Part 2 Steiner Tree & TSP (Traveling Salesman Problem)

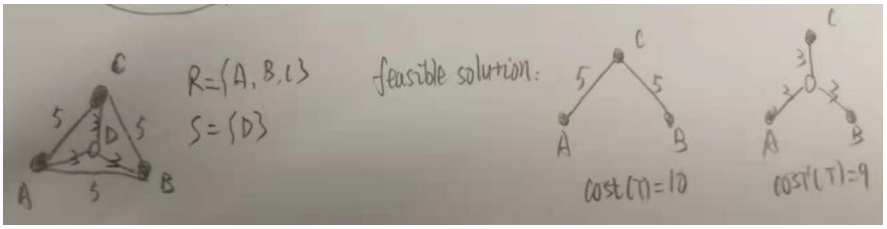
(**Definition 1**) **Metric Steiner Tree Problem**. Given a set $X = R \cup S$, where *R* is a set of **required points**, and *S* is a set of **optional points** (also called as the set of **Steiner vertexes**). Given a distance function $d : X \times X \to R_{\geq 0}$ (with $R_{\geq 0}$ as the set of non-negative real numbers), for all $x, y \in X$, $d(x, y)$ is a non-negative real number. Further, *d* satisfies the **triangle inequality** $d(x, z) \leq d(x, y) + d(y, z)$.

Alternatively, given a **weighted complete graph** on *X*, where the weights are non-negative and satisfy the triangle inequality.

The goal of **Steiner Tree Problem** is to find a **tree** (i.e., connected acyclic subgraph) $T = (V, E)$, where $R \subseteq V \subseteq X$, such that the **cost** of *T* is minimum.

Notes: The Metric Steiner Tree problem is **NP-hard**.

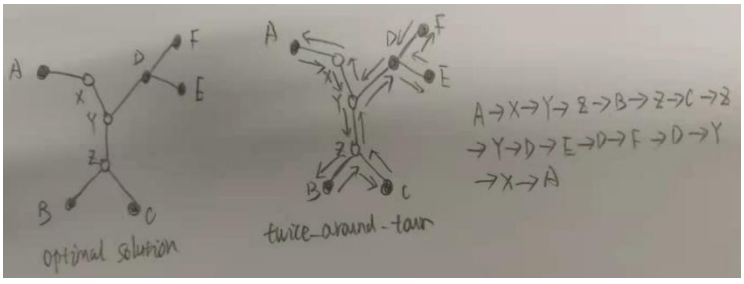
Notes: e.g.,



(**Algorithm 1**) **Approximated Algorithm** for **Metric Steiner Tree**. Consider the subgraph induced by *R* (but not a tree). Find the **minimum spanning tree** (**MST**) to be the output (i.e., the approximated metric Steiner tree).

(**Theorem 1**) The **approximation ratio** of **Algorithm 1** is 2. Namely, the output tree of **Algorithm 1** has the cost at most twice cost of the optimal metric Steiner tree.

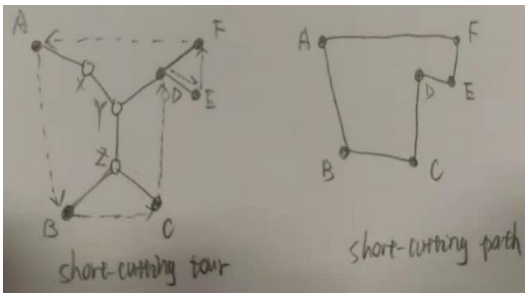
Proof. For the **optimal solution** of the **Metric Steiner Tree** problem, all the leaf nodes must be in *R* (i.e., the required vertices). Consider the **twice-around-tour**, which covers every edge twice and goes through all the required vertices and some optional vertices, e.g.,



Let *Opt* be the cost of the optimal Metric Steiner Tree. The cost of the **twice-around-tour** on the optimal Metric Steiner Tree satisfies that

$$cost(\text{twice-around-tour}) = 2 \cdot Opt .$$

We further **short-cut** the twice-around-tour, which obtain another tour that (i) covers all the required vertices (i.e., vertices in *R*) and (ii) skips all the all the (Steiner) optional vertices (i.e., vertices in *S*) as well as previously visited required vertexes (i.e., each required vertex is visited once), e.g.,



Since in the Metric Steiner Tree problem the distance function d satisfies the **triangle inequality**, the cost of the **short-cutting tour** satisfies that

$$cost(\text{short-cutting tour}) \le cost(\text{twice-around-tour}) = 2 \cdot Opt .$$

By **dropping an arbitrary edge** of the **short-cutting tour**, we obtain a **path** P (a feasible **spanning tree**) involving all the required vertexes (i.e., on R), with the cost satisfying

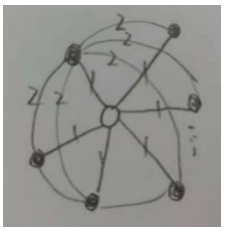
$$cost(P) < cost(\text{short-cutting tour}) \le 2 \cdot Opt .$$

Furthermore, **Algorithm 1** outputs the MST on R, which also satisfy

$$cost(\text{MST}) \le cost(P) < 2 \cdot Opt .$$

Recall that MST is the output of the approximation algorithm (i.e., **Algorithm 1**). Hence, the approximation ratio of **Algorithm 1** is 2.

(**Example 1**) **Tight Example** of **Metric Steiner Tree Problem**. Consider the following complete graph with 1 optional (Steiner) vertex and n required vertices (satisfying the **triangle inequality**!):



The cost of the approximated result (given by **Algorithm 1**) is

$$cost(\text{MST}) = 2(n-1) .$$

The cost of the optimal Metric Steiner Tree is

$$Opt = n .$$

The approximation ratio is

$$\frac{2(n-1)}{n} = 2(1-\frac{1}{n}) .$$

For very large n, we have

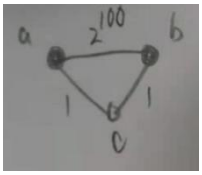
$$\lim_{n\to\infty} 2(1-\frac{1}{n}) = 2 .$$

Notes: The tight example indicates that one cannot improve the analysis much more, e.g., one cannot improve the approximation ratio to be 1.99. Namely, the analysis is already tight for the approximation algorithm. If someone needs to improve the approximation ratio, he/she needs to improve the algorithm.

(**Definition 2**) **General Steiner Tree Problem**. Like the Metric Stiner Tree problem, given a weighted complete graph on $X = R \cup S$, where R is a set of **required points**, and S is a set of **optional points**, and a distance function $d : X \times X \to R_{\ge 0}$, but d doesn't need to satisfy the triangle

inequality. The goal of General Steiner Tree problem is to find a tree $T=(V,E)$, where $R \subseteq V \subseteq X$, such that the **cost** of T is minimum.

(**Example 2**) **Algorithm 1** for **General Steiner Tree**. Consider the following graph:

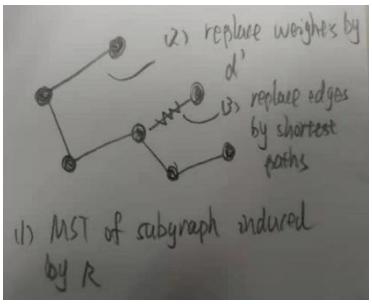


The cost of the MST output of **Algorithm 1** is 2^{100}. The cost of the optimal solution is 2. Hence, the approximation ration of **Algorithm 1** is $2^{100}/2=2^{99}$, which means that **Algorithm 1** is a very bad approximation algorithm for the General Steiner Tree problem.

(**Algorithm 2**) **Approximation Algorithm** for **General Steiner Tree**. For any pair of vertexes (x, y), let $d'(x, y)$ be the length of shortest path between x and y. Then, the distance function d' satisfies the triangle inequality, i.e.,

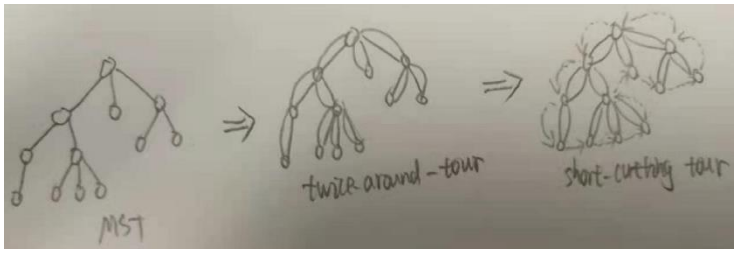
$$d'(x,y) \le d'(x,z) + d'(z,y).$$

Consider the **subgraph** induced by R. Replace the edge weights d with d' and find the MST. To obtain a tree output of the General Steiner Tree problem, further replace the edges of the MST output with the **shortest paths between the corresponding vertex pair** and drops some edges to **remove the remained cycles**.



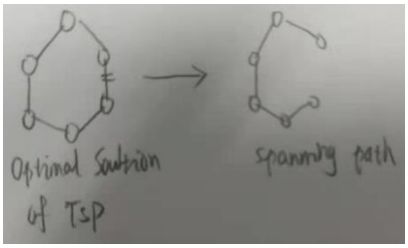
(**Definition 3**) **Travelling Salesman Problem** (**TSP**). Given a **complete weighted graph**, where weights are **non-negative** and satisfy the **triangle inequality**. Our goal is to find a **cycle** that goes through each vertex exactly once and the cost of this cycle is minimum.

(**Algorithm 3**) **Approximation Algorithm** of **TSP**. (1) Construct the MST of the given complete weighted graph. (2) Construct the twice-around-tour of the MST. (3) Short-cut the twice-around-tour and obtain a valid TSP tour.



Notes: By dropping any one edge of the **optimal solution** of TSP (the cycle with minimum cost),

we have a **snapping path** involving all the vertexes:



Note that a snapping path is also a feasible snapping tree, so we have

$$cost(\text{MST}) \le cost(\text{spanning path}) < Opt\ ,$$

i.e., the cost of MST is the **lower bound** of *Opt*. (!!)

We further have

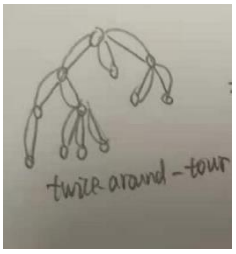
$$cost(\text{apx. output}) \le cost(\text{twice-around-tour}) = 2 \cdot cost(\text{MST}) < 2Opt\ .$$

Namely, **Algorithm 2** gives the result that is <u>within a factor of 2</u> of the **lower bound**, i.e., *cost*(MST).

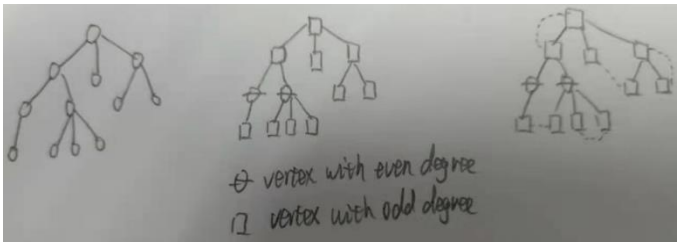
(**Lemma 1**) A graph *G* has a **Eulerian cycle**, i.e., a cycle that visit every **<u>edge</u>** exactly once, iff <u>*G* is **connected** and all the vertexes have **even degrees**</u>.

Notes: Every time one gets into a vertex, he/she needs to leave the vertex, which results in the even degrees of vertexes.

Notes: The **twice-around-tour** is a **Eulerian cycle** in the graph by doubling the edge of MST.



Notes: **Lemma 1** gives a new idea to construct the Euclidean cycle, i.e., we don't need to double all the edges but only need to double some of the edges. Concretely, one can <u>add edges to the MST to ensure all the vertexes are with even degrees</u>. Further, a TSP tour can be derived by <u>short-cutting the constructed Eulerian cycle</u>, which gives more flexibility to the approximation algorithm of TSP.



(**Theorem 2**) The **total degree** of a graph (i.e., sum of degrees of all the vertexes) is **even**. It's because that the total degree is equal to twice of the number of edges (i.e., 2×#Edges).

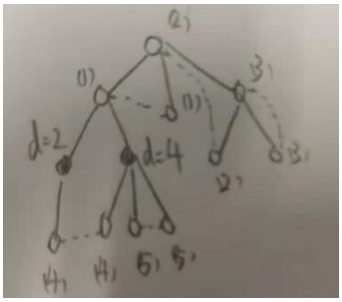
Notes: The **sum of degrees** of all the vertexes with **even degrees** is **even**.

Notes: The **sum of degrees** of all the vertexes with **odd degrees** is also **even**, i.e., total degree of the graph – total degree of vertexes with even degrees.

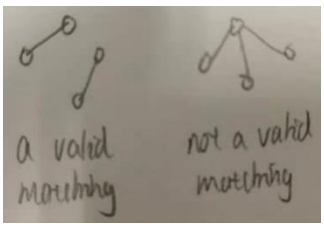
Notes: <u>In any graph *G*, the **number of vertexes** with **odd degrees** (#odd vertexes) is **even**</u>.

Notes: One can extract **pairs of odd vertexes** and add edges to construct the **Eulerian cycle**. For example, an MST with 10 odd vertexes has 5 pairs of odd vertexes. One can add 5 edges to ensure

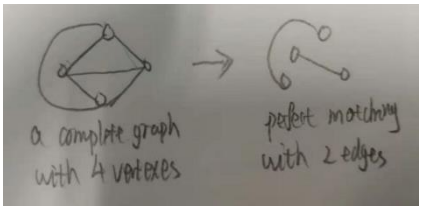
that all the edges in the MST have even degrees.



(**Definition 4**) **Matching**. Given a graph $G=(V, E)$, a subset of edges $M \subseteq E$ is a matching if **no pairs of edges** in M **share a common end vertex**.



(**Definition 5**) **Perfect Matching**. A **matching** is a **perfect matching**, if it covers all the vertexes of the graph. Namely, for **every vertex** of the graph, there should be **an edge** that is **induced** to it. For example, a graph with 10 vertexes, the perfect matching has 5 edges and covers all the vertexes.



(**Theorem 3**) A **complete graph** on end vertexes has a **perfect matching** when **the number of vertexes** is **even**. Any graph with **odd number of vertexes** cannot have a perfect matching.

(**Definition 6**) **Minimum-Weight Perfect Matching**. Given a weighted undirected graph, a minimum-weight perfect matching is a perfect matching with minimum sum of weights.

(**Theorem 4**) Given a **complete weighted graph** G with an **even number of vertexes**, it's possible to compute the **minimum-weight perfect matching** of G in **polynomial time**.

(**Algorithm 4**) **Approximation Algorithm** of **TSP**. In the **MST** T, let S denote the subset of vertexes with **odd degrees** in T. Consider the **subgraph** G_S induced by S in the **original input graph** G, where G_S is also a **complete graph**. Find the **minimum-weight perfect matching** in G_S.

Concretely, the approximation algorithm can be summarized as the following 5 steps:

(1) Compute the MST T w.r.t. G;

(2) Extract the subgraph G_S induced by S, i.e., vertexes with odd degrees;

(3) Compute the **minimum-weight perfect matching** of G_S;

(4) Add all the edges of the **minimum-weight perfect matching** to T, which forms a **Eulerian cycle**;

(5) **Short-cut** the constructed Eulerian cycle to extract the TSP tour.

(**Theorem 5**) The cost of **Algorithm 4** (i.e., the **approximation algorithm** of TSP) satisfies

$$\begin{aligned} cost(\text{apx. output}) &\le cost(\text{Eulerian cycle}) \\ &= cost(\text{MST}) + cost(\text{perfect matching of } G_S)\cdot \\ &\le Opt + Opt/2 = 1.5\cdot Opt \end{aligned}$$

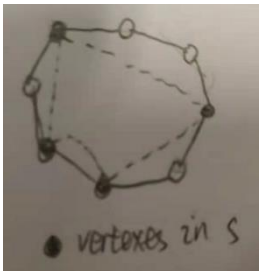
(**Lemma 2**) Let $G=(V, E)$ be a **complete weighted graph**. Let S be a subset of V with **even size** (i.e., even number of vertexes). Let G_S be the induced subgraph over S. Then, the weight of the minimum-weight perfect matching in G_S is at most half of the optimal TSP tour.

Proof. Consider the **optimal TSP tour** w.r.t. the **original graph** G. Some of the vertexes are in S. One can extract **a new tour** only involving vertexes in S by **short-cutting the optimal TSP tour**. We further have

$$cost(\text{new tour}) \le Opt\ .$$

The extracted new tour can be divided into 2 matchings, where we have

$$cost(\text{partitioned matching}) \le cost(\text{new tour})/2 \le Opt/2\ .$$



(**Theorem 6**) The TSP problem **without the triangle inequality** cannot be approximated in **polynomial time** within any constant factor C, unless $P=NP$.