Computability and Complexity
COSC 4200

Approximating TSP

Approximation Algorithms

Let X be a minimization problem.
An algorithm $\mathcal{A}$ is an $\alpha(n)$-approximation algorithm if for all n, for all instances of size n, $\mathcal{A}$ returns a solution with value at most $\alpha(n)$ times the value of the optimal solution.

For example:

- a $\log n$-approximation algorithm
- a 2-approximation algorithm

Approximation Algorithms

Let X be a minimization problem.
An algorithm $\mathcal{A}$ is an $\alpha(n)$-approximation algorithm if for all n, for all instances of size n, $\mathcal{A}$ returns a solution with value at most $\alpha(n)$ times the value of the optimal solution.

For example:

- a $\log n$-approximation algorithm
- a 2-approximation algorithm

Let X be a maximization problem.
An algorithm $\mathcal{A}$ is an $\alpha(n)$-approximation algorithm if for all n, for all instances of size n, $\mathcal{A}$ returns a solution with value at least $\alpha(n)$ times the value of the optimal solution.

For example:

- a $\frac{7}{8}$-approximation algorithm
- a $\frac{\log n}{n}$-approximation algorithm

Approximation of TSP

Theorem

If $\mathrm{P} \neq \mathrm{NP}$, then for any polynomial-time computable function $\alpha(n)$, there is no $\alpha(n)$-approximation algorithm for TSP.

Proof. We modify our reduction of $\mathrm{HAM\text{-}CYCLE}$ to TSP. Let $G = (V, E)$ be a graph.

Approximation of TSP

Theorem

If $\mathrm{P} \neq \mathrm{NP}$, then for any polynomial-time computable function $\alpha(n)$, there is no $\alpha(n)$-approximation algorithm for TSP.

Proof. We modify our reduction of $\mathrm{HAM\text{-}CYCLE}$ to TSP. Let $G = (V, E)$ be a graph.

Define $n = |V|$, $k = n$, and

$$c(i,j) = \begin{cases} 1 & \text{if } (i,j) \in E \\ \alpha(n) \cdot n & \text{if } (i,j) \notin E \end{cases}$$

for all $i, j \in \{1, \ldots, n\}$.

Notice that:

- if $G \in \mathrm{HAM\text{-}CYCLE}$, then there is a TSP tour with cost n.
- if $G \notin \mathrm{HAM\text{-}CYCLE}$, then any TSP tour has cost $> \alpha(n) \cdot n$.

Suppose $\mathcal{A}$ is a polynomial-time $\alpha(n)$-approximation algorithm for TSP. Consider the following Algorithm $\mathcal{B}$.

input: graph G, instance of $\mathrm{HAM\text{-}CYCLE}$

construct the TSP instance from G as described
run $\mathcal{A}$ on the TSP instance
let $c =$ cost of tour output by $\mathcal{A}$

if $c \leq \alpha(n) \cdot n$
    output 'yes'
else
    output 'no'

We claim that Algorithm $\mathcal{B}$ decides $\mathrm{HAM\text{-}CYCLE}$.

Suppose $G \in \mathrm{HAM\text{-}CYCLE}$.

- Then there is a TSP tour with cost n.
- The approximation algorithm $\mathcal{A}$ will find a tour with cost at most $\alpha(n) \cdot n$.
- Therefore algorithm $\mathcal{B}$ will output 'yes.'

Suppose $G \in \mathrm{HAM\text{-}CYCLE}$.

- Then there is a TSP tour with cost n.
- The approximation algorithm $\mathcal{A}$ will find a tour with cost at most $\alpha(n) \cdot n$.
- Therefore algorithm $\mathcal{B}$ will output 'yes.'

Now suppose $G \notin \mathrm{HAM\text{-}CYCLE}$.

- Then any TSP tour has cost $> \alpha(n) \cdot n$.
- The approximation algorithm $\mathcal{A}$ will return a tour with cost $> \alpha(n) \cdot n$.
- Therefore algorithm $\mathcal{B}$ will output 'no.'

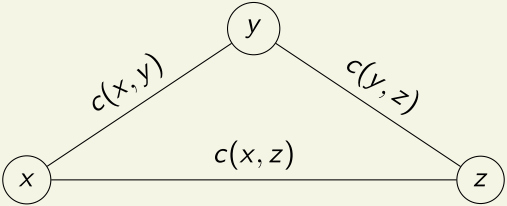
Therefore $\mathcal{B}$ decides $\mathrm{HAM\text{-}CYCLE}$ and $\mathrm{HAM\text{-}CYCLE} \in \mathrm{P}$.
Since $\mathrm{HAM\text{-}CYCLE}$ is NP-complete, $\mathrm{P} = \mathrm{NP}$. $\square$

Metric TSP

We showed there is no hope of approximating general TSP (unless P=NP). However, the cost function in the proof is wild: some costs are low and some are very high.

In the Metric TSP problem, the costs must satisfy the triangle inequality: for all x, y, and z,

$$c(x,z) \le c(x,y) + c(y,z).$$



Approximation Algorithm for Metric TSP

Consider the following algorithm to compute TSP tours:

On input a TSP instance as a weighted graph G:

1. Find a minimum spanning tree, T, of G.

Approximation Algorithm for Metric TSP

Consider the following algorithm to compute TSP tours:

On input a TSP instance as a weighted graph G:

1. Find a minimum spanning tree, T, of G.
2. Double every edge of T to obtain an Eulerian graph E.

Approximation Algorithm for Metric TSP

Consider the following algorithm to compute TSP tours:

On input a TSP instance as a weighted graph G:

1. Find a minimum spanning tree, T, of G.
2. Double every edge of T to obtain an Eulerian graph E.
3. Find an Eulerian tour, $\mathcal{T}$, of E.

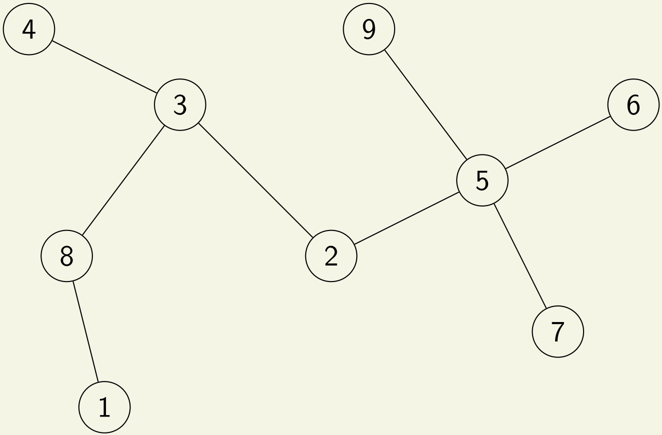
Approximation Algorithm for Metric TSP

Consider the following algorithm to compute TSP tours:

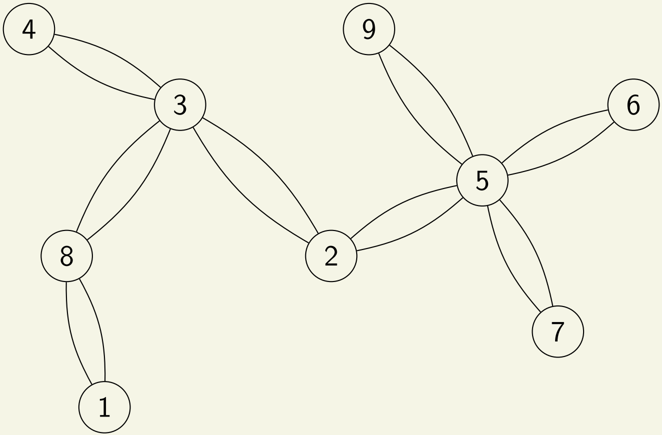
On input a TSP instance as a weighted graph G:

1. Find a minimum spanning tree, T, of G.
2. Double every edge of T to obtain an Eulerian graph E.
3. Find an Eulerian tour, $\mathcal{T}$, of E.
4. Output the tour $\mathcal{C}$ that visits vertices of G in the order of their first appearing in $\mathcal{T}$.

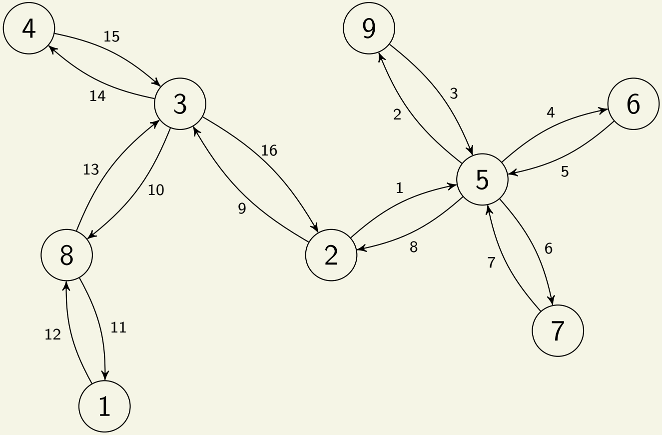
Minimum spanning tree T



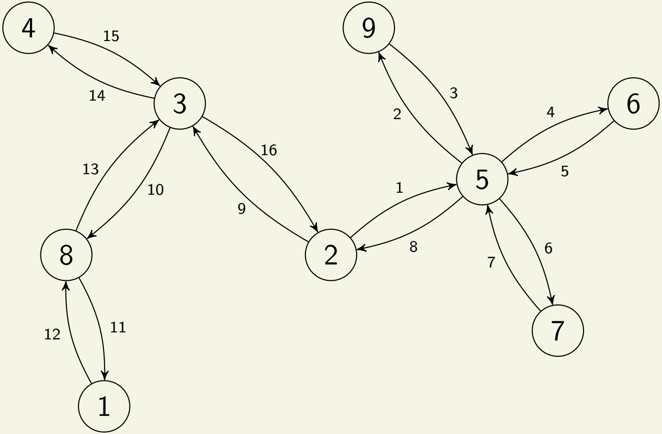
Double every edge to obtain E



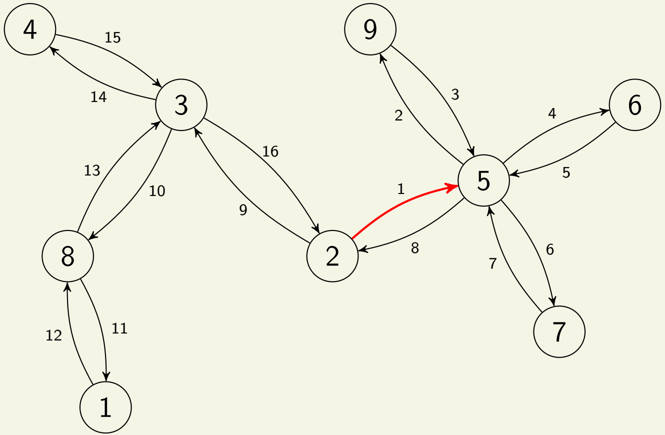
Find an Eulerian tour $\mathcal{T}$



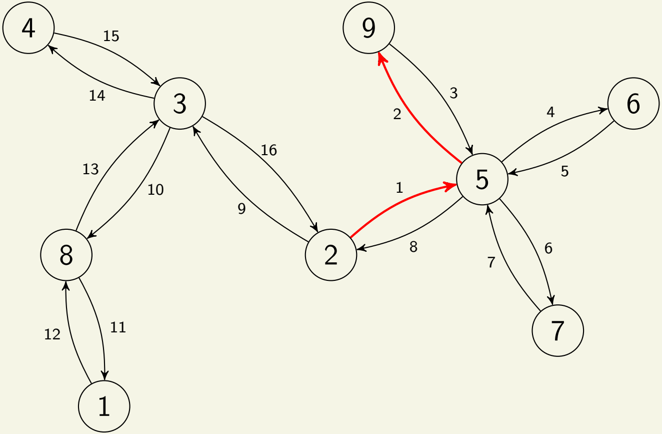
Take shortcuts to obtain final tour $\mathcal{C}$



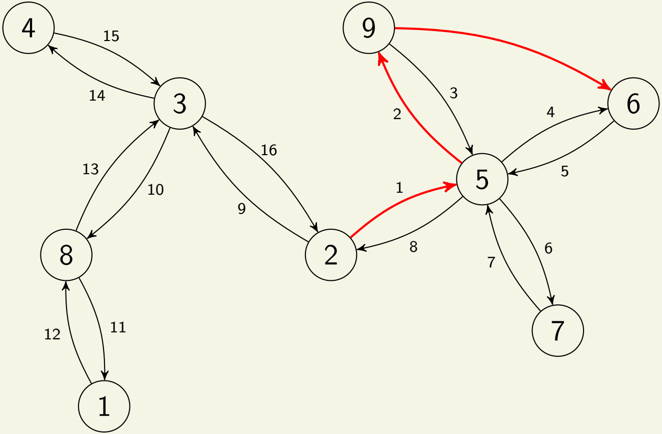
Take shortcuts to obtain final tour $\mathcal{C}$



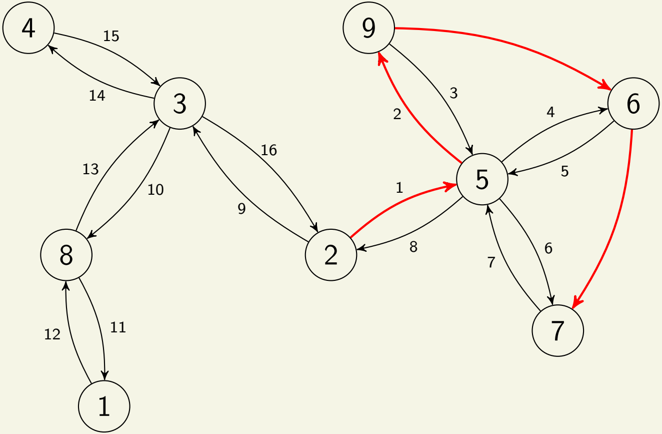
Take shortcuts to obtain final tour $\mathcal{C}$



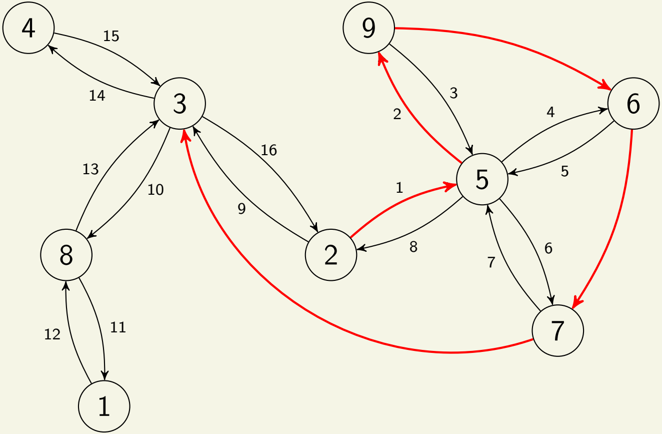
Take shortcuts to obtain final tour $\mathcal{C}$



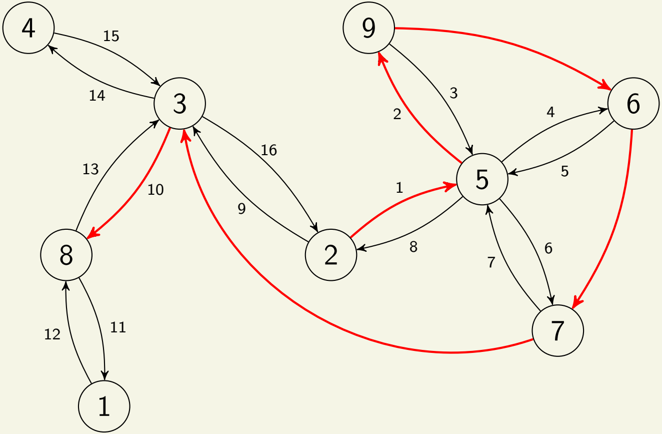
Take shortcuts to obtain final tour $\mathcal{C}$



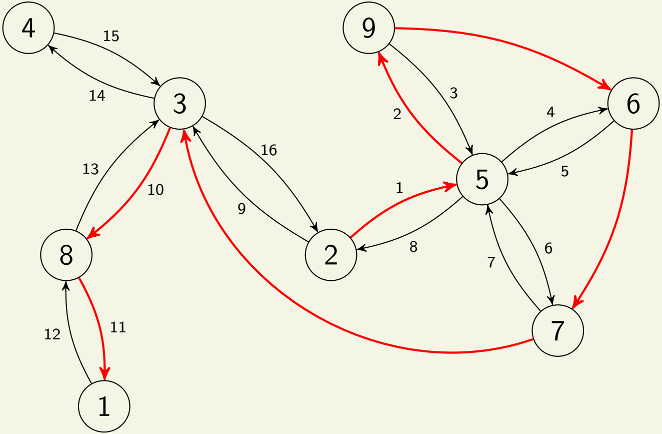
Take shortcuts to obtain final tour $\mathcal{C}$



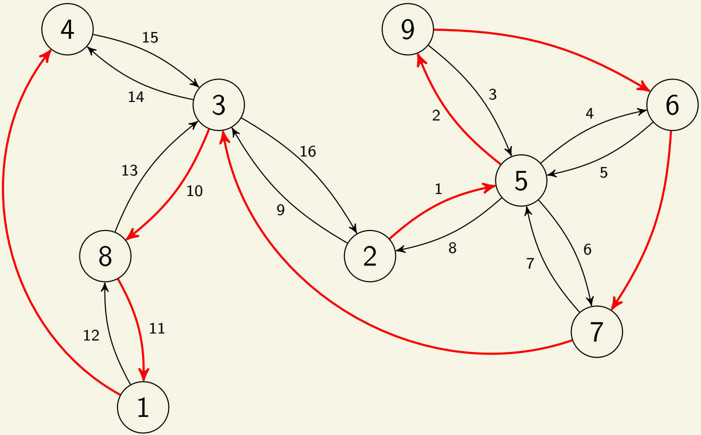
Take shortcuts to obtain final tour $\mathcal{C}$



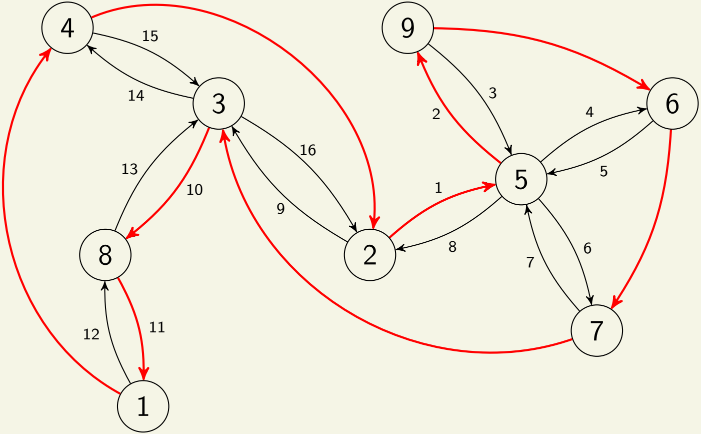
Take shortcuts to obtain final tour $\mathcal{C}$



Take shortcuts to obtain final tour $\mathcal{C}$



Take shortcuts to obtain final tour $\mathcal{C}$



Theorem. This is a 2-appoximation algorithm for Metric TSP.

Proof. Let OPT be the cost of an optimal TSP tour. Observe that:

- $\mathrm{cost}(T) \leq \mathrm{OPT}$. If we take any edge away from an optimal TSP tour, we have a spanning tree U. Thus $\mathrm{cost}(T) \leq \mathrm{cost}(U) \leq \mathrm{OPT}$.

Theorem. This is a 2-appoximation algorithm for Metric TSP.

Proof. Let OPT be the cost of an optimal TSP tour. Observe that:

- $\mathrm{cost}(T) \leq \mathrm{OPT}$. If we take any edge away from an optimal TSP tour, we have a spanning tree U. Thus $\mathrm{cost}(T) \leq \mathrm{cost}(U) \leq \mathrm{OPT}$.
- $\mathrm{cost}(\mathcal{T}) = \mathrm{cost}(E) = 2 \cdot \mathrm{cost}(T)$.

Theorem. This is a 2-appoximation algorithm for Metric TSP.

Proof. Let OPT be the cost of an optimal TSP tour. Observe that:

- $\mathrm{cost}(T) \leq \mathrm{OPT}$. If we take any edge away from an optimal TSP tour, we have a spanning tree U. Thus $\mathrm{cost}(T) \leq \mathrm{cost}(U) \leq \mathrm{OPT}$.
- $\mathrm{cost}(\mathcal{T}) = \mathrm{cost}(E) = 2 \cdot \mathrm{cost}(T)$.
- By the (extended) triangle inequality, $\mathrm{cost}(\mathcal{C}) \leq \mathrm{cost}(\mathcal{T})$.

Theorem. This is a 2-appoximation algorithm for Metric TSP.

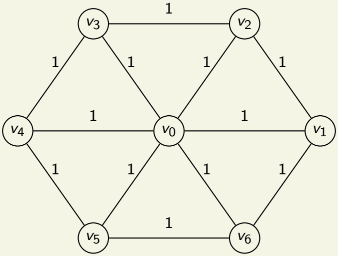
Proof. Let OPT be the cost of an optimal TSP tour. Observe that:

- $\mathrm{cost}(T) \leq \mathrm{OPT}$. If we take any edge away from an optimal TSP tour, we have a spanning tree U. Thus $\mathrm{cost}(T) \leq \mathrm{cost}(U) \leq \mathrm{OPT}$.
- $\mathrm{cost}(\mathcal{T}) = \mathrm{cost}(E) = 2 \cdot \mathrm{cost}(T)$.
- By the (extended) triangle inequality, $\mathrm{cost}(\mathcal{C}) \leq \mathrm{cost}(\mathcal{T})$.

Putting everything together, we have

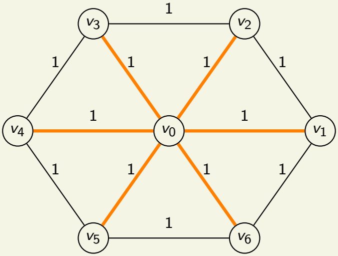
$$\mathrm{cost}(\mathcal{C}) \leq \mathrm{cost}(\mathcal{T}) = 2 \cdot \mathrm{cost}(T) \leq 2 \cdot \mathrm{OPT}. \quad \square$$

Tight Example for 2-Approximation Algorithm



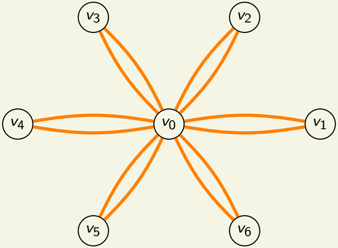
The distance for all other pairs of vertices is 2.

Tight Example for 2-Approximation Algorithm



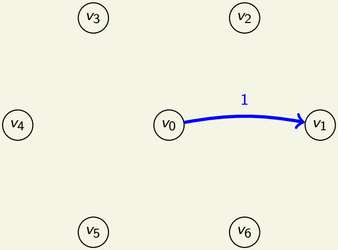
The MST consists of the orange edges and has cost 6.

Tight Example for 2-Approximation Algorithm



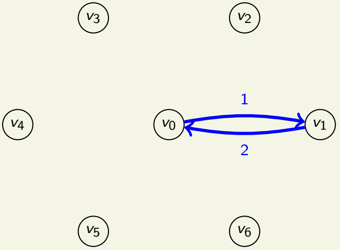
Double the edges of the MST to obtain an Eulerian graph.

Tight Example for 2-Approximation Algorithm



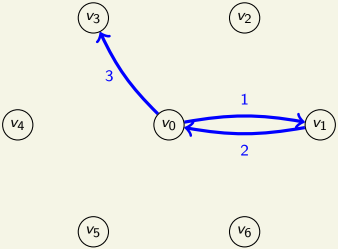
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



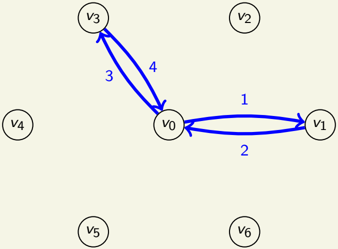
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



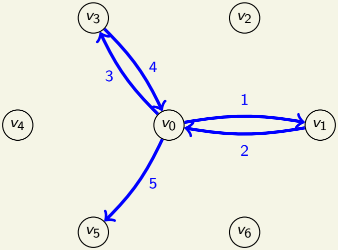
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



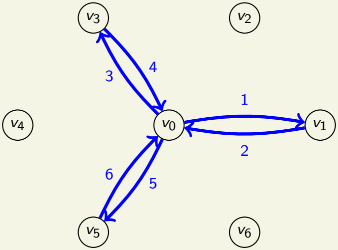
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



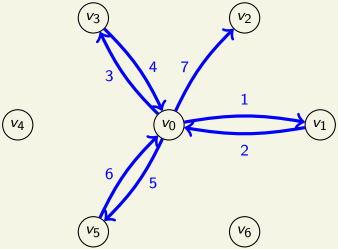
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



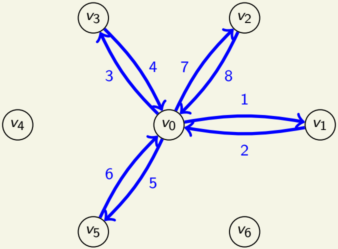
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



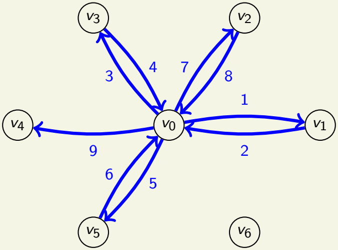
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



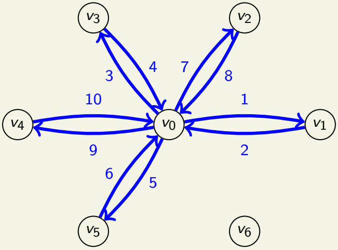
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



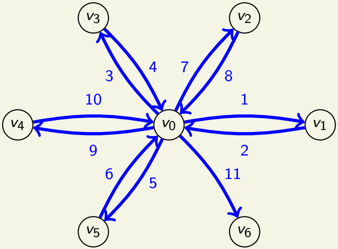
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



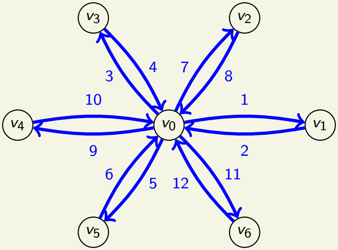
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



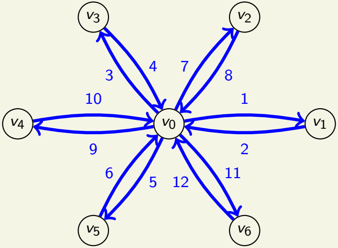
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



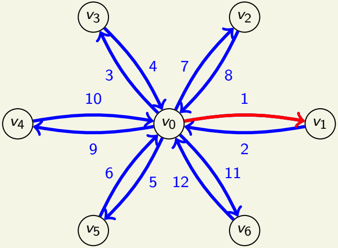
An Eulerian tour. The algorithm could find any of many Eulerian tours, but this one will give worst-case performance.

Tight Example for 2-Approximation Algorithm



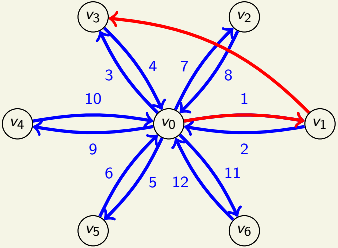
The shortcut tour.

Tight Example for 2-Approximation Algorithm



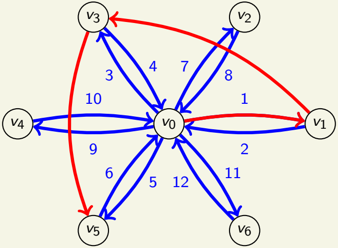
The shortcut tour.

Tight Example for 2-Approximation Algorithm



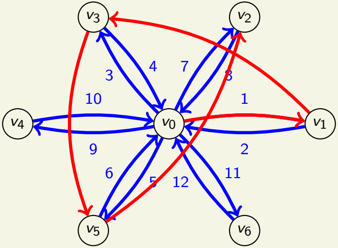
The shortcut tour.

Tight Example for 2-Approximation Algorithm



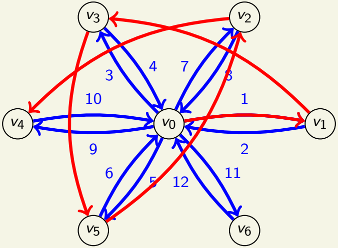
The shortcut tour.

Tight Example for 2-Approximation Algorithm



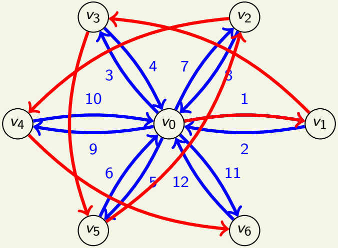
The shortcut tour.

Tight Example for 2-Approximation Algorithm



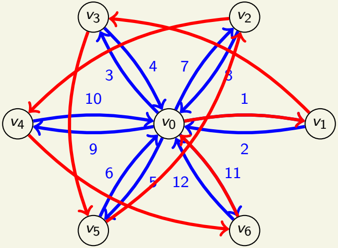
The shortcut tour.

Tight Example for 2-Approximation Algorithm



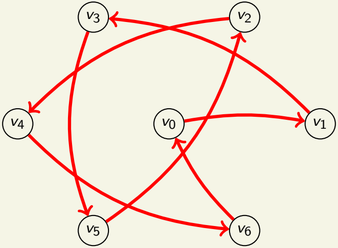
The shortcut tour.

Tight Example for 2-Approximation Algorithm



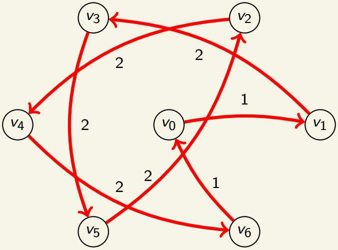
The shortcut tour.

Tight Example for 2-Approximation Algorithm



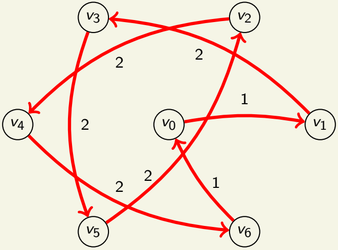
The shortcut tour.

Tight Example for 2-Approximation Algorithm



Tour cost: $2(6-1)+2=12$.

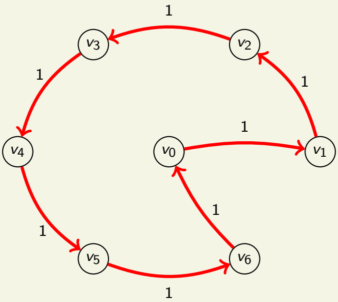
Tight Example for 2-Approximation Algorithm



Tour cost: $2(6-1)+2=12$.

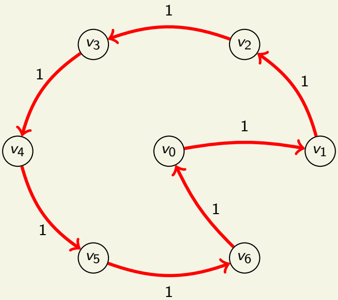
If we change 6 to n and obtain a analogous graph on $n+1$ vertices, the cost is $2(n-1)+2=2n$.

Tight Example for 2-Approximation Algorithm



Optimal tour cost: 7

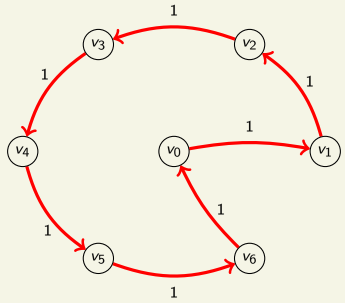
Tight Example for 2-Approximation Algorithm



Optimal tour cost: 7

If we change 6 to n and obtain a analogous graph on $n+1$ vertices, the optimal tour cost is $n+1$.

Tight Example for 2-Approximation Algorithm



Optimal tour cost: 7
If we change 6 to n and obtain a analogous graph on $n+1$ vertices, the optimal tour cost is $n+1$.

The performance ratio is $\frac{12}{7} \leq 2$ on the above instance. In general, for the graph on $n+1$ vertices, the performance ratio is:

$$\frac{2n}{n+1}.$$

This approaches 2 as $n \to \infty$.

Christofides' Algorithm

An improvement - Christofides' Algorithm
Consider the following algorithm to compute TSP tours:

On input a TSP instance as a weighted graph G:

1. Find a minimum spanning tree, T, of G.

Christofides' Algorithm

An improvement - Christofides' Algorithm
Consider the following algorithm to compute TSP tours:

On input a TSP instance as a weighted graph G:

1. Find a minimum spanning tree, T, of G.
2. Compute a minimum cost perfect matching, M, on the set of odd-degree vertices of T. A minimum cost perfect matching is a pairing (matching) of vertices at minimum total edge cost. Add M to T and obtain an Eulerian graph E.

Christofides' Algorithm

An improvement - Christofides' Algorithm
Consider the following algorithm to compute TSP tours:

On input a TSP instance as a weighted graph G:

1. Find a minimum spanning tree, T, of G.
2. Compute a minimum cost perfect matching, M, on the set of odd-degree vertices of T. A minimum cost perfect matching is a pairing (matching) of vertices at minimum total edge cost. Add M to T and obtain an Eulerian graph E.
3. Find an Eulerian tour, $\mathcal{T}$, of E.

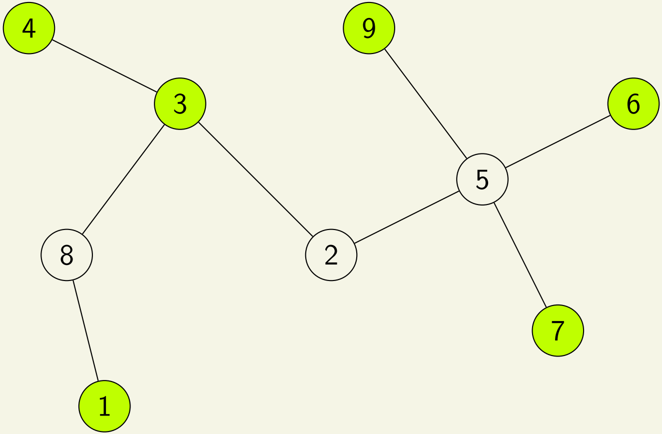
Christofides' Algorithm

An improvement - Christofides' Algorithm
Consider the following algorithm to compute TSP tours:

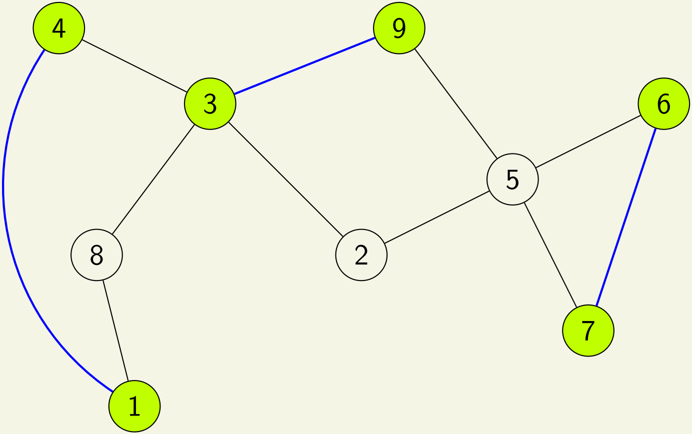
On input a TSP instance as a weighted graph G:

1. Find a minimum spanning tree, T, of G.
2. Compute a minimum cost perfect matching, M, on the set of odd-degree vertices of T. A minimum cost perfect matching is a pairing (matching) of vertices at minimum total edge cost. Add M to T and obtain an Eulerian graph E.
3. Find an Eulerian tour, $\mathcal{T}$, of E.
4. Output the tour $\mathcal{C}$ that visits vertices of G in the order of their first appearing in $\mathcal{T}$.

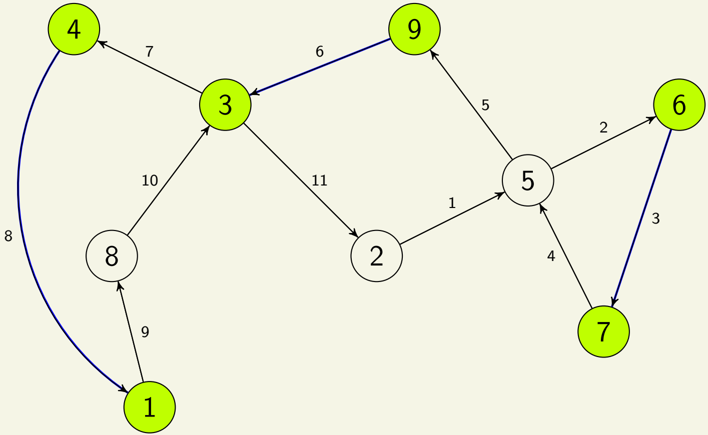
Minimum spanning tree T, odd degree vertices highlighted



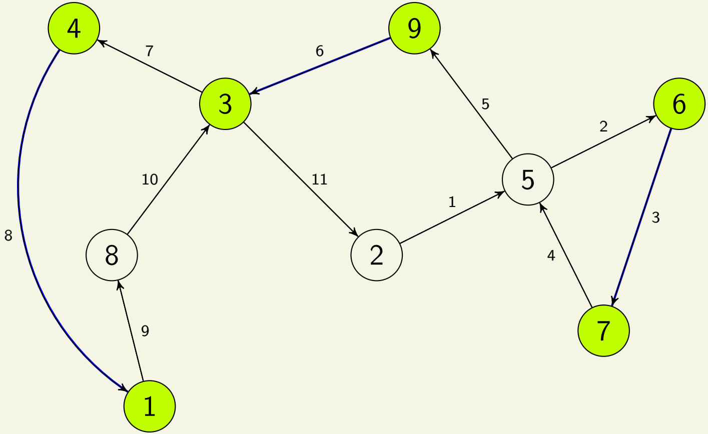
Minimum cost perfect matching added, now Eulerian graph



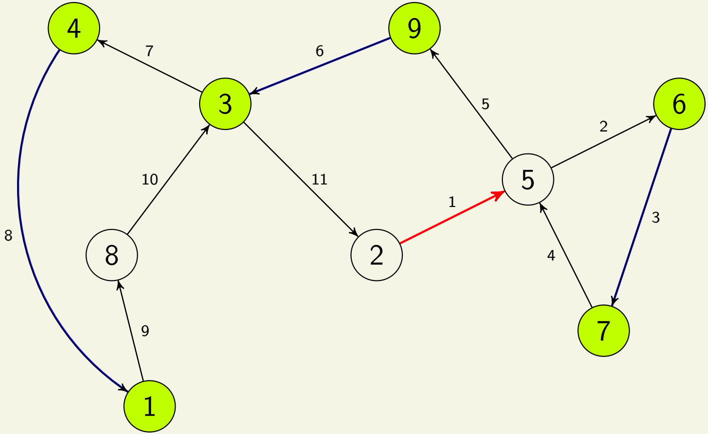
Minimum cost perfect matching added, now Eulerian graph



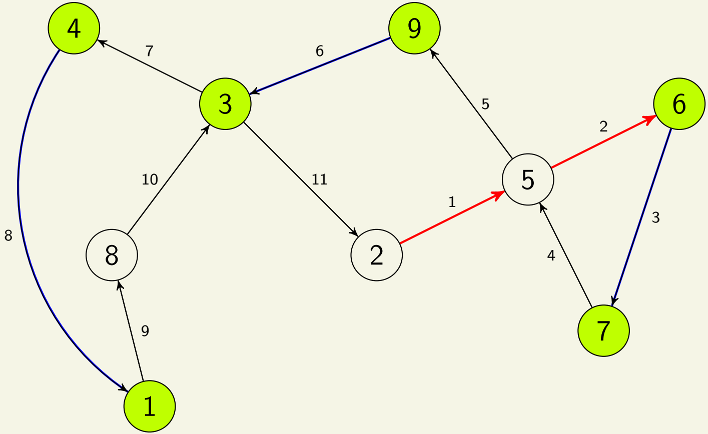
Take shortcuts to obtain final tour $\mathcal{C}$



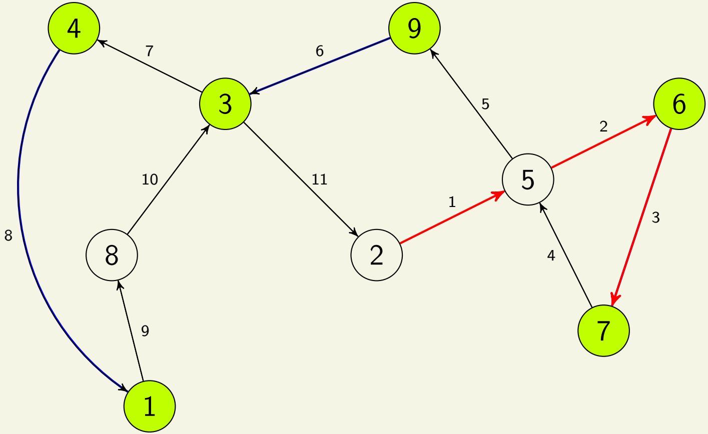
Take shortcuts to obtain final tour $\mathcal{C}$



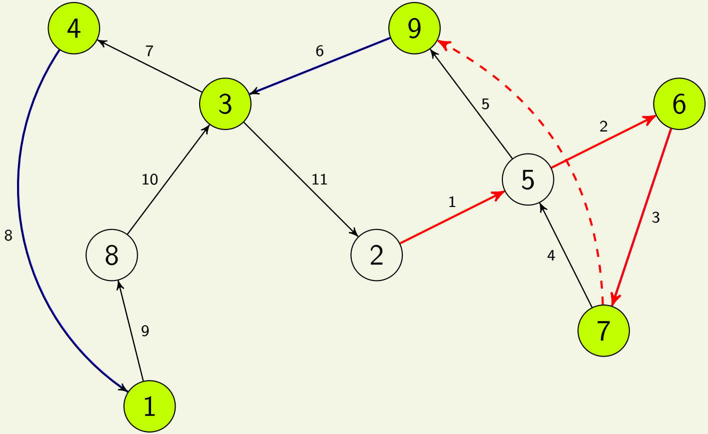
Take shortcuts to obtain final tour $\mathcal{C}$



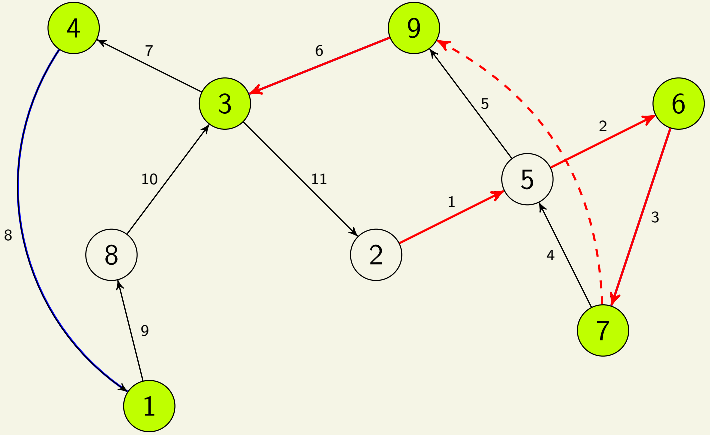
Take shortcuts to obtain final tour $\mathcal{C}$



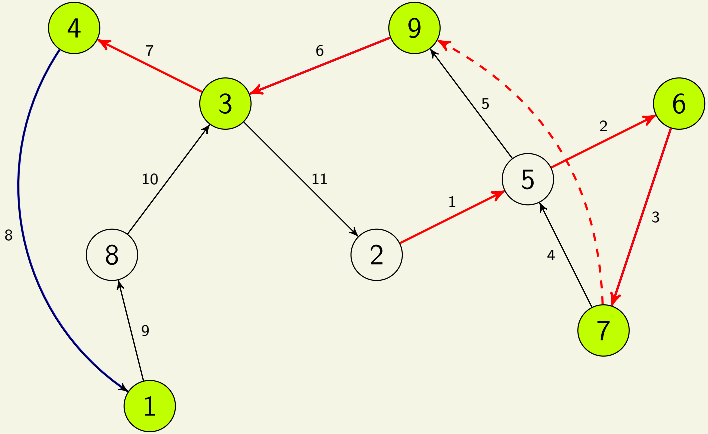
Take shortcuts to obtain final tour $\mathcal{C}$



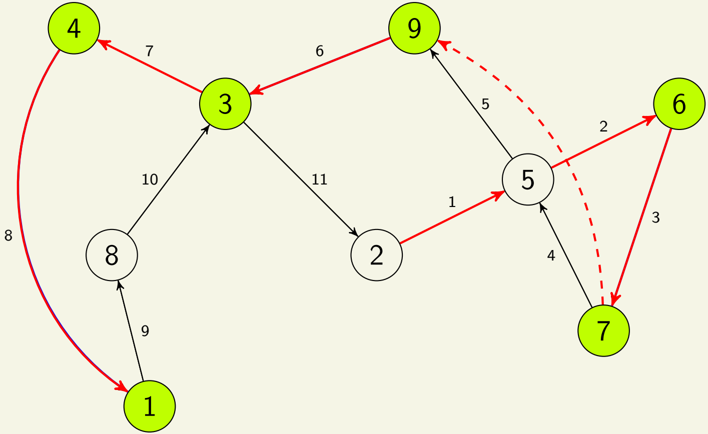
Take shortcuts to obtain final tour $\mathcal{C}$



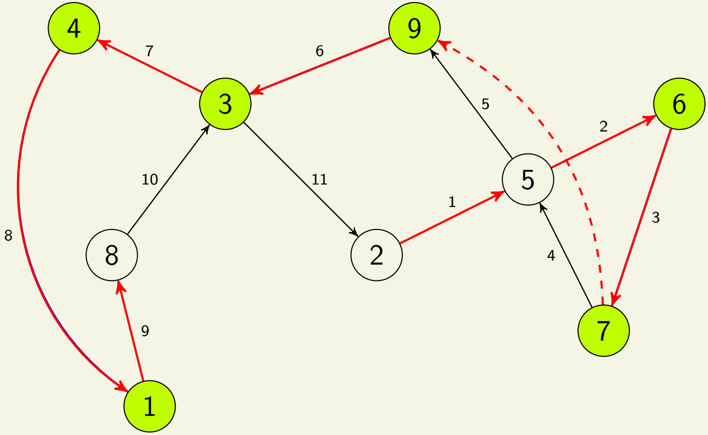
Take shortcuts to obtain final tour $\mathcal{C}$



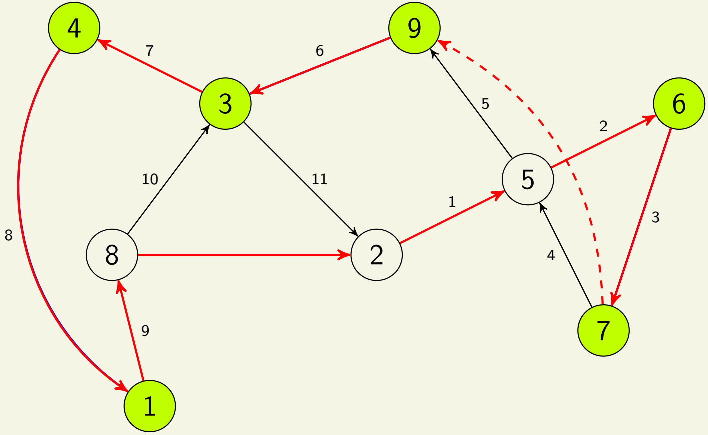
Take shortcuts to obtain final tour $\mathcal{C}$



Take shortcuts to obtain final tour $\mathcal{C}$



Take shortcuts to obtain final tour $\mathcal{C}$



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

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

Proof. Let τ be an optimal TSP tour on V.

- Let τ' be the tour on V' obtained by shortcutting τ.

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

Proof. Let τ be an optimal TSP tour on V.

- Let τ' be the tour on V' obtained by shortcutting τ.
- By the triangle inequality, $\mathrm{cost}(\tau') \leq \mathrm{cost}(\tau)$.

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

Proof. Let τ be an optimal TSP tour on V.

- Let τ' be the tour on V' obtained by shortcutting τ.
- By the triangle inequality, $\mathrm{cost}(\tau') \leq \mathrm{cost}(\tau)$.
- Now look at τ' as the union of two perfect matchings on V', each consisting of alternate edges of τ'.

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

Proof. Let τ be an optimal TSP tour on V.

- Let τ' be the tour on V' obtained by shortcutting τ.
- By the triangle inequality, $\mathrm{cost}(\tau') \leq \mathrm{cost}(\tau)$.
- Now look at τ' as the union of two perfect matchings on V', each consisting of alternate edges of τ'.
- The cheaper of these two perfect matchings has cost $\leq \mathrm{cost}(\tau')/2 \leq \mathrm{OPT}/2$.

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

Proof. Let τ be an optimal TSP tour on V.

- Let τ' be the tour on V' obtained by shortcutting τ.
- By the triangle inequality, $\mathrm{cost}(\tau') \leq \mathrm{cost}(\tau)$.
- Now look at τ' as the union of two perfect matchings on V', each consisting of alternate edges of τ'.
- The cheaper of these two perfect matchings has cost $\leq \mathrm{cost}(\tau')/2 \leq \mathrm{OPT}/2$.

Therefore the optimal perfect matching has cost $\leq \mathrm{OPT}/2$. $\square$

Theorem. Christofides' Algorithm is a $\frac{3}{2}$-approximation algorithm for Metric TSP.

Theorem. Christofides' Algorithm is a $\frac{3}{2}$-approximation algorithm for Metric TSP.

Proof. We have

$$\mathrm{cost}(\mathcal{C}) \quad \leq \quad \mathrm{cost}(\mathcal{T})$$

Theorem. Christofides' Algorithm is a $\frac{3}{2}$-approximation algorithm for Metric TSP.

Proof. We have

$$\begin{aligned}\mathrm{cost}(\mathcal{C}) &\leq \mathrm{cost}(\mathcal{T}) \\ &= \mathrm{cost}(T) + \mathrm{cost}(M)\end{aligned}$$

Theorem. Christofides' Algorithm is a $\frac{3}{2}$-approximation algorithm for Metric TSP.

Proof. We have

$$\begin{aligned} \mathrm{cost}(\mathcal{C}) &\leq \mathrm{cost}(\mathcal{T}) \\ &= \mathrm{cost}(T) + \mathrm{cost}(M) \\ &\leq \mathrm{OPT} + \frac{1}{2}\mathrm{OPT} \end{aligned}$$

Theorem. Christofides' Algorithm is a $\frac{3}{2}$-approximation algorithm for Metric TSP.

Proof. We have

$$\begin{aligned}
\mathrm{cost}(\mathcal{C}) &\leq \mathrm{cost}(\mathcal{T}) \\
&= \mathrm{cost}(T) + \mathrm{cost}(M) \\
&\leq \mathrm{OPT} + \frac{1}{2}\mathrm{OPT} \\
&= \frac{3}{2}\mathrm{OPT}. \quad \square
\end{aligned}$$

Theorem. Christofides' Algorithm is a $\frac{3}{2}$-approximation algorithm for Metric TSP.

Proof. We have

$$\begin{aligned}
\mathrm{cost}(\mathcal{C}) &\leq \mathrm{cost}(\mathcal{T}) \\
&= \mathrm{cost}(T) + \mathrm{cost}(M) \\
&\leq \mathrm{OPT} + \frac{1}{2}\mathrm{OPT} \\
&= \frac{3}{2}\mathrm{OPT}. \quad \square
\end{aligned}$$

Christofides' algorithm (1976) was the best approximation algorithm for Metric TSP until 2020!

In an 86 page paper, Karlin, Klein, and Gharan (2020) gave an algorithm with approximation ratio

$$\frac{3}{2} - 10^{-36}$$

Theorem. Christofides' Algorithm is a $\frac{3}{2}$-approximation algorithm for Metric TSP.

Proof. We have

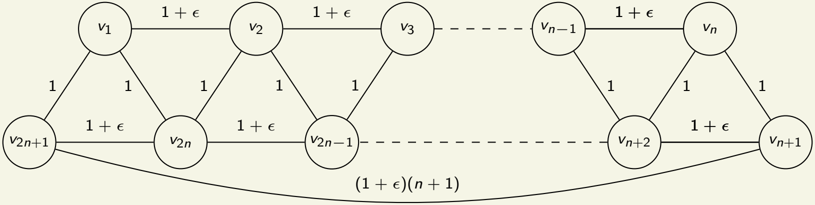
$$\begin{aligned}
\mathrm{cost}(\mathcal{C}) &\leq \mathrm{cost}(\mathcal{T}) \\
&= \mathrm{cost}(T) + \mathrm{cost}(M) \\
&\leq \mathrm{OPT} + \frac{1}{2}\mathrm{OPT} \\
&= \frac{3}{2}\mathrm{OPT}. \quad \square
\end{aligned}$$

Christofides' algorithm (1976) was the best approximation algorithm for Metric TSP until 2020!

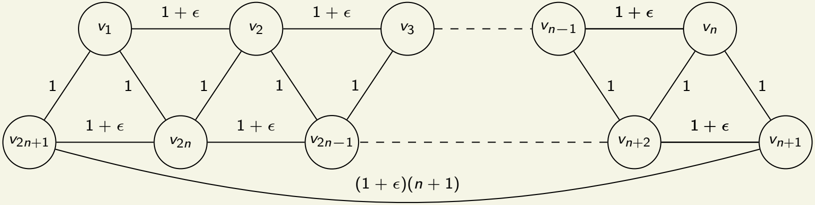
In an 86 page paper, Karlin, Klein, and Gharan (2020) gave an algorithm with approximation ratio

$$\frac{3}{2} - 10^{-36} = 1.499999999999999999999999999999999999.$$

Tight Example for Christofides' Algorithm

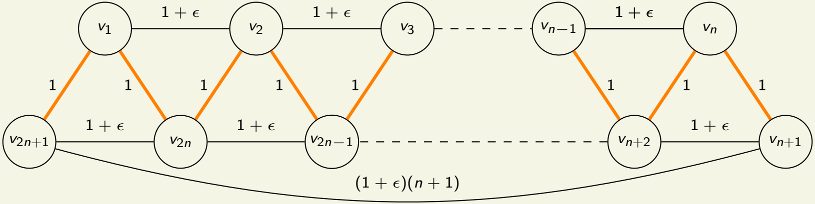


Tight Example for Christofides' Algorithm



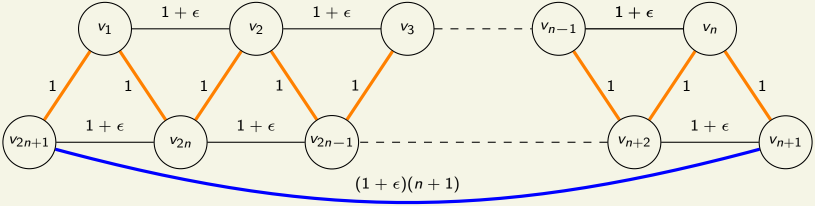
All other edge costs are the cost of the shortest path using the above edges.

Tight Example for Christofides' Algorithm



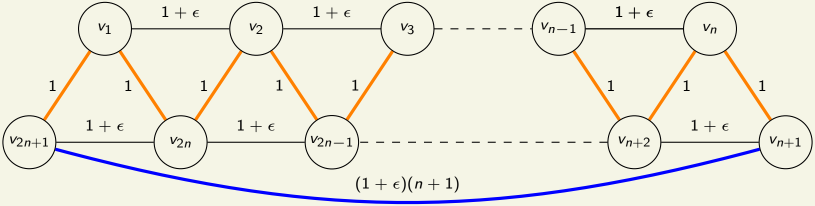
The MST consists of the orange edges and has cost $2n$.

Tight Example for Christofides' Algorithm



There are two odd degree vertices, v_{n+1} and v_{2n+1}.
They are matched by the minimum cost perfect matching algorithm.

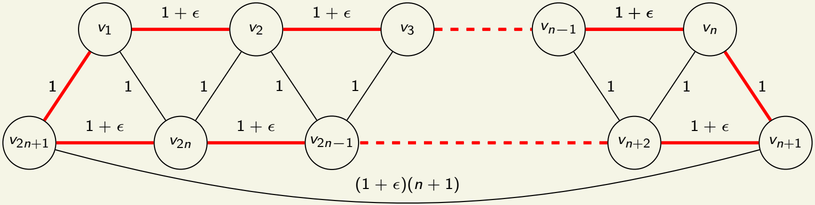
Tight Example for Christofides' Algorithm



There are two odd degree vertices, v_{n+1} and v_{2n+1}.
They are matched by the minimum cost perfect matching algorithm.

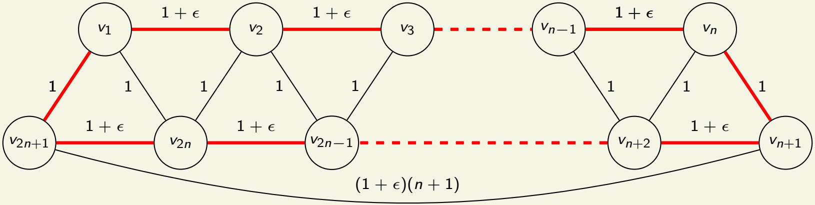
Tour cost: $2n + (1+\epsilon)(n+1) = (3+\epsilon)n + (1+\epsilon)$

Tight Example for Christofides' Algorithm



The optimal tour has cost $(2n-1)(1+\epsilon)+2 = 2(1+\epsilon)n+(1-\epsilon)$.

Tight Example for Christofides' Algorithm



The optimal tour has cost $(2n-1)(1+\epsilon)+2 = 2(1+\epsilon)n+(1-\epsilon)$.

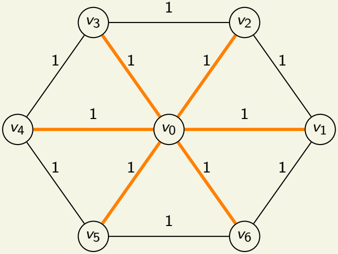
Performance ratio:

$$\frac{(3+\epsilon)n+(1+\epsilon)}{2(1+\epsilon)n+(1-\epsilon)}.$$

This approaches $\frac{3}{2}$ as $n \to \infty$ and $\epsilon \to 0$.

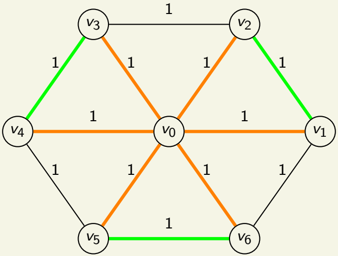
Revisiting Tight Example for 2-Approximation Algorithm

Let's see how Christofides' Algorithm performs on the tight instances for the 2-approximation algorithm.



The MST consists of the orange edges.

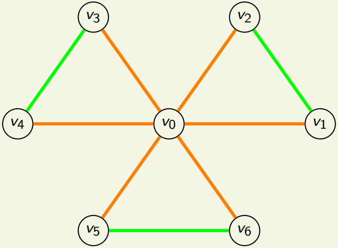
Revisiting Tight Example for 2-Approximation Algorithm



The MST consists of the orange edges.

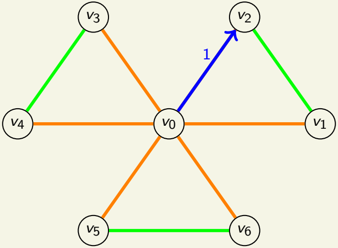
The green edges are a minimum cost perfect matching of the odd degree vertices in the MST.

Revisiting Tight Example for 2-Approximation Algorithm



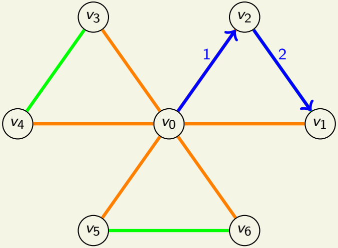
An Eulerian tour.

Revisiting Tight Example for 2-Approximation Algorithm



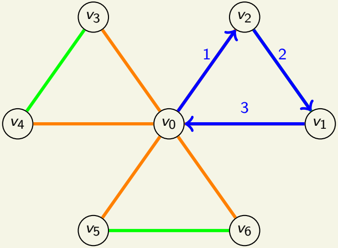
An Eulerian tour.

Revisiting Tight Example for 2-Approximation Algorithm



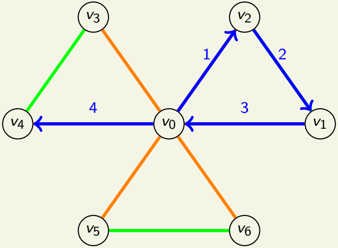
An Eulerian tour.

Revisiting Tight Example for 2-Approximation Algorithm



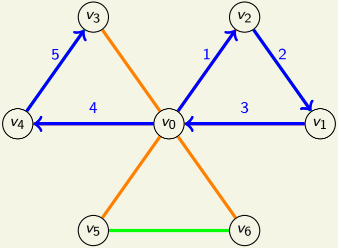
An Eulerian tour.

Revisiting Tight Example for 2-Approximation Algorithm



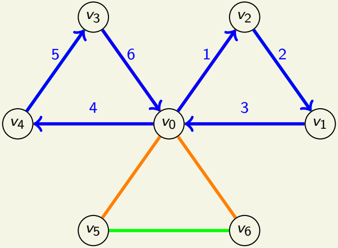
An Eulerian tour.

Revisiting Tight Example for 2-Approximation Algorithm



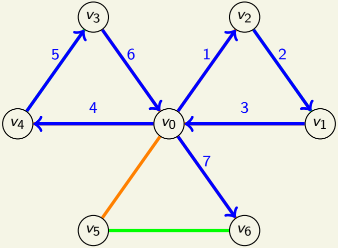
An Eulerian tour.

Revisiting Tight Example for 2-Approximation Algorithm



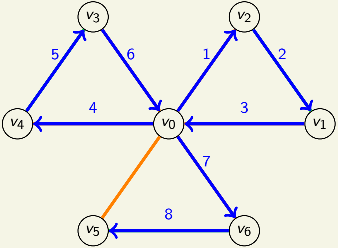
An Eulerian tour.

Revisiting Tight Example for 2-Approximation Algorithm



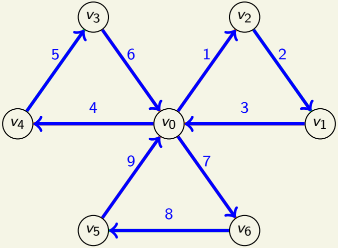
An Eulerian tour.

Revisiting Tight Example for 2-Approximation Algorithm



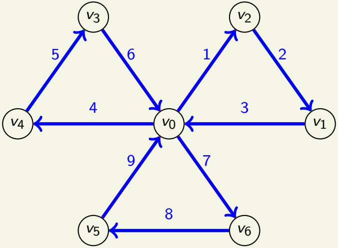
An Eulerian tour.

Revisiting Tight Example for 2-Approximation Algorithm



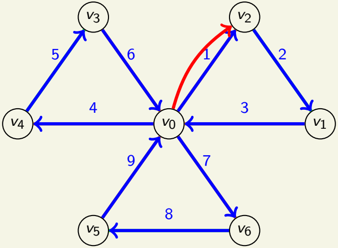
An Eulerian tour.

Revisiting Tight Example for 2-Approximation Algorithm



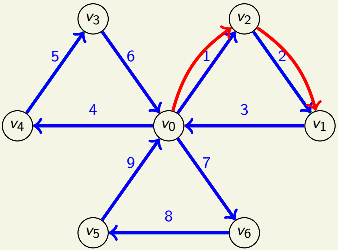
The shortcut tour.

Revisiting Tight Example for 2-Approximation Algorithm



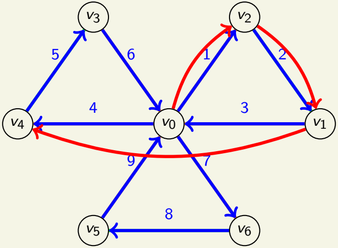
The shortcut tour.

Revisiting Tight Example for 2-Approximation Algorithm



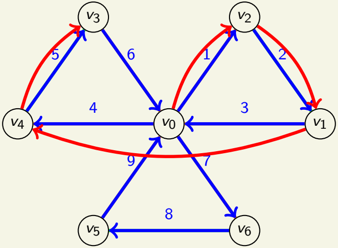
The shortcut tour.

Revisiting Tight Example for 2-Approximation Algorithm



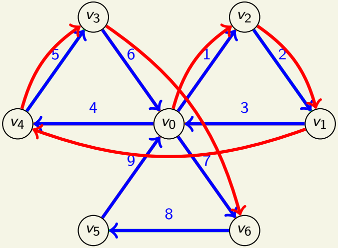
The shortcut tour.

Revisiting Tight Example for 2-Approximation Algorithm



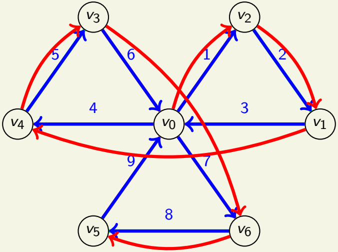
The shortcut tour.

Revisiting Tight Example for 2-Approximation Algorithm



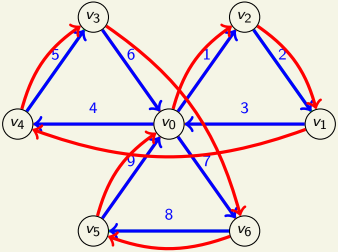
The shortcut tour.

Revisiting Tight Example for 2-Approximation Algorithm



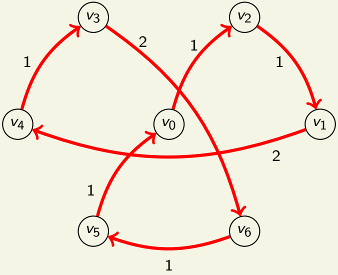
The shortcut tour.

Revisiting Tight Example for 2-Approximation Algorithm



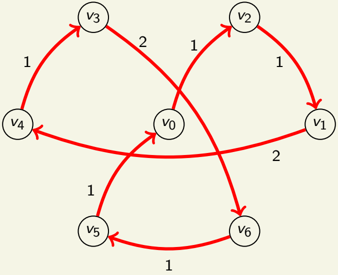
The shortcut tour.

Revisiting Tight Example for 2-Approximation Algorithm



Christofides' algorithm tour cost: 9

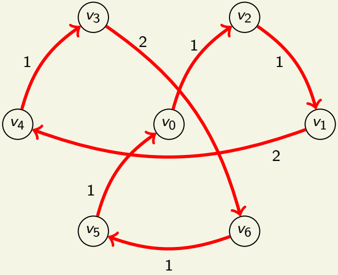
Revisiting Tight Example for 2-Approximation Algorithm



Christofides' algorithm tour cost: 9

Optimal tour cost: 7

Revisiting Tight Example for 2-Approximation Algorithm



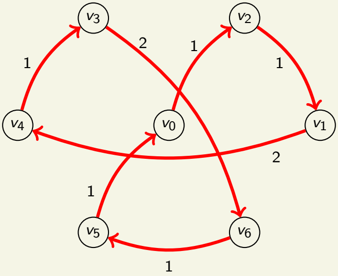
Christofides' algorithm tour cost: 9

Optimal tour cost: 7

Performance ratio: $\frac{9}{7} \leq \frac{3}{2}$

The performance depends on the Eulerian tour. With some Eulerian tours, the algorithm will obtain an optimal TSP tour.

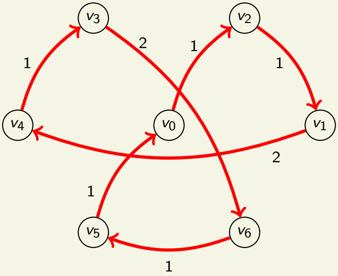
Revisiting Tight Example for 2-Approximation Algorithm



For the analogous graph on $n+1$ vertices with n even, Christofides' algorithm will obtain a tour with cost at most

$$\underbrace{2}_{v_0 \text{ edges}} \cdot 1 + \underbrace{\tfrac{n}{2}}_{\text{edges for pairs}} \cdot 1 + \underbrace{\left(\tfrac{n}{2} - 1\right)}_{\text{edges between pairs}} \cdot 2 = \frac{3n}{2}.$$

Revisiting Tight Example for 2-Approximation Algorithm



For the analogous graph on $n+1$ vertices with n even, Christofides' algorithm will obtain a tour with cost at most

$$\underbrace{2}_{v_0 \text{ edges}} \cdot 1 + \underbrace{\tfrac{n}{2}}_{\text{edges for pairs}} \cdot 1 + \underbrace{\left(\tfrac{n}{2} - 1\right)}_{\text{edges between pairs}} \cdot 2 = \frac{3n}{2}.$$

The optimal tour has cost $n+1$, so the performance ratio is

$$\frac{\frac{3n}{2}}{n+1} = \frac{3}{2} \cdot \frac{n}{n+1}.$$