Combinatorial Algorithms

Metric problems

Minimum Spanning Tree

- *Given* an undirected graph $G=(V,E)$ with nonnegative edge costs $c: E(G) \rightarrow \mathbf{R}$.
- *Find* a spanning tree in G of the minimum weight.

Optimality Conditions

Theorem 3.1

Let (G, c) be an instance of the Minimum Spanning Tree problem, and let T be a spanning tree in G. Then the following statements are equivalent:

a) T is optimum.

b) For every $e = \{x, y\} \in E(G) \setminus E(T)$, no edge on the x-y-path in T has higher cost than e.

c) For every $e \in E(T)$, e is a minimum cost edge of $\delta(V(C))$, where C is a connected component of $T - e$.

(c)⇒(a)

- (c) Suppose T satisfies (c), for every $e \in E(T)$, e is a minimum cost edge of $\delta(V(C))$, where C is a connected component of $T - e$.
- Let T^* be an optimum spanning tree with $E(T) \cap E(T^*)$ as large as possible. We show that $T = T^*$.
- Suppose there is an edge $e = \{x,y\} \in E(T) \setminus E(T^*)$.
- Let C be a connected component of $T - e$.
- $T^* + e$ contains a circuit D. Since $e \in E(D) \cap \delta(C)$, at least one more edge $f \neq e, f \in E(D) \cap \delta(C)$.
- Observe that $(T^* + e) - f$ is a spanning tree.
- Since T^* is optimum $\Rightarrow c(e) \geq c(f)$ and (c) $\Rightarrow c(f) \geq c(e)$.
- $c(f) = c(e)$ and $(T^* + e) - f$ is another optimum spanning tree.
- This is a contradiction, because $(T^* + e) - f$ has one edge more in common with T.

Prim's Algorithm (1957)

Input: A connected undirected graph G,
weights c: $E(G) \to \mathbf{R}$.

Output: Spanning tree T of minimum weight.

1) Choose $v \in V(G)$. $T := (\{v\}, \emptyset)$.
2) **While** $V(T) \neq V(G)$ **do**:
Choose an edge $e \in \delta_G(V(T))$ of minimum cost. Set $T := T + e$.

Steiner Tree

- *Given* an undirected graph $G = (V, E)$ with nonnegative edge costs and whose vertices are partitioned into two sets, ***required*** and ***Steiner***.
- *Find* a minimum cost tree in G that contains all the required vertices and any subset of the Steiner vertices.
- Let R denote the set of **required** vertices.
- Set V/R is called the **Steiner** set**.**

Metric Steiner Tree

- *Given* a complete undirected graph $G = (V, E)$ with nonnegative edge costs such that for any three vertices u, v and w, $cost(u,v) \leq cost(u,w) + cost(w,v)$ and whose vertices are partitioned into two sets, ***required*** and ***Steiner***.
- *Find* a minimum cost tree in G that contains all the required vertices and any subset of the Steiner vertices.

Factor Preserving Reduction

Theorem 3.2

There is an approximation factor preserving reduction from the Steiner tree problem to the metric Steiner tree problem.

Proof ($\Rightarrow$)

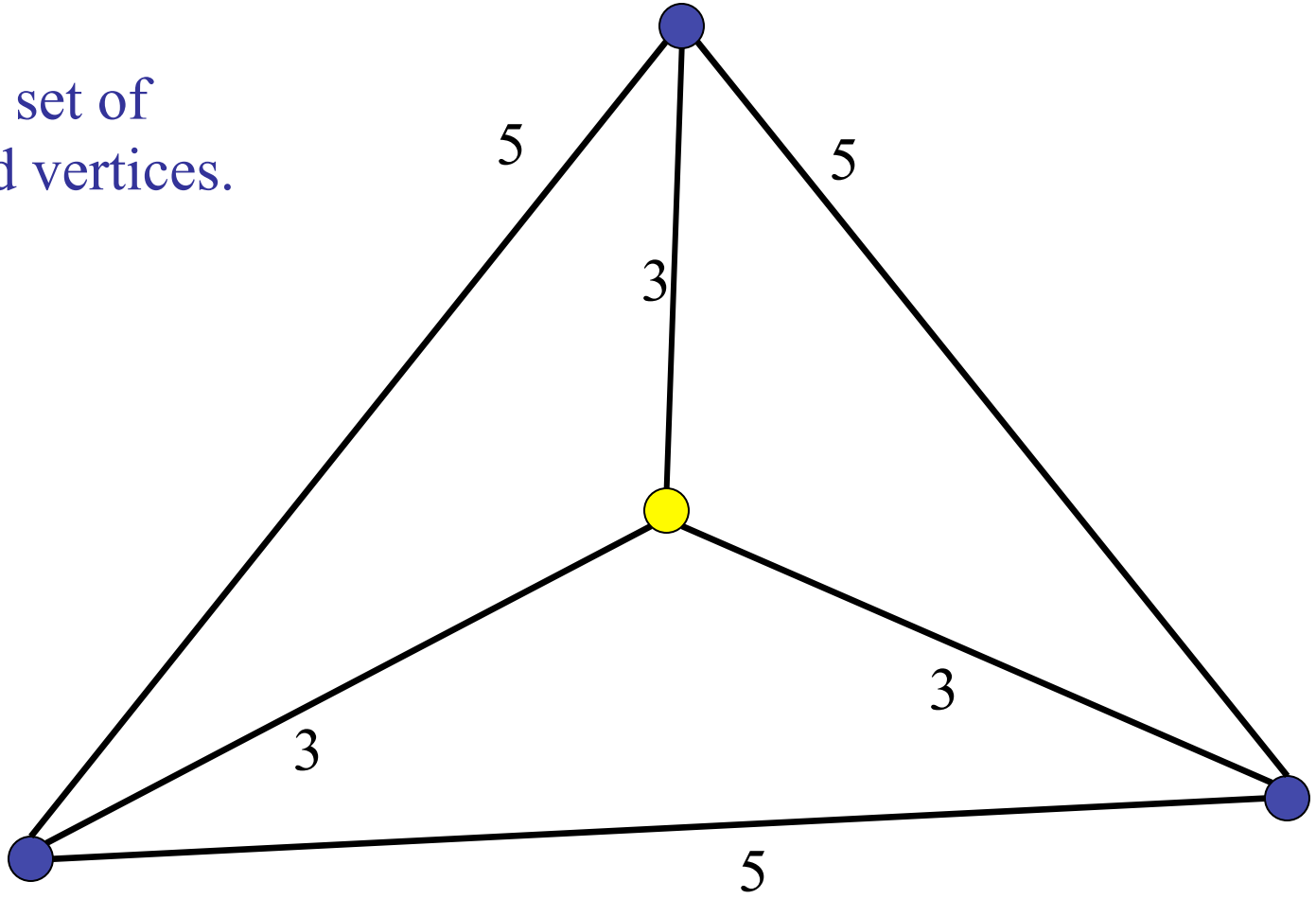
- We will transform, in polynomial time, an instance I of the Steiner tree problem, consisting of graph $G = (V,E)$, to an instance I' of the metric Steiner tree problem as follows.
- Let G' be the complete undirected graph on vertex set V. Define the cost of edge (u, v) in G' to be the cost of a shortest u-v path in G. G' is called the *metric closure* of G. The partition of V into required and Steiner vertices in I' is the same as in I.
- For any edge $(u, v) \in E$, its cost in G' is no more than its cost in G. Therefore, the cost of an optimal solution in I' does not exceed the cost of an optimal solution in I.

Proof ($\Leftarrow$)

- Next, given a Steiner tree T' in I', we will show how to obtain, in polynomial time, a Steiner tree T in I of at most the same cost.
- The cost of an edge (u, v) in G' corresponds to the cost of a path in G. Replace each edge of T' by the corresponding path to obtain a subgraph of G.
- In this subgraph, all the required vertices are connected. However, this subgraph may contain cycles. If so, remove edges to obtain tree T. This completes the approximation factor preserving reduction.
- As a consequence of Theorem 3.2, any approximation factor established for the metric Steiner tree problem carries over to the entire Steiner tree problem.

Steiner Tree and Minimum Spanning Tree (*MST*)

R is the set of required vertices.

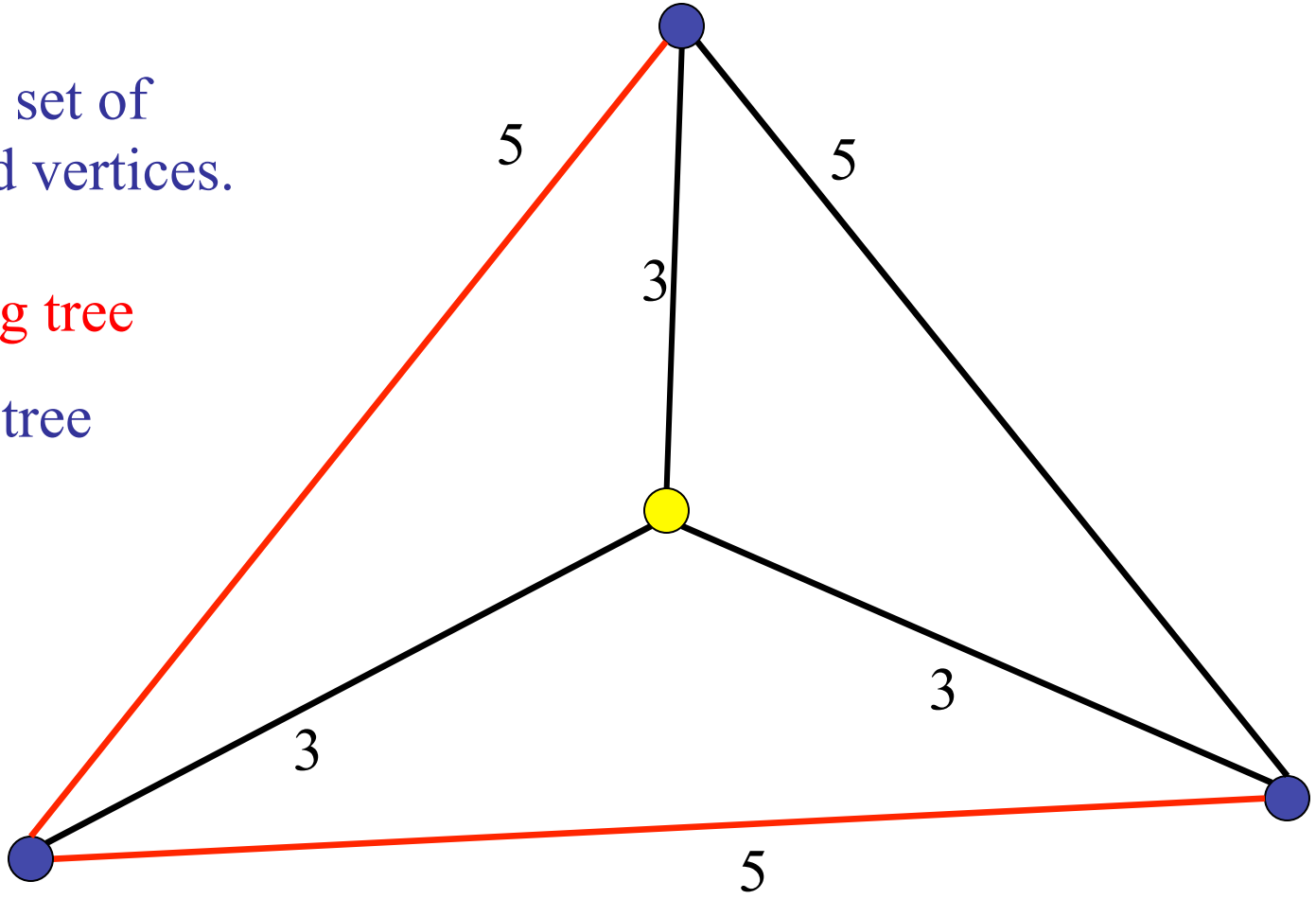


Steiner Tree and Minimum Spanning Tree (*MST*)

R is the set of required vertices.

Spanning tree

Steiner tree

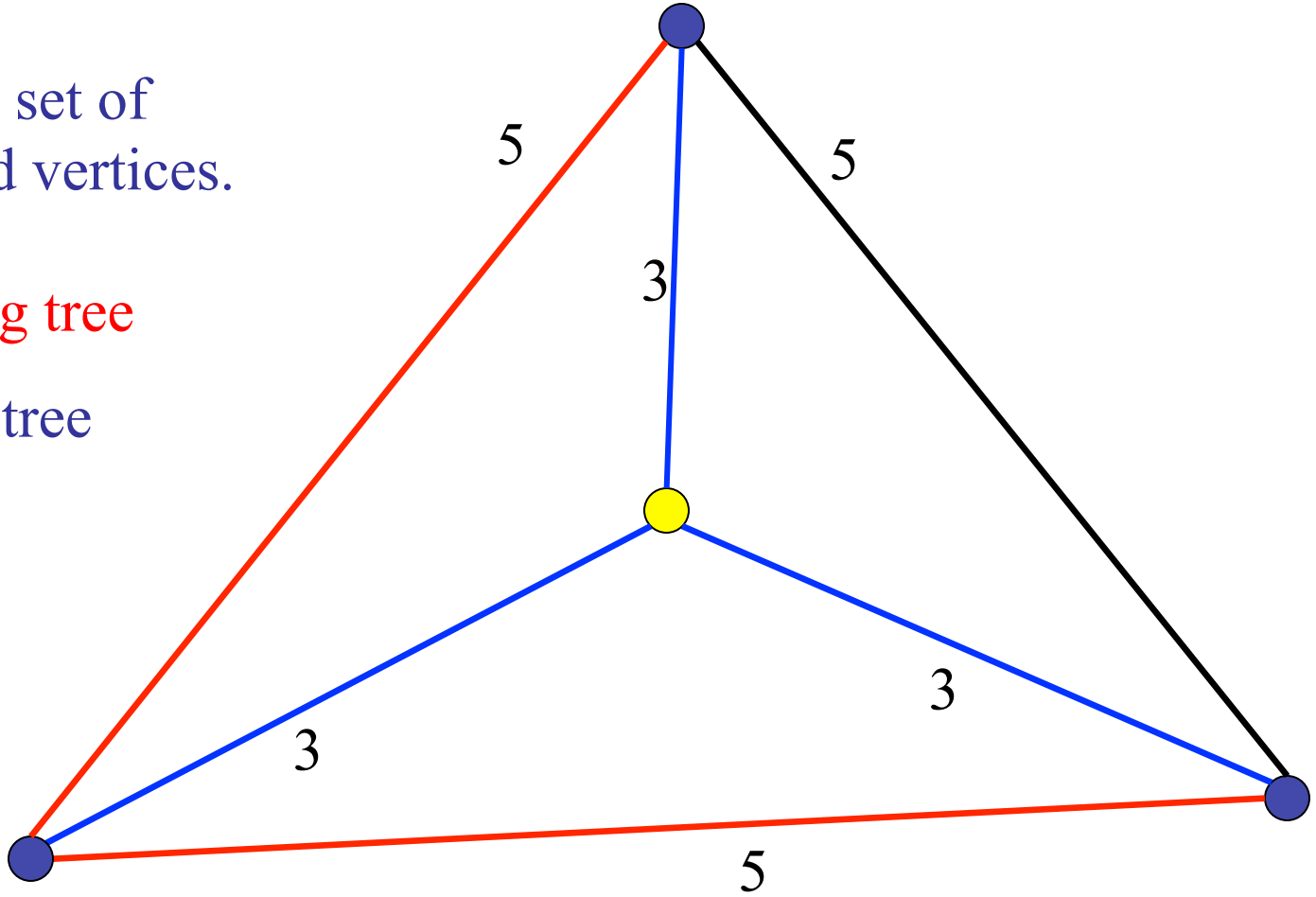


Steiner Tree and Minimum Spanning Tree (*MST*)

R is the set of required vertices.

Spanning tree

Steiner tree



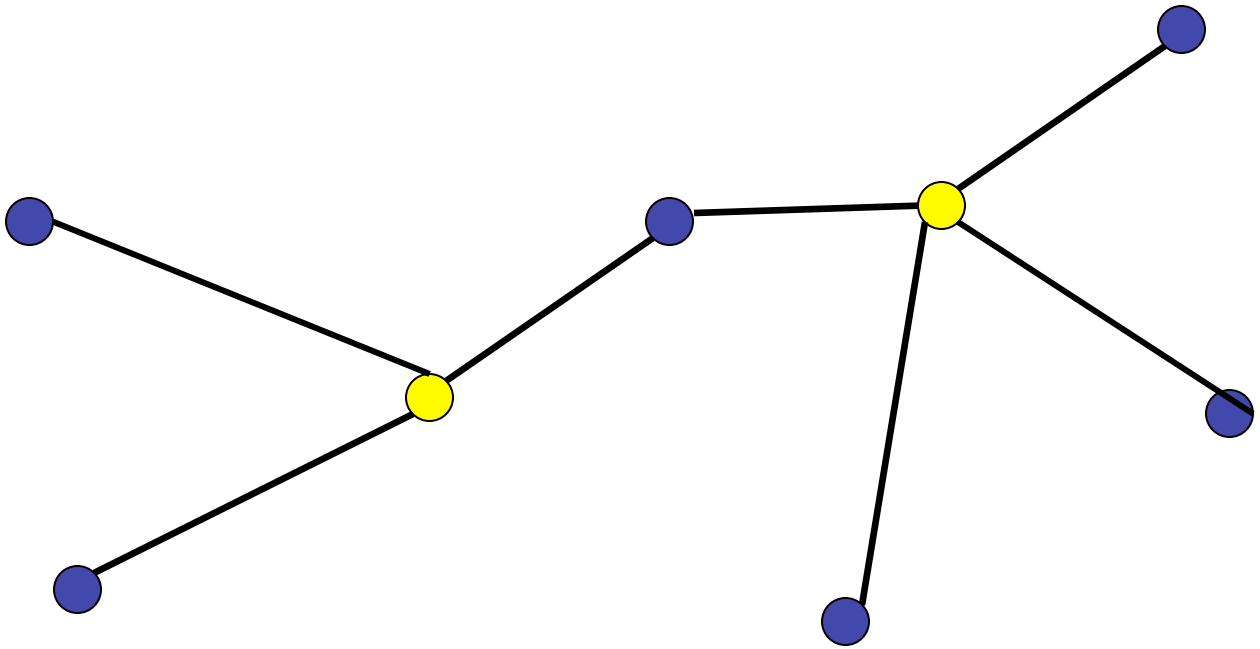
Lower Bound

Theorem 3.3

The cost of a minimum spanning tree on R is within 2OPT.

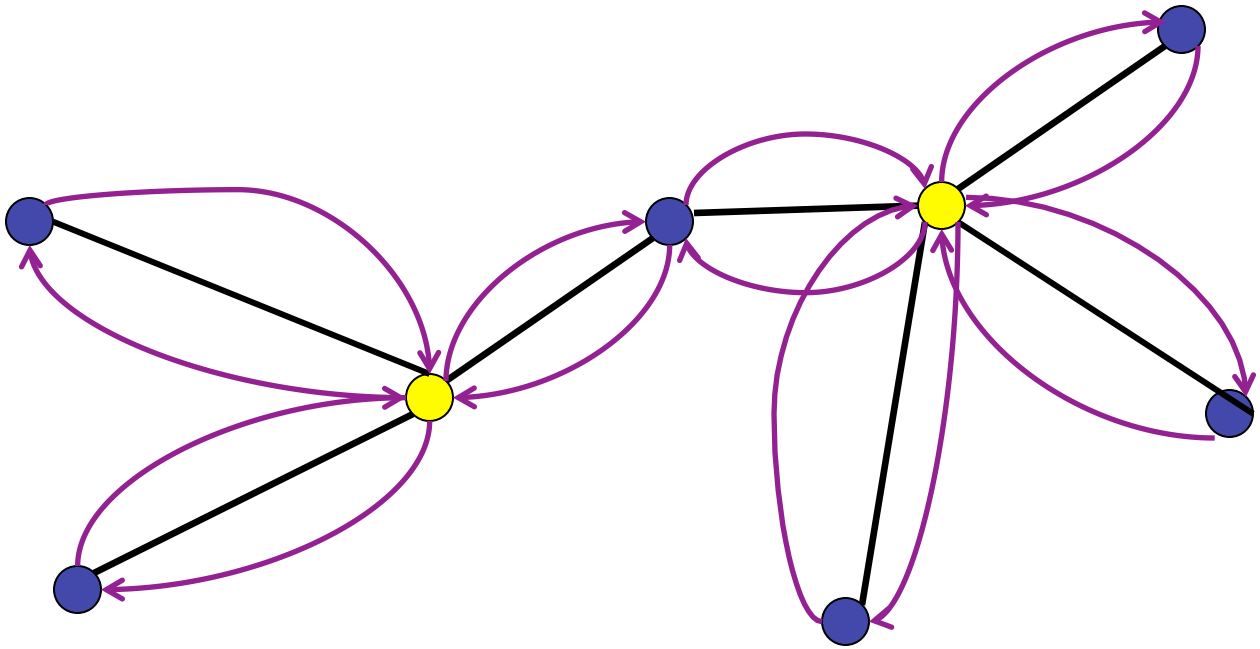
Proof

Steiner tree



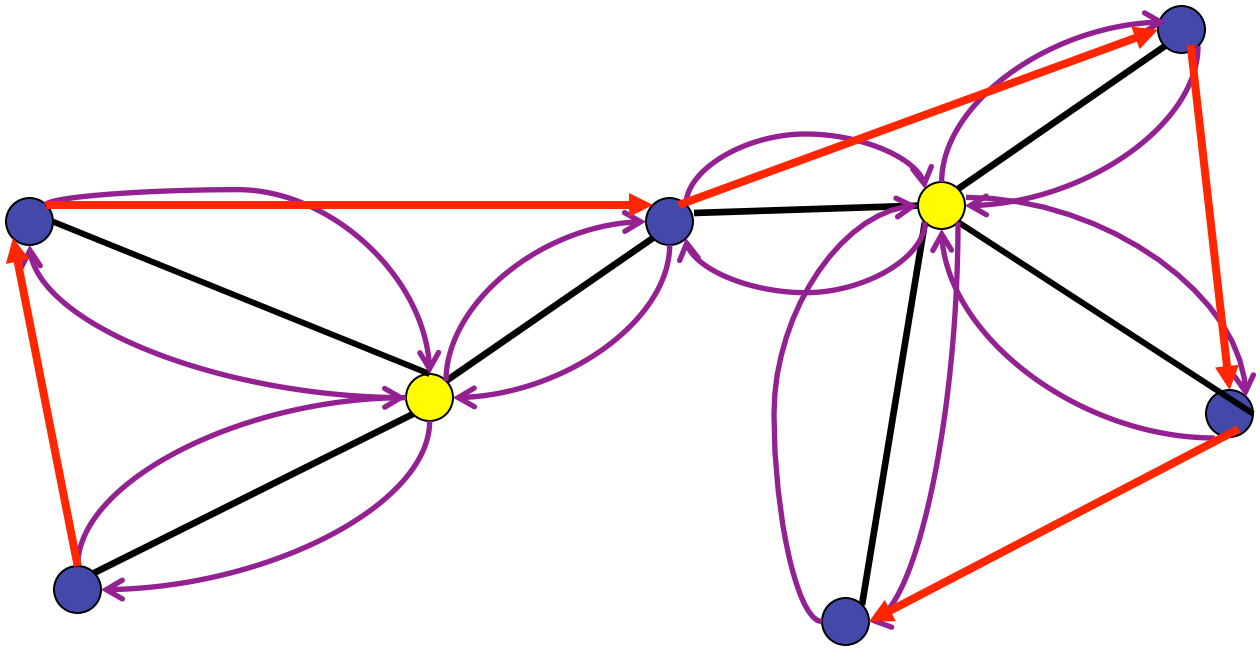
Proof

Steiner tree ⇒ Euler tour



Proof

Steiner tree ⇒ Euler tour ⇒ Spanning tree



Algorithm MST

Input (G, R, *cost*: $E \rightarrow \mathbf{Q}^+$)

1) Find a minimum spanning tree T on R.

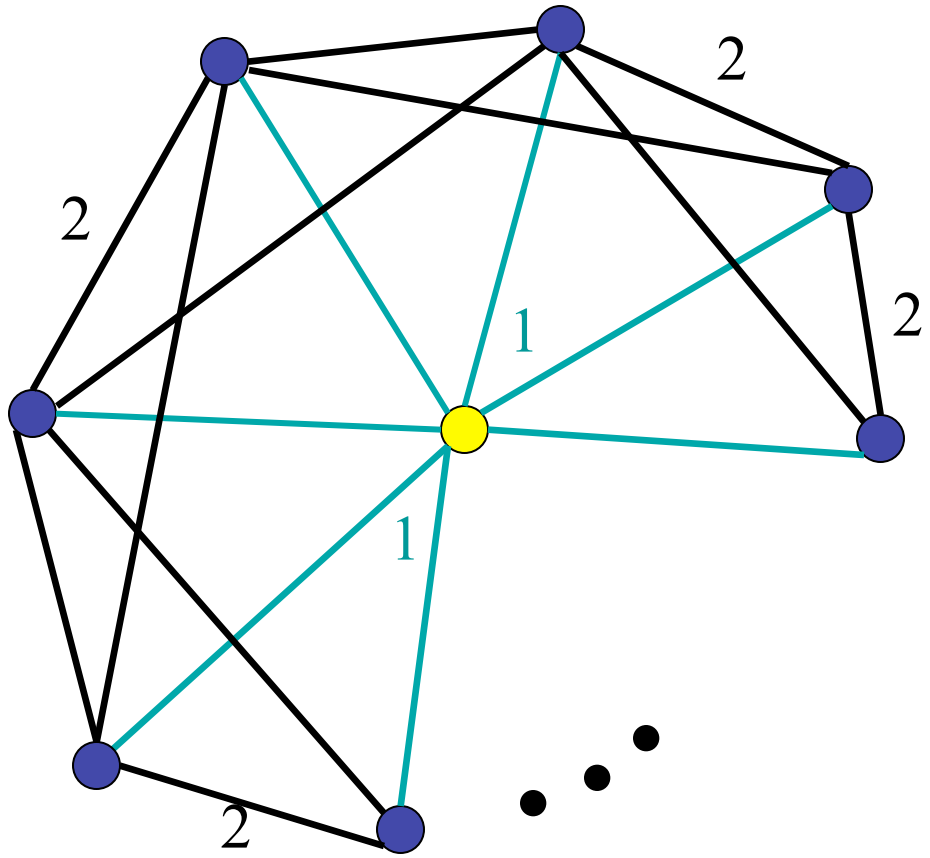
Output (T)

Approximation ratio of Algorithm MST

Corollary 3.4

Algorithm MST is a *2*-approximation algorithm for the Steiner problem.

Tight Example



n required vertices

one Steiner vertex

$$\frac{OPT_{MST}}{OPT} = \frac{2(n-1)}{n} \underset{n\to\infty}{\longrightarrow} 2$$

Travelling Salesman Problem (TSP)

- *Given* a complete undirected graph $G = (V, E)$ with nonnegative edge costs.
- *Find* a minimum cost cycle (Hamiltonian cycle) visiting every vertex exactly once.

Inapproximability

Theorem 3.5

For any polynomial time computable function $\alpha(n)$, TSP cannot be approximated within a factor of $\alpha(n)$, unless **P = NP**.

Sketch of Proof

- Assume, for a contradiction, that there is a factor $\alpha(n)$ polynomial time approximation algorithm A for the general TSP problem.
- We will show that A can be used for deciding the Hamiltonian cycle problem (which is NP-complete) in polynomial time, this implying **P = NP**.

Central Idea

Reduce the Hamiltonian cycle problem to TSP, i.e. transform a graph G on n vertices to an edge-weighted complete graph G' on n vertices such that

- If G has a Hamiltonian cycle, then the cost of an optimal TSP tour in G' is n, and
- If G does not have a Hamiltonian cycle, then an optimal TSP tour in G' is of cost greater than $n\alpha(n)$.

Reduction

- Assign a weight of 1 to edges of G, and a weight of $n\alpha(n)$ to nonedges, to obtain G'.
- Now, if G has a Hamiltonian cycle, then the corresponding tour in G' has cost n.
- If G has no Hamiltonian cycle, any tour in G' must use an edge of cost $n\alpha(n)$, and therefore has cost $> n\alpha(n)$.
- When run on G', algorithm A must return a solution of cost $\leq n\alpha(n)$ in the first case, and a solution of cost $> n\alpha(n)$ in the second case. Thus, it can be used for deciding whether G contains a Hamiltonian cycle.

Metric TSP

- *Given* a complete undirected graph $G = (V, E)$ with nonnegative edge costs, such that for any three vertices u, v and w,
  $cost(u,v) \leq cost(u,w) + cost(w,v)$.
- *Find* a minimum cost Hamiltonian cycle.

Algorithm MST-2

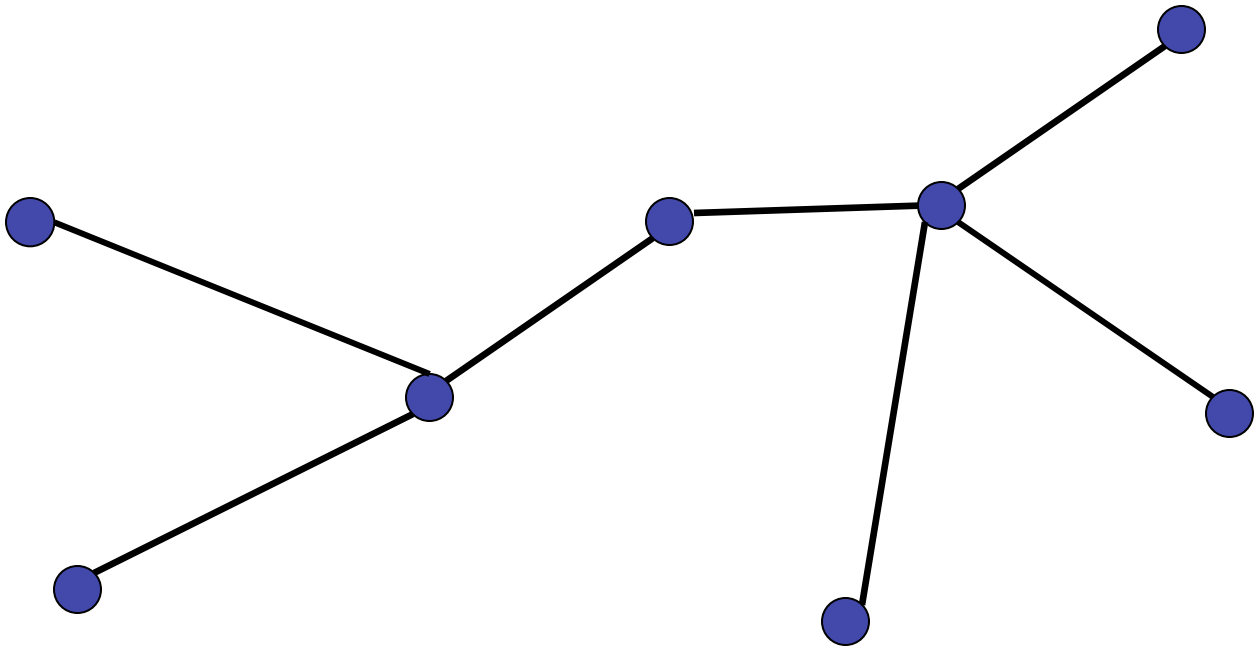
Input (G, *cost*: $E \rightarrow \mathbf{Q}^+$)

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

Output (C)

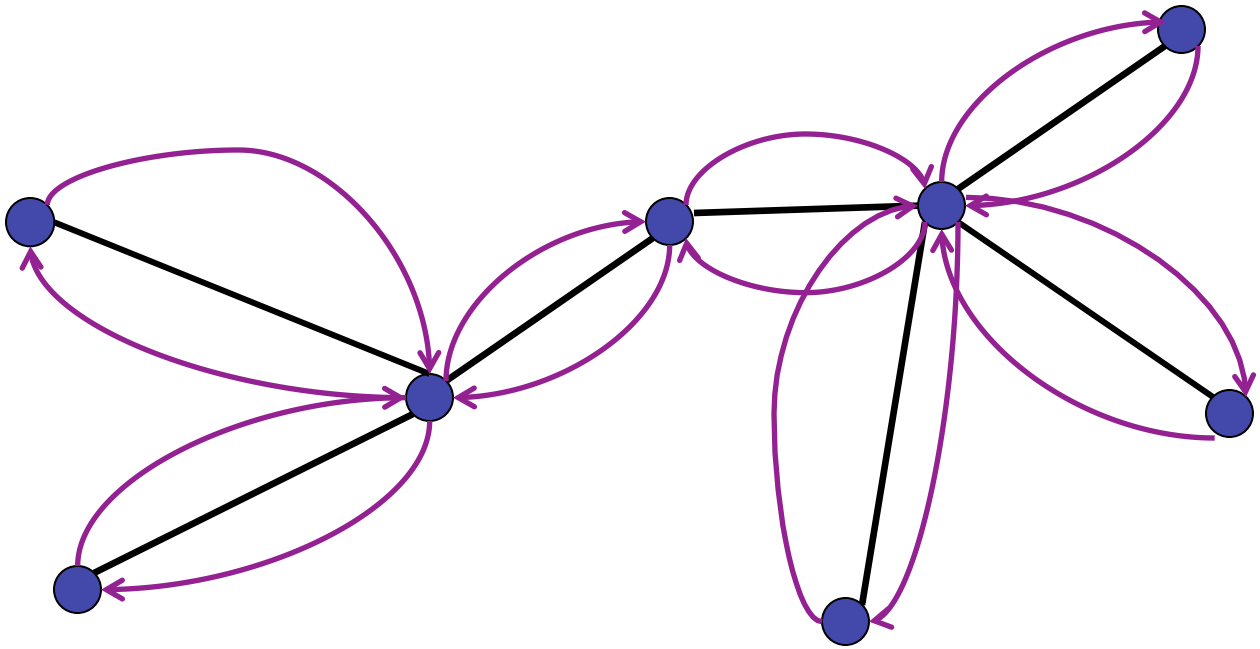
Example

Minimum spaning tree



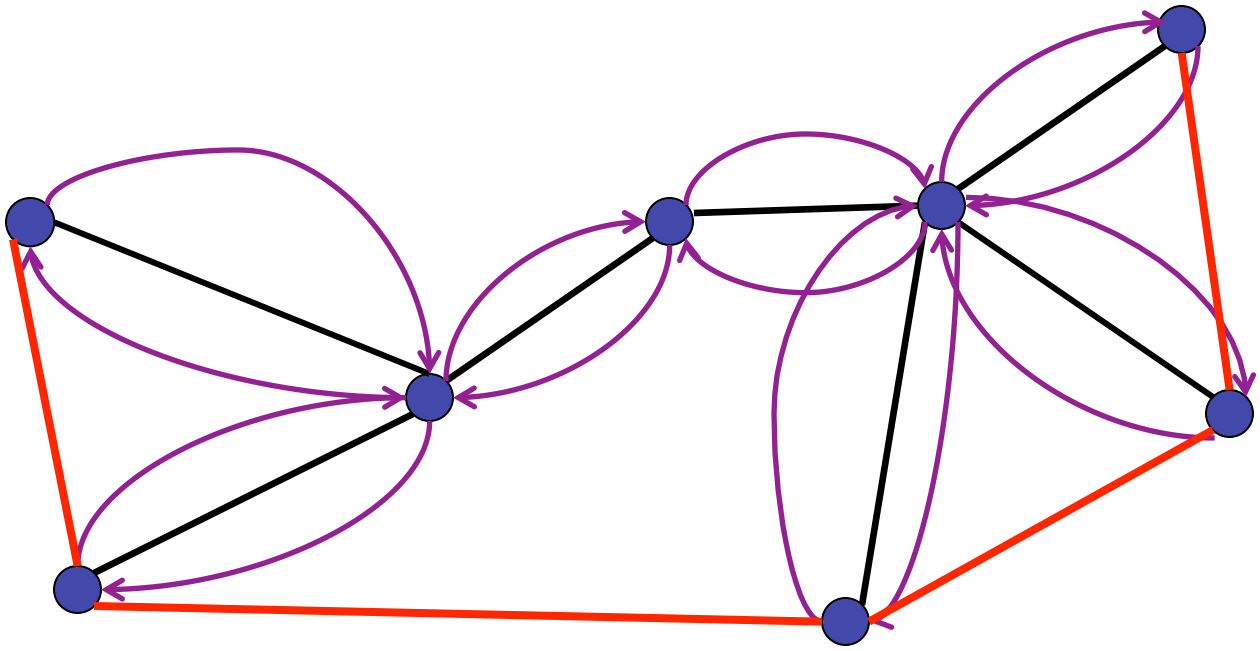
Example

Minimum spanning tree ⇨ Eulerian tour



Example

Minimum spanning tree ⇨ Eulerian tour ⇨ Hamiltonian cycle



Approximation ratio of Algorithm MST-2

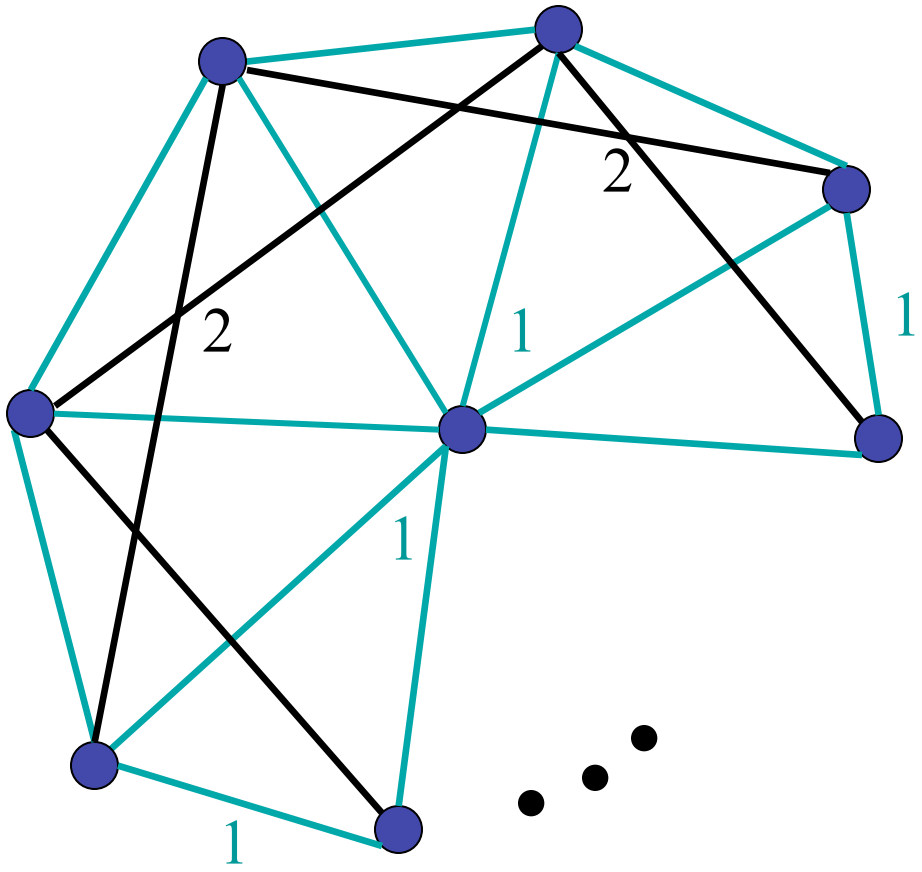
Theorem 3.6

Algorithm MST-2 is a factor 2 approximation algorithm for metric TSP.

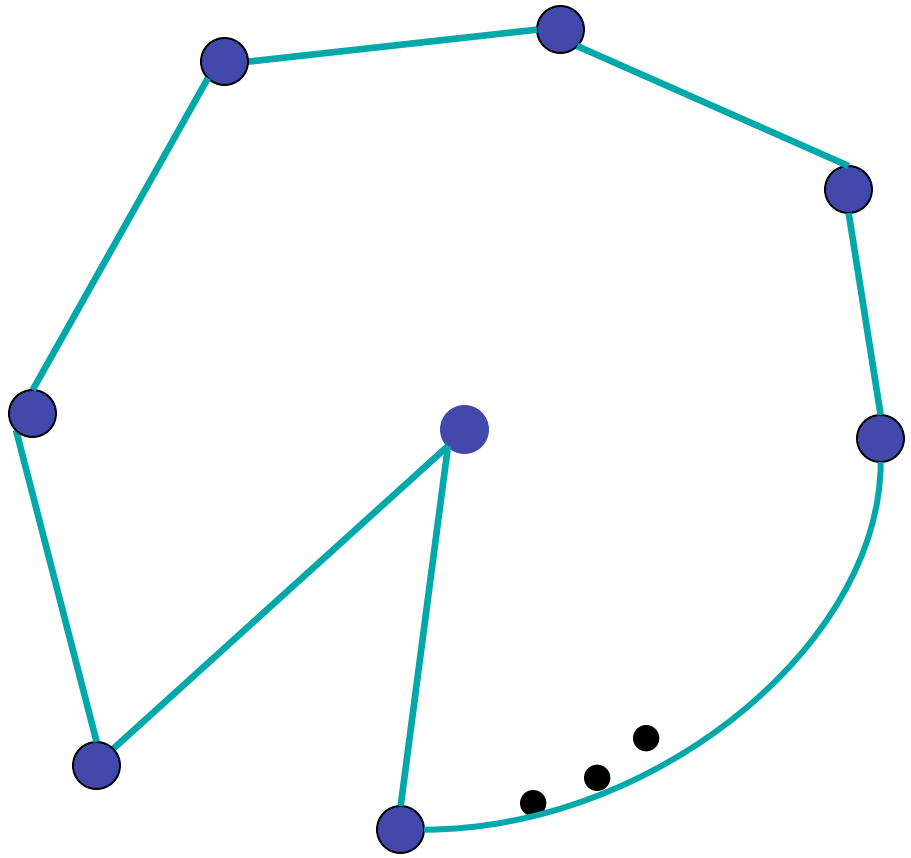
Proof

- It is obvious that cost(T) $\leq$ OPT. Since R contains each edge of T twice, cost(T) = 2cost(R). Because of triangle inequality, after the "short-cutting" step, cost(C) $\leq$ cost(R). Combining these inequalities we get that cost(C) $\leq$ 2OPT.

Tight Example

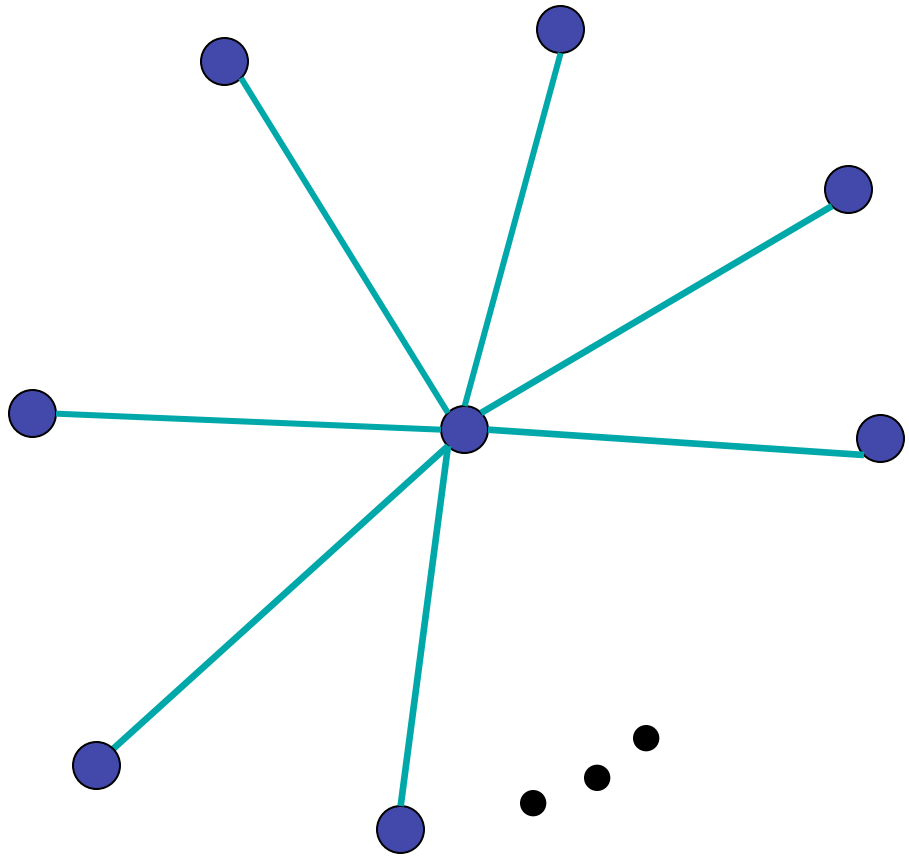


Optimal Tour

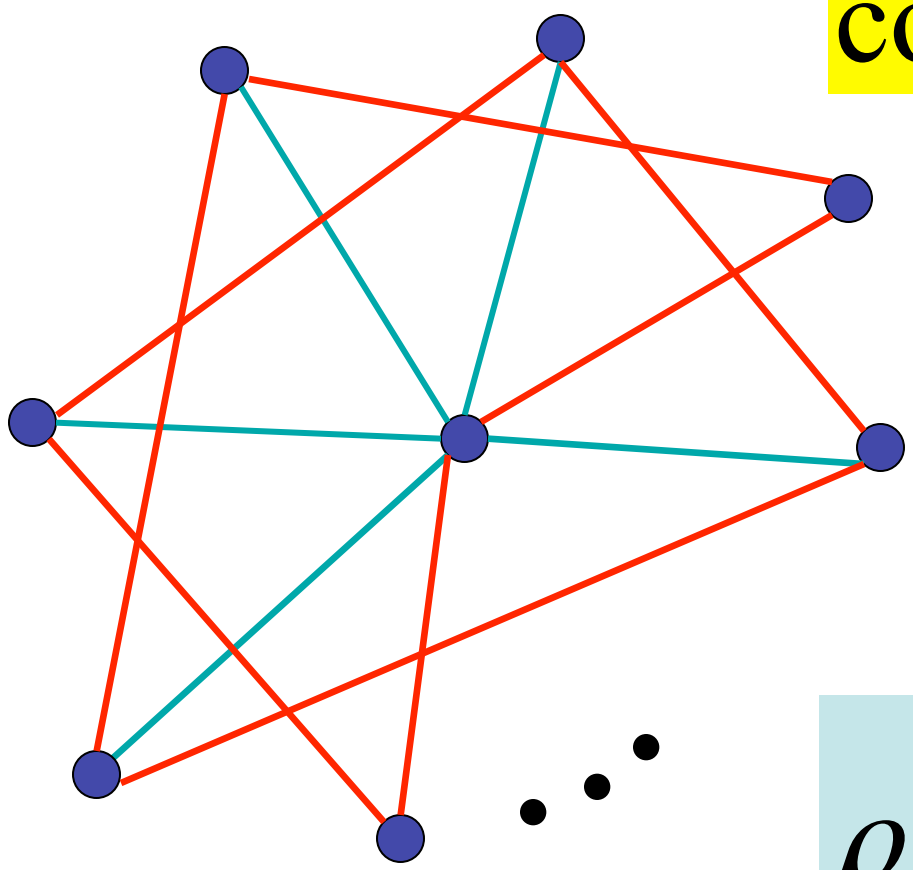


$$OPT = n$$

Minimal Spanning Tour



Hamiltonian cycle



$$\text{cost}(C) = 2n - 2$$

$$\rho = \frac{2n-2}{n} \xrightarrow[n\to\infty]{} 2$$

Christofides-Serdyukov Algorithm

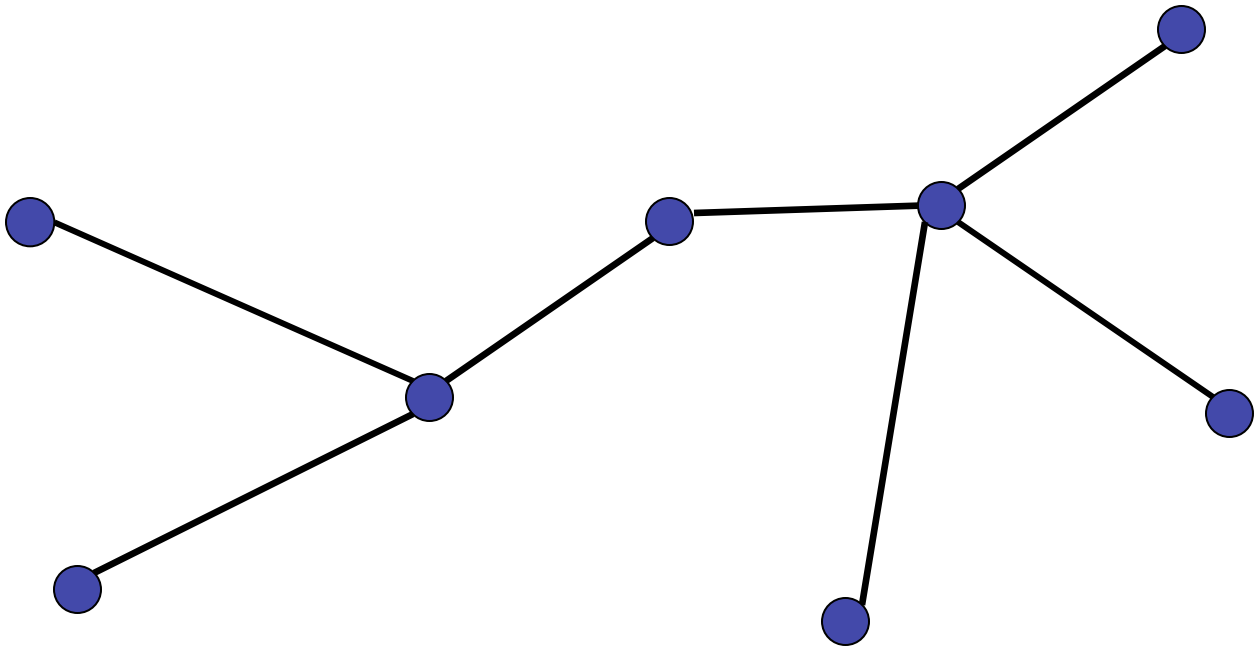
Input (G, *cost*: $E \rightarrow \mathbf{Q}^+$)

1) Find an MST of G, say T.
2) Compute a minimum cost perfect matching, M, on the set of odd degree vertices of T.
3) Add M to T and obtain an Eulerian graph H.
4) Find an Euler tour R of H.
5) Output tour C that visits vertices of G in order of their first appearance in R.

Output (C)

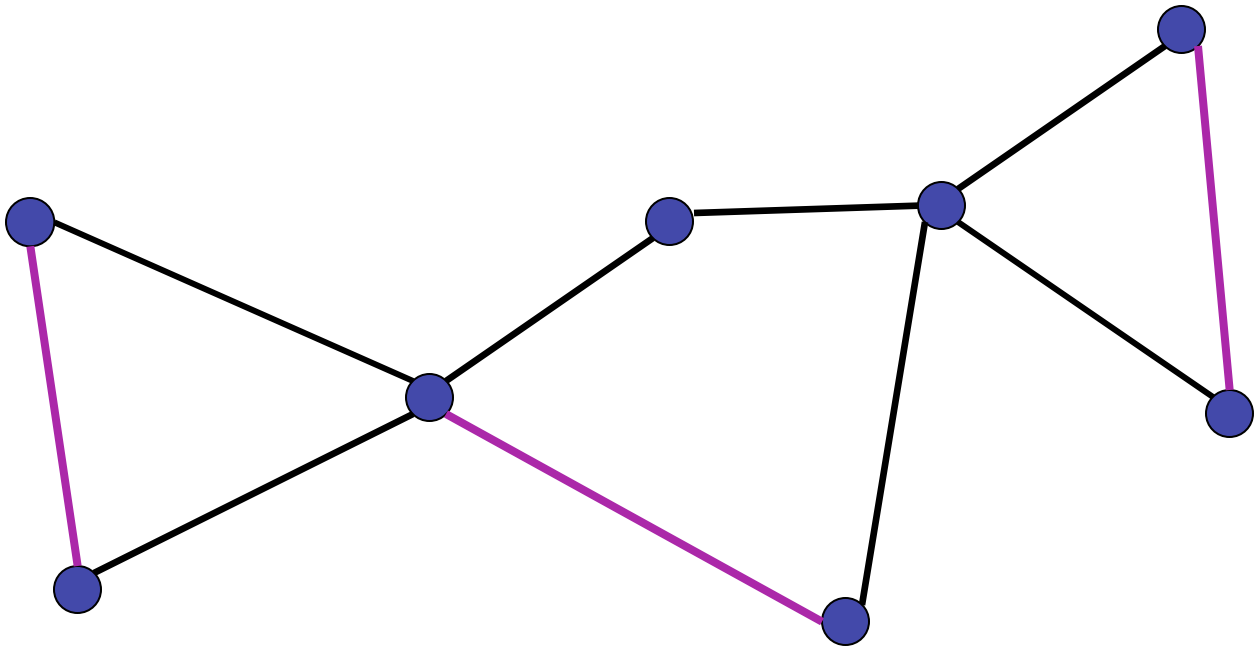
Example

Minimum spanning tree



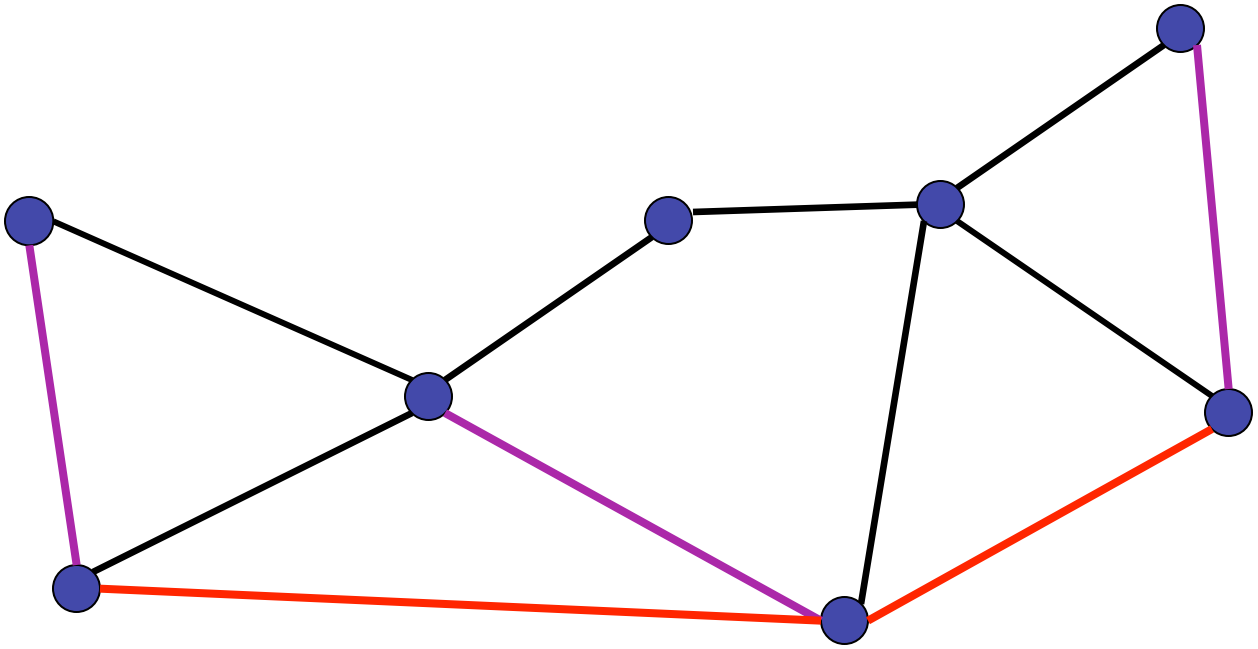
Example

Minimum spanning tree ⇨ Matching



Example

Minimum spanning tree ⇨ Matching ⇨ Hamiltonian cycle



Lower bound

Lemma 3.7

Let $V' \subseteq V$, such that $|V'|$ is even, and let M be a minimum cost perfect matching on V'. Then $\text{cost}(M) \leq \text{OPT}/2$.

Approximation ratio of Christofides-Serdyukov Algorithm

Theorem 3.8

Christofides-Serdyukov Algorithm achieves an approximation guarantee of 3/2 for metric TSP.

Proof:

$$\text{cost}(C) \le \text{cost}(H) = \text{cost}(T) + \text{cost}(M) \le OPT + \frac{OPT}{2} = \frac{3}{2} OPT$$

Metric k-center

- *Given* a complete undirected graph $G = (V, E)$ with nonnegative edge costs satisfying the triangle inequality, and k is a positive integer. For any set $S \subseteq V$ and vertex v define $connect(v,S) = \min\{cost(u,v) | u \in S\}$ (the cost of the cheapest edge from v to a vertex in S.)
- *Find* a set $S \subseteq V$, with $|S|=k$, so as to minimize $\max_v\{connect(v,S)\}$.
- The metric k-center problem is NP-hard.

Parametric pruning (1)

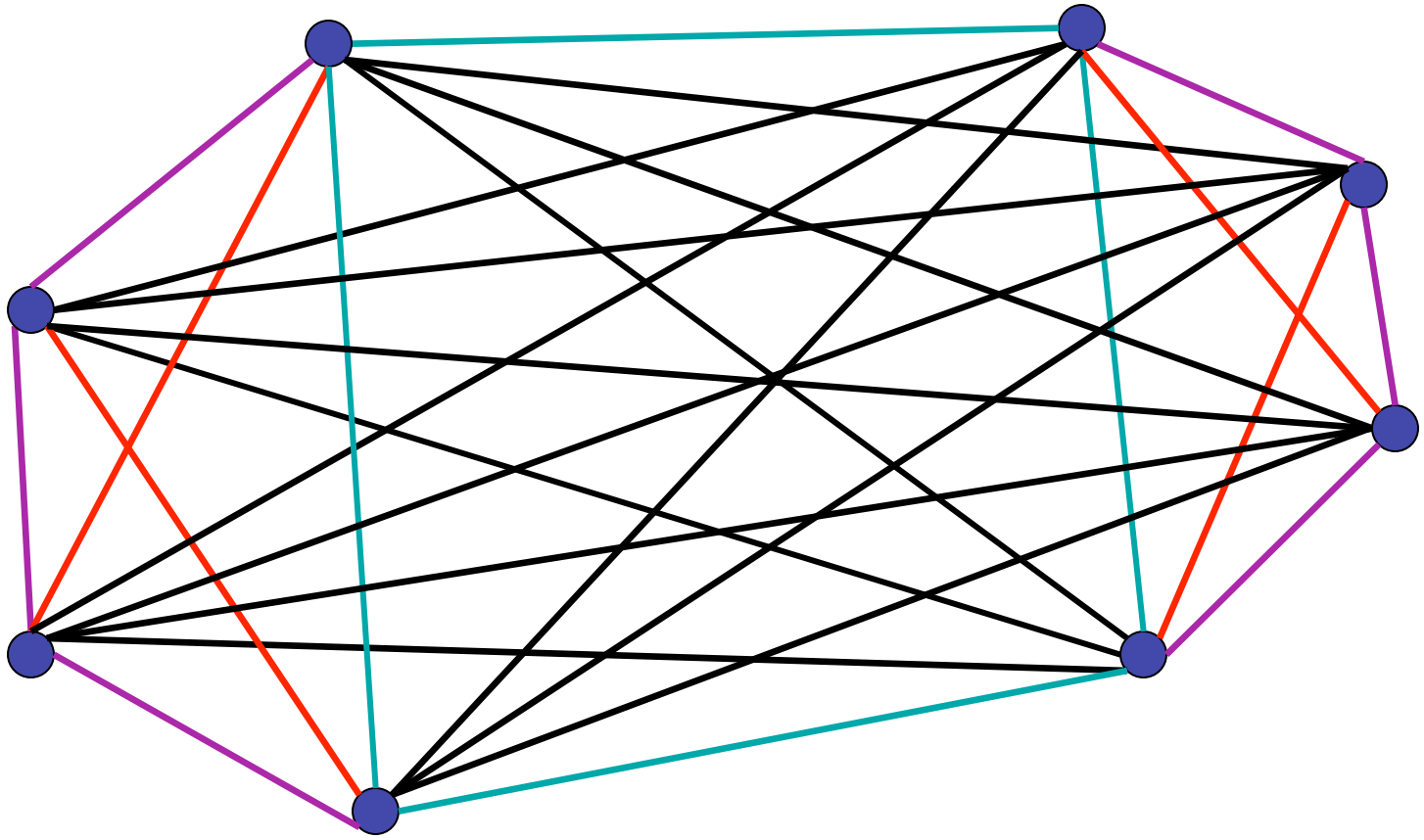
- If we know the cost of an optimal solution, we may be able to prune away irrelevant parts of the input and thereby simplify the search for a good solution.
- However computing the cost of an optimal solution is precisely the difficult core of **NP-hard NP-optimization problems.**
- The technique of parametric pruning gets around this difficulty as follows. A parameter *t* is chosen, which can be viewed as a "guess" on the cost of an optimal solution. For each value of *t,* the given instance *I* is pruned by removing parts that will not be used in any solution of cost > *t.*

Parametric pruning (2)

The algorithm consists of two steps.

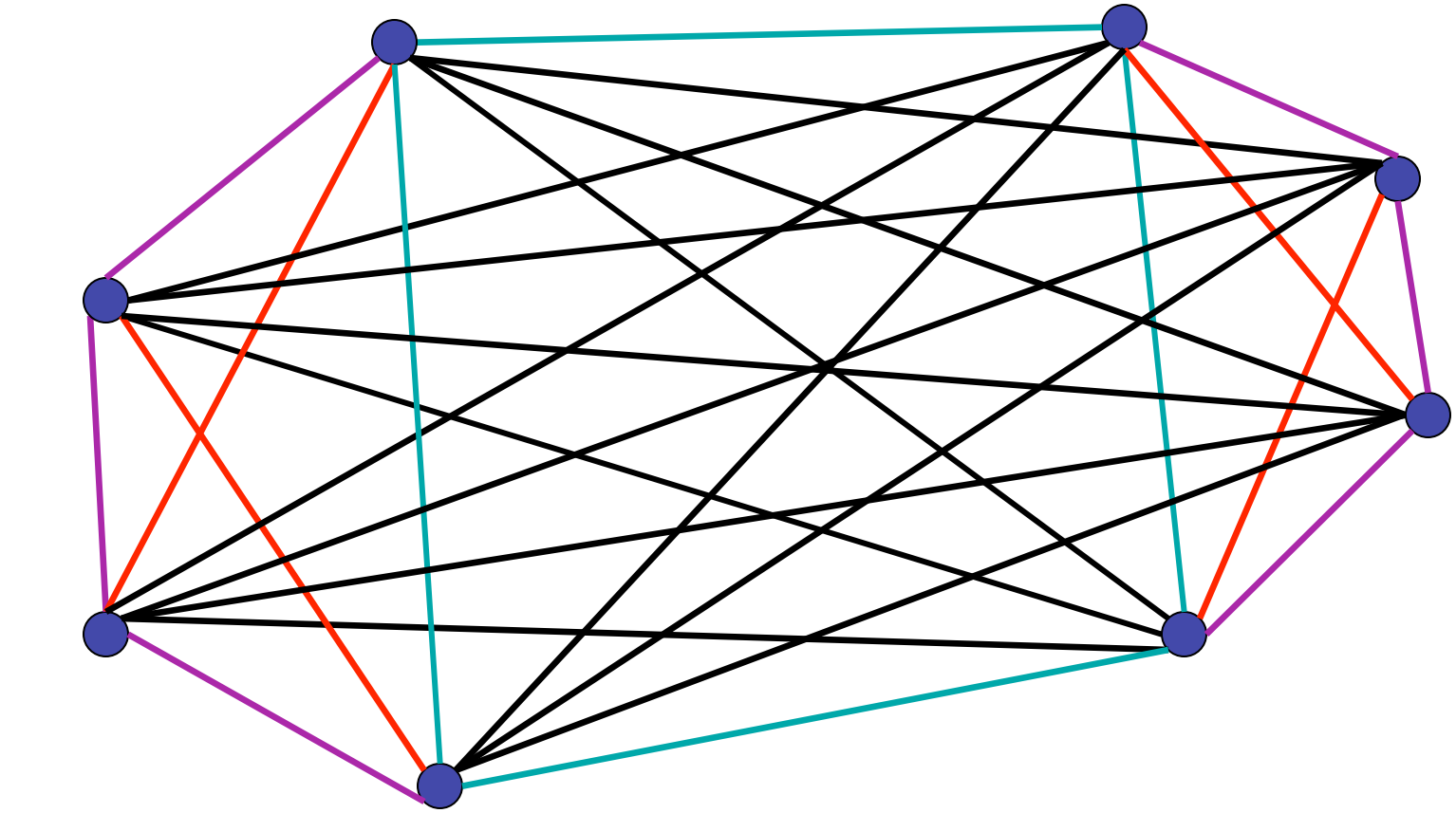
- In the first step, the family of instances $I(t)$ is used for computing a lower bound on *OPT*, say t^*.
- In the second step, a solution is found in instance $I(\alpha\, t^*)$, for a suitable choice of α.

Instance

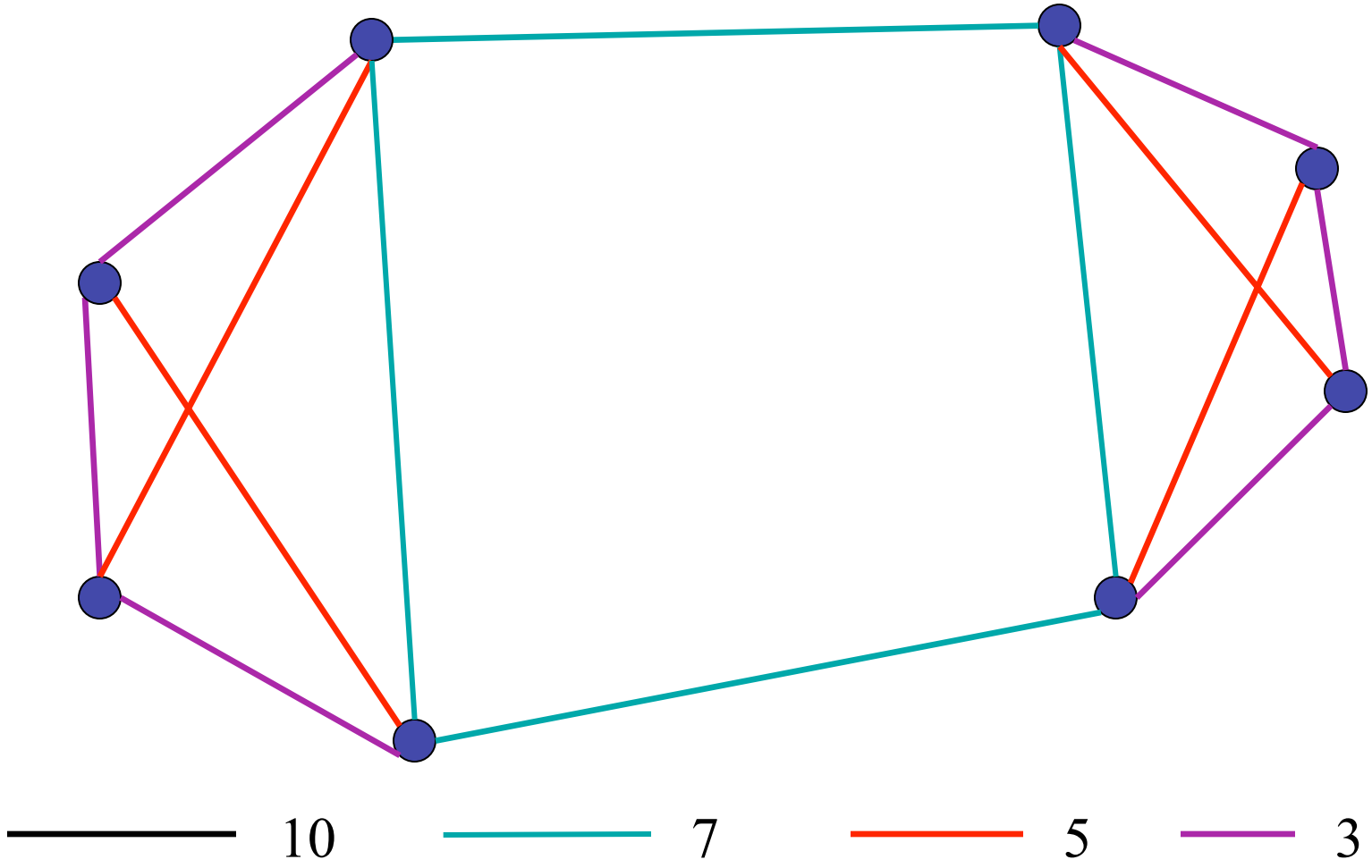


10 7 5 3

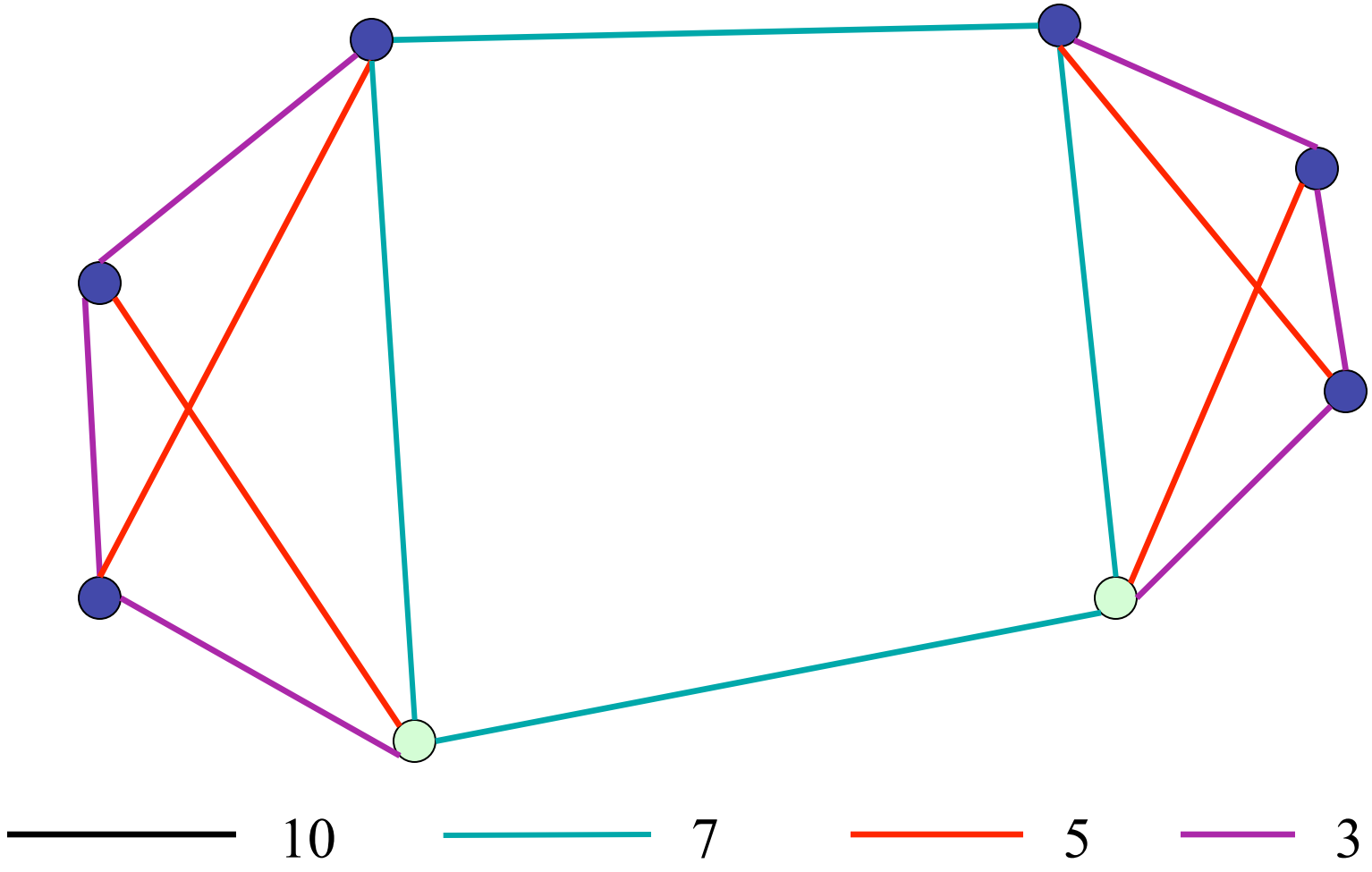
Idea of Algorithm (k=2, OPT ≤ 7)



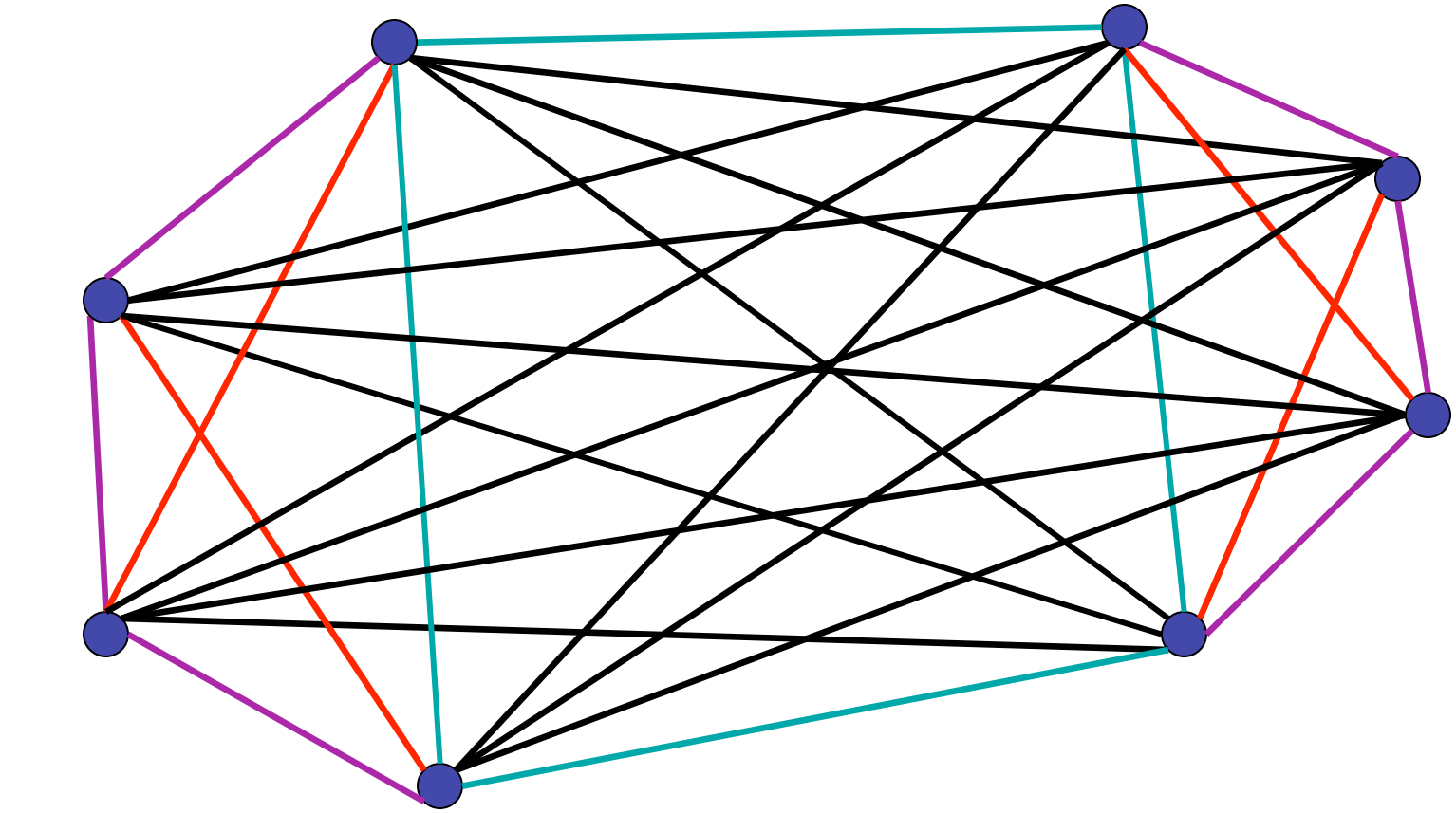
Idea of Algorithm (k=2, OPT ≤ 7)



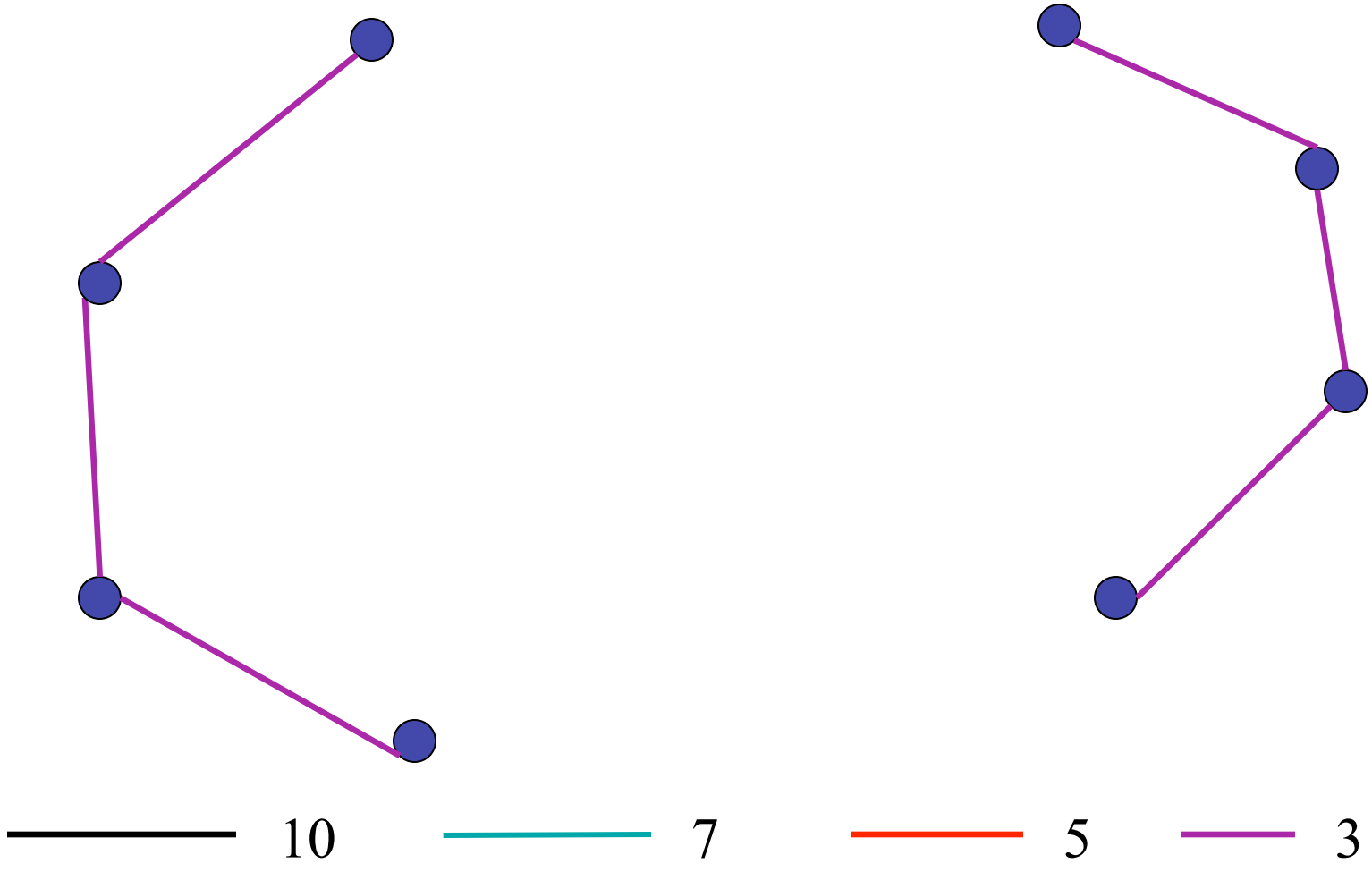
Idea of Algorithm (k=2, OPT ≤ 7)



Idea of Algorithm (k=2, OPT ≤ 3)



Idea of Algorithm (k=2, OPT ≤ 3)



Parametric pruning

- Sort the edges of G in nondecreasing order of cost, i.e. $cost(e_1) \leq cost(e_2) \leq \ldots \leq cost(e_m)$.
- Let $G_i = (V, E_i)$, where $E_i = \{e_1, e_2, \ldots, e_i\}$.
- For each G_i, we have to check whether there exists a subset $S \subseteq V$ such that every vertex in $V - S$ is adjacent to a vertex in S.

Dominating Set

- A dominating set in an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ such that every vertex in $V - S$ is adjacent to a vertex in S.
- Let dom(G) denote the size of minimum cardinality dominating set in G.

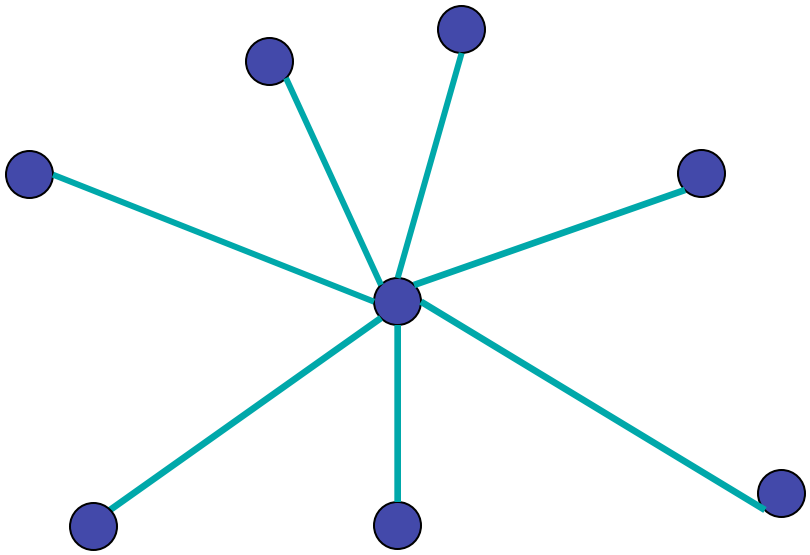
Dominating Set

- A dominating set in an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ such that every vertex in $V - S$ is adjacent to a vertex in S.
- Let dom(G) denote the size of minimum cardinality dominating set in G.
- Computing dom(G) is *NP*-hard.

k-Center

- The k-center problem is equivalent to finding the smallest index i such that G_i has a dominating set of size at most k.
- G_i contains k stars ($K_{1,p}$) spanning all vertices.

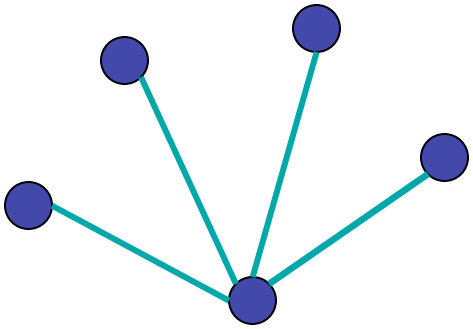
$K_{1,7}$



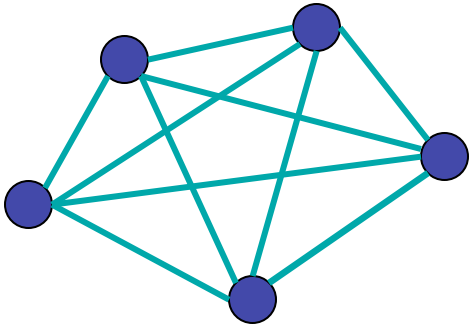
G^2

- **Independent set (stable set)** in $G = (V, E)$ is a subset $I \subseteq V$ of pairwise non-adjacent vertices.
- Define the **square of graph** $G = (V, E)$ to be the graph $G^2 = (V, E')$, containing an edge $(u,v) \in E'$ whenever G has a path of length at most 2 between.

$G=K_{1,4}$



$G^2=K_5$



Lower bound

- **Lemma 4.1**

  Given a graph H, let I be an independent set in H^2. Then, $|I| \leq \text{dom}(H)$.

Hochbaum-Shmoys Algorithm (1986)

Input (G, *cost*: $E \rightarrow \mathbf{Q}^+$)

1) Construct $G_1^2, G_2^2, \ldots, G_m^2$.
2) Compute a maximal independent set, I_r in each graph G_r^2.
3) Find the smallest index r such that $| I_r | \leq k$, say j.

Output (I_j)

Approximation ratio of Hochbaum-Shmoys Algorithm-1

Theorem 4.2

Hochbaum-Shmoys Algorithm achieves an approximation factor of 2 for the metric k-center problem.

Main Lemma

- **Lemma 4.3**

  For j as defined in the algorithm, $cost(e_j) \leq \text{OPT}$.

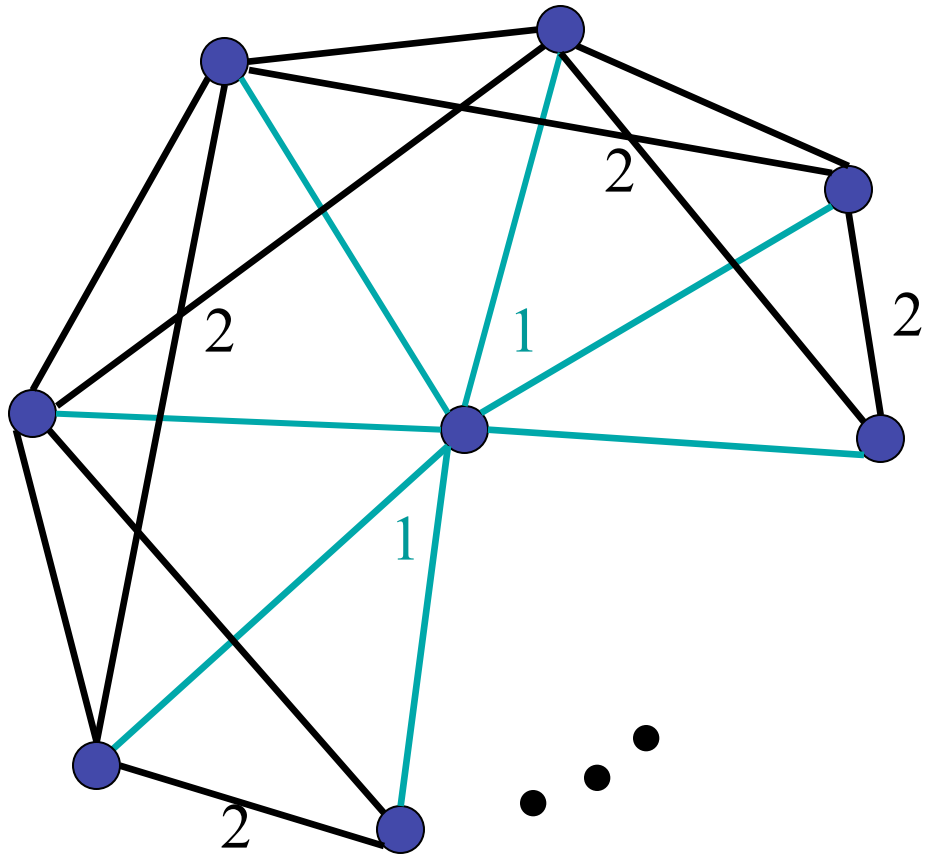
Proof.

- For evry $r < j$ we have that $|I_r| > k$.
- Now by Lemma 4.1 $\text{dom}(G_r) \geq |I_r| > k$.
- So $r^* > r$, and $r^* \geq j$.
- $cost(e_j) \leq \text{OPT}$

Proof of Theorem 4.2

- A maximal independent set I_j in a graph G_j^2 is also a dominating set.
- Thus there exist stars in G_j^2 centered on the vertices of I_j, covering all vertices.
- By the triangle inequality, each edge used in constructing these stars has cost at most $2cost(e_j)$.
- Lemma 6.3 implies $2\ cost(e_j) \leq 2$ OPT.

Tight Example ($k = 1$)



Metric weighted k-center

- *Given* a complete undirected graph $G = (V, E)$ with nonnegative edge costs satisfying the triangle inequality, a weight function on vertices, w: $V \rightarrow \mathbf{R}^+$ and a bound $W \in \mathbf{R}^+$. For any set $S \subseteq V$ and vertex v define $connect(v,S) = \min\{cost(u,v) | u \in S\}$.
- *Find* a set $S \subseteq V$ of total weight at most W, so as to minimize $\max_v\{connect(v,S)\}$.

Weight dominating set

- Let wdom(G) denote the weight of minimum weight dominating set in G.
- Calculating wdom(G) is *NP*-hard.

Parametric pruning

- Sort the edges of G in nondecreasing order of cost, i.e. $cost(e_1) \leq cost(e_2) \leq \ldots \leq cost(e_m)$.
- Let $G_i = (V, E_i)$, where $E_i = \{e_1, e_2, \ldots, e_i\}$.
- We need to find the smallest index индекс i such that $\text{wdom}(G_i) \leq W$. If i^* is this index, then the cost of the optimal solution is $\text{OPT} = cost(e_{i^*})$.

Lightest neighbors

- Given a vertex weighted graph $G = (V, E)$ let I be an independent set in G^2.
- For each $u \in I$, let $s(u)$ denote a lightest neighbor of u in G, where u is also considered a neighbor of itself.
- Let $S = \{s(u) \mid u \in I\}$.

Lower Bound

- **Lemma 4.4**

  Given graph H. Let I be an independent set in H^2. Then $w(S) \leq \text{wdom}(H)$.

Proof.

- Let D be a minimum weight dominating set of H.
- Then the exists a set of disjoint stars in H, centered on the vertices of D and covering all the vertices.
- Since each of these stars becomes a clique in H^2, the set I can pick at most one vertex from each of them.
- Thus each vertex in I has a center of the corresponding star available as a neighbor in H. Hence, $w(S) \leq \text{wdom}(H)$.

Hochbaum-Shmoys Algorithm-2

Input (G, $cost$: $E \rightarrow \mathbf{Q}^+$, w: $V \rightarrow \boldsymbol{R}^+$,W)

1) Construct $G_1^2, G_2^2, \ldots, G_m^2$.
2) Compute a maximal independent set I_r , in each graph G_r^2.
3) Compute $S_r = \{s_r(u) \mid u \in I_r \}$
4) Find the minimum index r such that $w(S_r) \leq W$, say j.

Output (S_j)

Approximation ratio of Hochbaum-Shmoys Algorithm-2

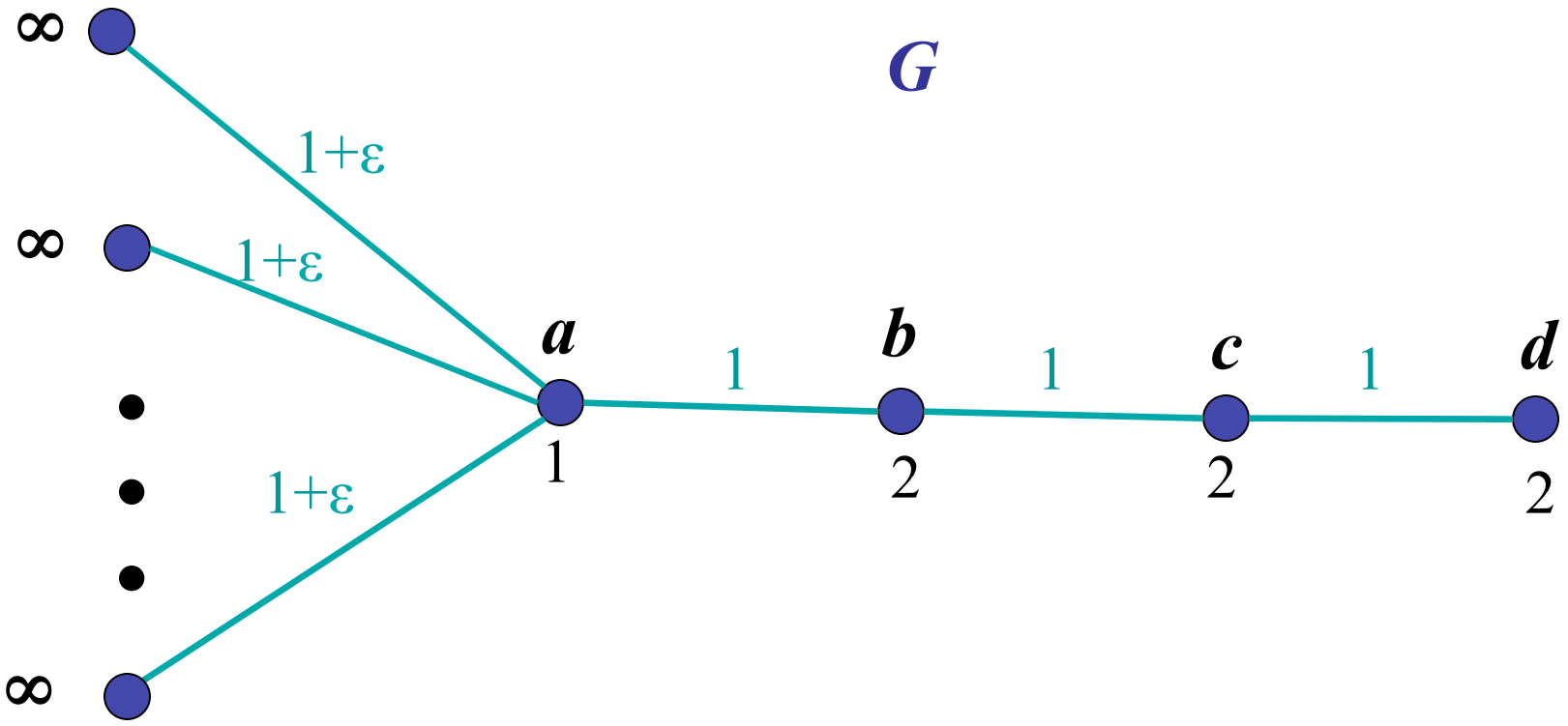
Theorem 4.5

Hochbaum-Shmoys Algorithm-2 achieves an approximation factor of 3 for the metric weighted k-center problem.

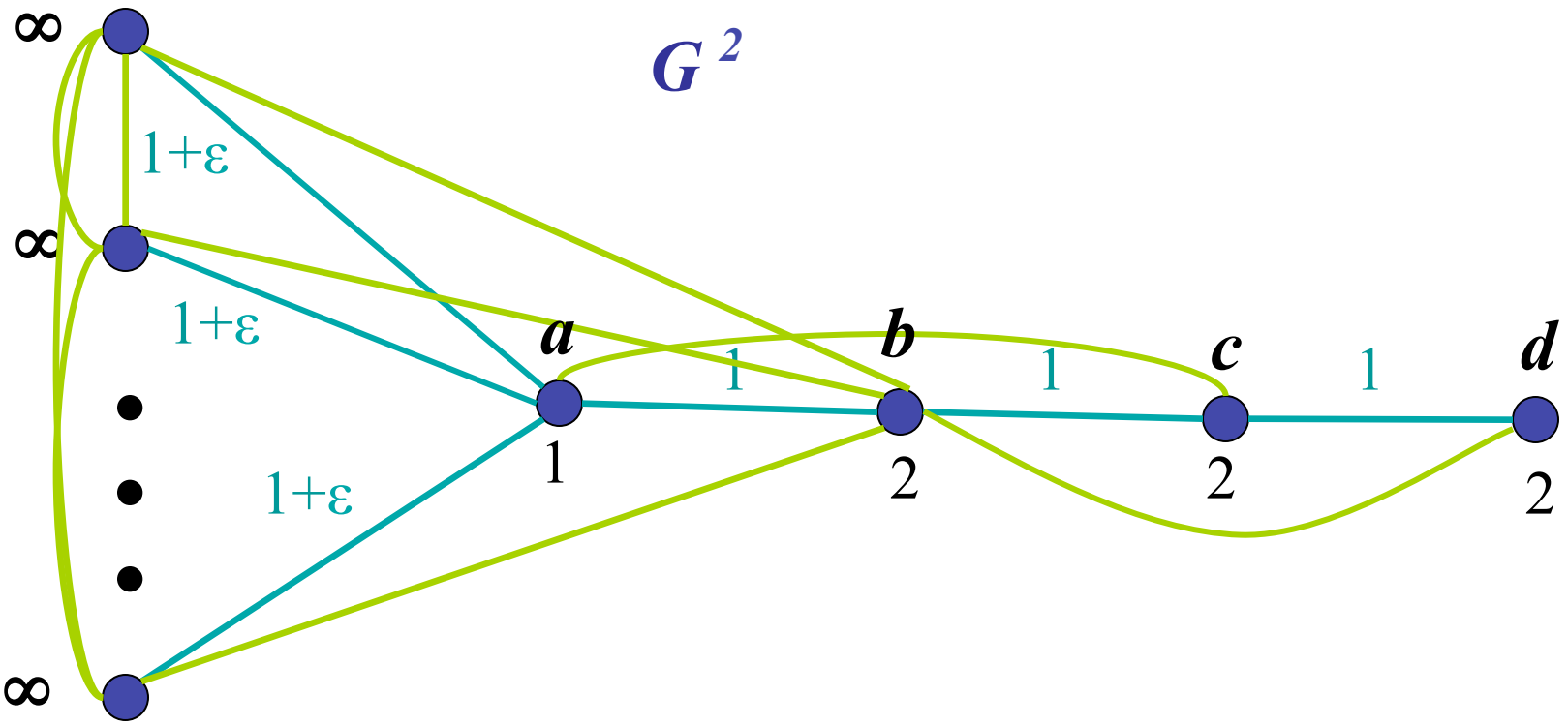
Proof

- By Lemma 4.4, $cost(e_j)$ is a lower bound on OPT; the argument is identical to that in Lemma 4.3. Since I_j is a dominating set in G_j^2, we can cover V with stars of G_j^2 centered in vertices of I_j. By the triangle inequality these stars use edges of cost at most 2 $cost(e_j)$.
- Each star center is adjacent to a vertex in S_j, using an edge of cost at most $cost(e_j)$. Move each of the centers to the adjacent vertex in S_j and redefine the stars. Again, by the triangle inequality, the largest edge cost used in constructing the final stars is at most $cost(e_j)$.

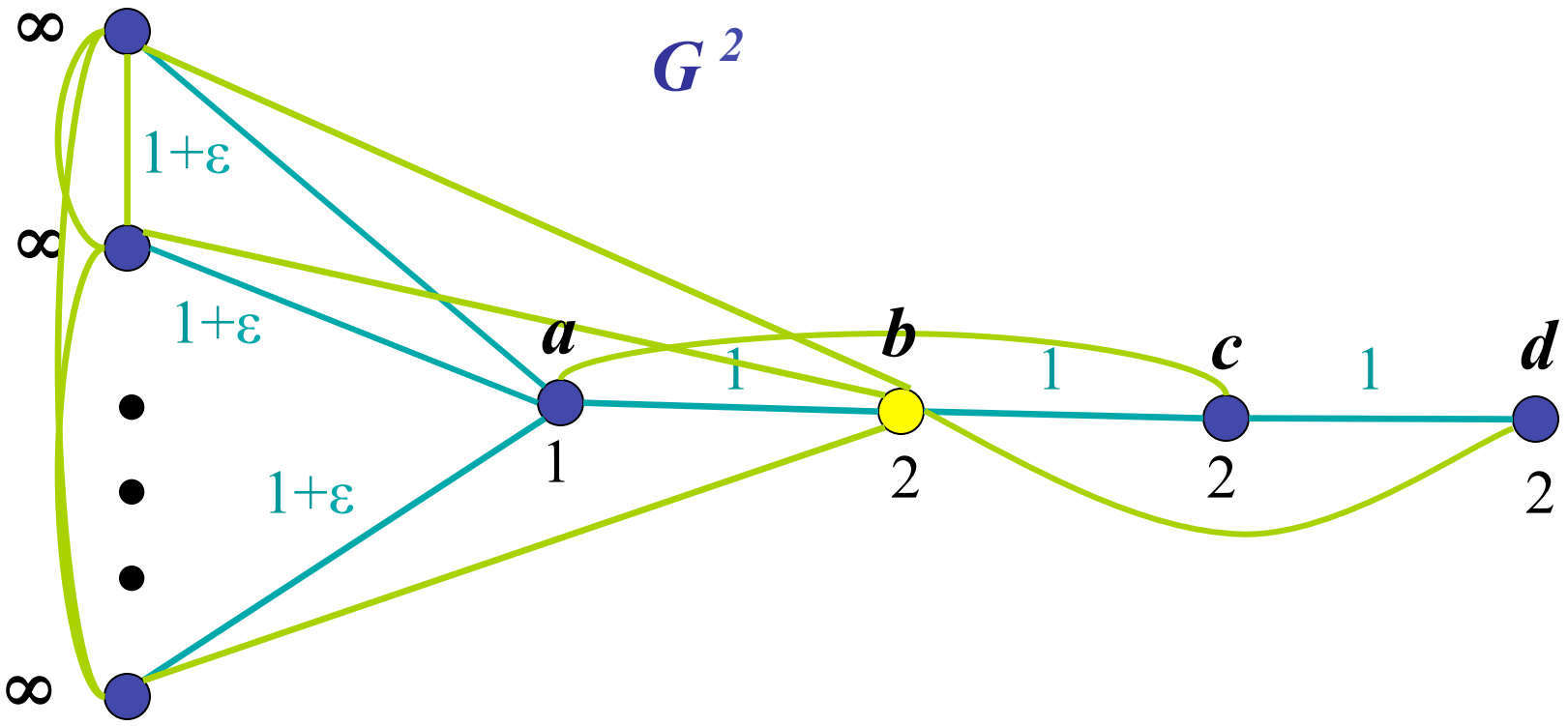
Tight Example ($W = 3$)



Tight Example

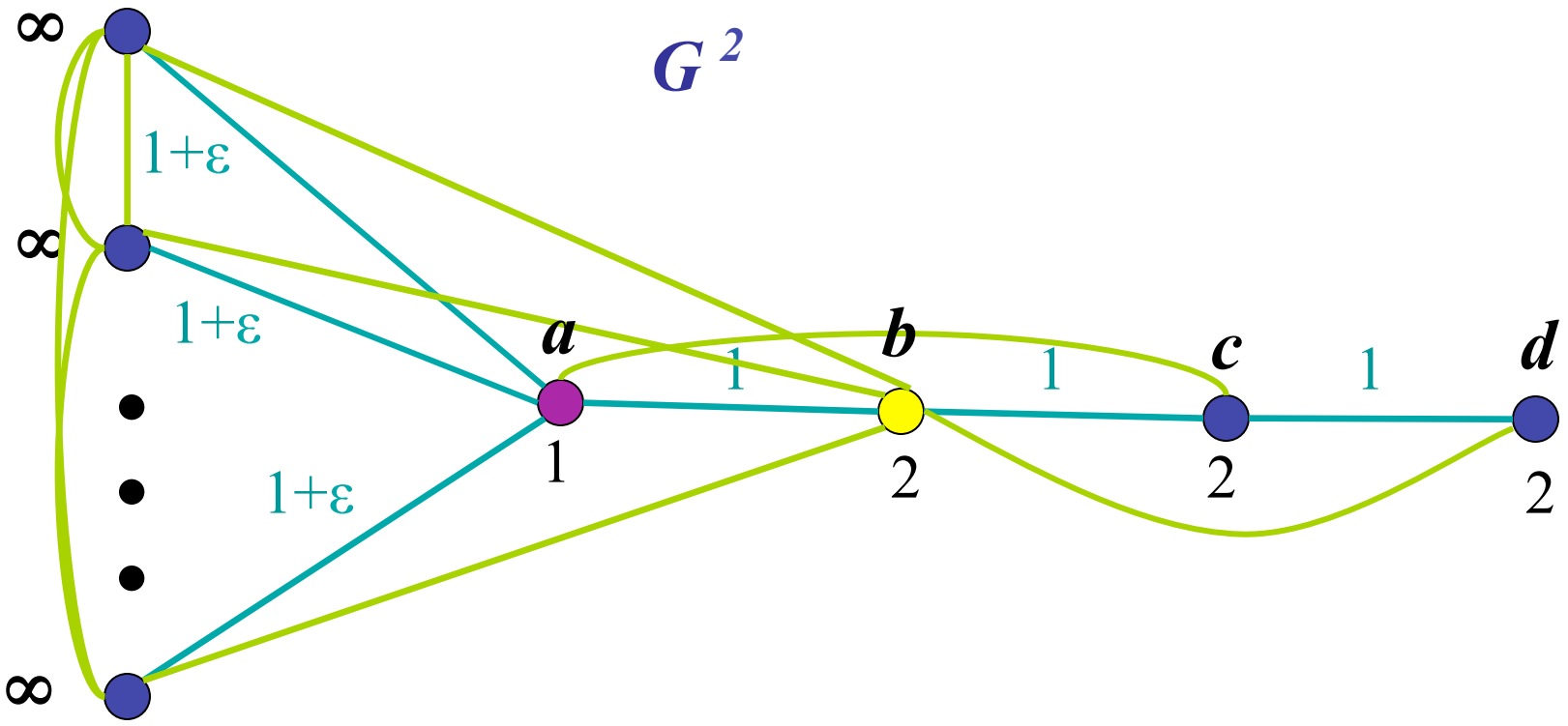


Tight Example



$I_{n+3}=\{b\}$

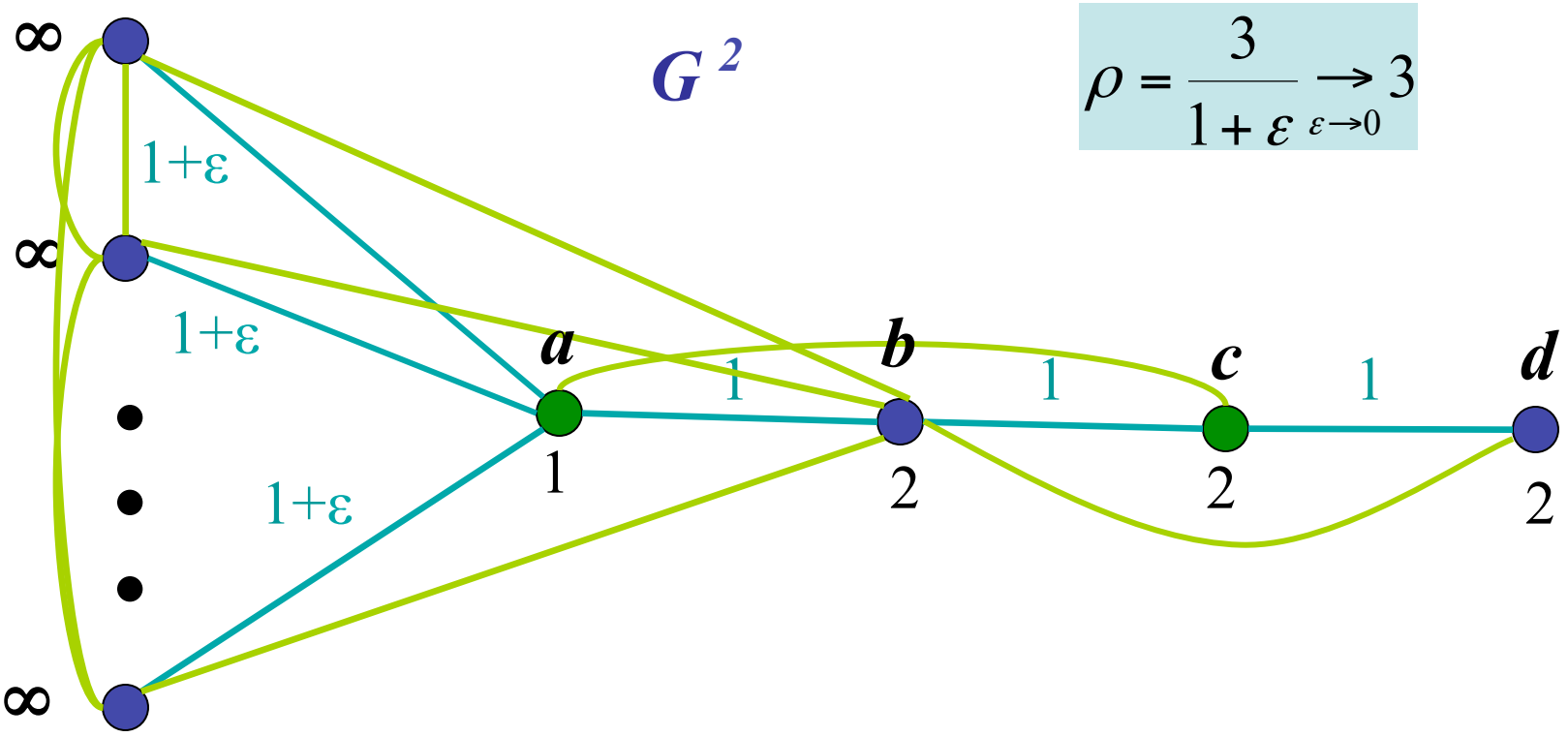
Tight Example



$I_{n+3}=\{b\}$

$S_{n+3}=\{a\}$

Tight Example



G^2

$$\rho = \frac{3}{1+\varepsilon} \xrightarrow[\varepsilon \to 0]{} 3$$

$I_{n+3}=\{b\}$

$S_{n+3}=\{a\}$

$OPT=\{a, c\}$

Exercise 3.1

- Consider the following greedy algorithm for metric TSP.
- Find the two closest cities, say v_i and v_j, and start by building a tour on that pair of cities; the tour consists of going from v_i to v_j and then back to v_i again. This is the first iteration. In each subsequent iteration, we extend the tour on the current subset S by including one additional city, until we include the full set of cities. In each iteration, we find a pair of cities $v_i \in S$ and $v_j \notin S$ for which the cost c_{ij} is minimum; let v_k be the city that follows v_i in the current tour on S. We add v_j to S, and insert v_j into the current tour between v_i and v_k.
- **Prove** that this greedy algorithm for metric TSP is a 2-approximation algorithm.

Exercise 3.2

- Let $G=(V,E)$ be a complete graph with edge costs satisfying the triangle inequality, and $V' \subseteq V$ be a set of even cardinality. Prove or disprove: The cost of a minimum cost perfect matching on V' is bounded above by the cost of a minimum cost perfect matching on V.