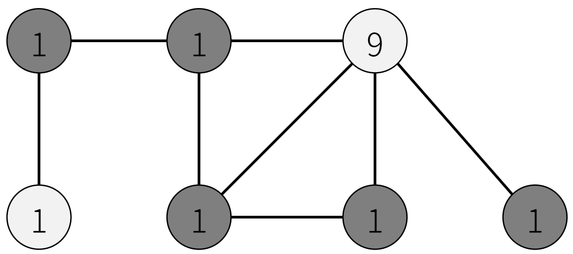
The Weighted-Vertex-Cover Problem (Optimization)

Given an undirected graph $G = (V, E)$, and a weight function $w(v)$ for all $v \in V$. Find a **vertex cover** whose total weight is minimized.





minimum vertex cover (size = 3)



minimum **weighted** vertex cover (weight = 5)

Q: Is the decision problem of weighted vertex cover a NPC problem?
Q: Is APPROX-VERTEX-COVER also a 2-approximation algorithm for finding minimum weighted vertex cover?

Modeling weighted vertex cover via integer programming

- Recall the Integer programming formulation for vertex cover
- Weighted vertex cover can be modeled similarly:
  - Variables: $x_i \in \{0,1\}$ represents whether vertex v_i is covered
  - Minimize: $z^* = \Sigma_{i=1}^{|V|} w(v_i) x_i$
  - Subject to
    - $x_i + x_j \geq 1, \forall e = (v_i, v_j) \in E$
    - $x_i \in \{0,1\}, \forall i = 1,2, \dots, |V|$

Approximating weighted vertex cover via linear programming

- Now we relax the integer programming formulation to linear programming, i.e., allowing non-integer variables
- Linear programming to approximate weighted vertex cover
  - Variables: $0 \le x_i \le 1$
  - Minimize: $\hat{z}^* = \sum_{i=1}^{|V|} w(v_i) x_i$
  - Subject to
    - $x_i + x_j \ge 1, \forall e = (v_i, v_j) \in E$
    - $0 \le x_i \le 1, \forall i = 1,2, \dots, |V|$

Q: Argue that $\hat{z}^* \le z^*$. That is, the LP solution is a lower bound of the IP solution.

An approximation algorithm to WEIGHTED-VERTEX-COVER

```
APPROX-WEIGHTED-VERTEX-COVER(G,w)
C = ∅
X = LinearProgram(G,w)
for each v ∈ V
    if x_v ≥ 1/2
        C = C ∪ {v}
return C
```

LinearProgram(G, w)

Variables: $0 \leq x_i \leq 1$

Minimize: $\sum_{i=1}^{|V|} w(v_i) x_i$

Subject to

$x_i + x_j \geq 1, \ \forall e = (v_i, v_j) \in E$

$0 \leq x_i \leq 1, \ \forall i = 1,2,\ldots,|V|$

Q: Show that APPROX-WEIGHTED-VERTEX-COVER outputs a vertex cover.

To satisfy the constraint $x_i + x_j \geq 1, \forall e = (v_i, v_j) \in E$, either x_i or x_j must be $\geq 1/2$. Thus, by choosing v_i whose $x_i \geq 1/2$, we ensure all edges are covered.

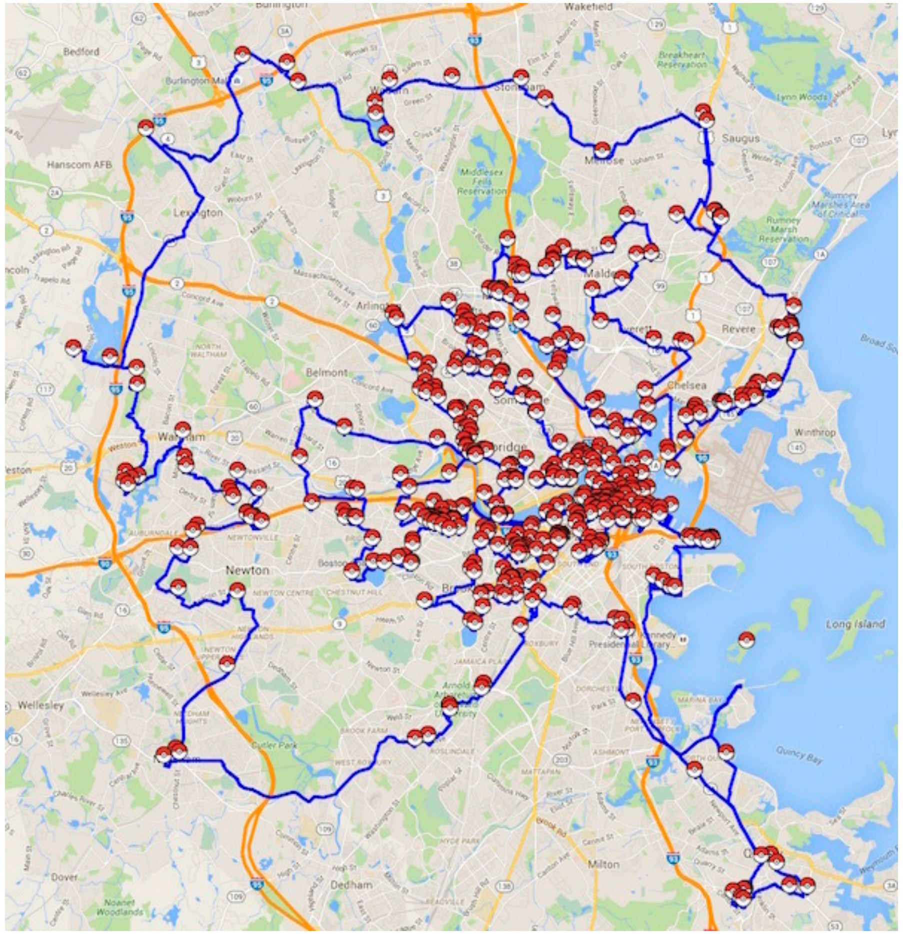
Q: show that APPROX-WEIGHTED-VERTEX-COVER is a 2-approximation algorithm for weighted vertex cover.

Hint: what's the relationship among $\hat{z}^*$, z^*, and $w(C)$?

Traveling Salesman Problem: Proving NP-Completeness

Traveling Salesman Problem (TSP)

- Optimization problem: Given an undirected complete graph $G = (V, E)$ and a non-negative edge cost function w, find a tour of lowest cost
- Decision problem: Given an undirected complete graph $G = (V, E)$ and a non-negative edge cost function w, find a tour of cost at most k
- Tour = visit each vertex exactly once and return to the beginning



A tour to catch every (518) Pokémon in Boston
https://www.math.uwaterloo.ca/tsp/poke/index.html

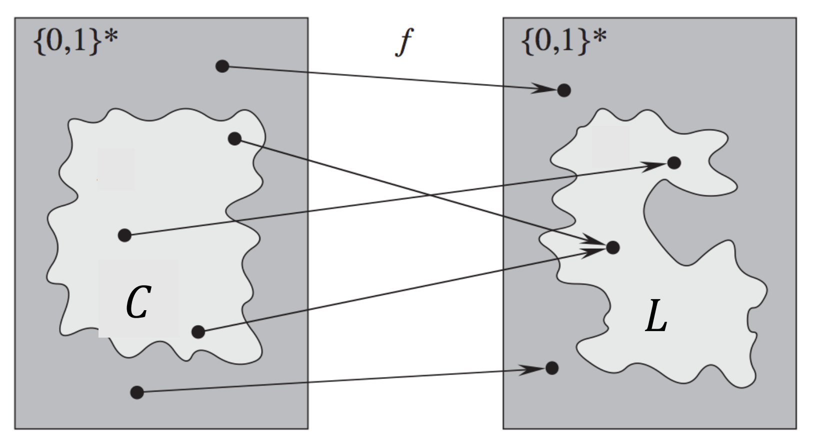
The TSP Problem

$\mathrm{TSP} = \{\langle G, w, k\rangle : G = (V, E)$ is a complete graph, w is a non-negative cost function for edges, G has a traveling-salesman tour with cost at most $k\}$

- Prove that $\mathbf{TSP} \in$ NP-COMPLETE
- Polynomial-time reduction: HAM-CYCLE $\leq_p$ $\mathbf{TSP}$

Step-by-step approach for proving L in NPC:

1. Prove $L \in$ NP
2. Prove $L \in$ NP-hard ($C \leq_p L$)
   ① Select a known NPC problem C
   ② Construct a reduction f transforming every instance of C to an instance of L
   ③ Prove that x in C if and only if $f(x)$ in L for all x in $\{0,1\}^*$
   ④ Prove that f is a polynomial time transformation

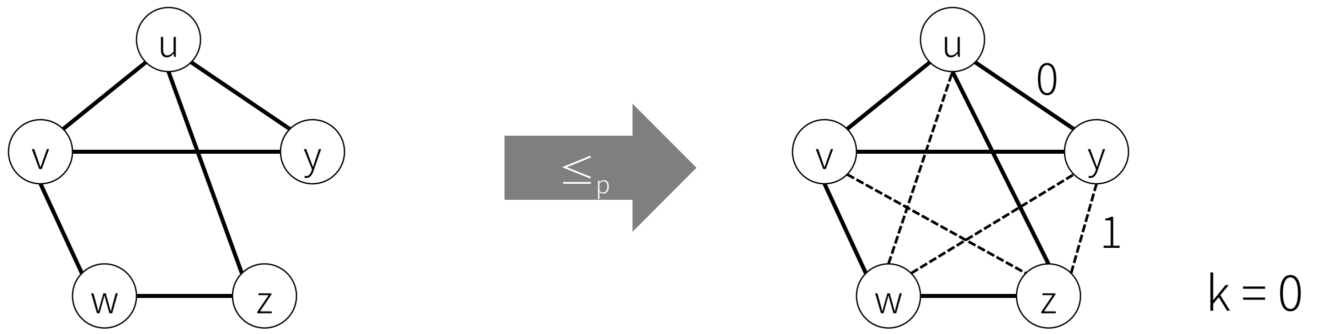


The TSP Problem

TSP $= \{\langle G, w, k\rangle : G = (V, E)$ is a complete graph, w is a non-negative cost function for edges, G has a traveling-salesman tour with cost at most $k\}$

② Construct a reduction f transforming every HAM-CYCLE's instance $G_H = (V_H, E_H)$ to a TSP instance with cost at most k

- We construct a TSP instance in which G is a complete graph with $V = V_H$, and $w(i,j) = 0$ if $(i,j) \in E_H$; $w(i,j) = 1$, otherwise.
- With this reduction function, we set $k = 0$



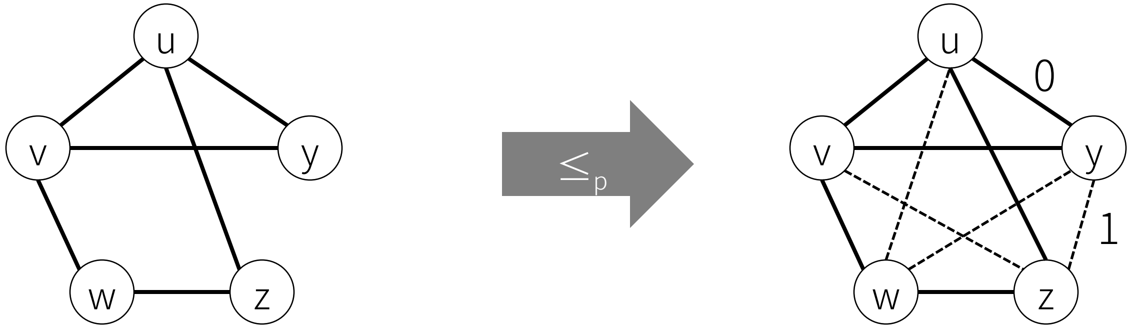
HAM-CYCLE $\leq_p$ TSP

③ Prove that $x \in$ HAM_CYCLE $\Leftrightarrow f(x) \in$ TSP

Correctness proof: $x \in$ HAM-CYCLE $\Leftrightarrow f(x) \in \textbf{TSP}$

- More specifically, we want to prove that G contains a Hamiltonian cycle $h = \langle v_1, v_2, \dots, v_n, v_1 \rangle$ if and only if $\langle v_1, v_2, \dots, v_n, v_1 \rangle$ is a traveling-salesman tour with cost at most 0

u, y, v, w, z, u is a Hamiltonian cycle ⇔ u, y, v, w, z, u is a traveling-salesman tour with cost 0

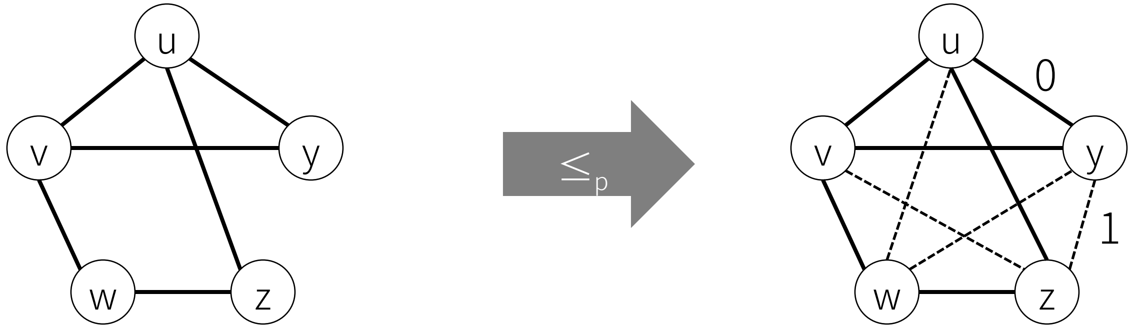


HAM-CYCLE $\leq_p$ TSP

③ Prove that x ∈ HAM_CYCLE ⇔ $f(x)$ ∈ TSP

Correctness proof: x ∈ HAM-CYCLE ⇒ $f(x) \in \mathbf{TSP}$

- Suppose the Hamiltonian cycle is $h = \langle v_1, v_2, \dots, v_n, v_1 \rangle$
- ⇒ h is also a tour in the transformed TSP instance
- ⇒ The cost of the tour h is 0 since there are n consecutive edges in E, and so has cost 0 in $f(x)$
- ⇒ $f(x) \in$ TSP ($f(x)$ has a TSP tour with cost ≤ 0)



u, y, v, w, z, u is a Hamiltonian cycle

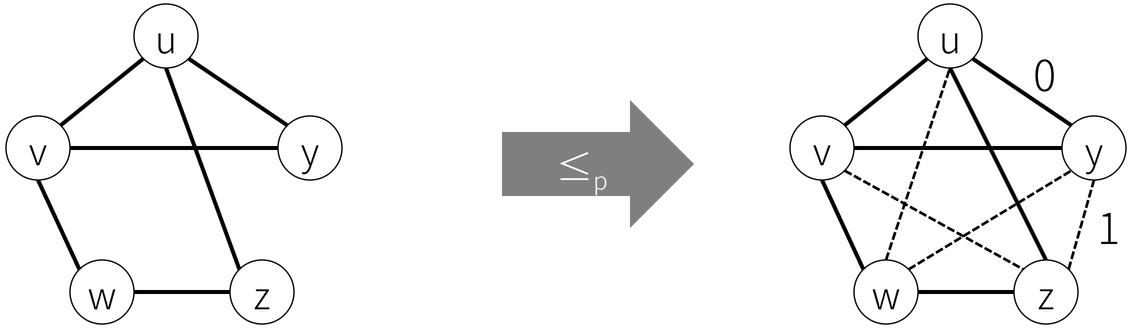
u, y, v, w, z, u is a traveling-salesman tour with cost 0

HAM-CYCLE $\leq_p$ TSP

③ Prove that x ∈ HAM_CYCLE ⇔ $f(x)$ ∈ TSP

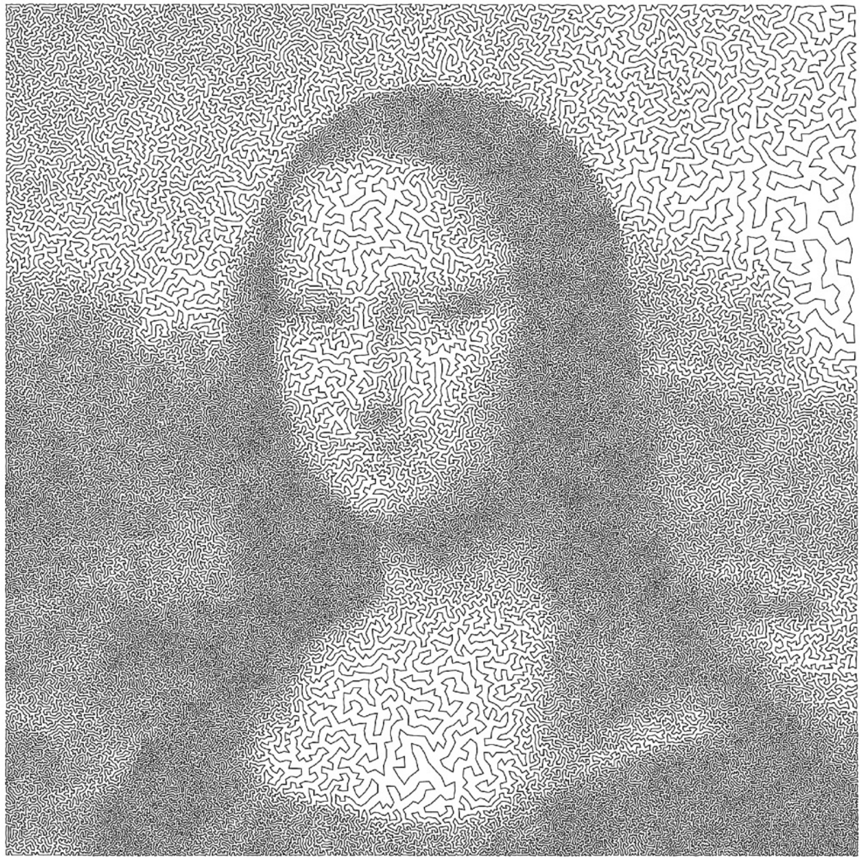
Correctness proof: $f(x)$ ∈ **TSP** ⇒ x ∈ HAM−CYCLE

- Suppose after reduction, there is a TSP tour with cost ≤ 0. Let it be $\langle v_1, v_2, \dots, v_n, v_1 \rangle$
- ⇒ The TSP tour contains only edges in E_H
- ⇒ Thus, $\langle v_1, v_2, \dots, v_n, v_1 \rangle$ is a Hamiltonian cycle (x ∈ HAM-CYCLE).

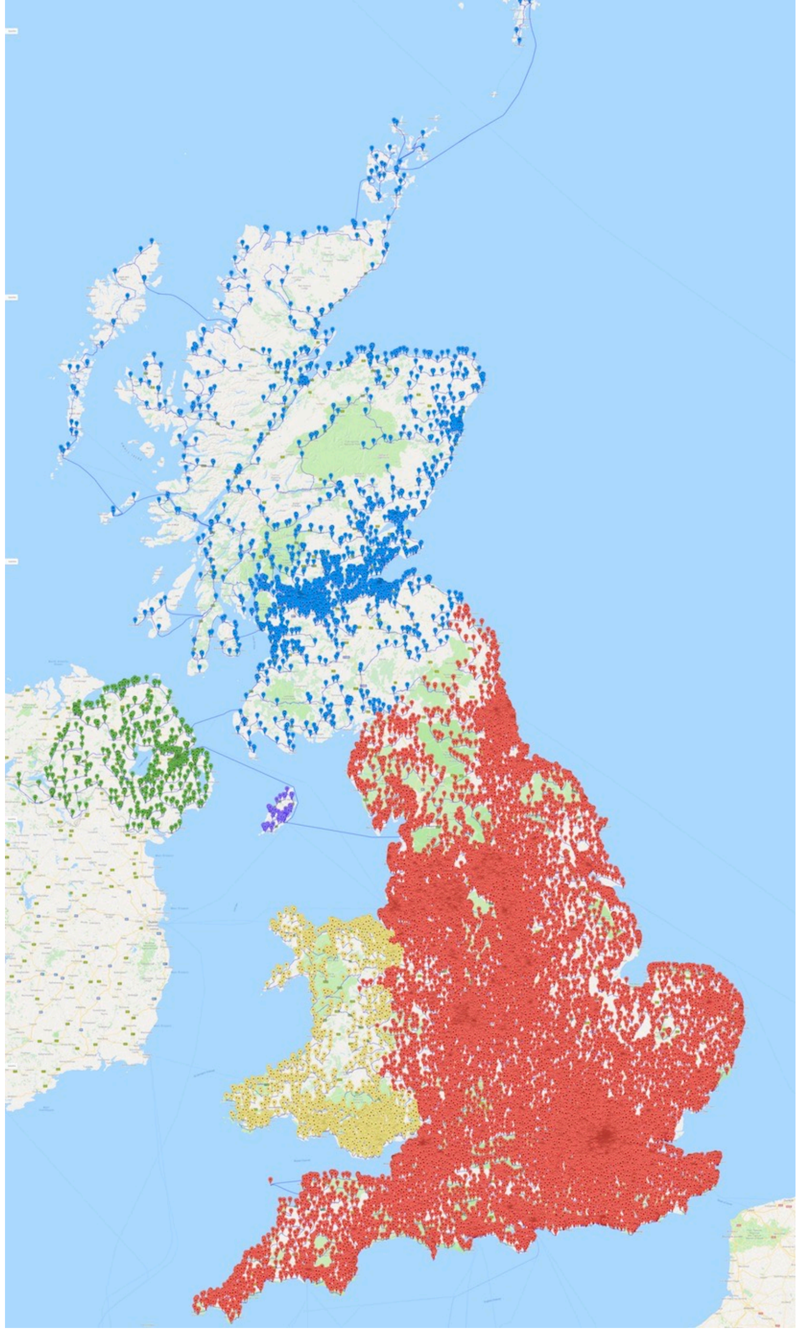


u, y, v, w, z, u is a Hamiltonian cycle   u, y, v, w, z, u is a traveling-salesman tour with cost 0

TSP arts and challenges



Mona Lisa TSP: $1,000 Prize for a 100,000-city challenge problem
http://www.math.uwaterloo.ca/tsp/



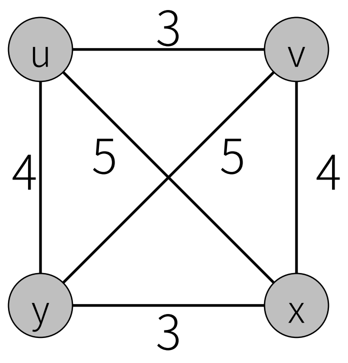
Shortest possible tour to nearly every (49687) pub in the United Kingdom
https://www.math.uwaterloo.ca/tsp/uk/

Traveling Salesman Problem: Approximation for Metric TSP

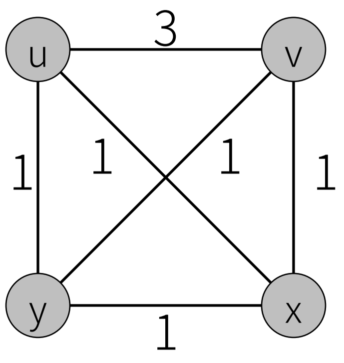
Metric TSP

- Optimization problem: Given a set of cities and their pairwise distances, find a tour of lowest cost that visits each city exactly once, and the pairwise distances satisfy triangle inequality.
  - Triangle inequality: $\forall u, v, w \in V, d(u, w) \leq d(u, v) + d(v, w)$.

Satisfy triangle inequality



Do not satisfy triangle inequality



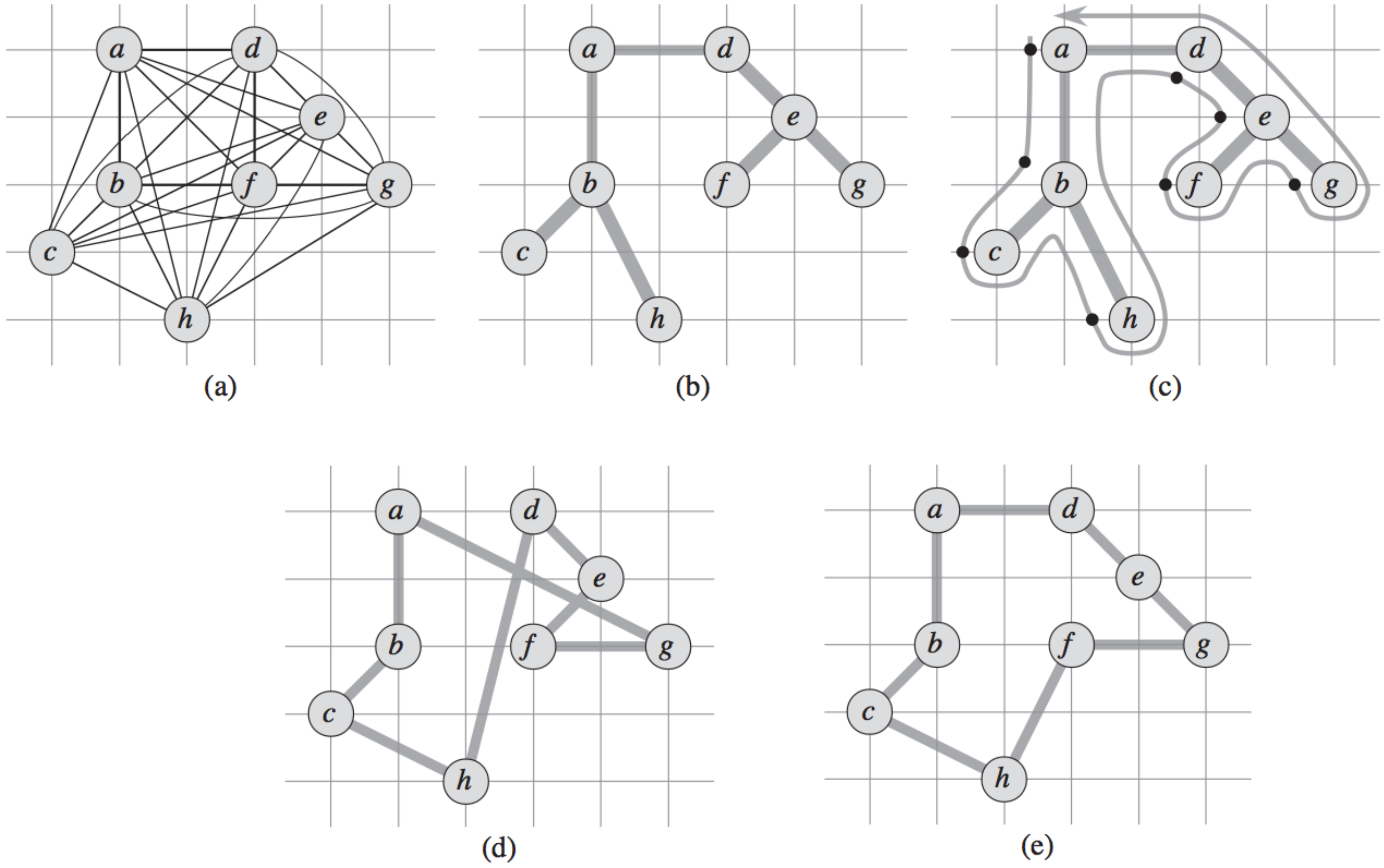
Show that Metric TSP is also in NPC

Hint: reduce from either HAM-CYCLE or general TSP; make sure the metric TSP instance satisfies the triangle inequality

2-approximation algorithm for Metric TSP

```
APPROX-TSP-TOUR(G)
1. select a vertex r ∈ G.V to be a "root" vertex
2. grow a minimum spanning tree T for G
3. let H be the list of vertices visited in a preorder tree walk of T
4. return H
```

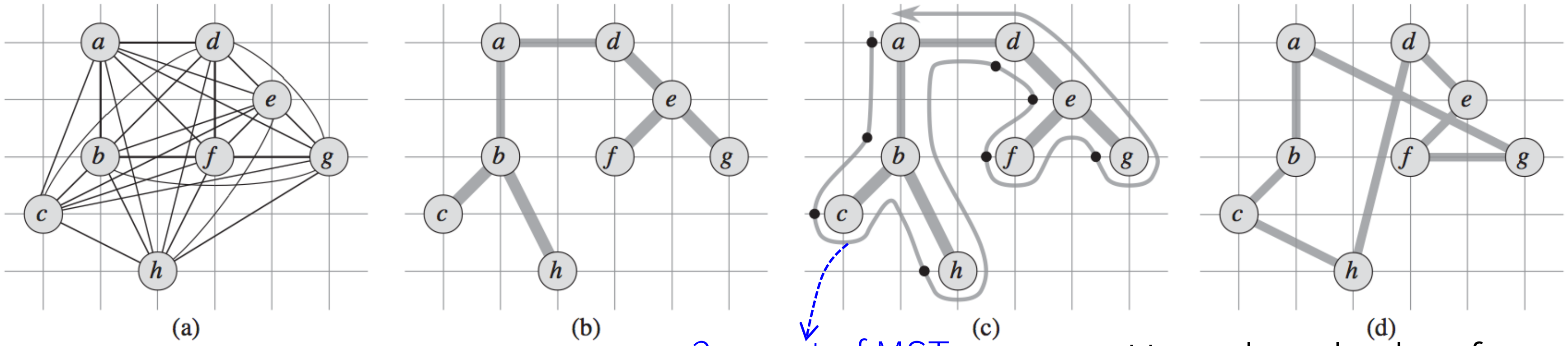
- Running time is dominated by finding a MST
  - MST is in P: $O(V^2)$ when using adjacency matrix
- Claim: Approximation ratio $\rho(n) = 2$



H = a, b, c, h, d, e, f, g, a

OPT H^* = a, b, c, h, f, g, e, d, a

2-approximation algorithm for Metric TSP



$2 \times$ cost of MST

H = a, b, c, h, d, e, f, g, a

- Let H^* denote an optimal tour, H the TSP tour found by the algorithm, T^* a MST
- With triangle inequality, immediately we have $w(H) \leq 2w(T^*)$
- Also, H^* is formed by some tree T plus some edge e, i.e., $w(H^*) = w(T) + w(e)$
- $\Rightarrow w(T^*) \leq w(H^*)$
- Combining $w(H) \leq 2w(T^*)$ and $w(T^*) \leq w(H^*)$, we have $w(H) \leq 2w(T^*) \leq 2w(H^*)$
- $\Rightarrow \rho(n) = 2$

Traveling Salesman Problem: Inapproximability of General TSP

Theorem 35.3 General TSP (when triangle inequality may not hold)

If P $\neq$ NP, there is no polynomial-time approximation algorithm with a constant ratio bound ρ for the general TSP.

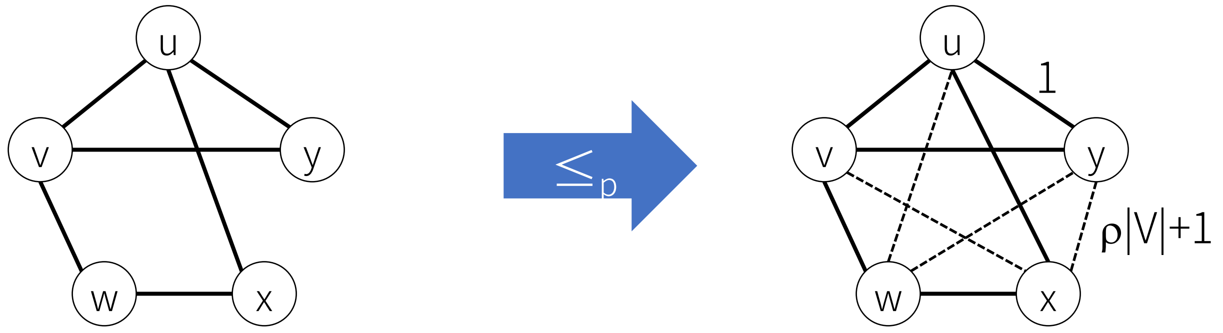
Proof by contradiction

- Suppose there is such an algorithm Alg_{AT} to approximate TSP with a constant ρ. We will use Alg_{AT} to construct Alg_{HC} to solve HAM-CYCLE in polynomial time.
- Consider the following reduction algorithm f converting an instance of HAM-CYCLE α into an instance of TSP $f(\alpha)$
  - $\alpha = \{G = (V, E)\}; f(\alpha) = \{G' = (V, E'), w, k = \rho|V|\}$
  - That is, we construct a TSP instance with a complete graph $G' = (V, E')$, where $w(u, v) = 1$ if $(i, j) \in E$; $w(u, v) = \rho|V| + 1$, otherwise.
  - Run Alg_{AT} on $f(\alpha)$
  - If $Alg_{AT}\big(f(\alpha)\big)$ returns a tour whose cost $\leq \rho|V|$, then $Alg_{HC}(\alpha) = 1$ (i.e., G contains a Hamiltonian cycle); otherwise, $Alg_{HC}(\alpha) = 0$.

Proof by contradiction (cont'd)

- Correctness of reduction
  - If G has an HC: G' contains a tour of cost $|V|$ by picking edges in E, each with cost of 1. Then, Alg_{AT} guarantees to return a tour whose cost ≤ $\rho|V|$.
  - If G' has a tour whose cost ≤ $\rho|V|$: the tour must consist of edges in E only, and thus it's also a HC in G

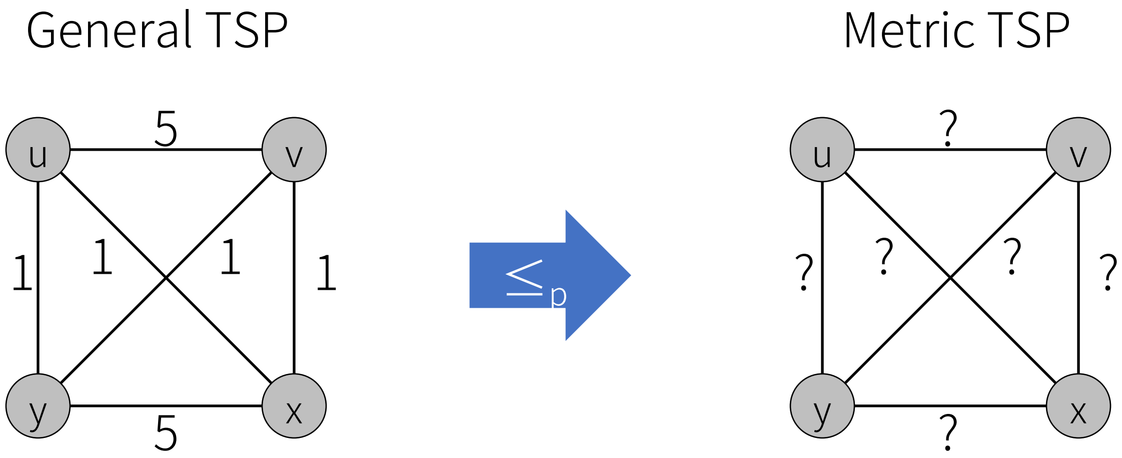
⇒ Alg_{HC} can solve HAM-CYCLE in polynomial time, contradiction!



u, y, v, w, x, u is a Hamiltonian Cycle

u, y, v, w, x, u is a traveling-salesman tour with cost |V|

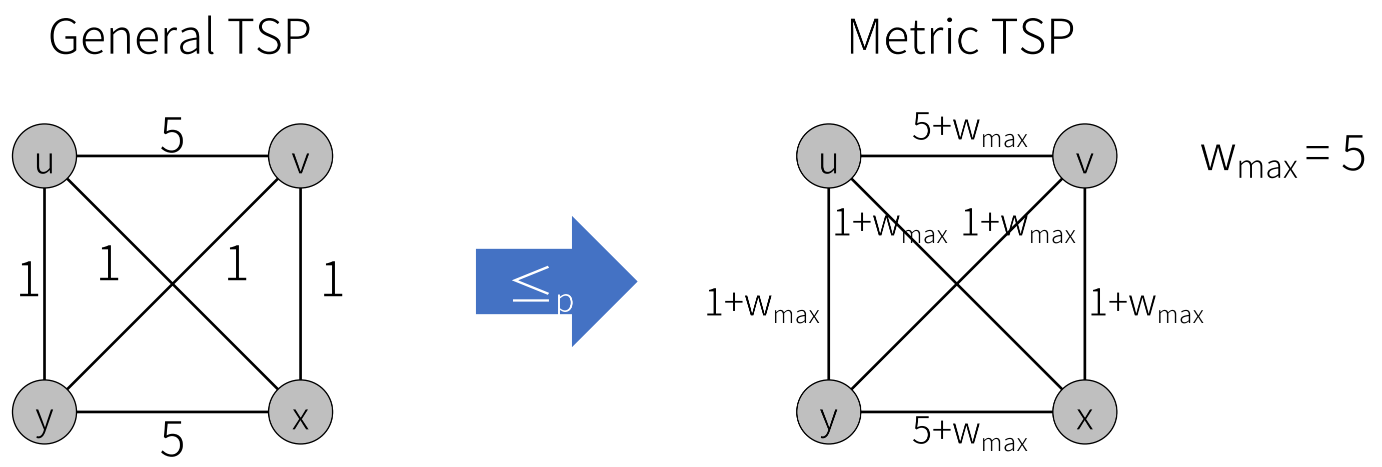
Exercise 35.2-2 Show how in polynomial time we can transform one instance of the traveling-salesman problem into another instance whose cost function satisfies the triangle inequality. The two instances must have the same set of optimal tours. Explain why such a polynomial-time transformation does not contradict Theorem 35.3, assuming that P≠NP.



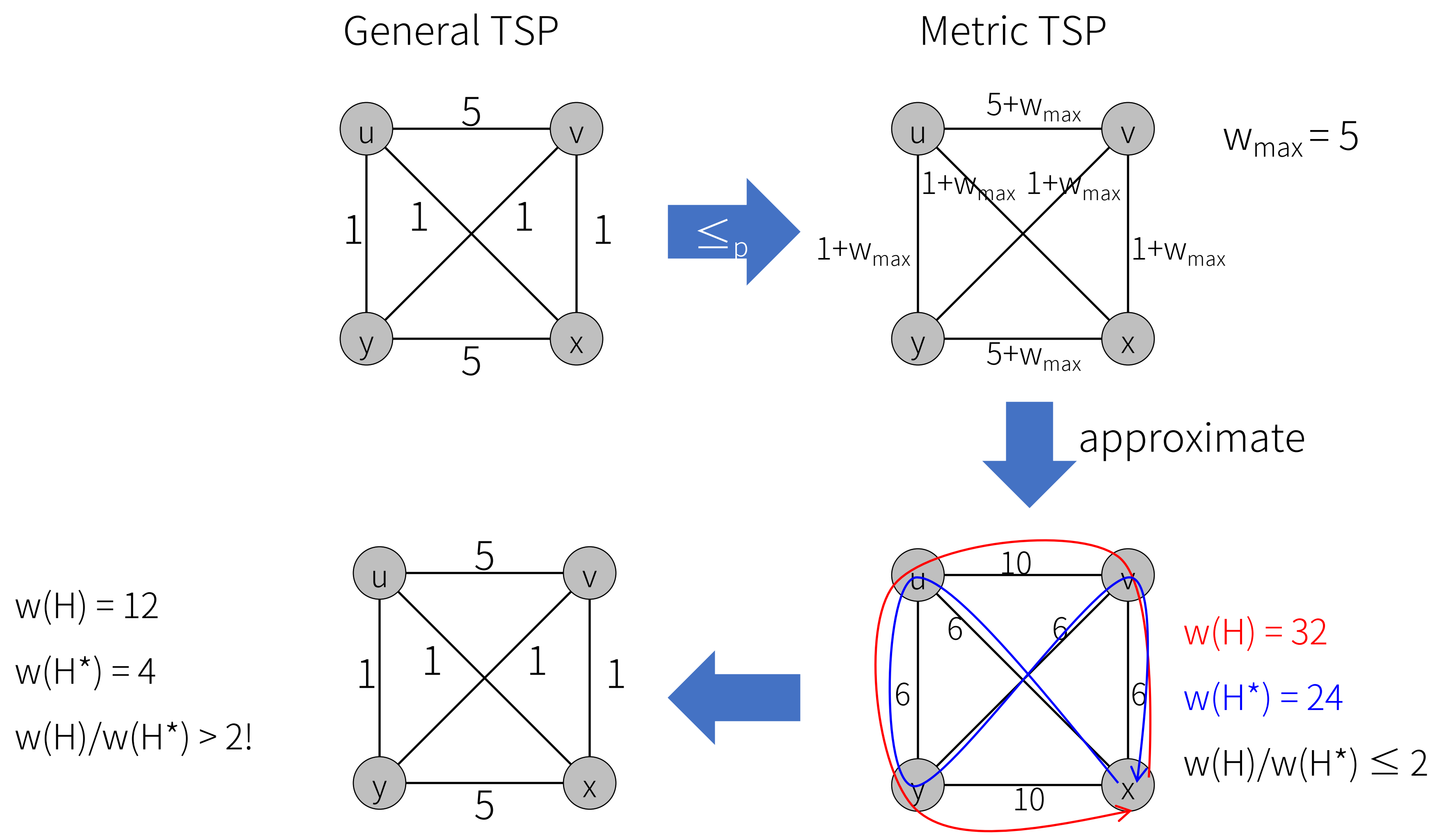
Exercise 35.2-2

- For example, we can add w_{max} (the largest cost) to each edge
- G contains a tour of minimum cost k ⇔ G' contains a tour of minimum cost $k + w_{max} * |V|$
- G' satisfies triangle inequality because $\forall t, u, v \in V$,

$$w'(u,v) = w(u,v) + w_{max} \leq 2 * w_{max} \leq w'(t,u) + w'(t,v)$$



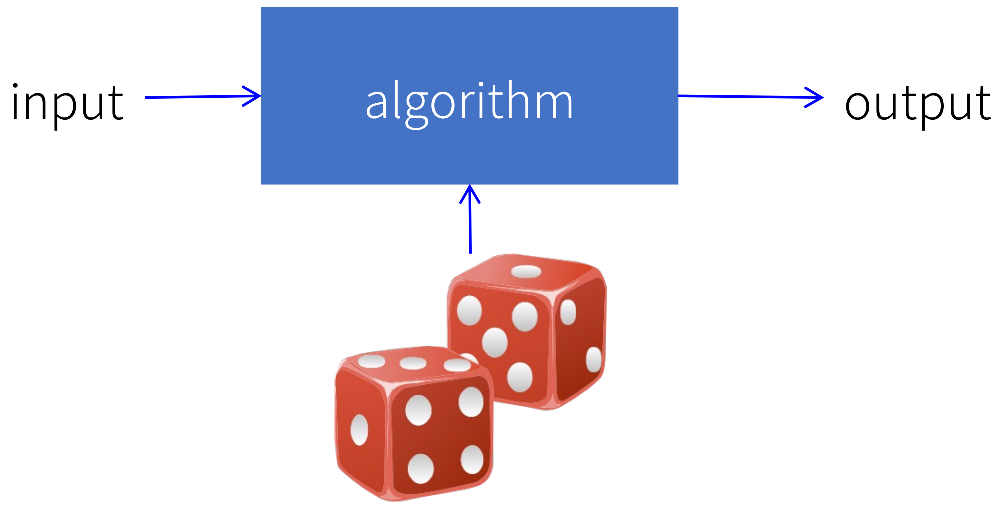
Exercise 35.2-2



Randomized Approximation Algorithms

Randomness

- A randomized algorithm is an algorithm that employs a degree of randomness as part of its logic
- A randomized data structure is a data structure that employs a degree of randomness as part of its logic
- A randomized algorithm's behavior is determined not only by its input but also by values produced by a random-number generator



Randomized approximation algorithm

	Exact	Approximate
Deterministic	Prim's (MST)	APPROX-TSP-TOUR
Randomized	Quick Sort	MAX-3-CNF-SAT MAX-CUT

MAX-3-CNF Satisfiability

- **3-CNF-SAT**: Satisfiability of Boolean formulas in 3-conjunctive normal form (3-CNF)
  - 3-CNF = AND of clauses, each is the OR of exactly 3 distinct literals
  - A literal is an occurrence of a variable or its negation, e.g., x_1 or $\neg x_1$
- 3-CNF-SAT is a decision problem. What should be an optimization version of 3-CNF-SAT?

MAX-3-CNF Satisfiability

- MAX-3-CNF-SAT: find an assignment of the variables that satisfies as many clauses as possible
  - Closeness to optimum is measured by the fraction of satisfied clauses
- Can you design a randomized 8/7-approximation algorithm?

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

$\langle x_1, x_2, x_3, x_4 \rangle = \langle 0, 0, 1, 1 \rangle$ satisfies 3 clauses
$\langle x_1, x_2, x_3, x_4 \rangle = \langle 1, 0, 1, 1 \rangle$ satisfies 2 clauses

Randomized 8/7-approximation algorithm for MAX-3-CNF-SAT

- A randomized 8/7-approximation algorithm:
  - 丟硬幣決定變數要設成 0 或是 1
  - That's it!

Theorem 35.6

Given an instance of MAX-3-CNF-SAT with n variables $x_1, x_2, \dots, x_n$ and m clauses, the randomized algorithm that independently sets each variable to 1 with probability 1/2 and to 0 with probability 1/2 is a randomized 8/7-approximation algorithm.

* Satisfying 7/8 of the clauses in expectation

Theorem 35.6

Given an instance of MAX-3-CNF-SAT with n variables $x_1, x_2, \ldots, x_n$ and m clauses, the randomized algorithm that independently sets each variable to 1 with probability 1/2 and to 0 with probability 1/2 is a randomized 8/7-approximation algorithm.

Proof

- A clause that contains both a variable and its negation is always evaluated to 1
- The rest of the clauses is the OR of exactly 3 distinct literals, and no variable and its negation appear at the same time
  - $\Pr[x_i = 0] = \Pr[x_i = 1] = 1/2$
  - $\Rightarrow$ for all $x_1 \neq x_2 \neq x_3$, $\Pr[(x_1 \vee x_2 \vee \neg x_3) = 0] = 1/8$
- $\Rightarrow$ E[# of satisfied clauses] = m * E[clause j is satisfied] $\geq m * (1 - \frac{1}{8}) = \frac{7}{8}m$
- $\Rightarrow$ $\rho(n)$ = max # of satisfied clauses / E[# of satisfied clauses] $\leq$ 8/7

MAX-CUT

- Optimization problem: Given an unweighted undirected graph $G = (V, E)$, find a cut whose size is maximized
  - A cut partitions V into V_0 and V_1; a cut consists of the edges across the partition
- Decision problem: Given an unweighted undirected graph $G = (V, E)$, there exists a cut whose size is k
- MAX-CUT problem is NP-complete
  - C.f. MIN-CUT is in P
- Can you design a randomized 2-approximation algorithm?

Randomized 2-approximation algorithm for MAX-CUT

- Randomly assign each vertex to either V_1 and V_2 with equal probablity
- Done!

Proof

- Let C be the cut found by the algorithm; $C^*$is the maximum cut
- For any edge $e = (u, v)$ on G, the probability that $e \in C$ is ½
- Let x_e be an indicator variable for event $e \in C$; that is, $x_e = 1$ if $e \in C$, otherwise, 0.
- => $\mathrm{E}[|C|] = E[\Sigma_{e \in E} x_e] = \Sigma_{e \in E} E[x_e]$

$$= \Sigma_{e \in E}(Pr[e \in C] * 1 + Pr[e \notin C] * 0) = \frac{|E|}{2} \geq \frac{|C^*|}{2}$$

Suppose vertices are numbered from 1 to n, show that the following greedy algorithm achieves 2-approximation max cut:

1. Initially $V_0^1 = \{1\}, \mathrm{V}_1^1 = \{\}$
2. Adding the ith vertex to the subset that results in a better cut, say V_x: $V_x^i = V_x^{i-1} \cup \{i\}, V_{1-x}^i = V_{1-x}^{i-1}$
3. Output V_0^n, V_1^n

Hint: consider the edges introduced by adding the ith vertex