CSE 6140/ CX 4140:

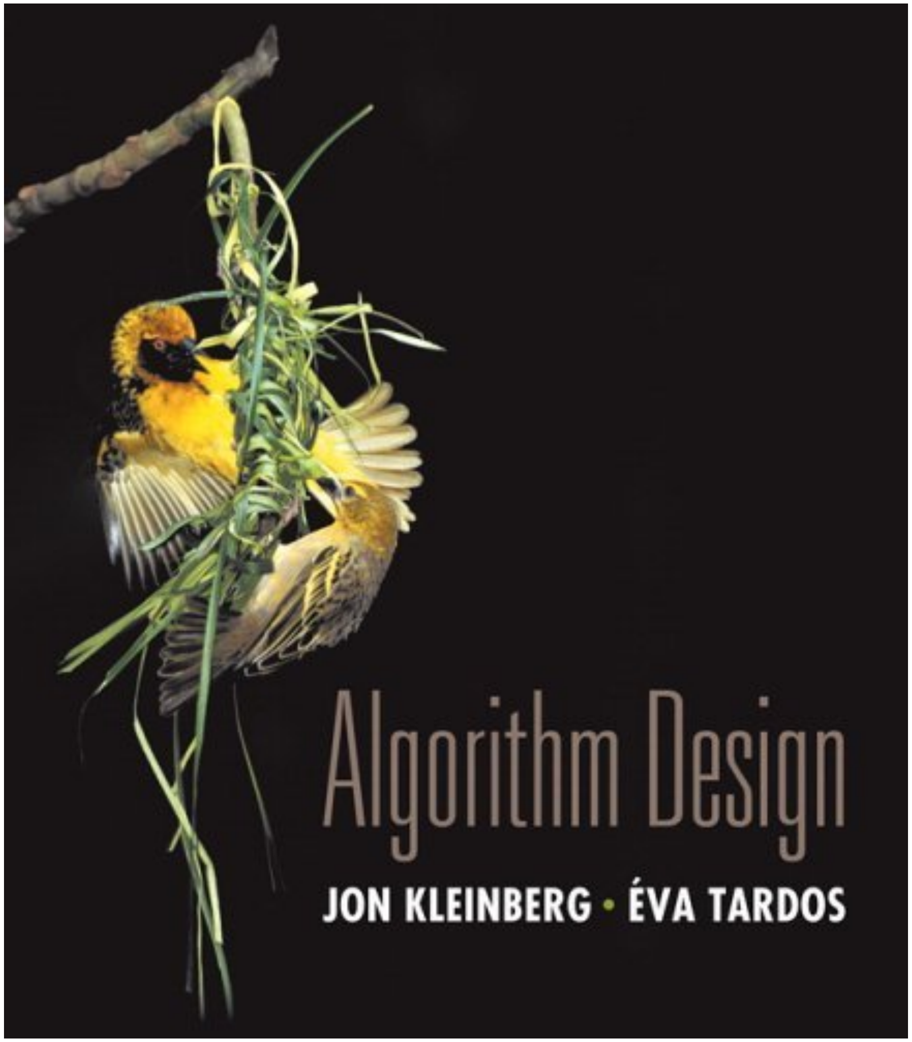
Computational Science and Engineering ALGORITHMS

Instructor: Anne Benoit

Visiting Associate Professor, CSE

Based on slides by Bistra Dilkina

KT 11.2 Clustering

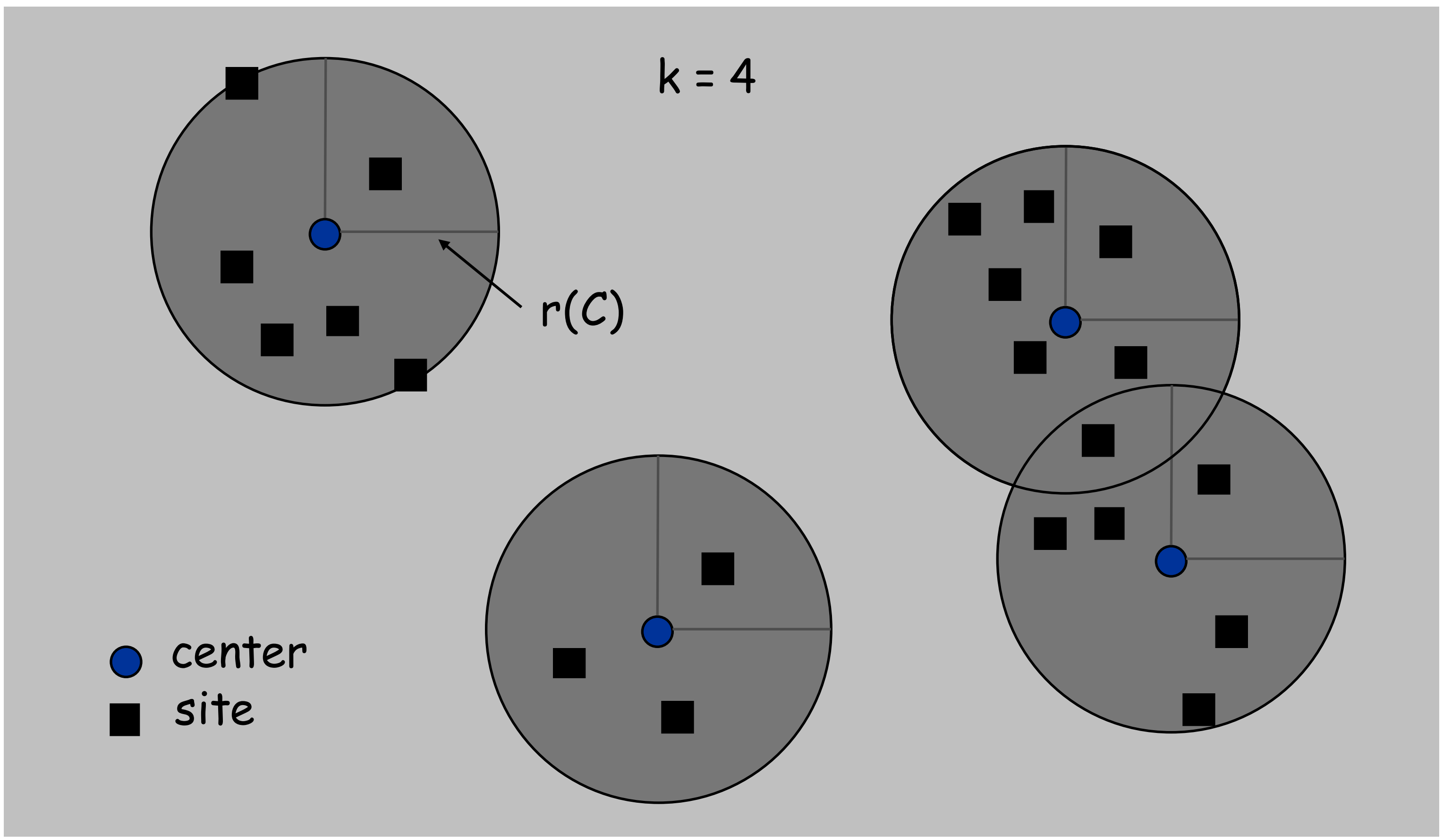


Center Selection Problem

Input. Set of n sites $s_1, \ldots, s_n$ and integer $k > 0$.

Center selection problem. Select k centers C so that maximum distance from a site to nearest center is minimized.

Application: where to put the branch offices w.r.t. clients?



Center Selection: Analysis of Greedy Algorithm

Theorem. Let C^* be an optimal set of centers. Then $r(C) \leq 2r(C^*)$.

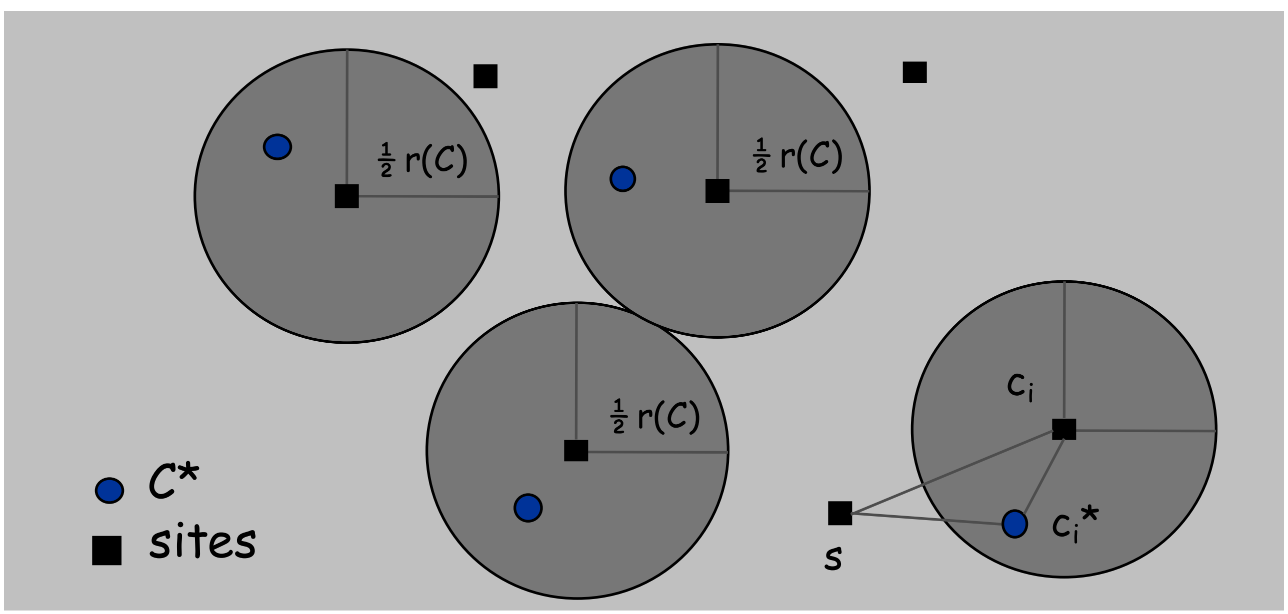
Pf. (by contradiction)

Assume $r(C) > 2r(C^*)$, i.e. $r(C^*) < \frac{1}{2}\, r(C)$.

- For each center c_i in C, consider ball of radius $\frac{1}{2}\, r(C)$ around it.
- $\text{dist}(c_i, C^*) \leq r(C^*) < \frac{1}{2}\, r(C)$, so <u>at least</u> one c_i^* in each ball in C

By definition of radius

By our assumption

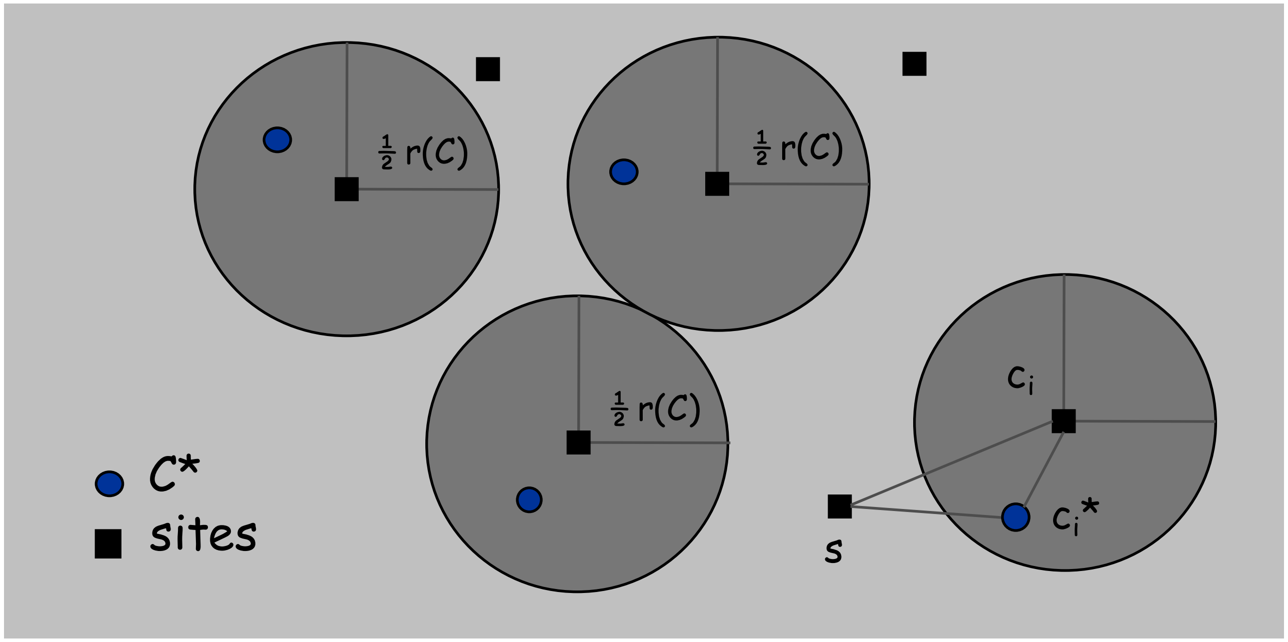


Center Selection: Analysis of Greedy Algorithm

Theorem. Let C^* be an optimal set of centers. Then $r(C) \le 2r(C^*)$.

Pf. (by contradiction) Assume $r(C) > 2r(C^*)$, i.e. $r(C^*) < \frac{1}{2} r(C)$.

- at least one c_i^* in each ball in C
- Every pair of c_i's in C are at least $r(C)$ apart (by alg.), so each ball around a c_i in C does not intersect with any other ball
- Each ball has at least one c_i^* and $|C|=|C^*|=k$, so at most one c_i^* in each ball
- Therefore exactly one c_i^* in each ball



Center Selection: Analysis of Greedy Algorithm

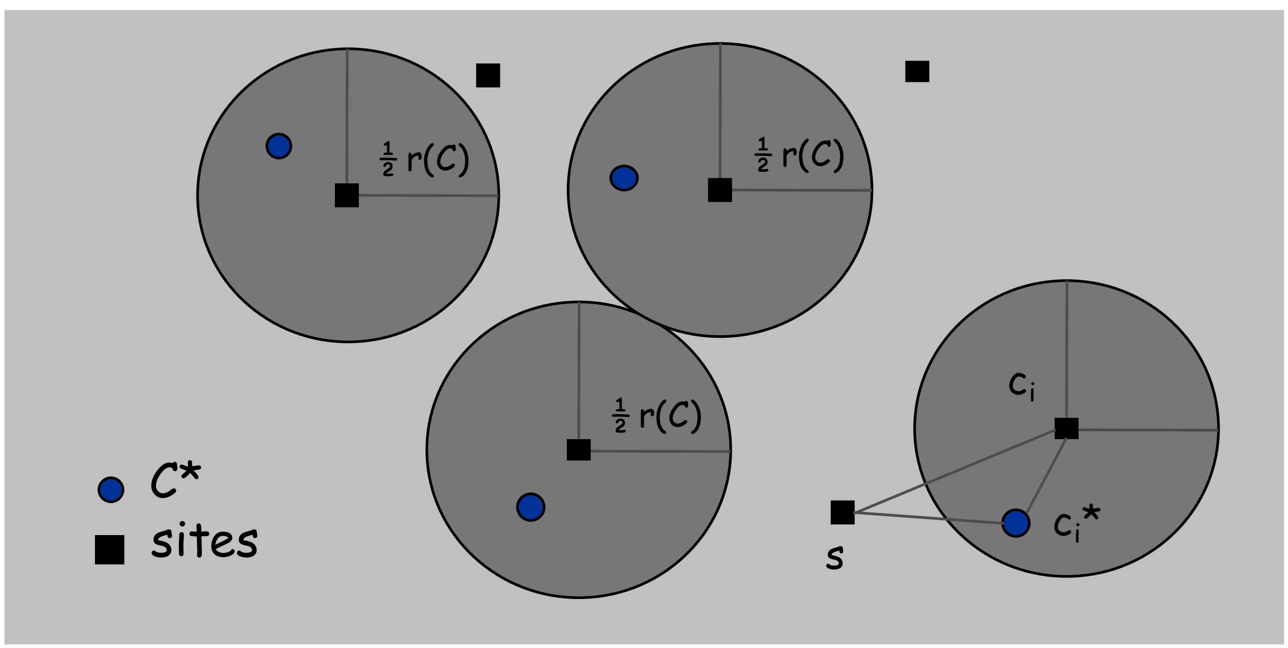
Theorem. Let C^* be an optimal set of centers. Then $r(C) \le 2r(C^*)$.

Pf. (by contradiction) Assume $r(C) > 2r(C^*)$, i.e. $r(C^*) < \frac{1}{2} r(C)$.

- exactly one c_i^* in each ball in C ; let c_i be the site paired with c_i^*
- Consider **any** site s and its closest center c_i^* in C^*.
- $\underbrace{dist(s, C)}_{\text{min across all } c_i} \le \underbrace{dist(s, c_i) \le dist(s, c_i^*) + dist(c_i^*, c_i)}_{\Delta\text{-inequality}} \le 2r(C^*)$.

  $\le r(C^*)$ since c_i^* is closest center to both s and c_i

- True for any site s including the one that has $dist(s,C)=r(C)$
- Thus $r(C) \le 2r(C^*)$, this is a contradiction with our assumption ▪



Center Selection

Theorem. Let C^* be an optimal set of centers. Then $r(C) \leq 2r(C^*)$.

Theorem. Greedy algorithm is a 2-approximation for center selection problem.

Remark. Greedy algorithm always places centers at sites, but is still within a factor of 2 of best solution that is allowed to place centers anywhere.

e.g., points in the plane

Theorem. There is no better approximation algorithm (show next).

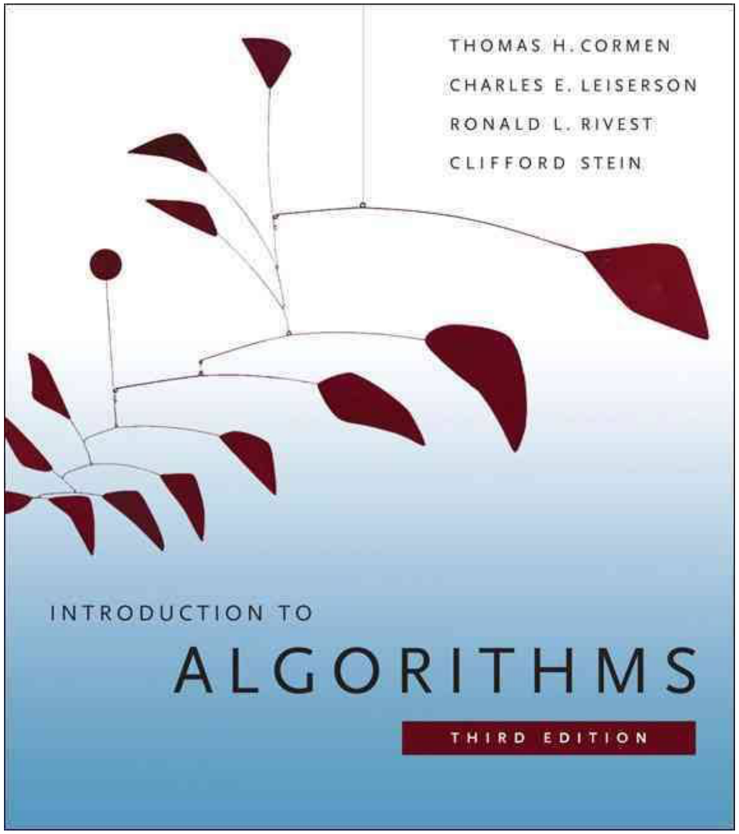
Center Selection: Hardness of Approximation

Theorem. Unless P = NP, there is no ρ-approximation algorithm for metric k-center problem for any $\rho < 2$.

Pf. We show how we could use a $(2 - \varepsilon)$ approximation algorithm for k-center to solve DOMINATING-SET in poly-time.

- DOMINATING-SET: Given a graph G, is there a set of vertices U of size at most k such that every other vertex has a neighbor in U
- Let [G = (V, E), k] be an instance of DOMINATING-SET.
- Construct instance [G',k'=k,r=1] of k-CENTER with sites V and distances
  - $d(u, v) = 1$ if $(u, v) \in E$
  - $d(u, v) = 2$ if $(u, v) \notin E$
- Note that G' satisfies the triangle inequality.
- Claim: G has dominating set of size k iff there exists k centers C^* with $r(C^*) = 1$ in G'. (how do we show this?)
- Thus, if G has a dominating set of size k, a $(2 - \varepsilon)$-approximation algorithm on [G',k] must find a solution C^* with $r(C^*) = 1$ since it cannot use any edge of distance 2.

TRAVELING SALESMAN PERSON (TSP) – [CLRS 35.2]



Types of TSP

TSP: Given a complete graph G with nodes V and edge cost c(u,v) defined for every pair of nodes, find the shortest simple cycle that visits all nodes in V.

General TSP: No restrictions on the cost function.

Metric TSP: All edge cost are symmetric and fulfill the triangle inequality:

$$c(u,v) \leq c(u,w) + c(w,v), \ \forall u,v,w \in V$$

Euclidean TSP: The vertices correspond to points in a d-dimensional space, and the cost function is the Euclidean distance.
The Euclidean distance between two points $x = (x_1, x_2, \ldots, x_d)$ and $y = (y_1, y_2, \ldots, y_d)$ is

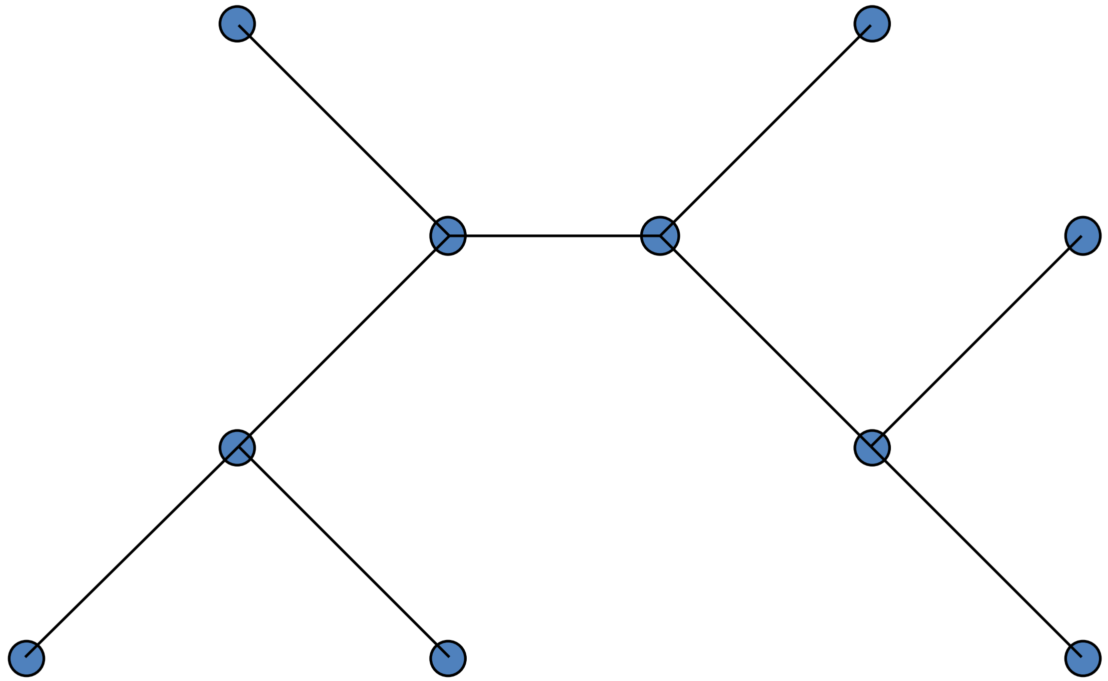
$$\left(\sum_{i=1}^{d} (x_i - y_i)^2 \right)^{1/2}$$

Spanning Tree and TSP

Strategy: Construct the TSP tour from a minimum spanning tree.

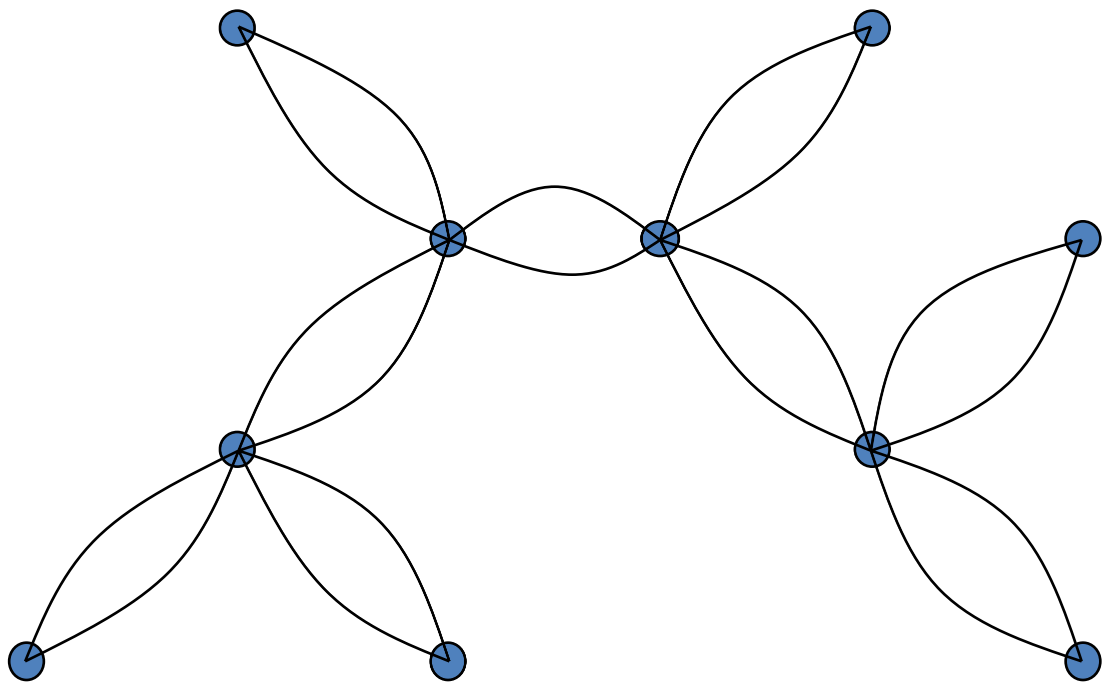
Use the edges in the minimum spanning tree as much as possible but still keeping the tour a simple cycle.

How to formalize the idea of "following" a minimum spanning tree?



Spanning Tree and TSP

How to formalize the idea of "following" a minimum spanning tree?

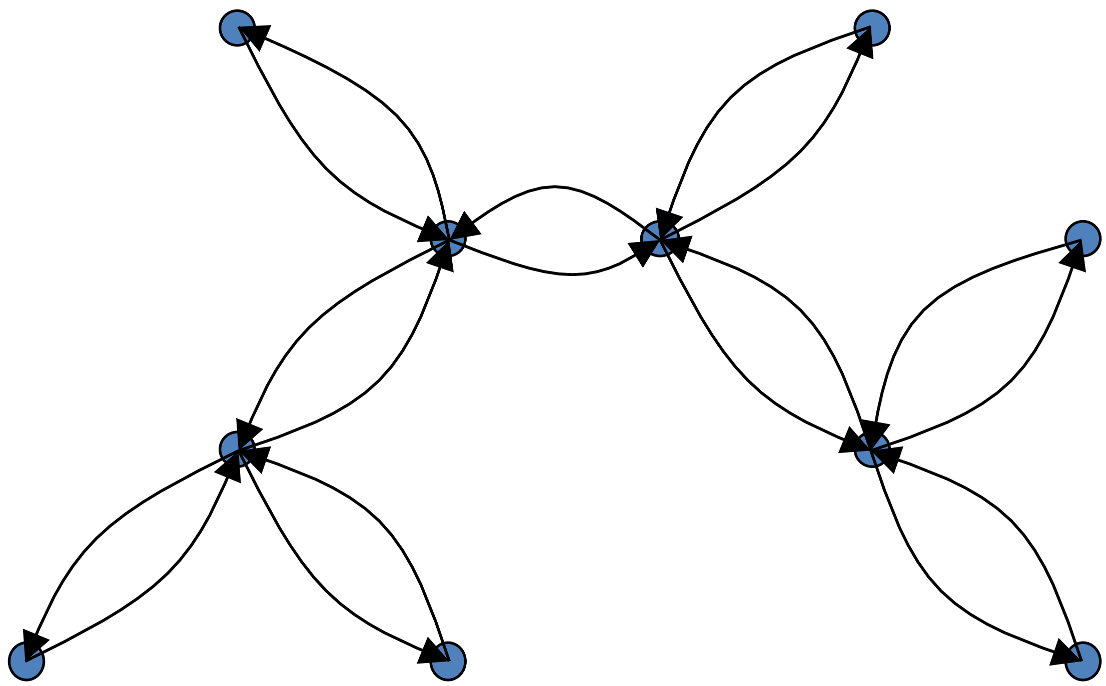


Key idea: double all the edges and find an Eulerian tour.

This graph has cost 2MST.

Spanning Tree and TSP

How to formalize the idea of "following" a minimum spanning tree?

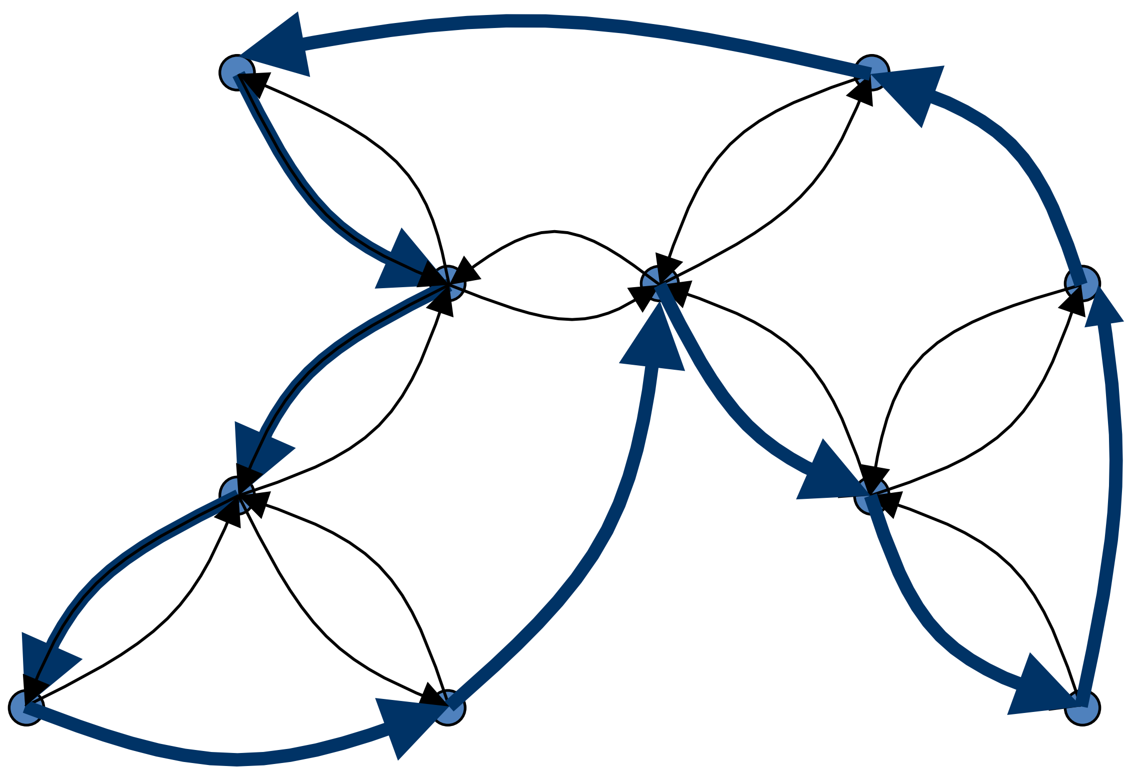


Key idea: double all the edges and find an Eulerian tour

This graph has cost 2MST.

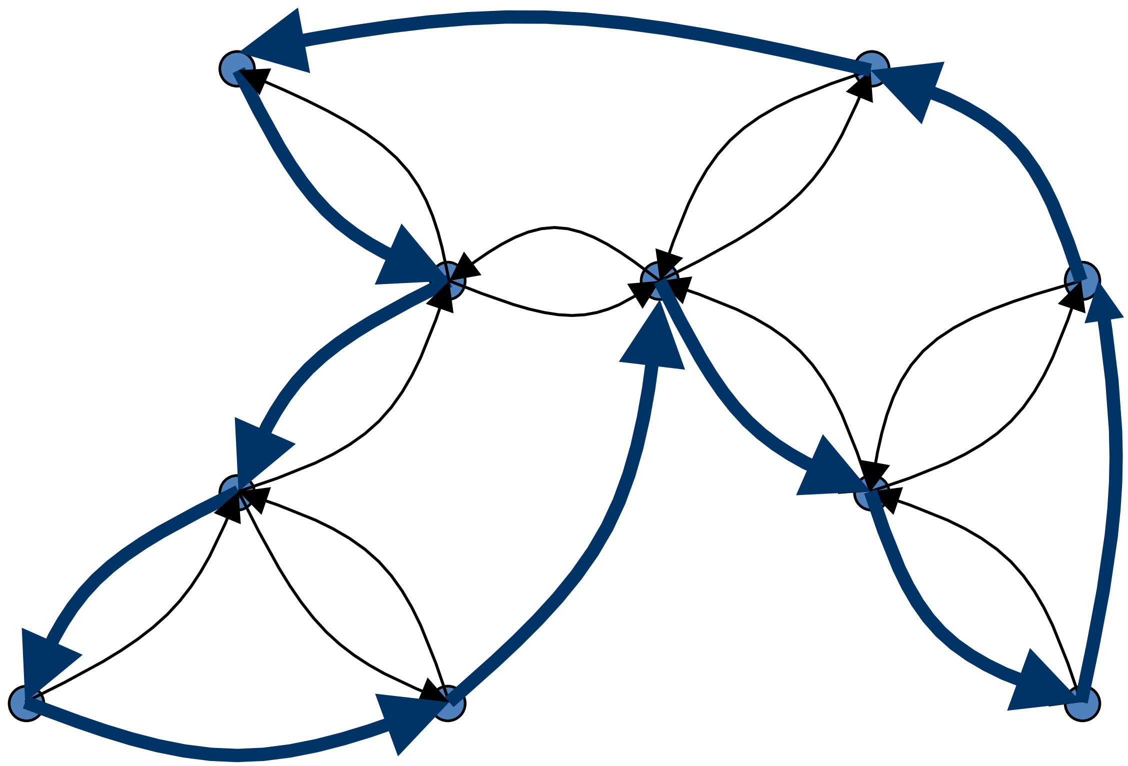
Spanning Tree and TSP

Strategy:
Choose a root node.
Follow MST along the Eulerian tour.
Only add a vertex as part of the TSP tour the first time it is encountered, i.e., shortcut this Eulerian tour whenever there are repeated vertices.



Spanning Tree and TSP

By **triangle inequalites**, the shortcut tour is <u>no longer</u> than the Eulerian tour.



Each directed edge is used/shortcuted exactly once in the TSP tour.

2-Approximation Algorithm for Metric TSP

Georgia Tech

(Metric TSP – Factor 2)

1. Find an MST, **T**, of G.
2. Double every edge of the MST to obtain an Eulerian graph.
3. Pick a root, find an Eulerian tour, **T***, on this graph.
4. Output the tour that visits vertices of G in the order of their first appearance in T*. Let **C** be this tour.

   (That is, shortcut T*)

Analysis:

1. cost(T) ≤ OPT (because MST is a lower bound of TSP)
2. cost(T*) = 2cost(T) (because every edge appears twice)
3. cost(C) ≤ cost(T*) (because of triangle inequalities, **shortcutting**)
4. So, cost(C) ≤ cost(T*) = 2cost(T) ≤ 2OPT

Approximation Algorithms for TSP

APPROX-TSP-TOUR(G)

Find an MST T;

Choose a vertex as root r;

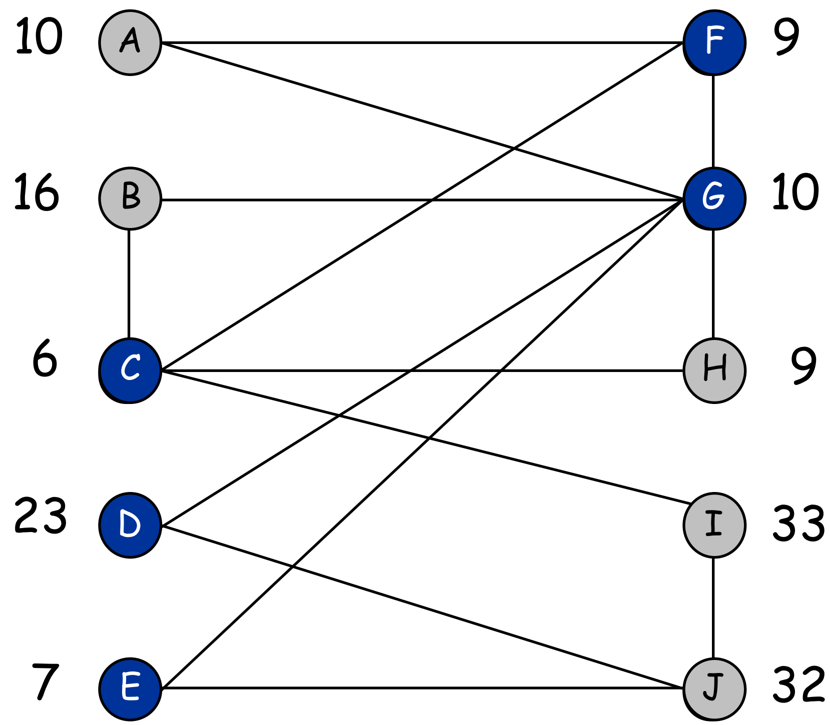
return preorderTreeWalk(T, r);

- preorderTreeWalk(T,r)
  - Depth first search in the tree, and output each node the first time that you enter it
  - Exactly the same order as the Eulerian tour and the shortcutting
- For any constant $\lambda \geq 1$, there does not exist any λ-approx algo for TSP unless P=NP.
- There is a 1.5 approximation algorithm for Metric TSP.
- There is a $(1+\varepsilon)$ approximation for Euclidean TSP (PTAS) .
  - Distances follow triangle ineq., and further follow Euclidean dist. metric

11.6 LP Rounding: Vertex Cover

Weighted Vertex Cover

Weighted vertex cover. Given an undirected graph $G = (V, E)$ with vertex weights $w_i \geq 0$, find a minimum weight subset of nodes S such that every edge is incident to at least one vertex in S.



total weight = 55

Weighted Vertex Cover: ILP Formulation

Weighted vertex cover. Given an undirected graph $G = (V, E)$ with vertex weights $w_i \geq 0$, find a minimum weight subset of nodes S such that every edge is incident to at least one vertex in S.

Integer linear programming formulation.

- Model inclusion of each vertex i using a 0/1 variable x_i.

$$x_i = \begin{cases} 0 & \text{if vertex } i \text{ is not in vertex cover} \\ 1 & \text{if vertex } i \text{ is in vertex cover} \end{cases}$$

- Vertex covers in 1-1 correspondence with 0/1 assignments: $S = \{i \in V : x_i = 1\}$

- Objective function: minimize $\Sigma_i w_i x_i$.

- Must take either i or j for each edge (i,j) in E: $x_i + x_j \geq 1$.

Weighted Vertex Cover: ILP Formulation

Weighted vertex cover. Integer linear programming (ILP) formulation.

$$
\begin{array}{llllll}
(ILP) & \min & \sum_{i \in V} w_i \, x_i & & & \\
& \text{s. t.} & x_i + x_j & \geq & 1 & (i,j) \in E \\
& & x_i & \in & \{0,1\} & i \in V
\end{array}
$$

Observation. If x^* is optimal solution to (ILP), then $S = \{i \in V : x^*_i = 1\}$ is a min weight vertex cover.

Integer Linear Programming

INTEGER-LINEAR-PROGRAMMING. Given integers a_{ij}, b_i, and c_i (parameters), find integers x_j (variables) that satisfy:

$$
\begin{array}{lrcl}
\min & c^t x & & \\
\text{s.t.} & Ax & \geq & b \\
 & x & & \text{integral}
\end{array}
$$

vector/matrix notation

$$
\begin{array}{rcll}
\min \displaystyle\sum_{j=1}^{n} c_j x_j & & & \\
\displaystyle\sum_{j=1}^{n} a_{ij} x_j & \geq & b_i & 1 \leq i \leq m \\
x_j & \geq & 0 & 1 \leq j \leq n \\
x_j & & \text{integral} & 1 \leq j \leq n
\end{array}
$$

Observation. Vertex cover formulation proves that integer linear programming is an NP-hard search problem.

even if all coefficients are 0/1 and
at most two variables per inequality

Recipe. Determine the variables, write the objective function, write the constraints. Distinguish variables from parameters.

MILP for Maximum Satisfiability

Goal: Find a truth assignment to satisfy all clauses

$$(x_1 \vee x_2 \vee x_3) \wedge \ldots \wedge (x_3 \vee \overline{x_4} \vee \overline{x_1})$$

$$\begin{aligned} x_1 + x_2 + x_3 &\geq 1 \\ x_3 + (1 - x_4) + (1 - x_1) &\geq 1 \\ x_i &= \{0, 1\} \end{aligned}$$

MILP for Knapsack

KNAPSACK: Given a finite set X, nonnegative weights w_i, nonnegative values v_i, a weight limit W, find a subset $S \subseteq X$ such that the value of S is maximum.

$$\max \sum_{i=1..n} v_i x_i$$

$$\sum_{i=1..n} w_i x_i \leq W$$

$$x_i \in \{0,1\}, \text{ for } i = 1..n$$

Linear Programming

Linear programming. Max/min linear objective function subject to linear inequalities (constraints).

- Input: parameters c_j, b_i, a_{ij} .
- Output (variables): real numbers x_j.

$$\begin{array}{llrcl} (P) & \min & c^t x & & \\ & \text{s.t.} & Ax & \geq & b \\ & & x & \geq & 0 \end{array}$$

$$\begin{array}{llrcll} (P) & \min & \sum_{j=1}^{n} c_j x_j & & & \\ & \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i & 1 \leq i \leq m \\ & & x_j & \geq & 0 & 1 \leq j \leq n \end{array}$$

Linear. No x^2, xy, $\arccos(x)$, $x(1-x)$, etc.

Simplex algorithm. [Dantzig 1947] Can solve LP in practice.

Ellipsoid algorithm. [Khachian 1979] Can solve LP in poly-time.

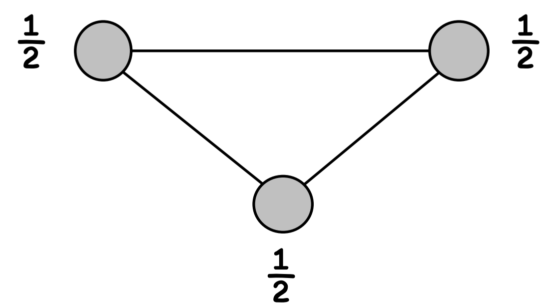
Weighted Vertex Cover: LP Relaxation

Weighted vertex cover. Linear programming formulation.

$$
\begin{array}{llllll}
(LP) & \min & \sum_{i \in V} w_i x_i & & & \\
& \text{s. t.} & x_i + x_j & \geq & 1 & (i,j) \in E \\
& & x_i & \geq & 0 & i \in V
\end{array}
$$

Observation. Optimal value of (LP) is ≤ optimal value of (ILP).
Pf. LP has fewer constraints. Any solution to ILP is also solution to LP

Note. LP is not equivalent to vertex cover.



Q. How can solving LP help us find a small vertex cover?
A. Solve LP and round fractional values: x_i>=1/2 become 1, x_i < ½ become 0

Weighted Vertex Cover

Theorem. If x^* is optimal solution to (LP), then $S = \{i \in V : x^*_i \geq \frac{1}{2}\}$ is a vertex cover whose weight $\Sigma_{i \in S} w_i$ is at most twice OPT(Vertex Cover).

Pf. [S is a vertex cover]

- Consider an edge $(i, j) \in E$.
- Since $x^*_i + x^*_j \geq 1$, either $x^*_i \geq \frac{1}{2}$ or $x^*_j \geq \frac{1}{2}$ $\Rightarrow$ (i, j) covered.

Pf. [S has desired cost, $w(S) <= 2w(S^{VCOPT})$]

- Let S^{VCOPT} be optimal vertex cover. Corresponds to a soln of LP with x_i=1 if i in S^{VCOPT}, and 0 otherwise. Then

$$w(S^{VCOPT}) = \sum_{i \in S^{VCOPT}} w_i 1 \geq \sum_{i \in V} w_i x_i^* \geq \sum_{i \in S} w_i x_i^* \geq \frac{1}{2} \sum_{i \in S} w_i = \frac{1}{2} w(S)$$

soln corresponding to S^{VCOPT} cannot be better than opt LP solution x^*, since LP is a relaxation

Drop i with $x^*_i < \frac{1}{2}$, Keep $x^*_i >= \frac{1}{2}$

$x^*_i \geq \frac{1}{2}$ For all i in S

Theorem. 2-approximation algorithm for weighted vertex cover.

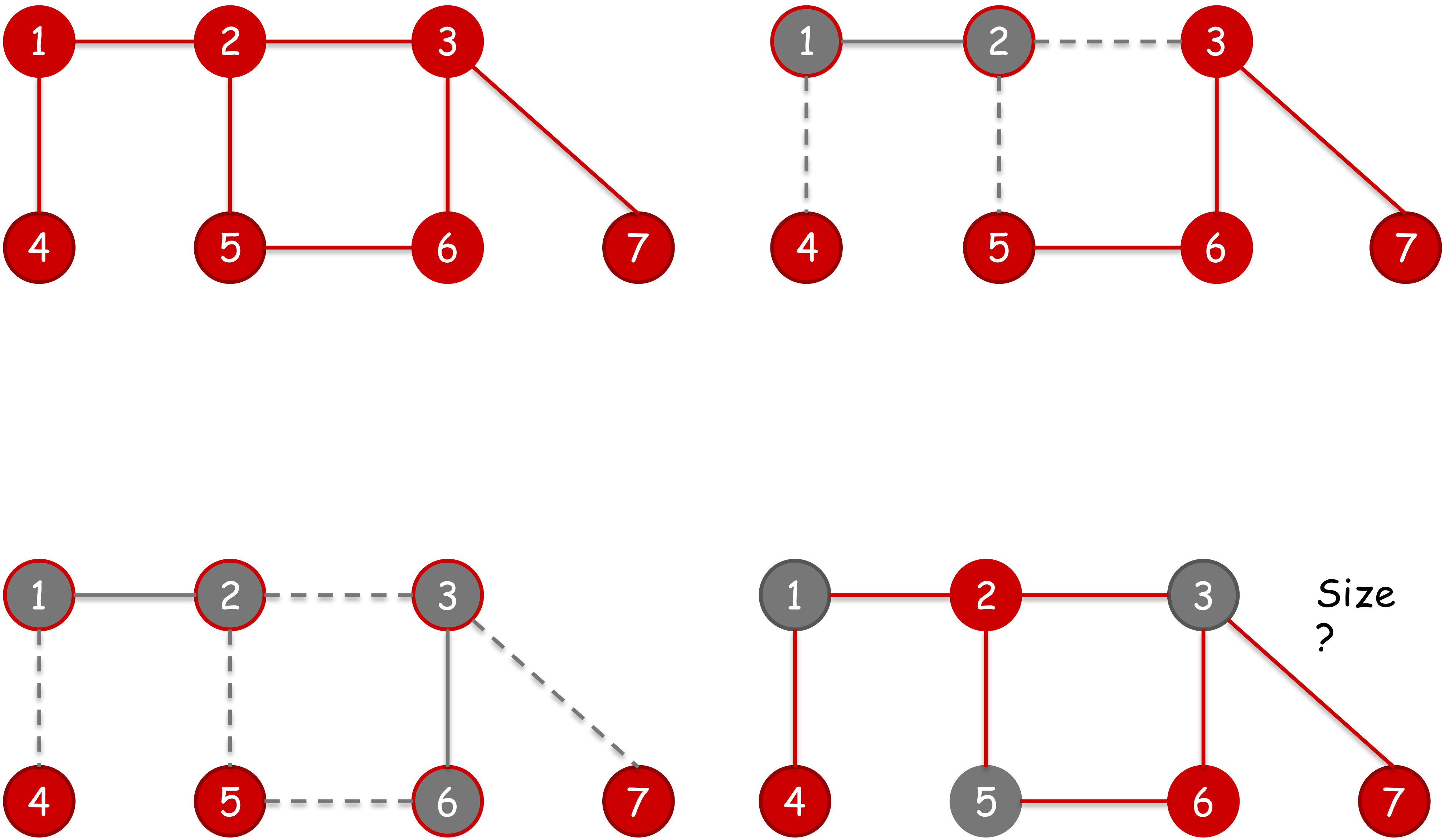
Greedy approximation algorithm for Vertex Cover [CLRS35.1]

Vertex cover: given a graph $G=(V,E)$, find the *smallest* number of vertices that cover *each edge*
(each edge has at least one endpoint in the vertex cover set)

Greedy algorithm:

$C \leftarrow \phi$ (the vertex cover)
$E' \leftarrow E$ (uncovered edges)
while $E' \neq \phi$
  do let (u,v) be an arbitrary edge of E'
    $C \leftarrow C \cup \{u,v\}$
    remove from E' every edge incident to either u or v.
return C

Example



Solution = 4

Optimal = 3

Ratio=1.33

2-approximation algorithm for Vertex Cover

Theorem.

- APPROX-VERTEX-COVER is a poly-time 2-approximation algorithm, i.e., the size of returned vertex cover set is at most twice of the size of optimal vertex-cover.

Proof:

- It runs in poly time (linear time)
- The returned set C is a vertex cover
  - every selected or deleted edge has endpoint in C,
  - and we continue until every edge is either selected or deleted

2-approximation algorithm for Vertex Cover

Proof continued

- We will show $|C| \leq 2|C^*|$
- Let A be the set of edges picked in line 4 of Algorithm and C^* be the optimal vertex cover.
  - C^* must include at least one end of each edge in A, since C^* is a vertex cover
  - no two edges in A are covered by the same vertex in C^*, since edges in A do not share endpoints (due to line 6)
  - so $|C^*| \geq |A|$ (at least one vertex from every edge in A)

  - Moreover, $|C| = 2|A|$
  - (for each edge in A, we add 2 nodes to C, and edges in A do not share endpoints so each endpoint counts towards $|C|$)

  - so $|C| = 2|A| \leq 2|C^*|$