Gems of TCS

Approximation Algorithms

Sasha Golovnev

January 28, 2020

Approximation Algorithms

- Optimal exact solution OPT (ex: shortest TSP cycle)

Approximation Algorithms

- Optimal exact solution OPT (ex: shortest TSP cycle)
- OPT is too hard to find (ex: **NP**-hard)

Approximation Algorithms

- Optimal exact solution OPT (ex: shortest TSP cycle)
- OPT is too hard to find (ex: **NP**-hard)
- A *k*-approximation algorithm finds a solution $\leq k \times \mathrm{OPT}$

Approximation Algorithms

- Optimal exact solution OPT (ex: shortest TSP cycle)
- OPT is too hard to find (ex: **NP**-hard)
- A *k*-approximation algorithm finds a solution $\leq k \times \text{OPT}$

2-apx will always
solution $\leq 2 \cdot \text{OPT}$

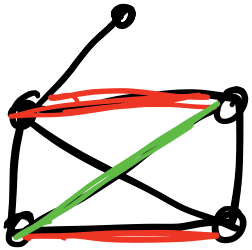
- Possibly efficiently! (ex: poly time)

Approximation Algorithms

- Optimal exact solution OPT (ex: shortest TSP cycle)
- OPT is too hard to find (ex: **NP**-hard)
- A *k-approximation* algorithm finds a solution $\leq k \times \text{OPT}$
- Possibly efficiently! (ex: poly time)
- When do we use approximation algorithms?

Matchings

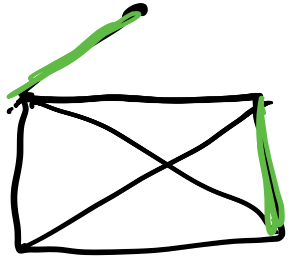
- A Matching in a graph is a set of edges without common vertices



Matchings

- A Matching in a graph is a set of edges without common vertices

- A Maximal Matching is a matching which cannot be extended to a larger matching



Matchings

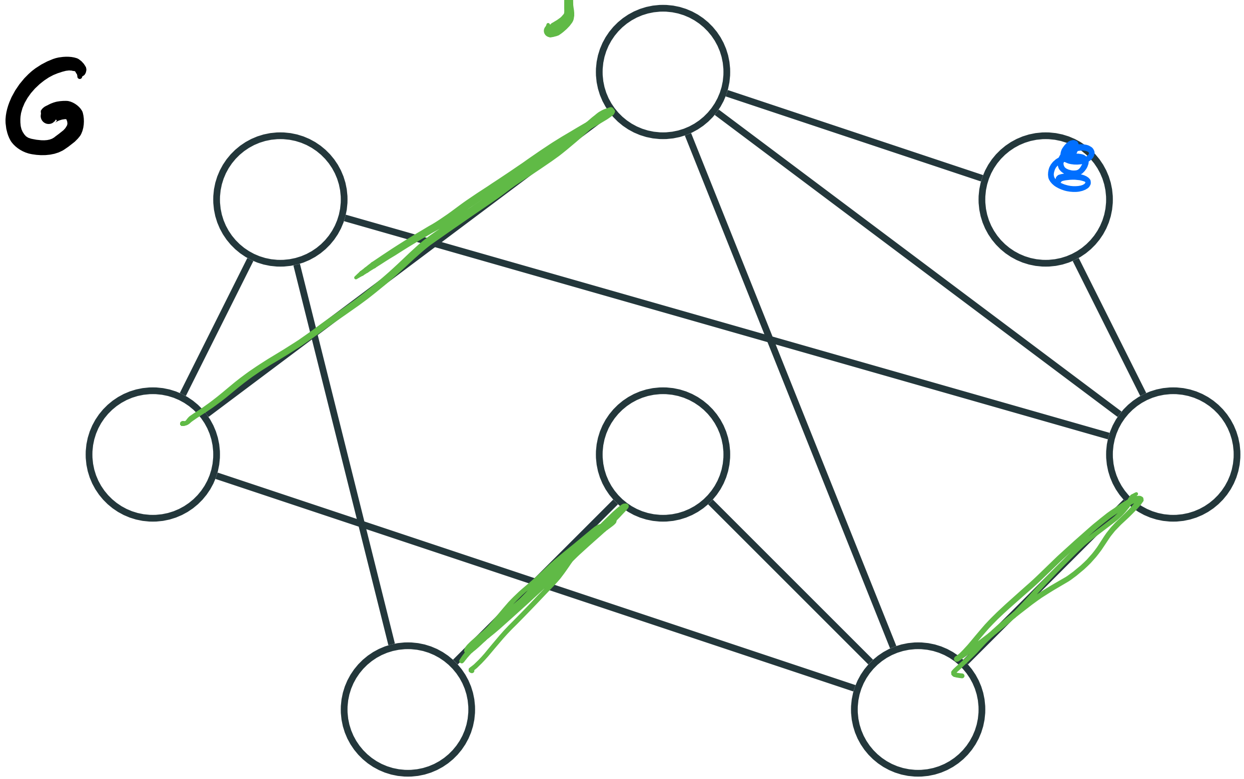
- A Matching in a graph is a set of edges without common vertices
- A Maximal Matching is a matching which cannot be extended to a larger matching
- A Maximum Matching is a matching of the largest size

is a Maximal matching too

Matchings. Examples

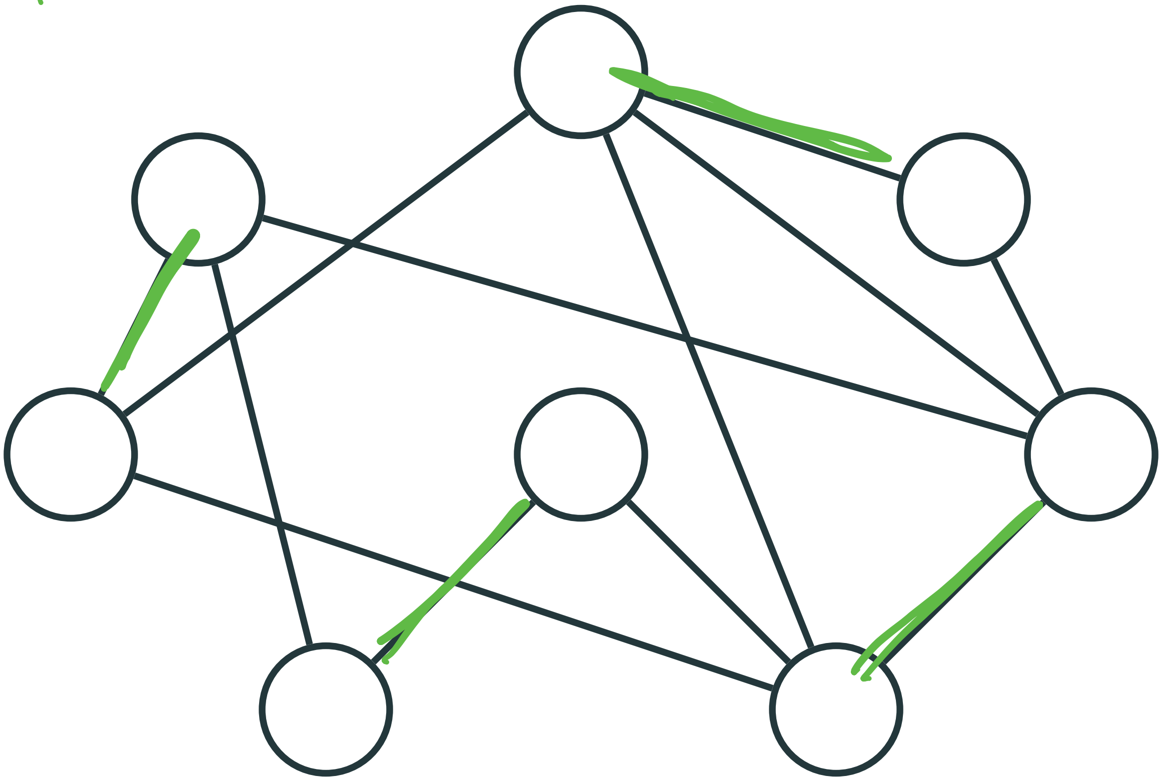
Maximal matching

not Maximum matching



Matchings. Examples

Maximum matchings

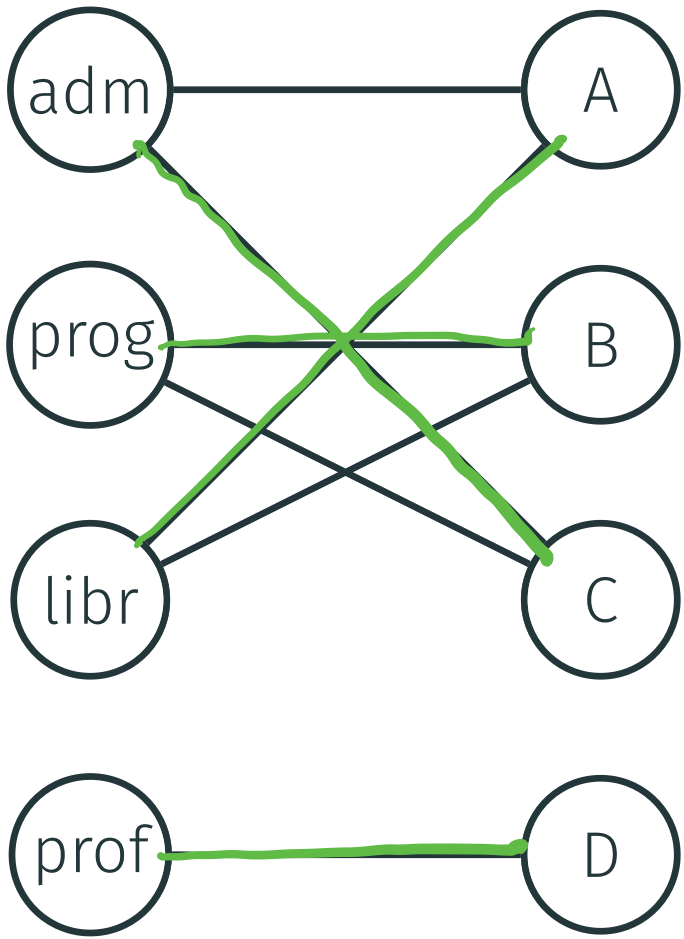


Job Assignment

	Alice	Ben	Chris	Diana
Administrator	+		+	
Programmer		+	+	
Librarian	+	+		
Professor				+

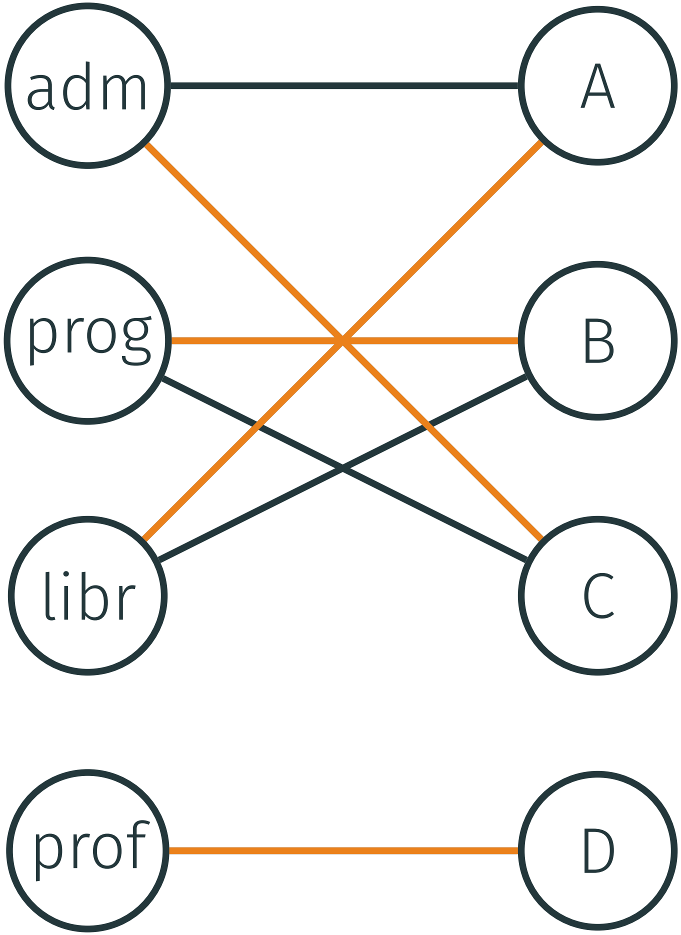
Job Assignment

Job Assignment



Perfect matching
=
Matching that covers all vertices

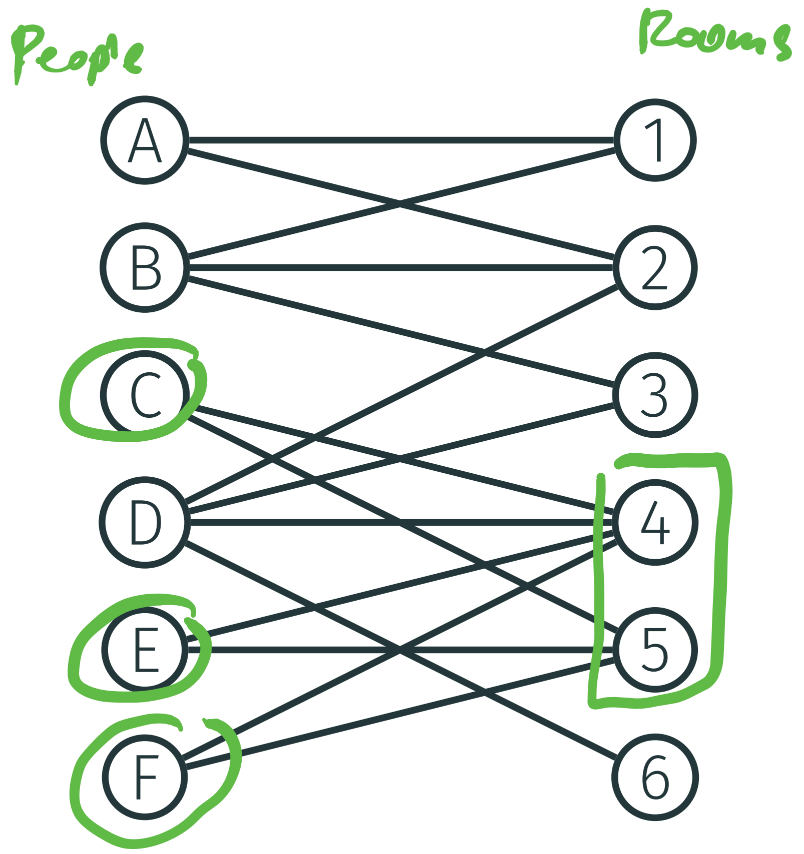
Job Assignment



Room Assignment

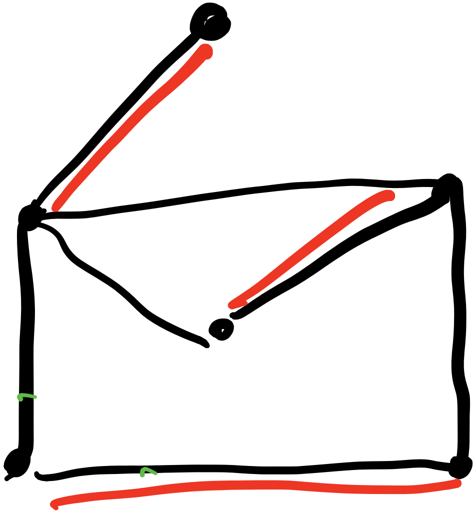
	R# 1	R# 2	R# 3	R# 4	R# 5	R# 6
Aaron	+	+				
Bianca	+	+	+			
Carol				+	+	
Dana		+	+	+		+
Emma				+	+	
Francis				+	+	

Room Assignment



Maximal Matching

Can be found in polynomial time by a greedy algorithm



Algorithms

Maximal Matching

Can be found in polynomial time by a greedy algorithm

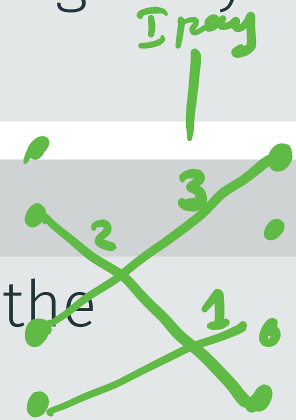
Maximum Matching

Can be found in polynomial time by the blossom algorithm

Algorithms

Maximal Matching

Can be found in polynomial time by a greedy algorithm



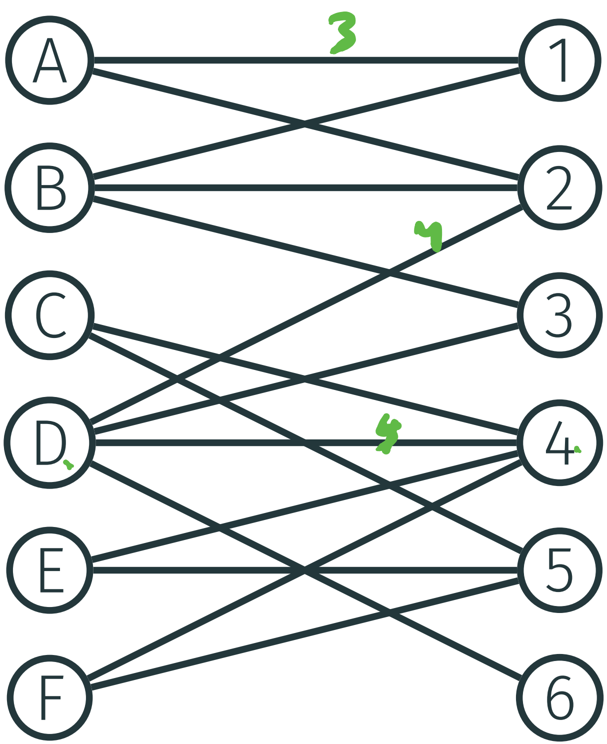
Maximum Matching

Can be found in polynomial time by the blossom algorithm

Minimum Weight Perfect Matching

Can be found in polynomial time by Edmonds' algorithm

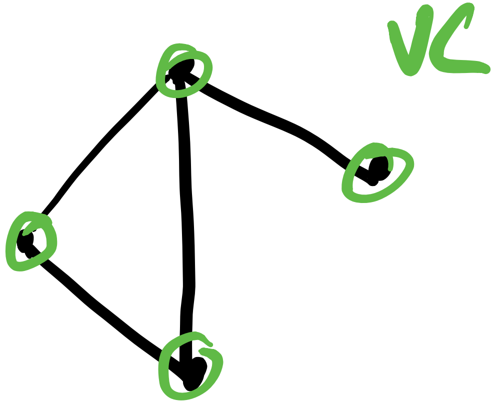
Room Assignment



Vertex Cover

Vertex Covers

- A Vertex Cover of a graph G is a set of vertices C such that every edge of G is connected to some vertex in C.



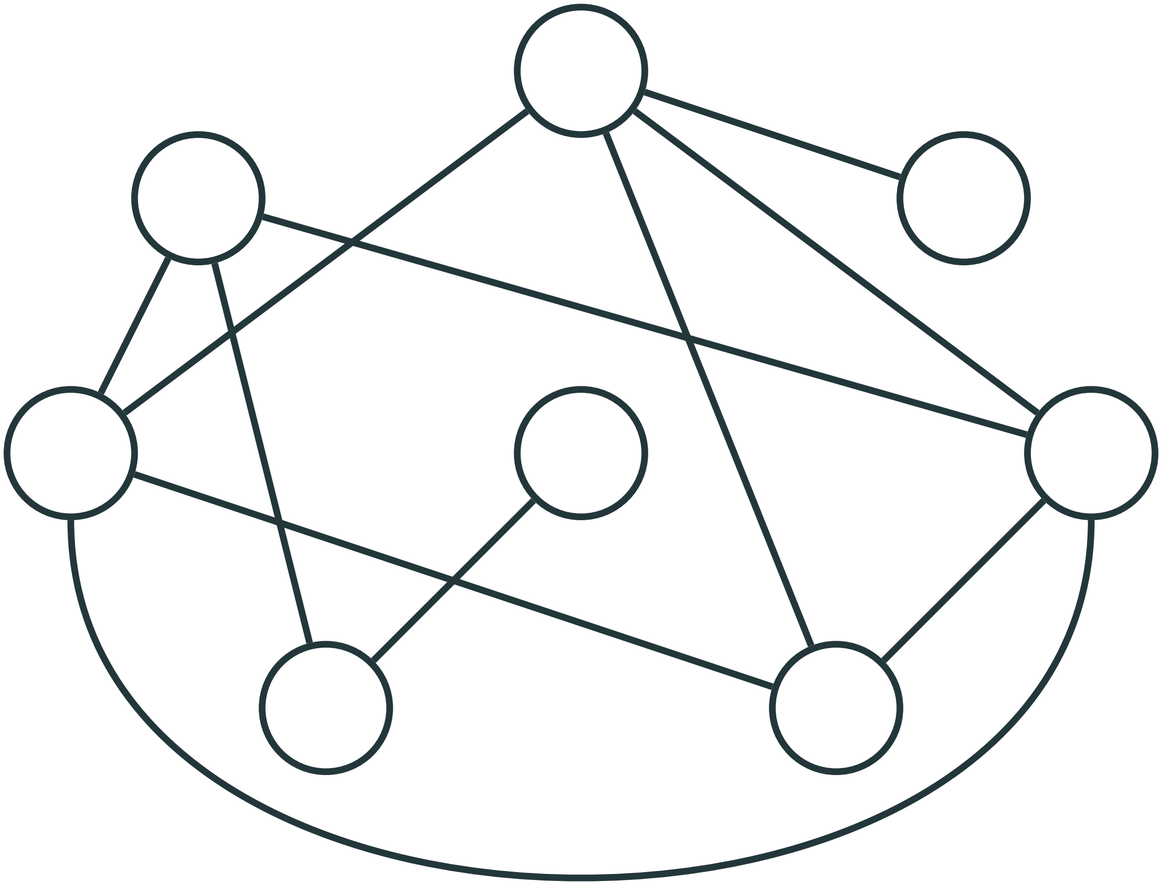
Vertex Covers

- A Vertex Cover of a graph G is a set of vertices C such that every edge of G is connected to some vertex in C.
- A Minimal Vertex Cover is a vertex cover which does not contain other vertex covers.

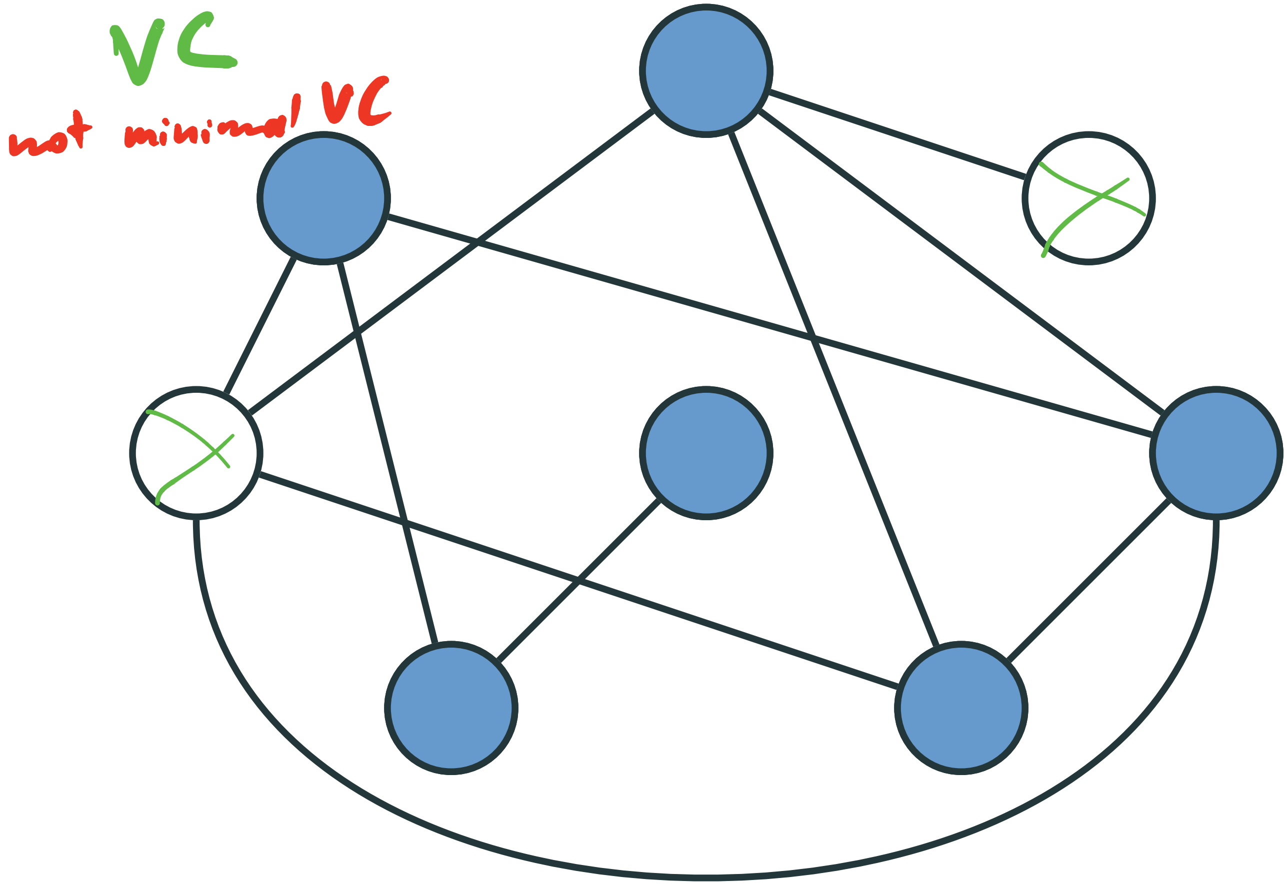
Vertex Covers

- A Vertex Cover of a graph G is a set of vertices C such that every edge of G is connected to some vertex in C.
- A Minimal Vertex Cover is a vertex cover which does not contain other vertex covers.
- A Minimum Vertex Cover is a vertex cover of the smallest size.

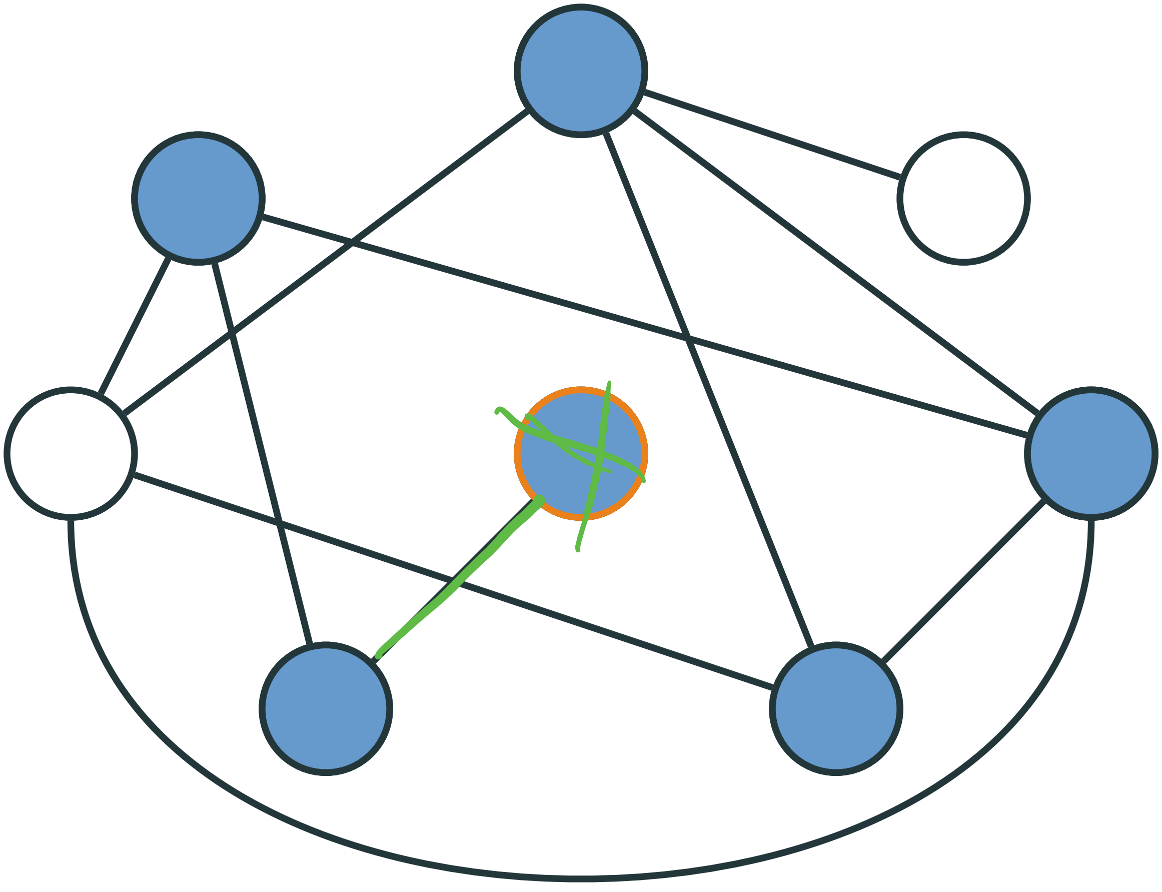
Vertex Covers: Examples



Vertex Covers: Examples

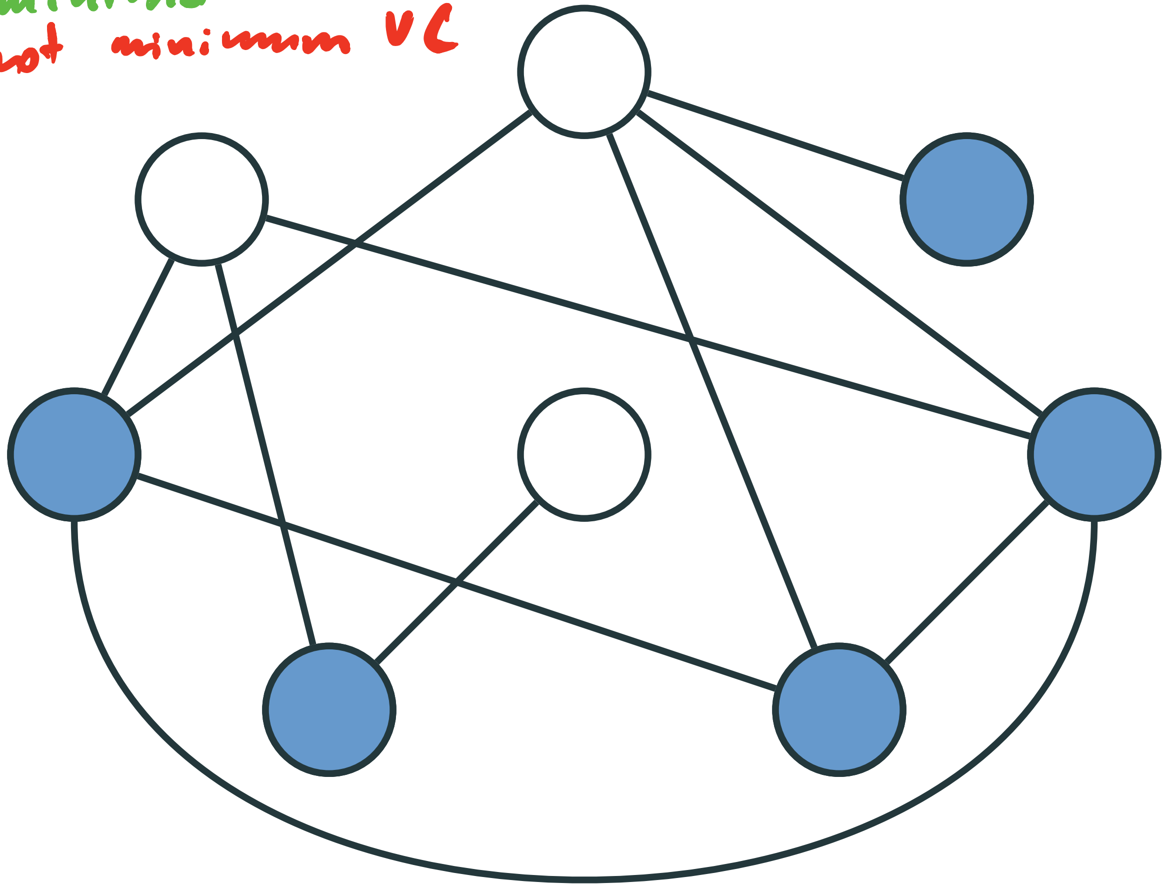


Vertex Covers: Examples

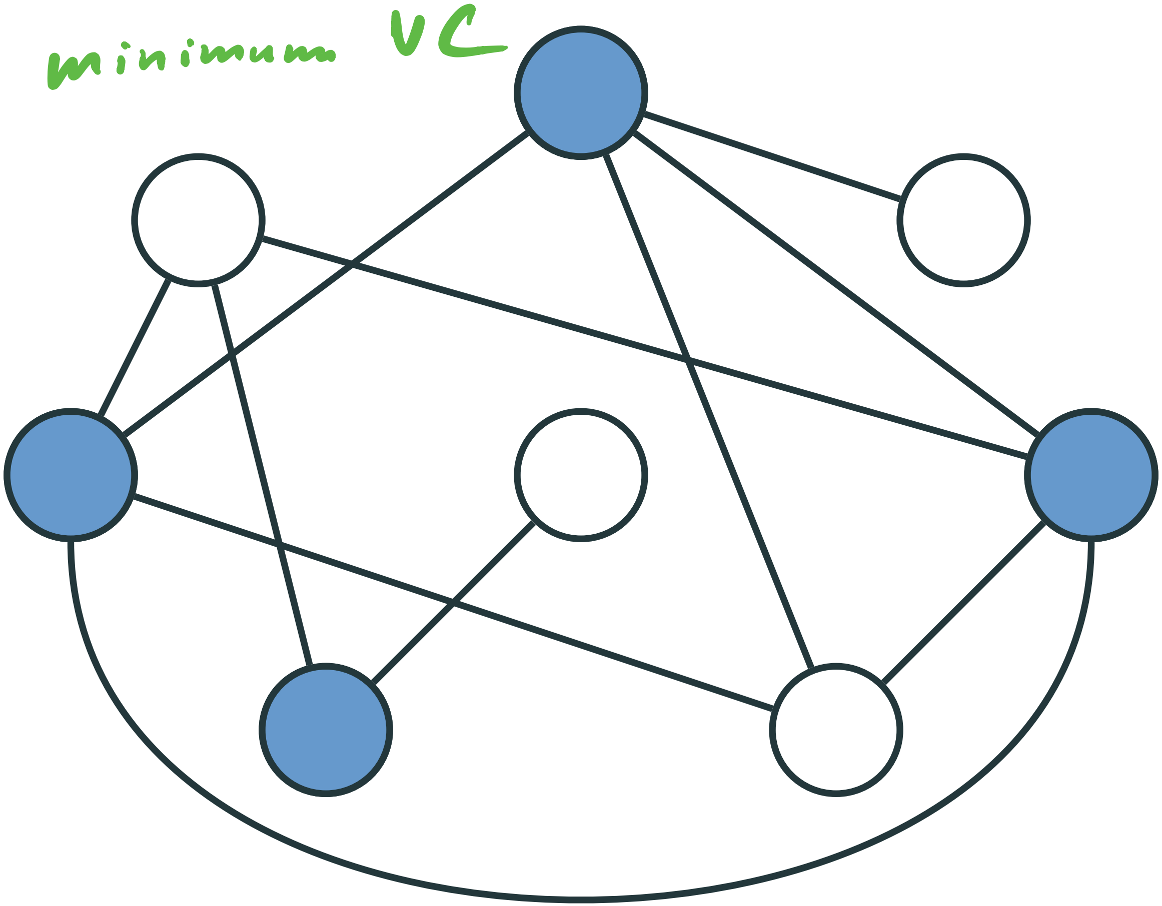


Vertex Covers: Examples

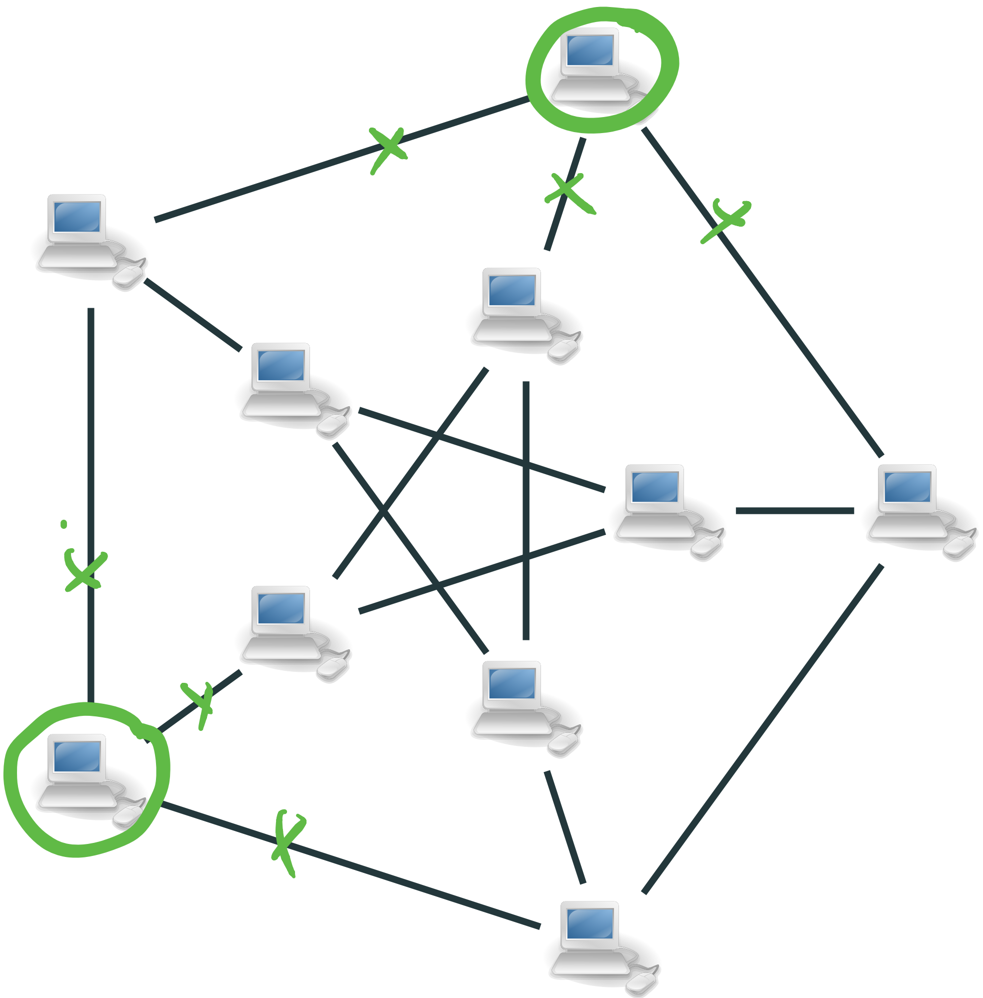
minimal VC
not minimum VC



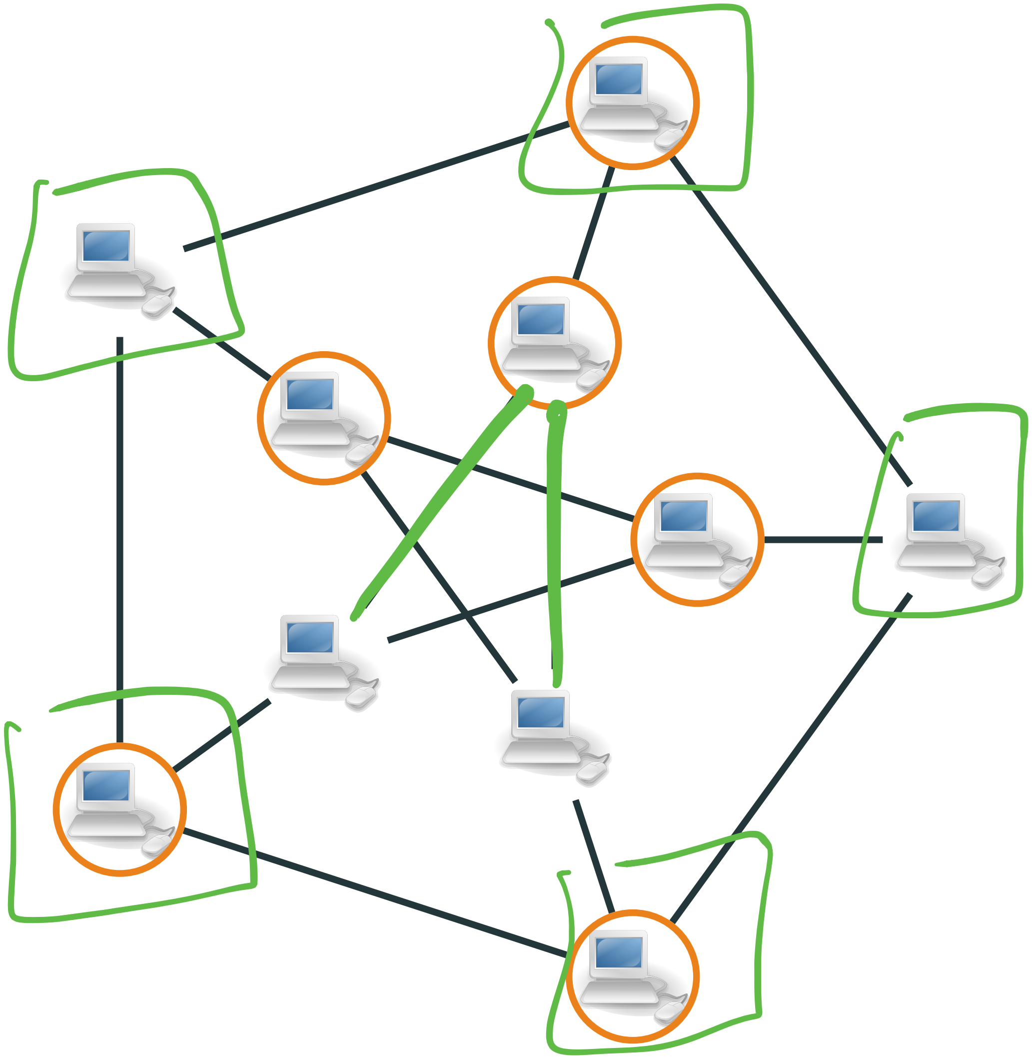
Vertex Covers: Examples



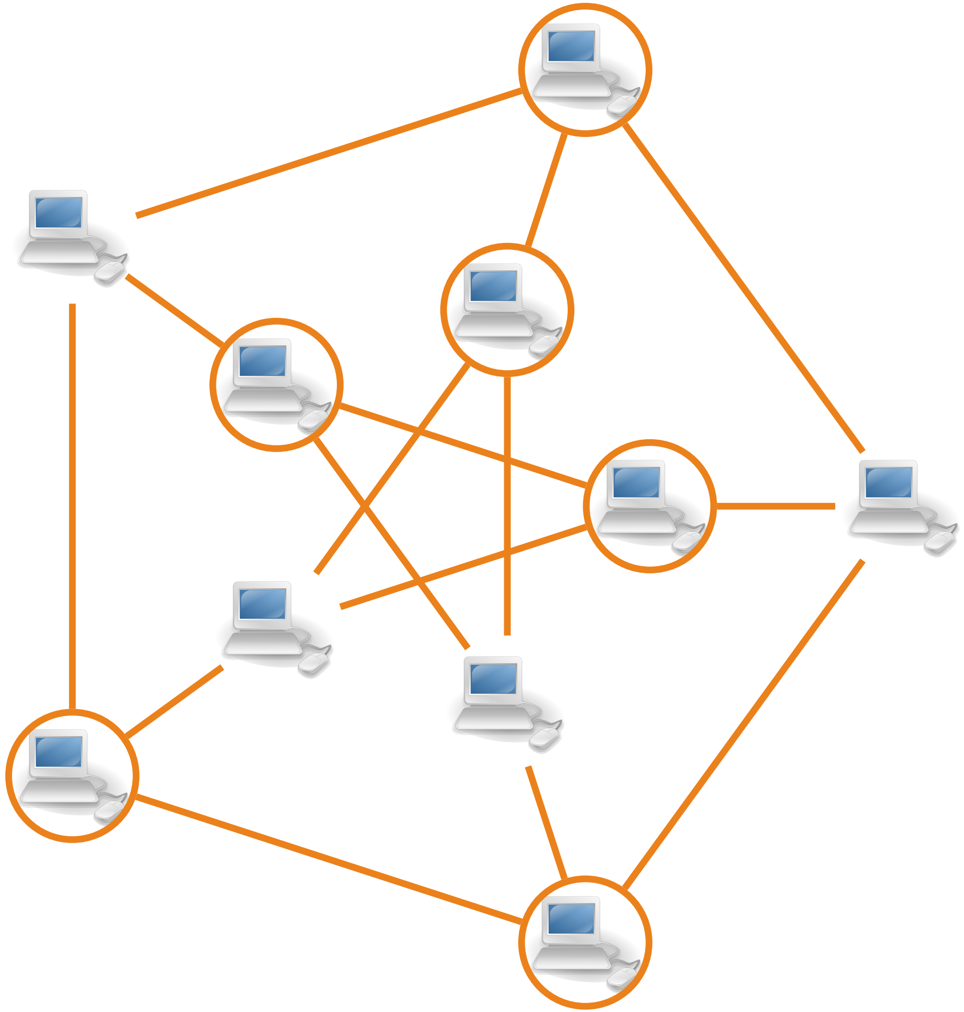
Antivirus System



Antivirus System



Antivirus System

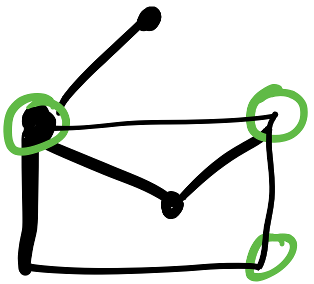


Algorithms

Minimal Vertex Cover

Can be found in polynomial time by a greedy algorithm

EASY



Algorithms

Minimal Vertex Cover

Can be found in polynomial time by a greedy algorithm

Minimum Vertex Cover

Is **NP**-hard. We only know exponential-time algorithms

Approximation Algorithm

- $M \leftarrow$ maximal matching in G

Approximation Algorithm

- $M \leftarrow$ maximal matching in G
- return all vertices in M

1. It runs in poly time
2. It's 2-approximate

Equivalent Algorithm

- $C \leftarrow \emptyset$

Equivalent Algorithm

Maximal matching

$G = (V, E)$

- $C \leftarrow \emptyset$
- while $E \neq \emptyset$

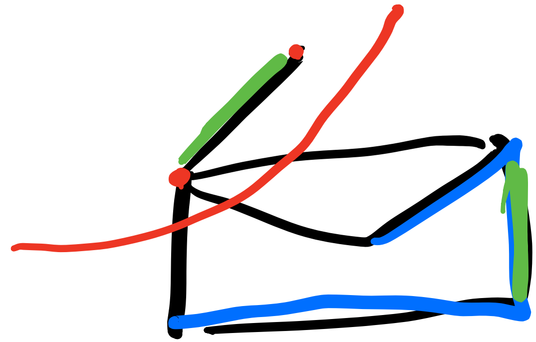
Equivalent Algorithm

- $C \leftarrow \emptyset$
- while $E \neq \emptyset$
    - $\{u, v\} \leftarrow$ any edge from E

Equivalent Algorithm

- $C \leftarrow \emptyset$
- while $E \neq \emptyset$
  - $\{u, v\} \leftarrow$ any edge from E
  - add u, v to C

Equivalent Algorithm



- $C \leftarrow \emptyset$
- while $E \neq \emptyset$
  - $\{u, v\} \leftarrow$ any edge from E
  - add u, v to C
  - delete from E all edges incident to u or v
- return C

Runs in poly time

Proof

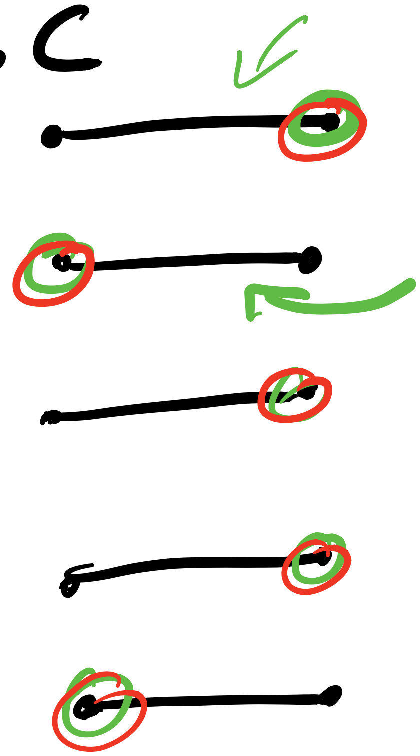
Lemma

This algorithm runs in polynomial time and is 2-approximate: it returns a vertex cover that is at most twice larger then a minimum vertex cover.

OPT is the size of minimum vertex

We select $\leq 2OPT$ vertices

Matching C



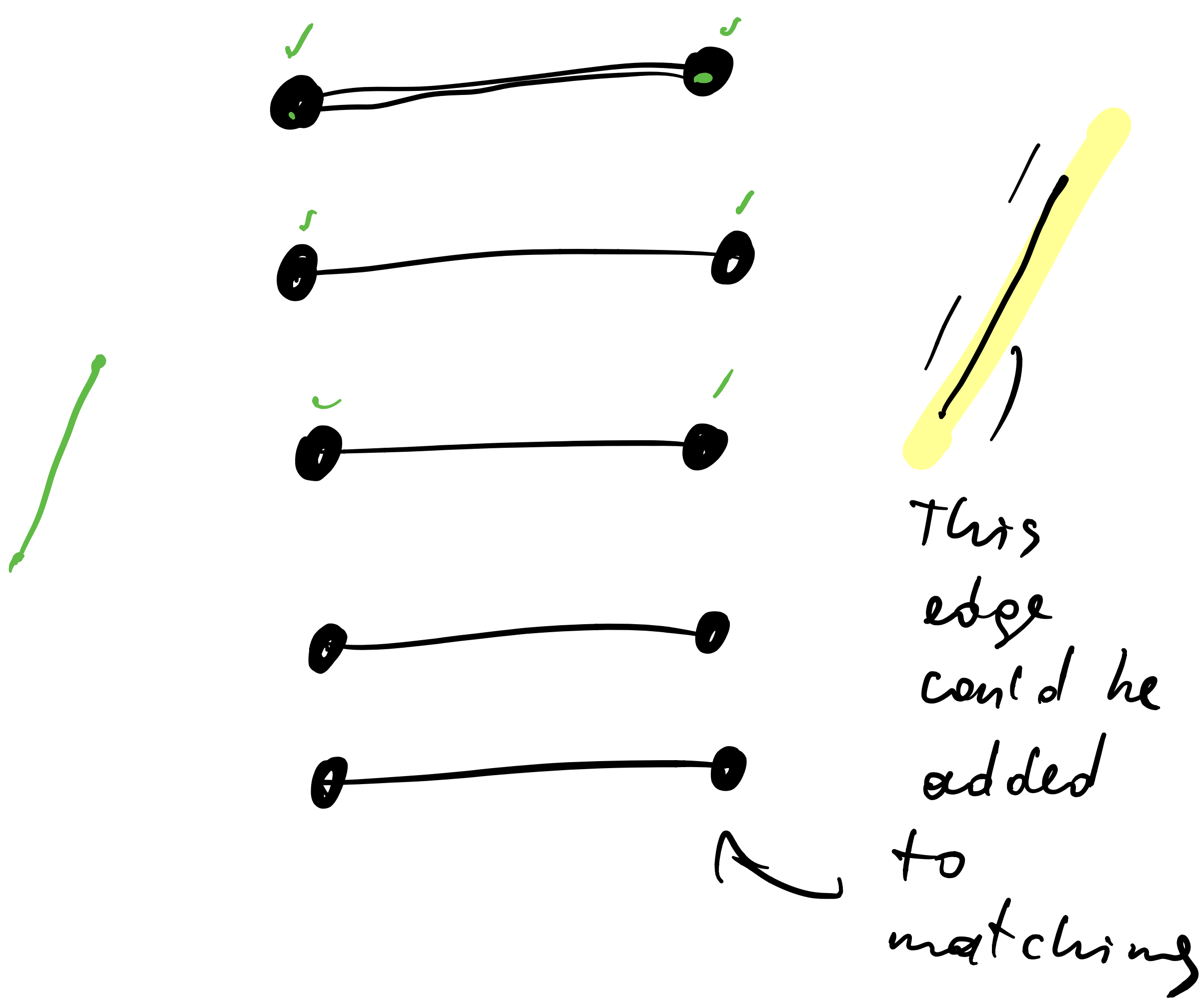
Minimum VC has size OPT

OPT ≥ # edges in matching ≥ $\frac{1}{2}$ # vertices in matching

our output = # vertices matching ≤

≤ 2 OPT

1. Runs in poly-time
2. Outputs some vertex cover
3. Its VC ≤ 2 · Minimum VC



Final Remarks

- The analysis is tight: there are graphs with matchings twice larger than vertex covers

Final Remarks

- The analysis is tight: there are graphs with matchings twice larger than vertex covers

- No 1.99-approximation algorithm is known

Under reasonable conj, there is no better apx-alg in poly time

Break

Matchings:

http://bit.ly/job_assignment

Vertex covers:

http://bit.ly/antivirus_system

Traveling Salesman

Approximation

- If $\mathsf{P} \neq \mathsf{NP}$, then there is no k-approximation algorithm for the general version of TSP for any constant k

Approximation

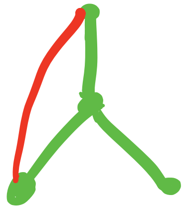
- If $\mathbf{P} \neq \mathbf{NP}$, then there is no k-approximation algorithm for the general version of TSP for any constant k
- Euclidean TSP: $w(u, v) = w(v, u)$ and $w(u, v) \leq w(u, z) + w(z, v)$

Approximation

- If $\mathbf{P} \neq \mathbf{NP}$, then there is no k-approximation algorithm for the general version of TSP for any constant k
- Euclidean TSP: $w(u, v) = w(v, u)$ and $w(u, v) \leq w(u, z) + w(z, v)$
- We will design a 2-approximation algorithm: it quickly finds a cycle that is at most twice longer than an optimal one

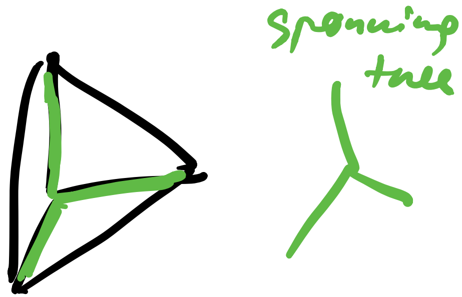
Definition

- A tree is a connected graph without cycles



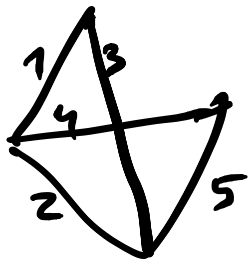
Definition

- A tree is a connected graph without cycles
- A tree is a connected graph on n vertices with $n - 1$ edges



Definition

- A tree is a connected graph without cycles
- A tree is a connected graph on n vertices with $n-1$ edges
- A Spanning Tree of a graph G is a subgraph of G that (i) is a tree and (ii) contains all vertices of G

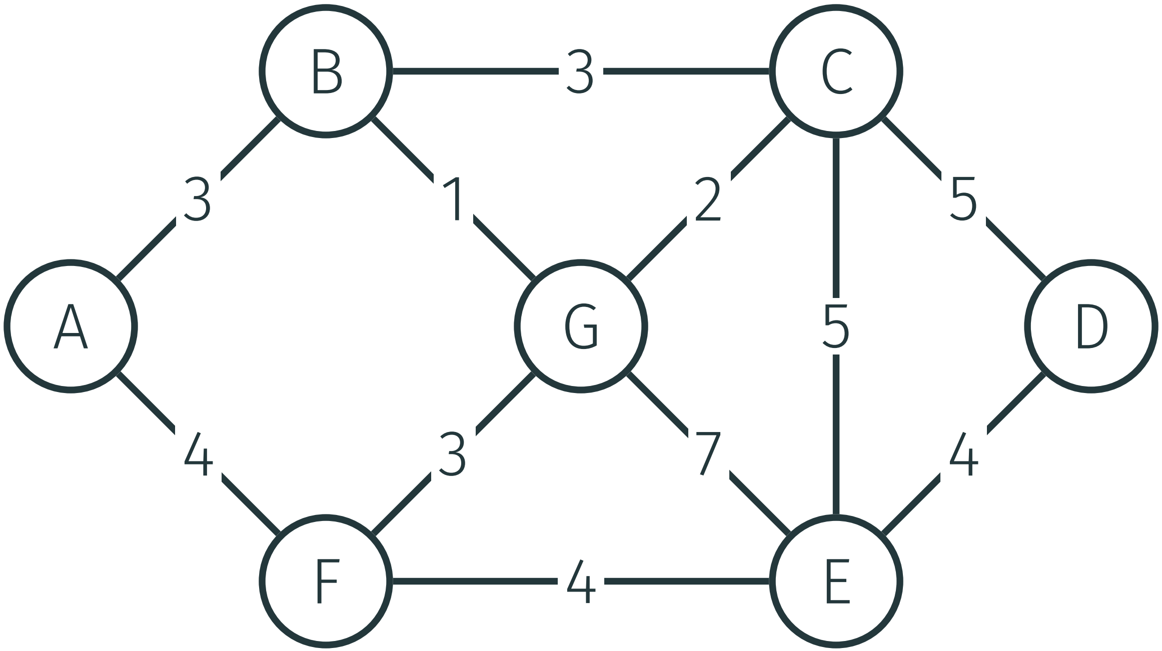


Definition

- A tree is a connected graph without cycles
- A tree is a connected graph on n vertices with $n - 1$ edges
- A Spanning Tree of a graph G is a subgraph of G that (i) is a tree and (ii) contains all vertices of G
- A Minimum Spanning Tree (MST) of a weighted graph G is a spanning tree of the smallest weight

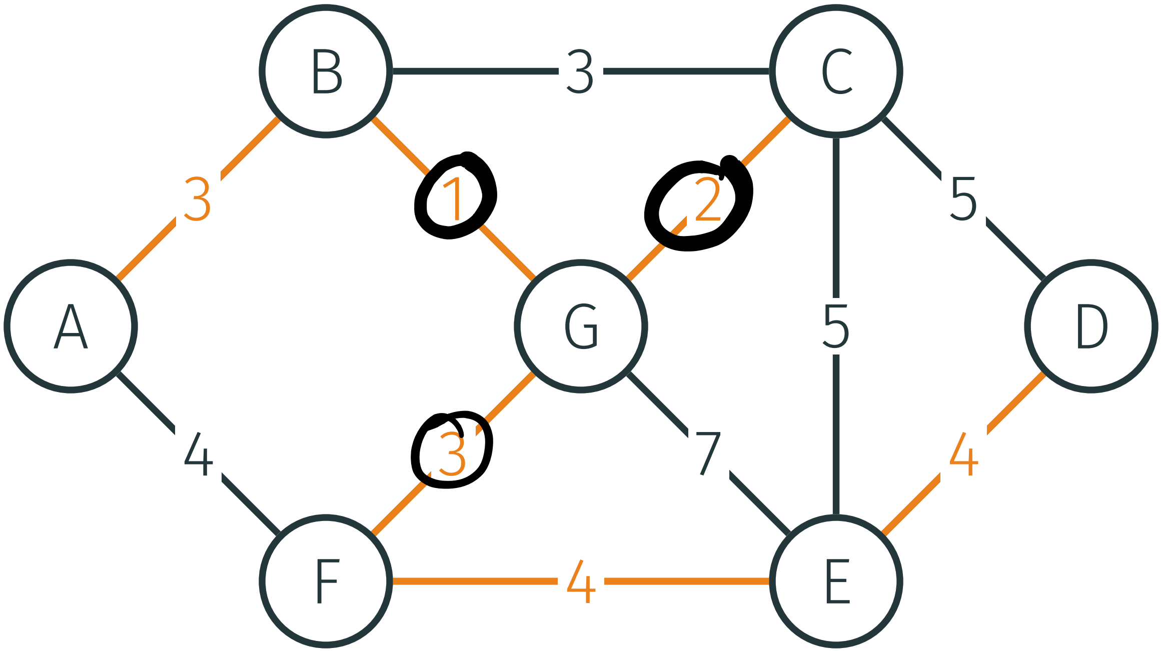
Kruskal's, Prim's

Minimum Spanning Tree: Examples



Minimum Spanning Tree: Examples

MST



Lemma

Let G be an undirected graph with non-negative edge weights. Then $MST(G) \leq TSP(G)$.

If I can find some cycle of length $\leq 2MST \leq 2TSP$

TSP

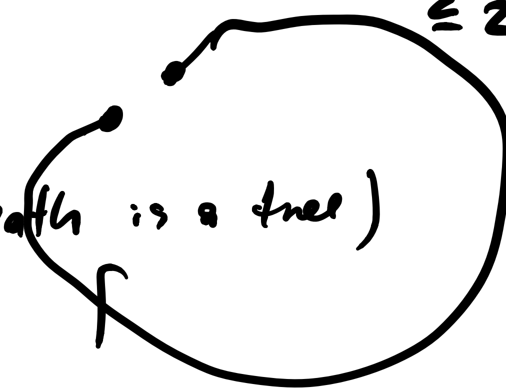
$\geq$ weight of this path

$\geq$ min weight of a path (path is a tree)

$\geq$ min weight of a tree

$= MST(G)$

TSP



Minimum Spanning Trees

Lemma

Let G be an undirected graph with non-negative edge weights. Then $\mathrm{MST}(G) \leq \mathrm{TSP}(G)$.

Proof

By removing any edge from an optimum TSP cycle one gets a spanning tree of G.

Eulerian Cycle

An Eulerian cycle (or path) visits every edge exactly once

Eulerian Cycle

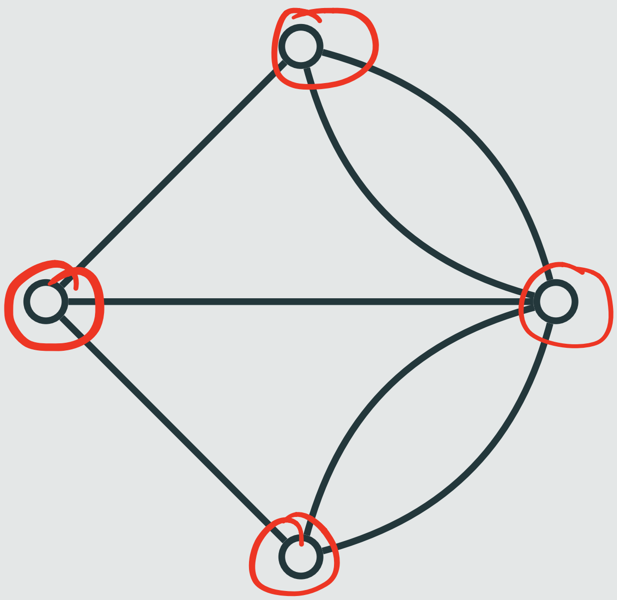
An Eulerian cycle (or path) visits every edge exactly once

Criteria

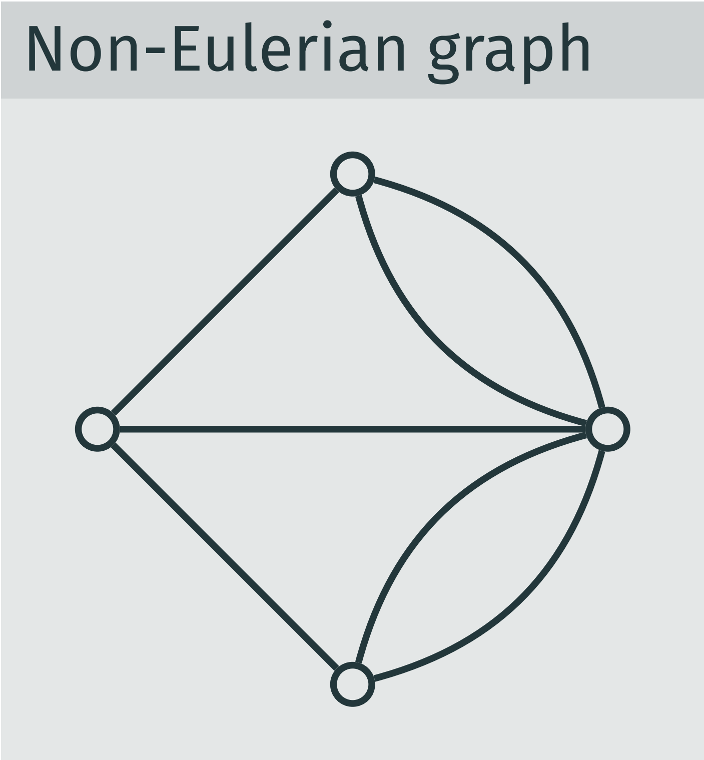
A connected undirected graph contains an Eulerian cycle, if and only if the degree of every node is even

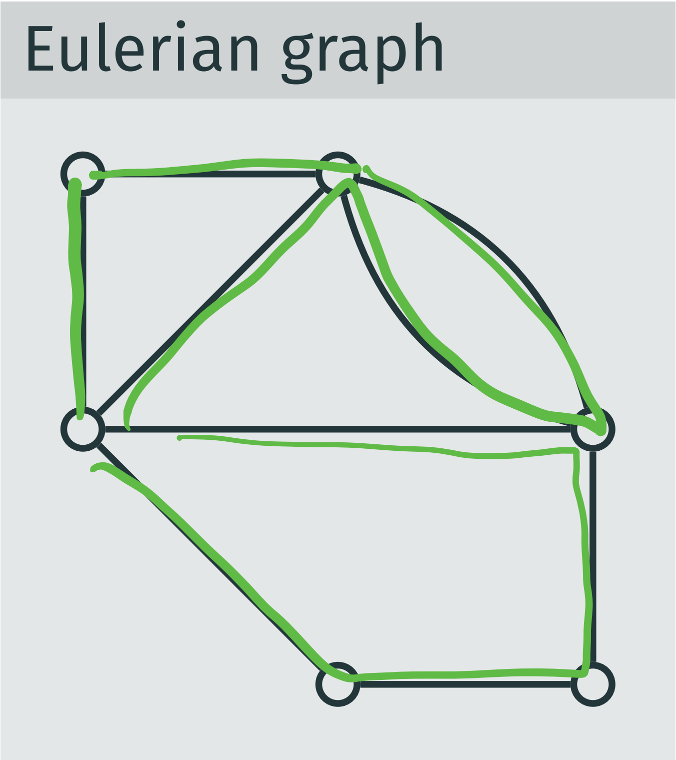
Example

Non-Eulerian graph



Example



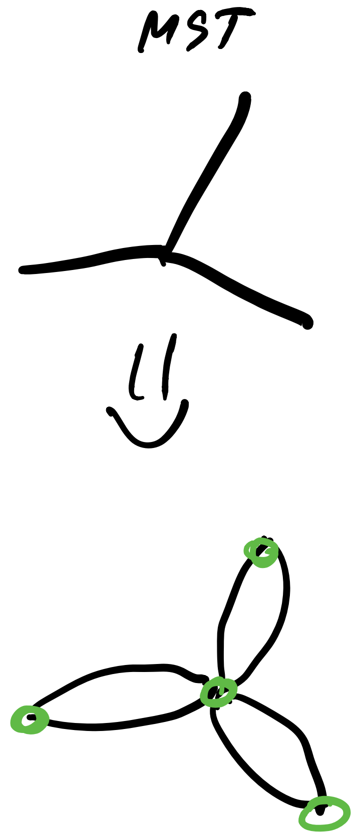


Algorithm

- $T \leftarrow$ minimum spanning tree of G

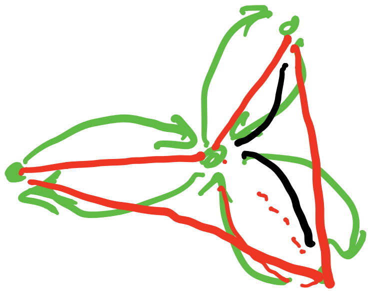
Algorithm

- $T \leftarrow$ minimum spanning tree of G
- $D \leftarrow T$ with each edge doubled



Algorithm

- $T \leftarrow$ minimum spanning tree of G
- $D \leftarrow T$ with each edge doubled
- find an Eulerian cycle C in D

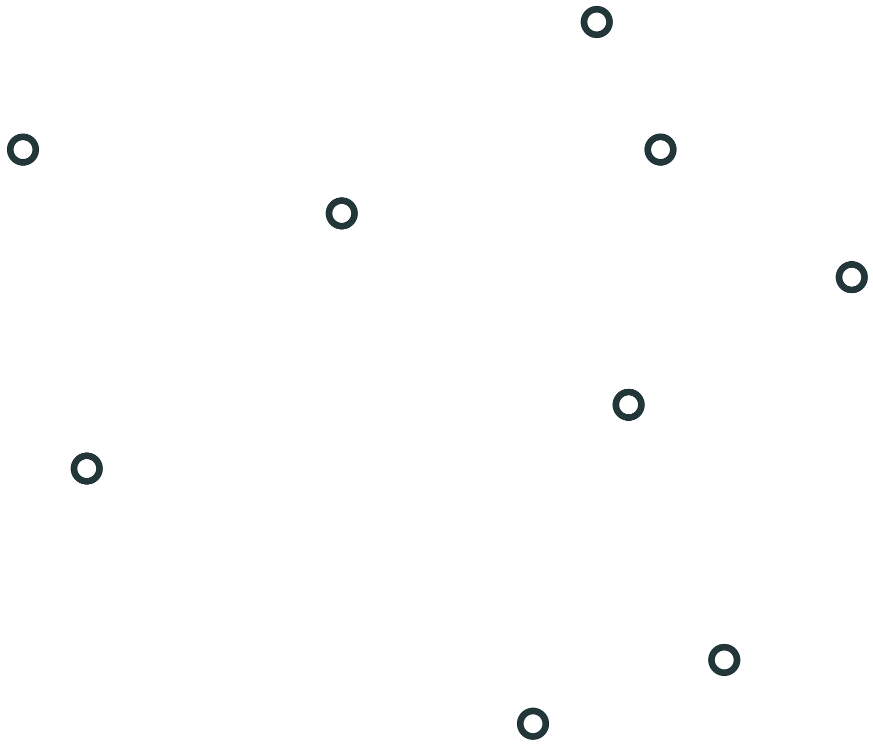


Algorithm

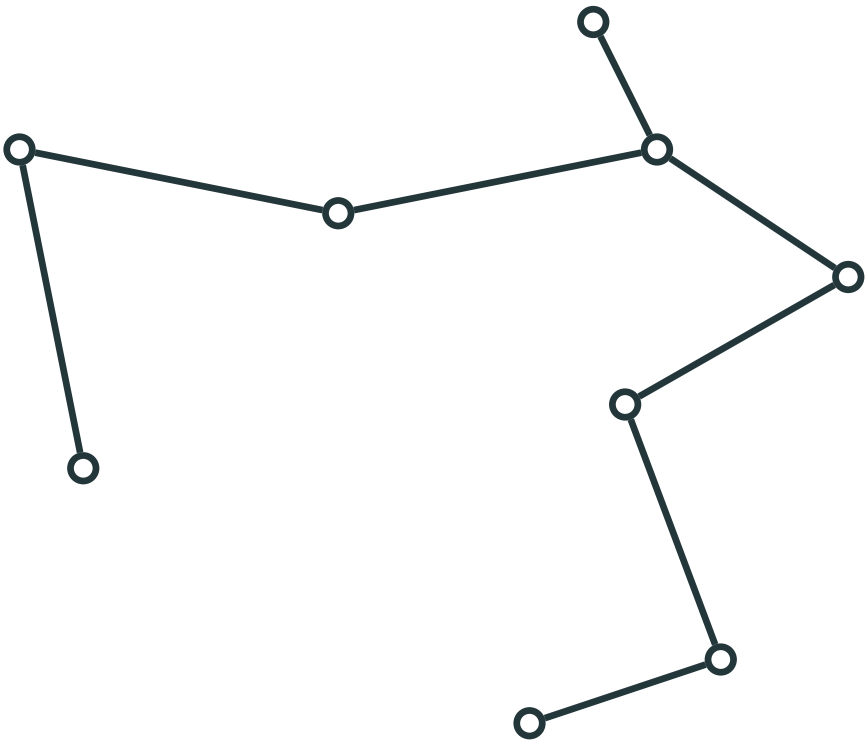
- $T \leftarrow$ minimum spanning tree of G
- $D \leftarrow T$ with each edge doubled
- find an Eulerian cycle C in D
- return a cycle that visits the nodes in the order of their first appearance in C

Example

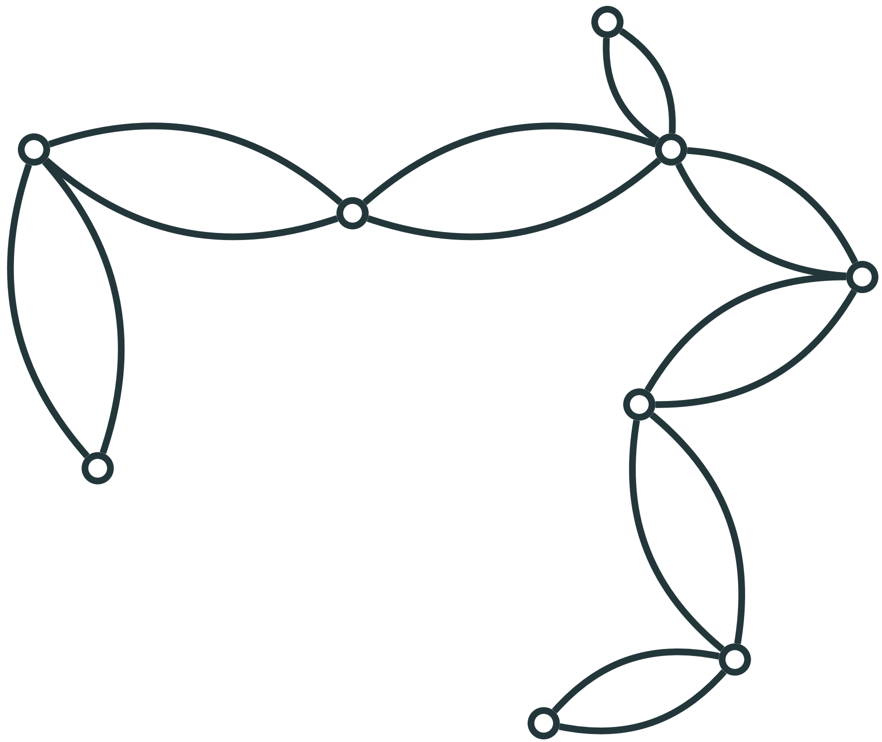
Euclidean TSP



Example



Example



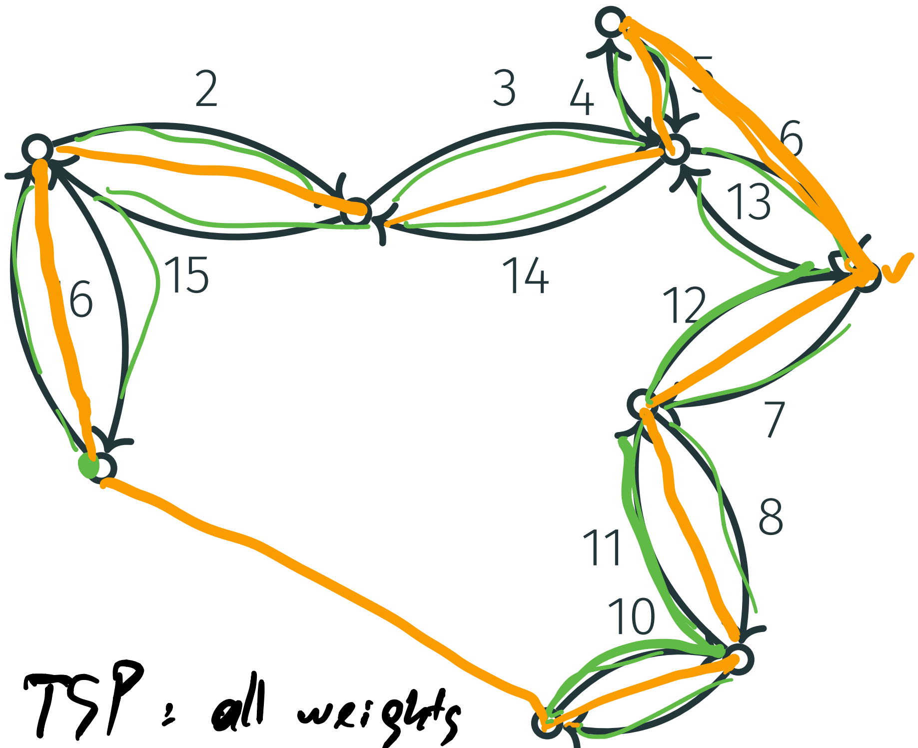
EXAMPLE

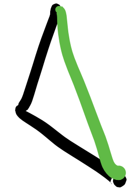
our solution

$\leq MST \cdot 2 \leq TSP \cdot 2$

Recall Lemma

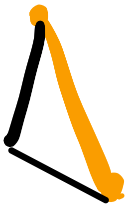
$MST \leq TSP$

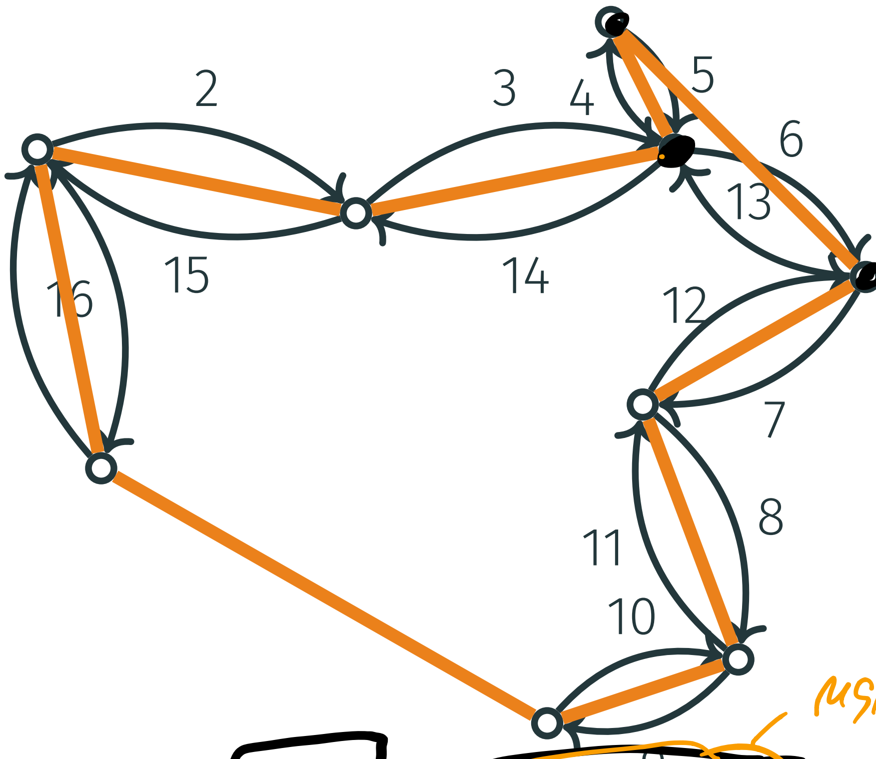




Metric TSP: all weights in the graph satisfy the Δ inequality

Example





TSP ≥ MST

our sol ≤ 2TSP

⇑

our sol ≤ 2MST

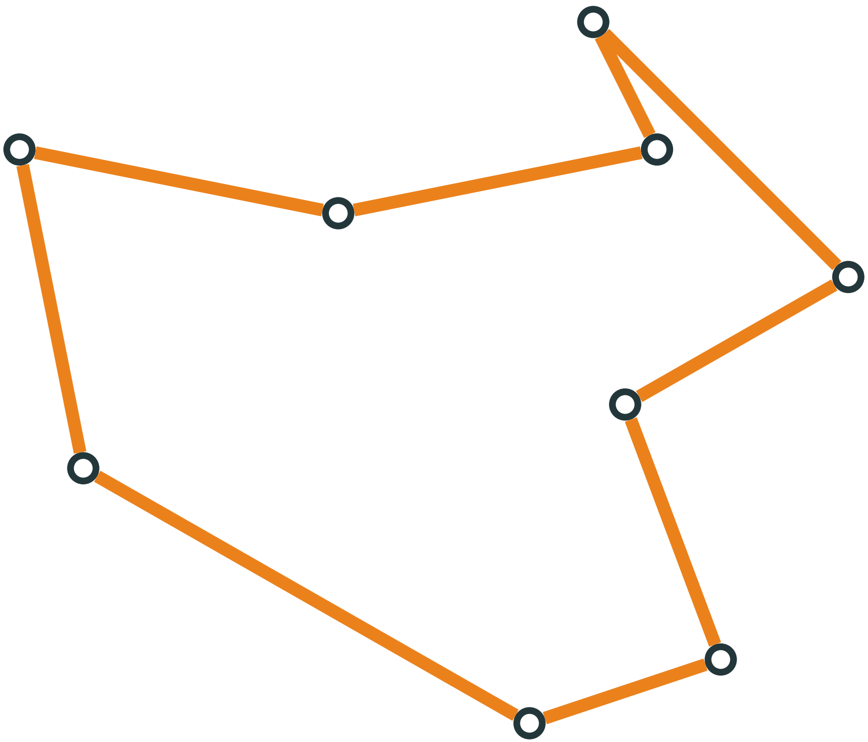
n edges   2n−2

MST ≤ 2·TSP

OUR sol ≤ MST + huch edge ≤ c · TSP

1·TSP

Example



Approximation Guarantee

Lemma

The algorithm is 2-approximate.

Approximation Guarantee

Lemma

The algorithm is 2-approximate.

Proof

- The total length of the MST $T \leq \mathrm{OPT}$ = TSP

Approximation Guarantee

Lemma

The algorithm is 2-approximate.

Proof

- The total length of the MST $T \leq \mathrm{OPT}$
- We start with Eulerian cycle of length $2|T|$

Approximation Guarantee

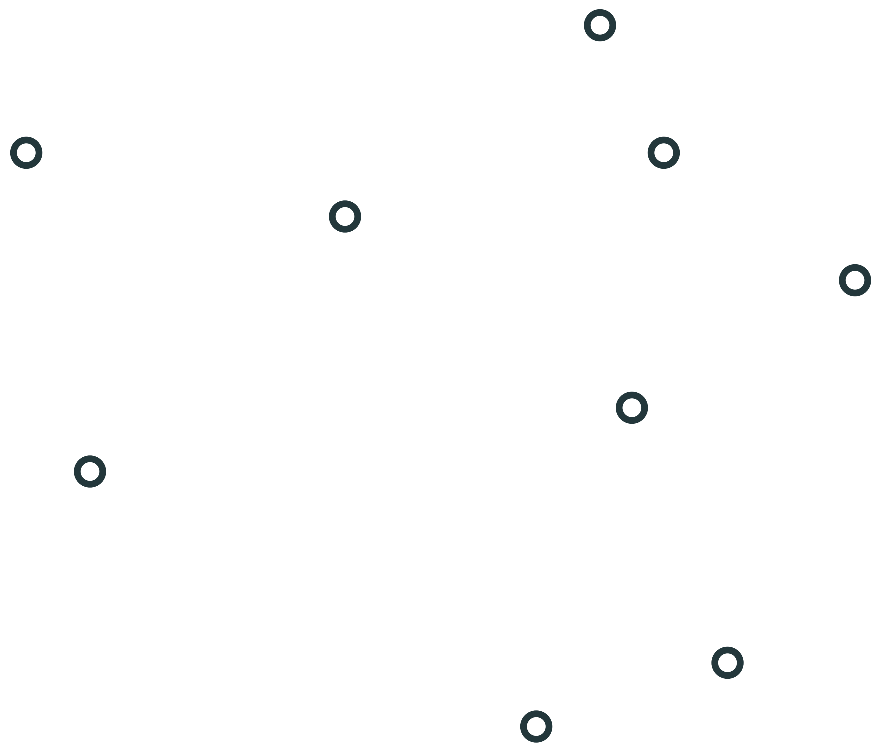
Lemma

The algorithm is 2-approximate.

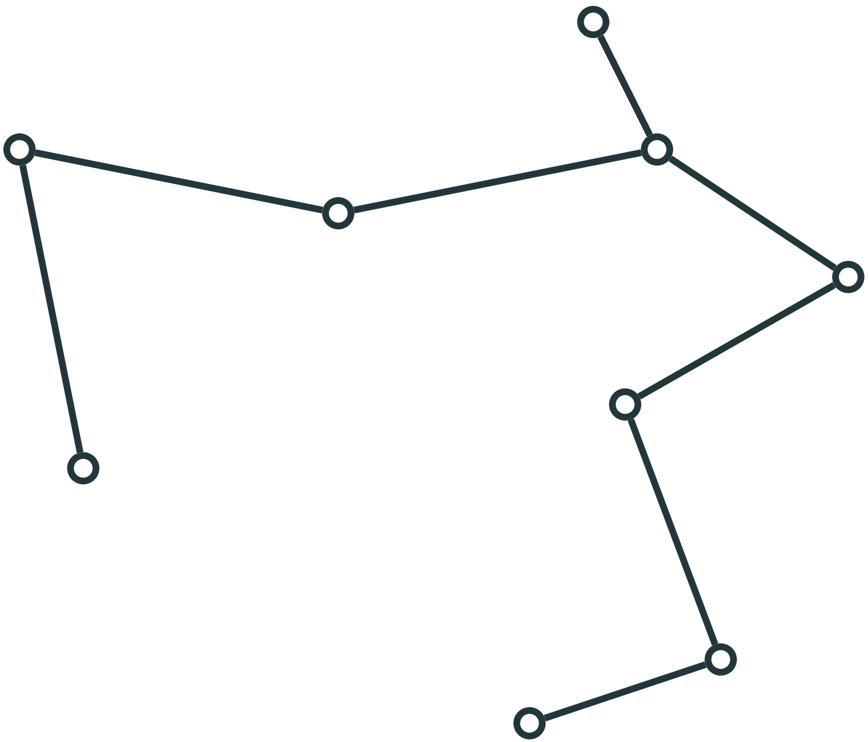
Proof

- The total length of the MST $T \leq \mathrm{OPT}$
- We start with Eulerian cycle of length $2|T|$
- Shortcuts can only decrease the total length

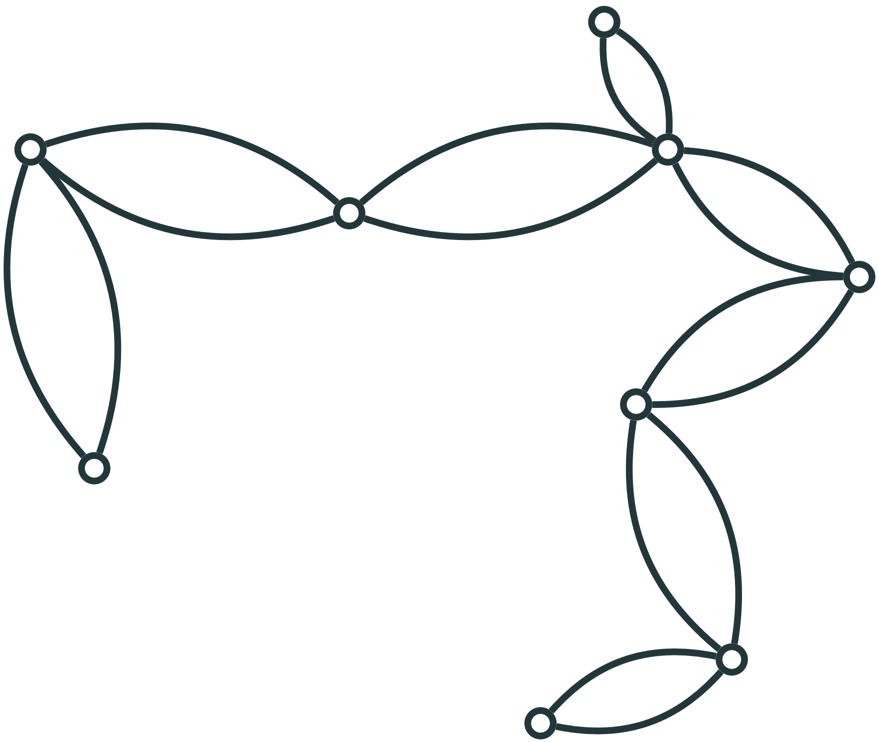
Improvement



Improvement

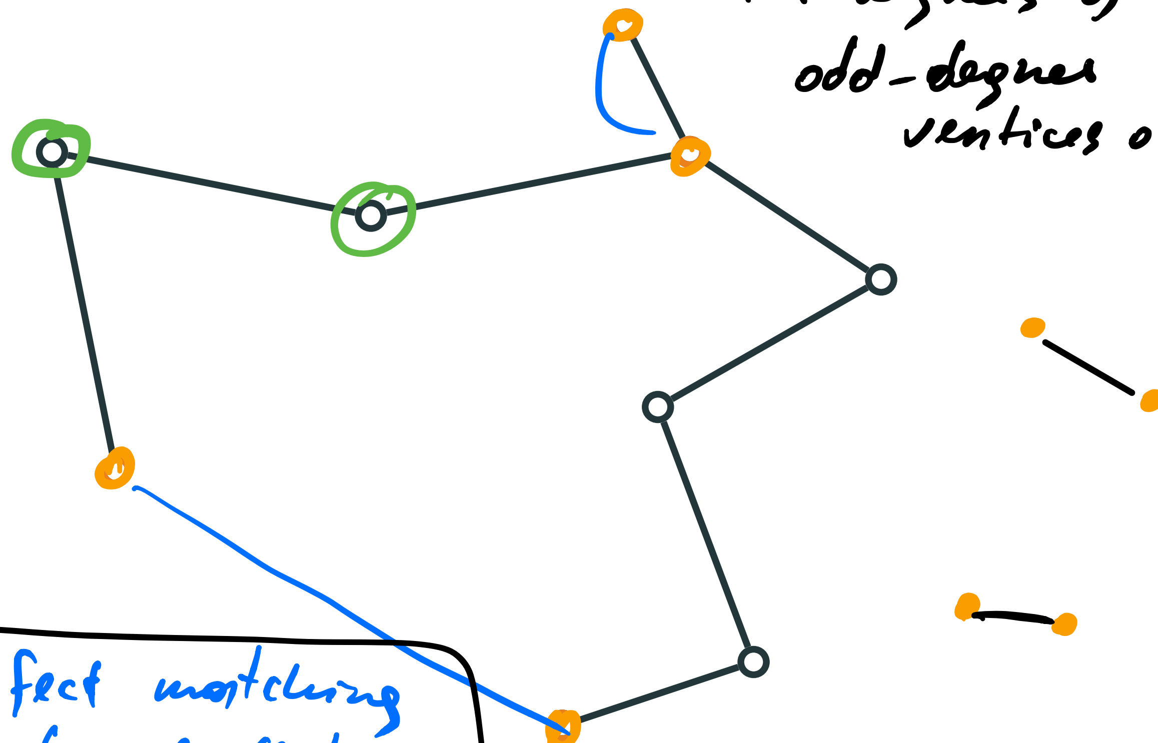


Improvement



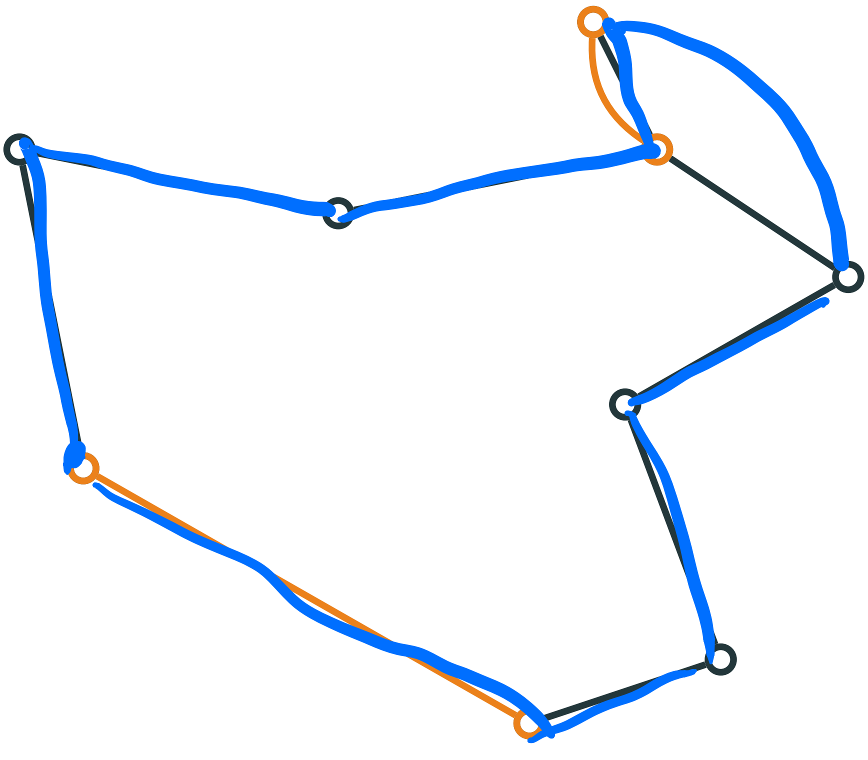
Improvement

Instead of doubling all edges, increase by 1 the degrees of odd-degree vertices only

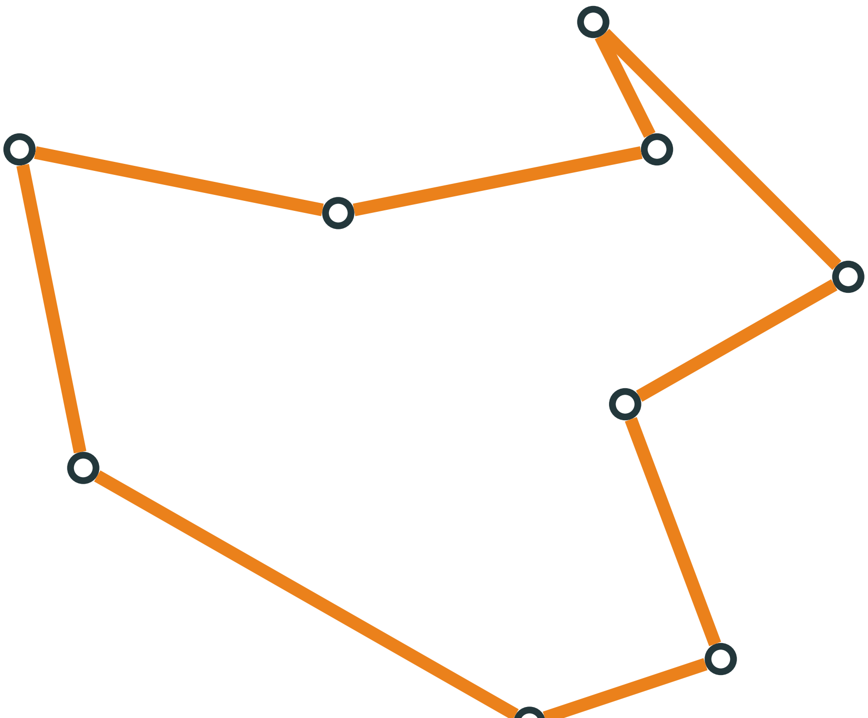


Perfect matching odd-degree vertices of minimum weight — poly time

Improvement



Improvement



Algorithm

- $T \leftarrow$ minimum spanning tree of G

Algorithm

- $T \leftarrow$ minimum spanning tree of G
- $M \leftarrow$ minimum weight perfect matching on odd-degree vertices of T

Algorithm

- $T \leftarrow$ minimum spanning tree of G
- $M \leftarrow$ minimum weight perfect matching on odd-degree vertices of T
- find an Eulerian cycle C in $T \cup M$

Algorithm

- $T \leftarrow$ minimum spanning tree of G
- $M \leftarrow$ minimum weight perfect matching on odd-degree vertices of T
- find an Eulerian cycle C in $T \cup M$
- return a cycle that visits the nodes in the order of their first appearance in C

Approximation Guarantee

Lemma

The algorithm is 3/2-approximate.

Approximation Guarantee

MST ≤ TSP

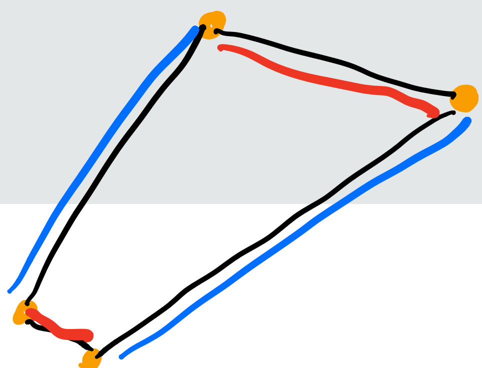
Matching ≤ $\frac{TSP}{2}$

Lemma

The algorithm is 3/2-approximate.

Proof

- The total length of the MST $T \leq$ OPT



TSP ≥ Matching1 + Matching2

⇓

Min weight matching ≤ TSP/2

Approximation Guarantee

Lemma

The algorithm is 3/2-approximate.

Proof

- The total length of the MST $T \leq \mathrm{OPT}$
- The weight of the matching $M \leq \mathrm{OPT}/2$

Approximation Guarantee

Lemma

The algorithm is 3/2-approximate.

Proof

- The total length of the MST $T \leq \mathrm{OPT}$
- The weight of the matching $M \leq \mathrm{OPT}/2$
- Shortcuts can only decrease the total length

Final Remarks

- Euclidean TSP can be approximated to within any factor $(1 + \varepsilon)$ 1.00001

Final Remarks

- Euclidean TSP can be approximated to within any factor $(1 + \varepsilon)$
- The currently best known approximation algorithm for TSP with triangle inequality is has approximation factor of $3/2 - 10^{-36}$ (July 2020)