Large-scale Knowledge Processing
Lecture 8

Kazuhisa Seto

Today's Lecture

Study algorithms for NP-complete problems.

- Exact Algorithms and Approximation Algorithms
- 2-Approximation Algorithm for the Metric TSP
- Kruskal's Algorithm for the Minimum Spanning Tree

Exact Algorithms and Approximation Algorithms

Review: NP-complete Problem

A NP-complete problem is the problem A satisfying the following two conditions.

- A is in NP.
- Any problem in NP is reducible to A in polynomial time.

It is believed that NP-complete problems cannot be solved in deterministic polynomial time.

⇒ The size of inputs grows large, solving NP-complete problem takes huge time.

Review: NP-complete Problem

But, many problems in the real world are formulated to some NP-complete problem.

Thus, we cannot give up to solve the problem, and we must try to solve in some way.

There exists two main approaches : Exact Algorithms and Approximation Algorithms.

Exact Algorithms vs. Approximation Algorithms

Exact Algorithm

- Search an exact solution
- It may run in exponential time
- In many cases of the real world, we must get an exact solution. The other solutions has no meanings.
- We need a faster algorithm as possible

Approximation Algorithm

- Search an approximate solution
- It must run in polynomial time
- In many cases of the real world, we need not an exact solution, but we get good approximation solution as fast as possible.
- We need a good approximation algorithm as possible.

Trade Off !

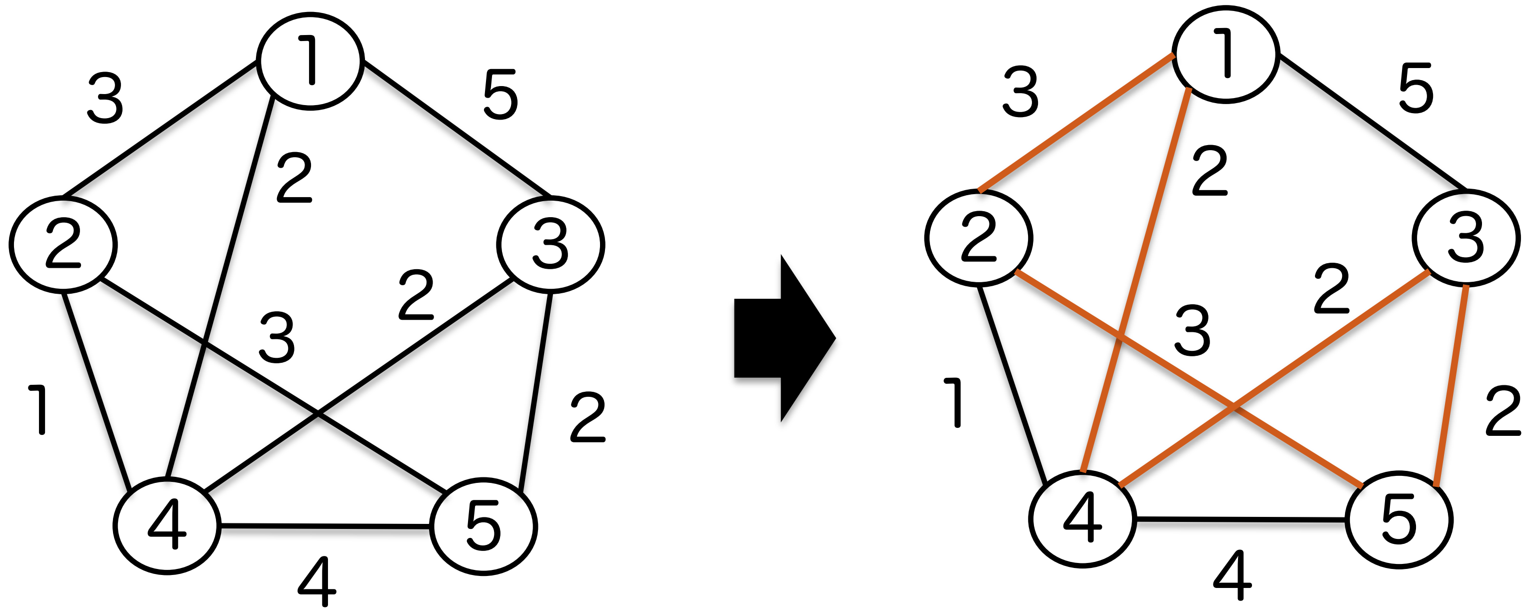
Traveling Salesman Problem（TSP）

Input：a weighted graph G

Ask：Search a minimum Hamiltonian cycle

Output：a minimum Hamiltonian cycle (MHC)

MHC: A cycle C that visits every vertex of G exactly at once and the total weight of edges in C is minimum.

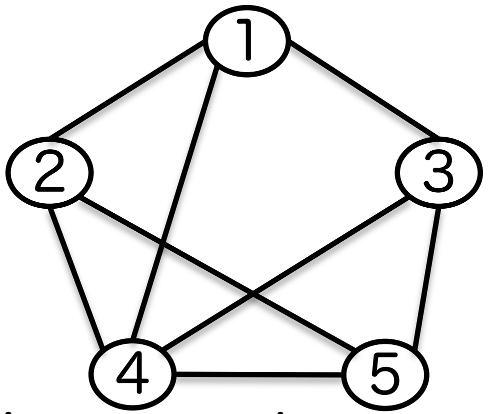


Metric TSP

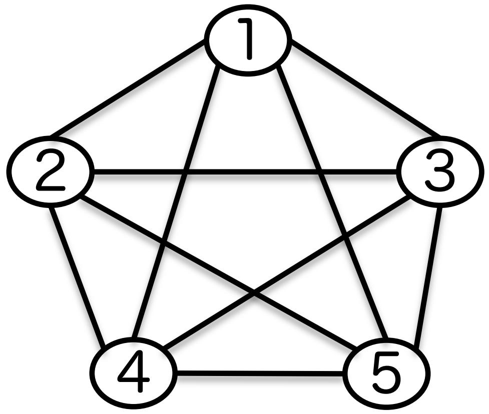
Metric TSP：TSP satisfies the following two conditions

- ➢ Input graph is a complete graph
  - ✓ Complete graph: any pair of vertices has an edge.
- ➢ Let $\mathrm{w}(x, y)$ is a weight between x and y. Any three vertices A, B, C satisfies the triangle inequality.

$$w(A, B) \leq w(A, C) + w(B, C)$$



Not complete graph



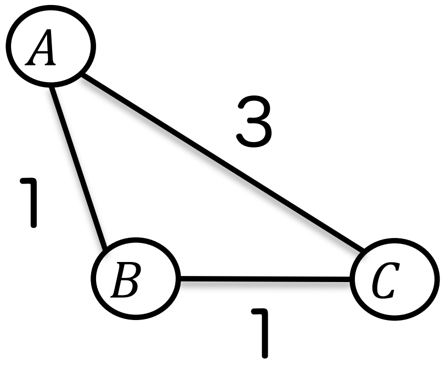
complete graph

Metric TSP

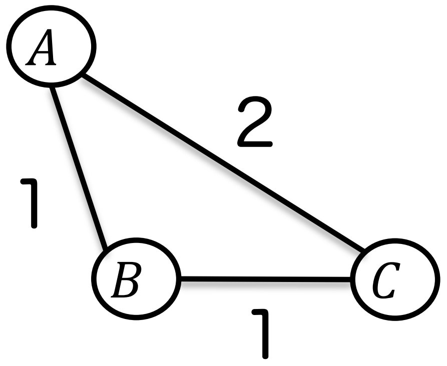
Metric TSP：TSP satisfies the following two conditions

- ➢ Input graph is a complete graph
  - ✓ Complete graph: any pair of vertices has an edge.
- ➢ Let $w(x, y)$ is a weight between x and y. Any three vertices A, B, C satisfies the triangle inequality.

$$w(A, B) \leq w(A, C) + w(B, C)$$



triangle inequality doesn't hold



triangle inequality holds

The definition of Approximation Ratio

Because Metric TSP is NP-hard problem, it seems to be impossible to get an optimal solution in polynomial time.
⇒ We try to get a good approximation solution in polynomial time.

We estimate the performance of approximation algorithms by approximation ratios.

The definition of Approximation Ratio

Approximation ratio is the worst value of

$$\frac{\text{Cost of the solution solved by Algorithm}}{\text{Cost of the optimal solution}}$$

A c-approximation algorithm is defined as an approximation algorithm whose approximation ratio is less than or equal c.

History of Approximation ratio for Metric TSP

Upper Bounds of approximation ratio

- 2-approximation [Folklore]
- 1.5-approximation [Cristofides 1976]
- $1.5\text{-}10^{-36}$ – approximation [Karlin, Klein, and Graham 2020]

Hardness of approximation

- Unless P=NP, there in no 123/122-approximation polynomial-time algorithm. [Karpinski, Lampis, and Schmied 2006]

An Exact Algorithm for TSP

There exists an $O(n^2 2^n)$ time exact algorithm for TSP by using a dynamic programming.
However, it has yet known an exact algorithm essentially faster than $O(n^2 2^n)$ time.

If you know the algorithm by using a dynamic programming, you search in web sites.
There are many articles about it.

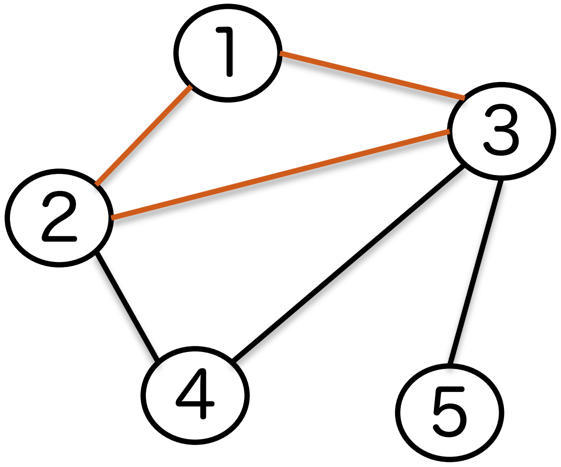
2-approximation Algorithm for TSP

The definition of Tree

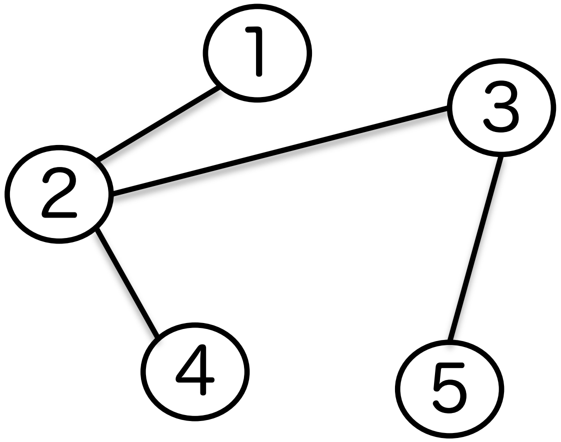
A tree is a graph that has no cycle.

In the following figures,

- The graph G_1 has a cycle (1-2-3-1)
- The graph G_2 has no cycle, thus it is a tree.



G_1

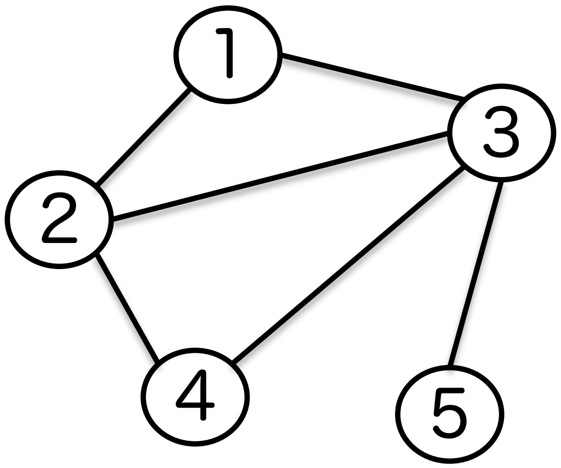


G_2

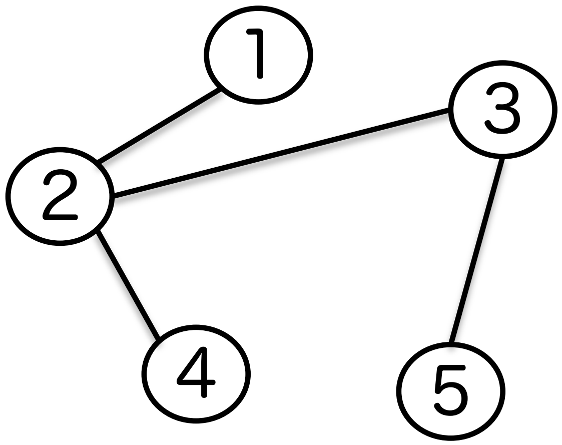
The definition of subgraph

A subgraph H is a graph using a subset of vertices and edges in G.

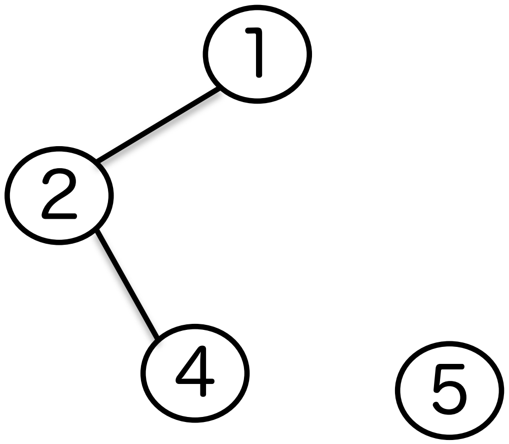
- In the following figures, H_1 and H_2 are subgraph of G.



(i) G



(ii) H_1

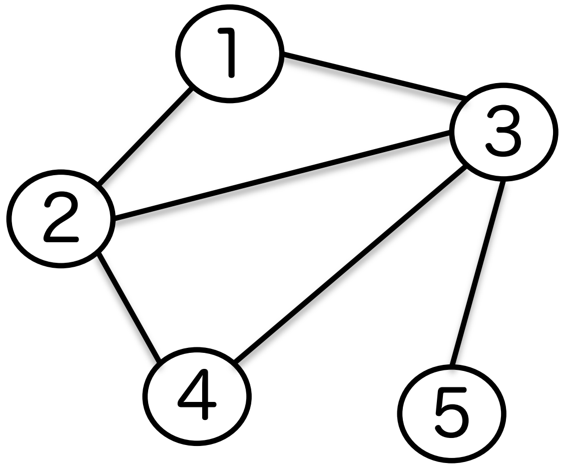


(iii) H_2

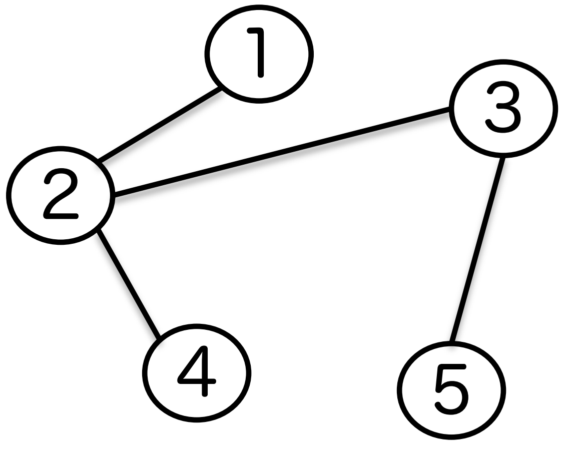
The definition of spanning tree

A spanning tree is a subgraph such that it is a tree and all vertices are connected.

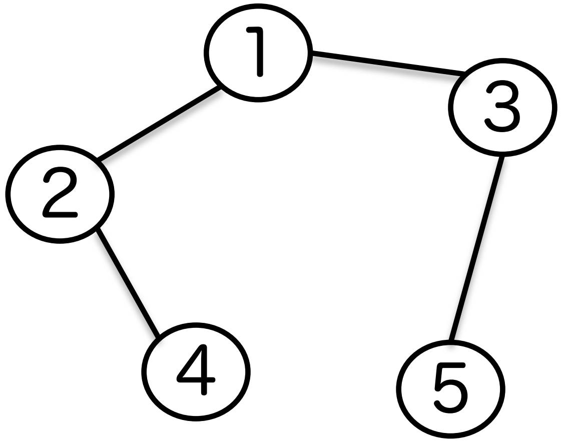
- In the following figures, H_1 and H_2 are spanning trees of G.



(i) G

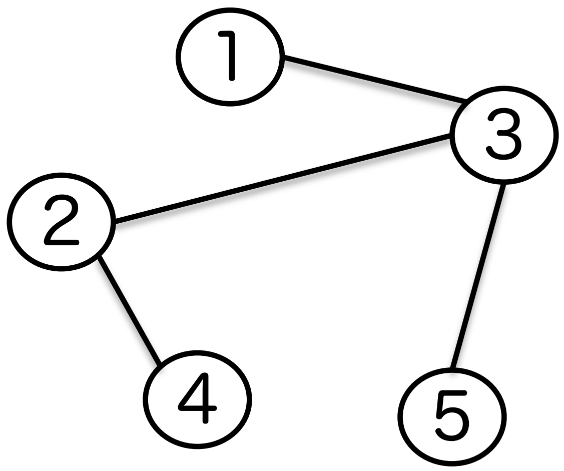


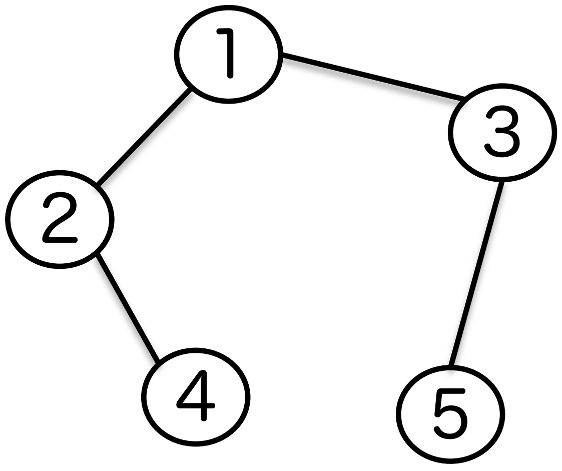
(ii) H_1

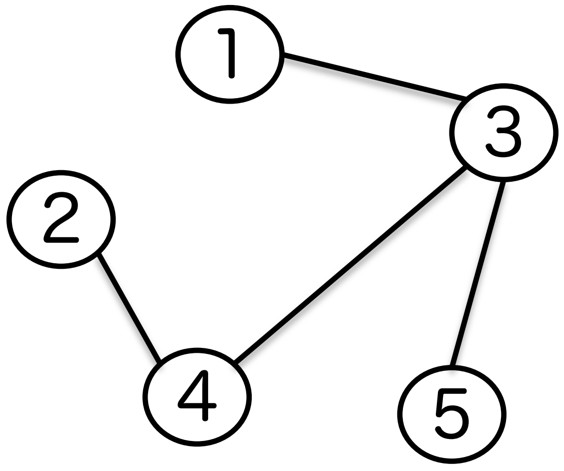


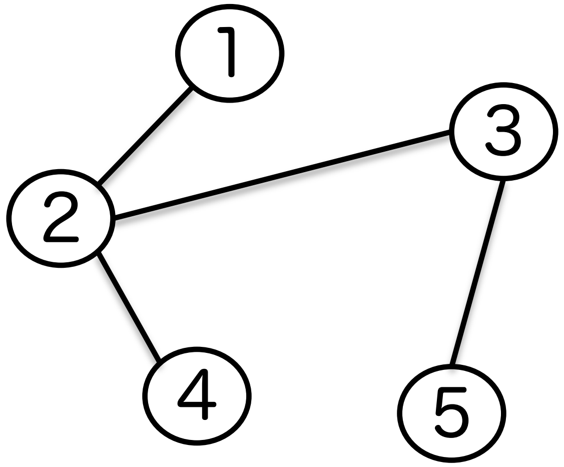
(iii) H_2

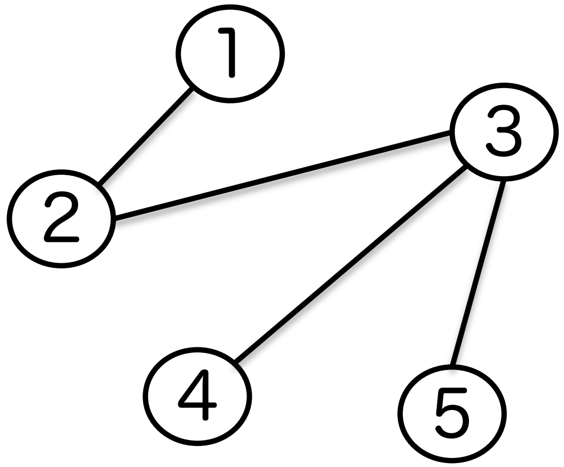
Enumerating all spanning trees of G in the previous slide.

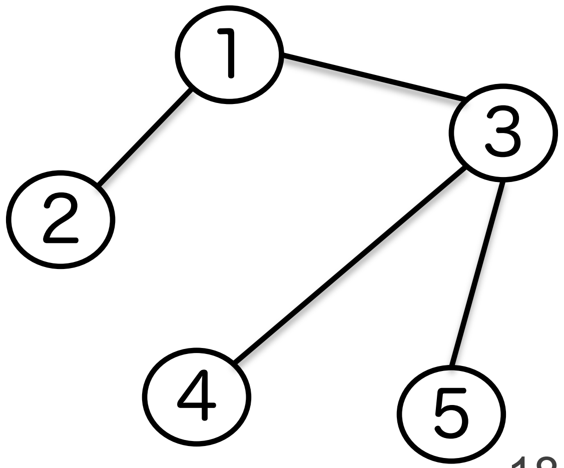








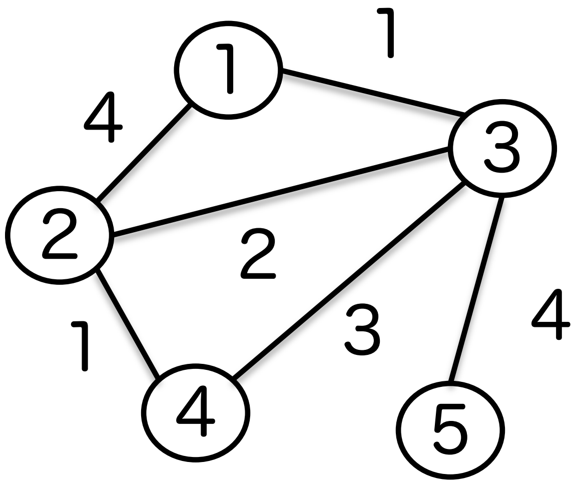




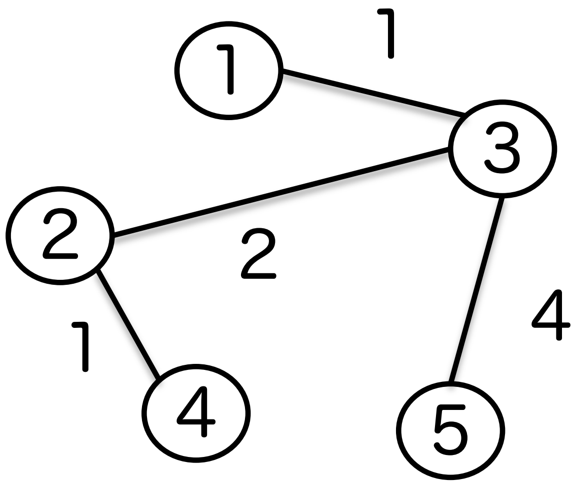
The definition of Minimum Spanning Tree

Given a weighted graph G, a minimum spanning tree T is a spanning tree such that the total edge weight of T is minimum between all spanning trees.

- H is a minimum spanning tree of G.



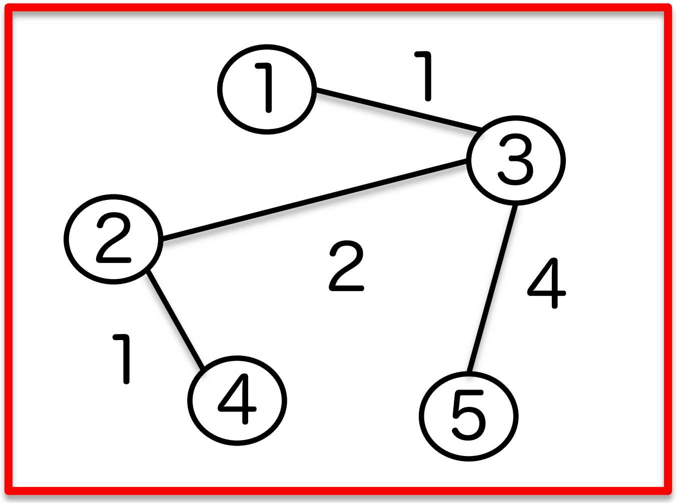
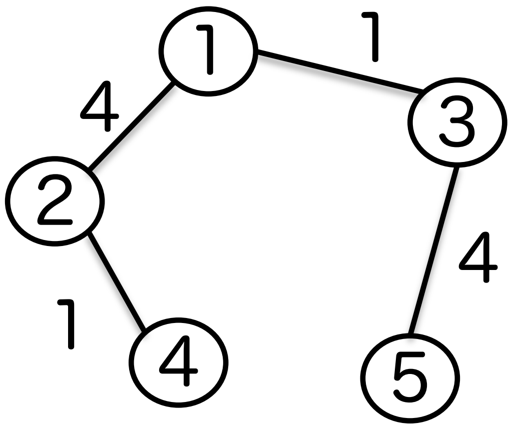
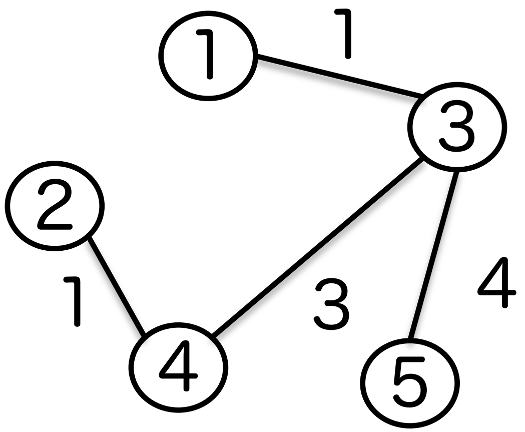
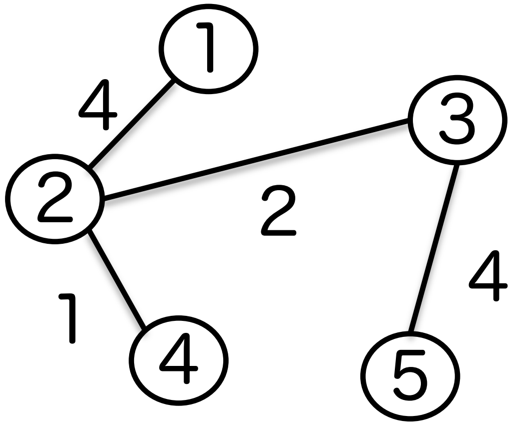
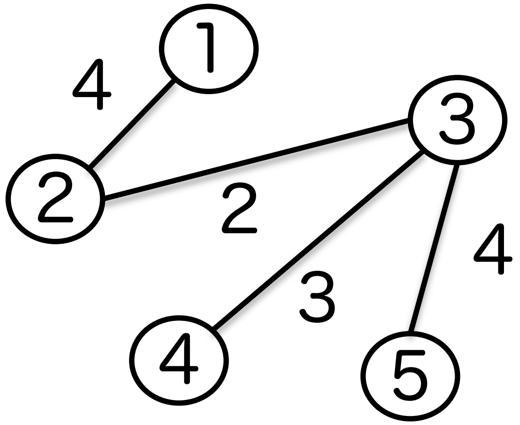
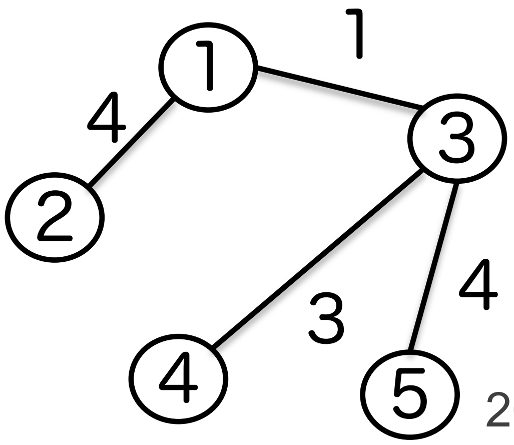
G



T

An example of minimum spanning trees

The spanning tree in the red rectangle is the minimum spanning tree of G in the previous slide.

2-Approximation Algorithm for Metric TSP

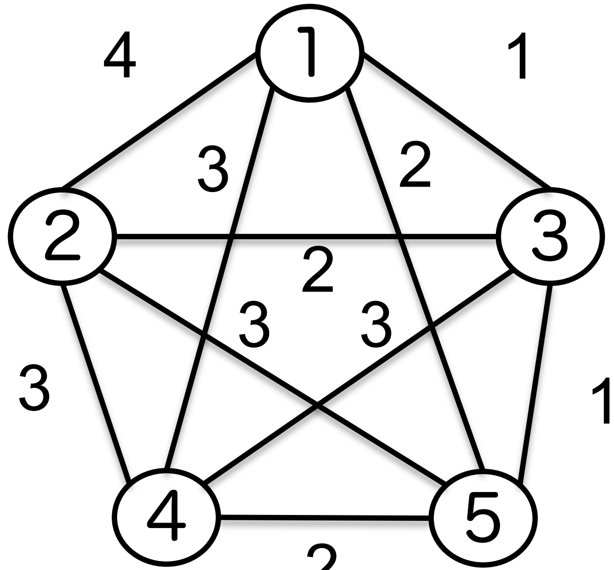
We use a minimum spanning tree.

1. Compute a minimum spanning tree T of a given graph G.
2. Construct the graph T' from T by doubling every edge.
3. Compute a cycle C throughout all vertices on T'.

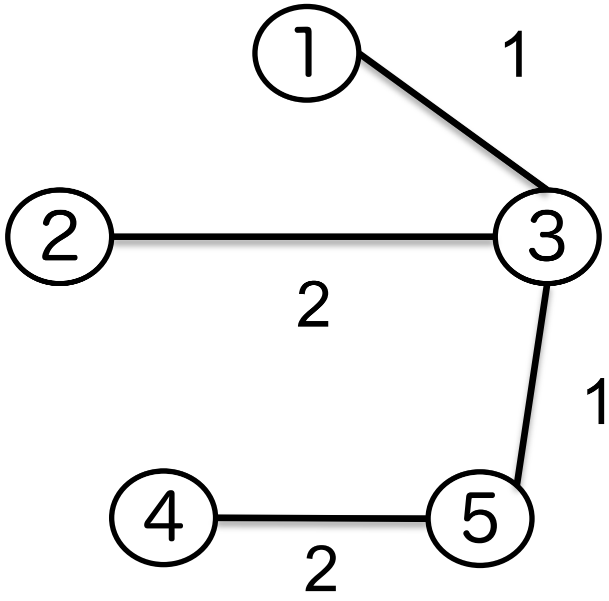
   ※ The same vertex may be passed through twice.
4. Compute the Hamiltonian cycle C' from C to shortcut vertices already passed.
5. Output C' and the total weight of edges in C’.

An example of Algorithm's procedure

1. Compute a minimum spanning tree T of a given graph G.
   It's done by Kruskal's algorithm（explain it later）or Prim's algorithm



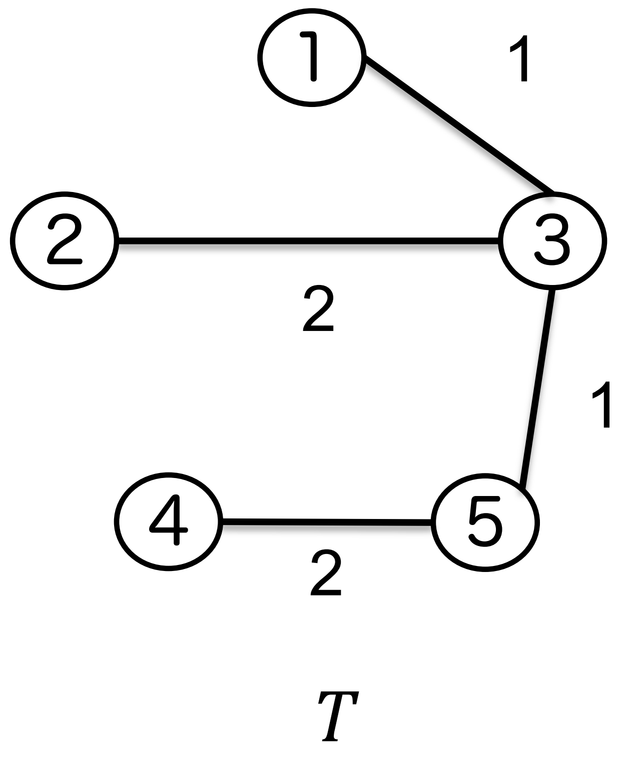
G

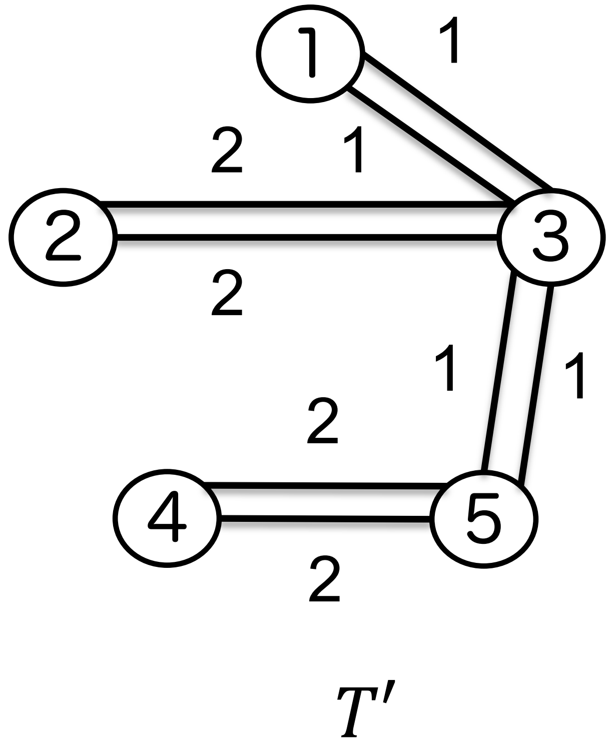


T

An example of Algorithm’s procedure

2. Construct the graph T' from T by doubling every edge.

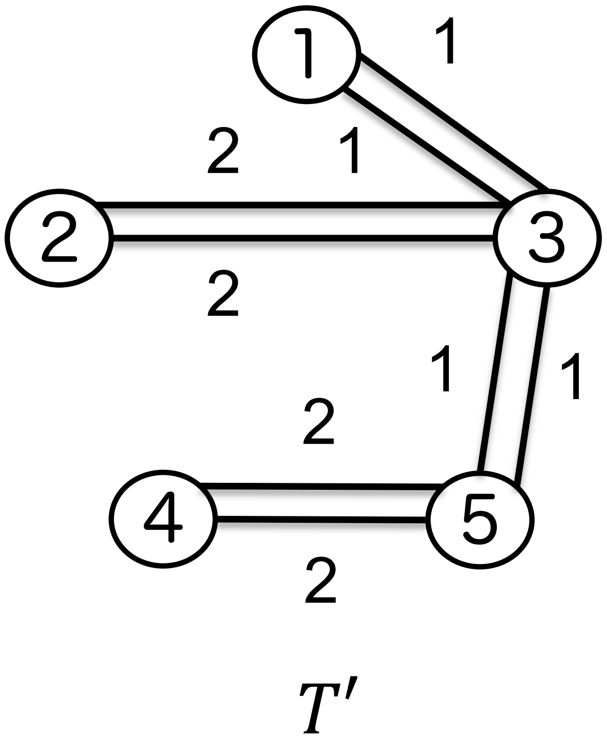




An example of Algorithm's procedure

3. Compute a cycle C throughout all vertices on T.

   ※ The same vertex may be passed through twice.



$1 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 5$

$\rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 1$

C

An example of Algorithm's procedure

4. Compute the Hamiltonian cycle C' from C to shortcut vertices already passed.

$1 \to 3 \to 2 \to 3 \to 5$
$\to 4 \to 5 \to 3 \to 1$

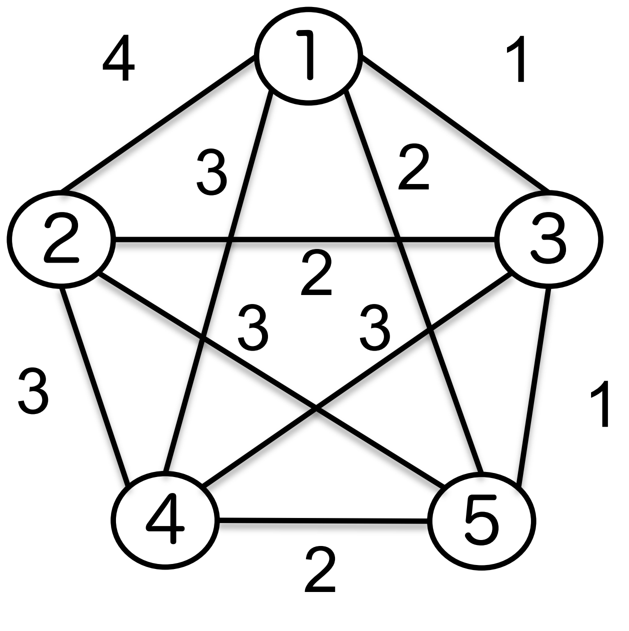
$1 \to 3 \to 2 \to 5 \to 4 \to 1$

C

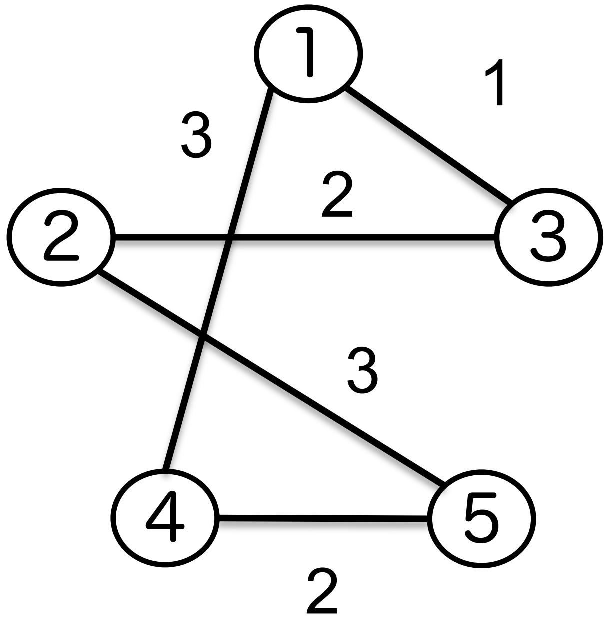
C'

An example of Algorithm's procedure

5. Output C' and the total weight of edges in C'.



G



C' :1 → 3 → 2 → 5 → 4 → 1

The total weights : 11

The running time of the algorithm

Let the number of vertices and that of edges be n and m, respectively.

1. Compute a minimum spanning tree T of a given graph G.
   $\rightarrow O(n \log_2 m)$ (explain it later)
2. Construct the graph T' from T by doubling every edge.
   $\rightarrow O(n)$
3. Compute a cycle C throughout all vertices on T'. $\rightarrow O(n)$
4. Compute the Hamiltonian cycle C' from C to shortcut vertices already passed. $\rightarrow O(n)$
5. Output C' and the total weight of edges in C'. $\rightarrow O(n)$

The total running time is $O(n \log_2 m)$.

The proof of 2-approximation

We define the following about a given graph G.

OPT: The total weights of a min. Hamiltonian cycle of G

MST: The total weights of a min. spanning tree of G

First, it is easy to see that

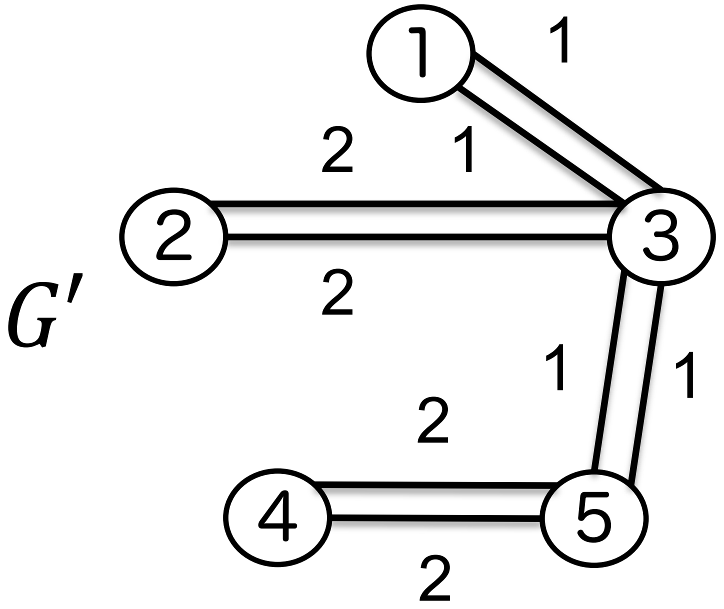
$$\mathrm{MST} \leqq \mathrm{OPT}$$

holds because any Hamiltonian cycle has one more edge than a minimum spanning tree has.

The proof of 2-approximation

We consider a cycle C throughout all vertices on T' constructed from a minimum spanning tree T by doubling every edge.

The total weights of edges in C is 2×MST.



C: 1 → 3 → 2 → 3 → 5 → 4 → 5 → 3 → 1

The proof of 2-approximation

We consider the Hamiltonian cycle C' from C to shortcut vertices already passed.

Let ALG be the total weights of edges in C'.

Then,

$$\text{ALG} \leqq 2\times\text{MST}$$

holds.

Why？？

→ Metric TSP has triangle inequality constraints.

The proof of 2-approximation

See example. We consider C and C' in slide 25.

C : $1 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 1$

C': $1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 1$

First, we shortcut $2 \rightarrow 3 \rightarrow 5$ to $2 \rightarrow 5$.

Note that triangular inequality between the weights between any three vertices, then

$$w(2,5) \leq w(2,3) + w(3,5)$$

holds. If $w(2,3) \leq w(3,5)$, then $w(2,5) \leq 2w(3,5)$.

Otherwise, $w(2,5) \leq 2w(2,3)$.

Repeating this argument, the following holds.

$$\text{ALG} \leqq 2 \times \text{MST}$$

The proof of 2-approximation

From the previous discussion,

- MST ≦ OPT
- ALG ≦ 2×MST

hold, and then

$$\mathrm{ALG} \leqq 2\times\mathrm{MST} \leqq 2\times\mathrm{OPT}$$

holds.

Thus, for any graph G, ALG/OPT ≦ 2 holds.

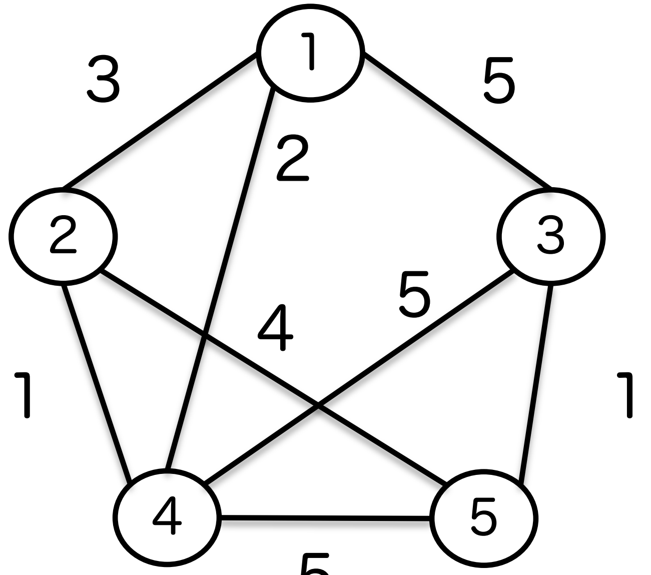
The proof is completed.

Kruskal’s Algorithm for MST

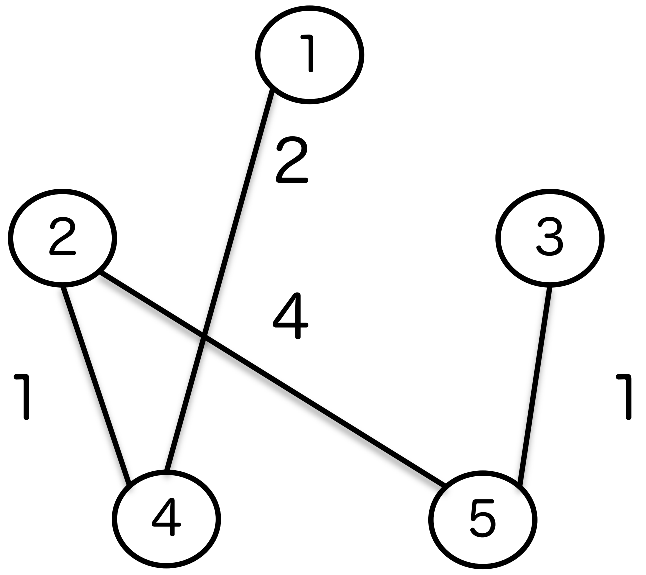
Kruskal's Algorithm

In the figure, T is a minimum spanning tree of G.

Our aim is to understand Kruskal's algorithm.



G



T

Kruskal's Algorithm

Let n be the number of vertices of G.

1. We set T to be an empty graph (no vertex and no edge).
2. Until the number edges in T is $n-1$, we repeat the following operations.

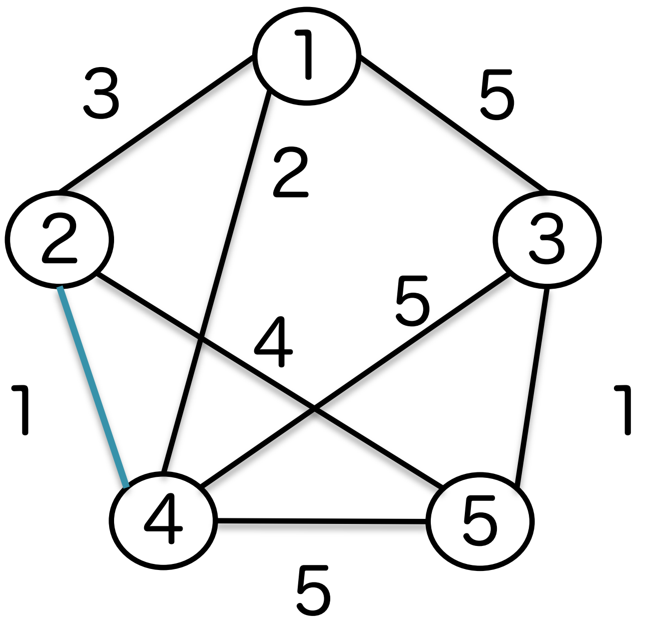
   A) Pick up an edge e having the minimum weight.

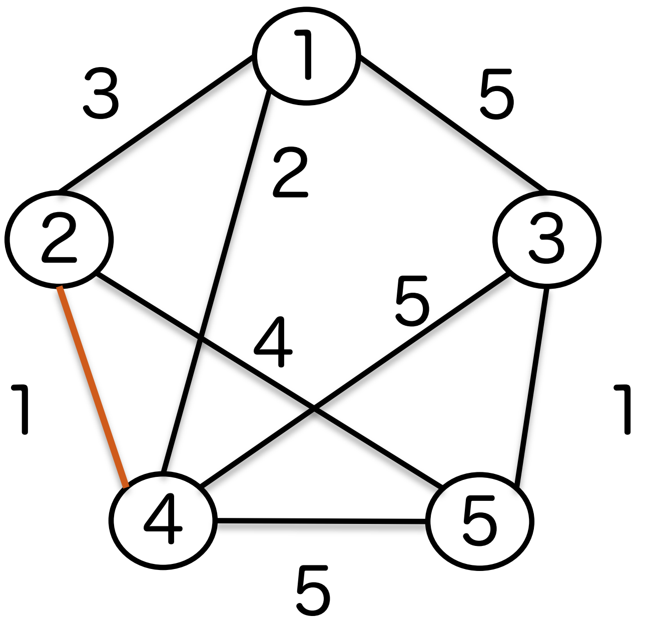
   - If the edge e and edges in T make a cycle, we don't adopt it as an edge of T.
   - Otherwise, we adopt e as an edge of T.
3. Output a tree T with $n-1$ edges.

An example of Algorithm's procedure

From 2-(A), we must pick up an edge e with the minimum weight. Now, there are two edges (2, 4) and (3, 5) with the minimum weight. We can choose both edges.

First, we pick up the edge (2,4) and adopt it as an edge of T.



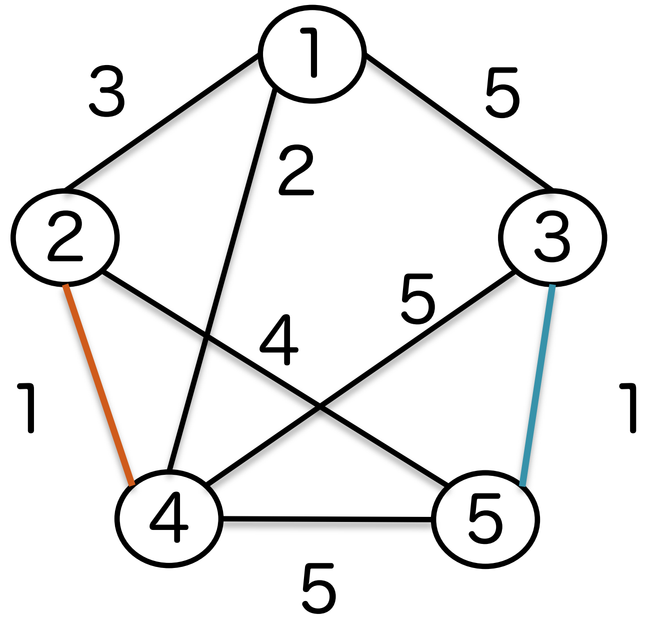


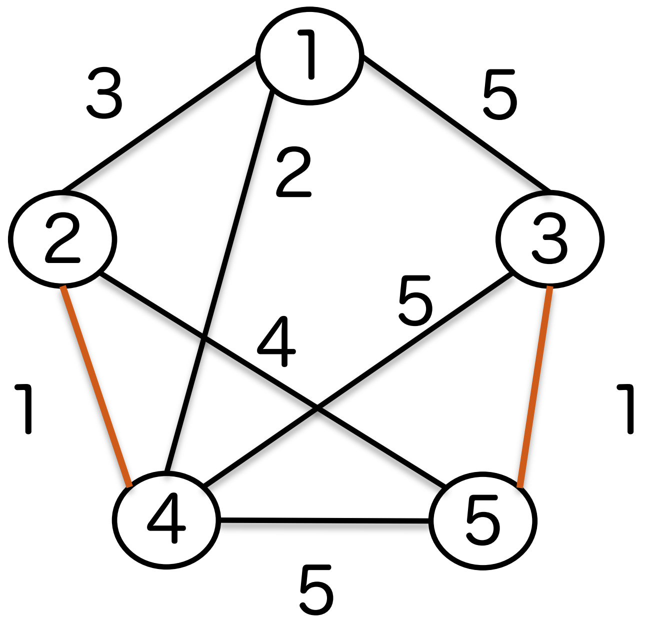
An example of Algorithm's procedure

Next, there is one edge (3,5) with the min. weight.

We pick up it and it doesn't make a cycle.

Thus, we adopt it as an edge of T.



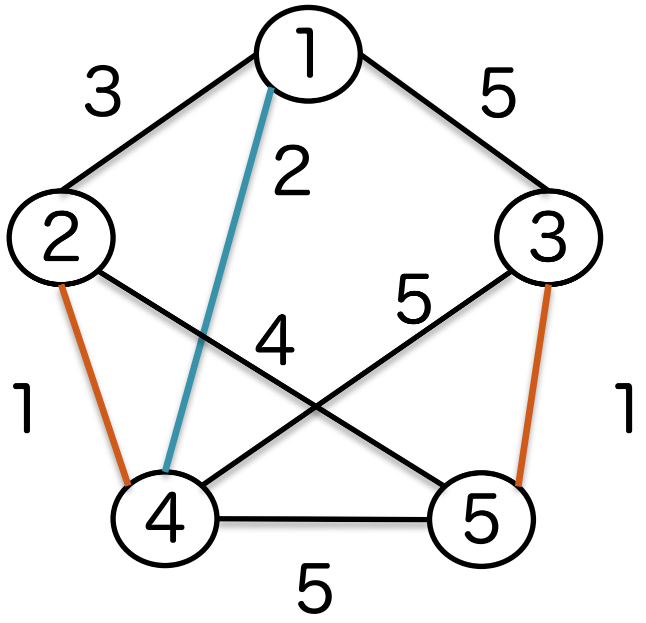


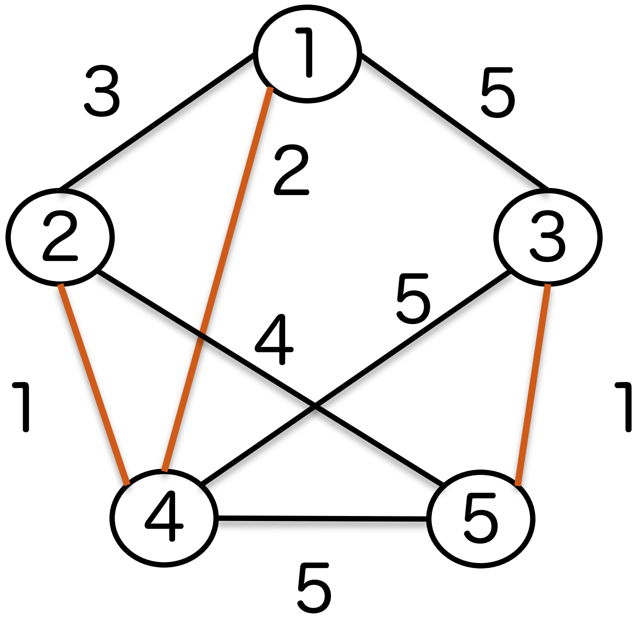
An example of Algorithm's procedure

Next, there is one edge (1,4) with the min. weight.

We pick up it and it doesn't make a cycle.

Thus, we adopt it as an edge of T.



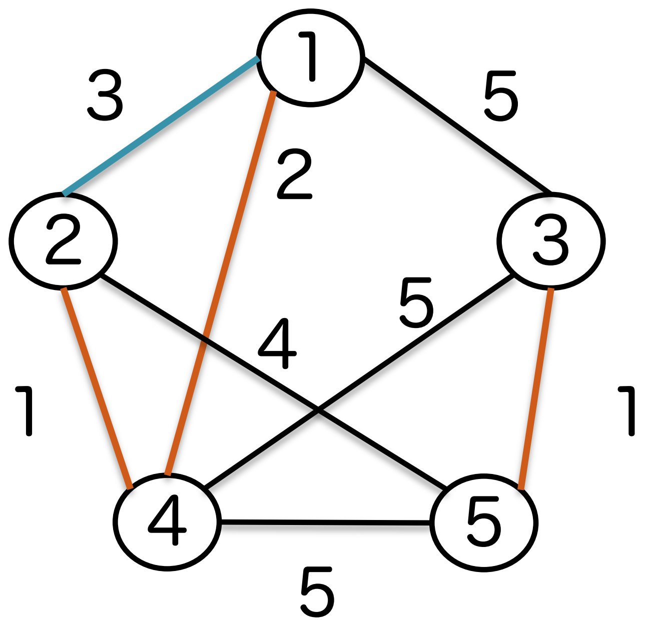


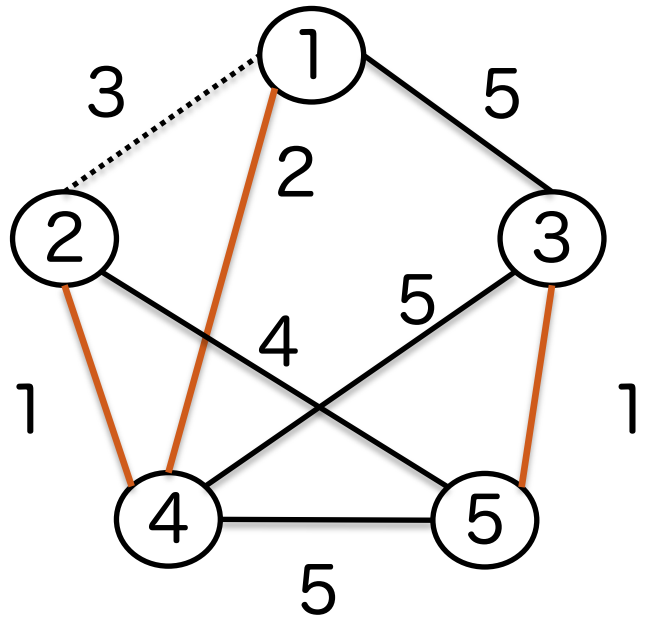
An example of Algorithm’s procedure

Next, there is one edge (1, 2) with the minimum weight.

We pick up it, but it makes a cycle with edges (1,4) and (2, 4) which have already been adopted.

Thus, we discard it and pick up the other edge.



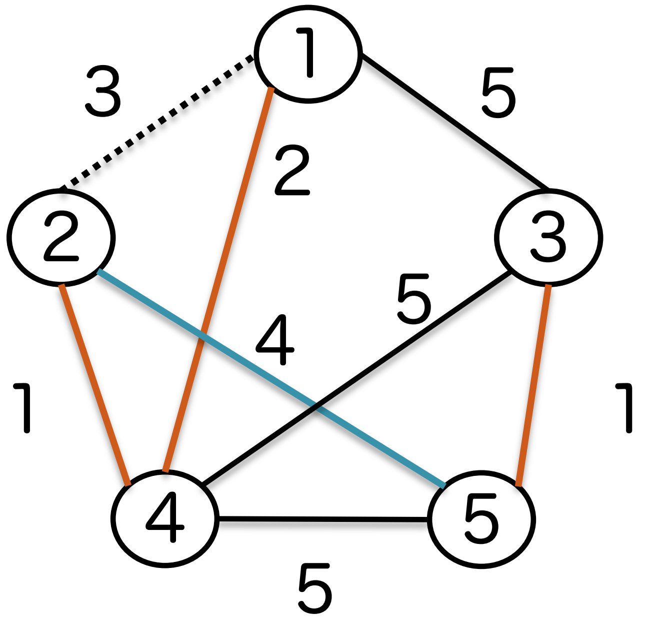


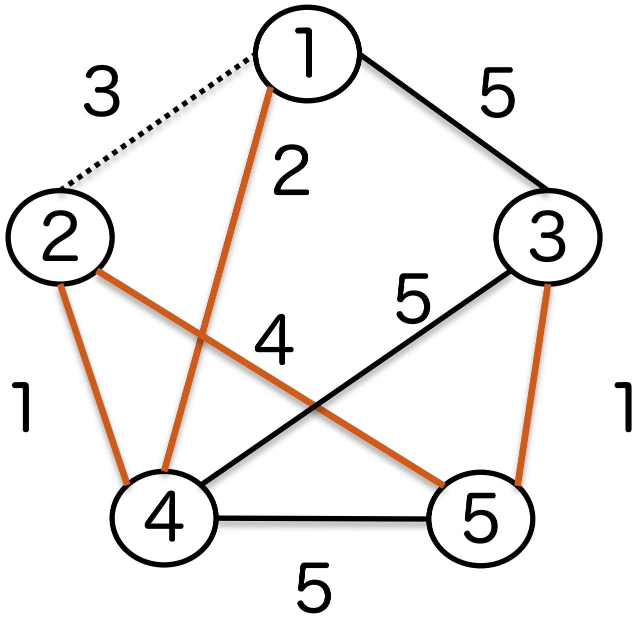
An example of Algorithm's procedure

Next, there is one edge (2,5) with the min. weight.

We pick up it and it doesn't make a cycle.

Thus, we adopt it as an edge of T.



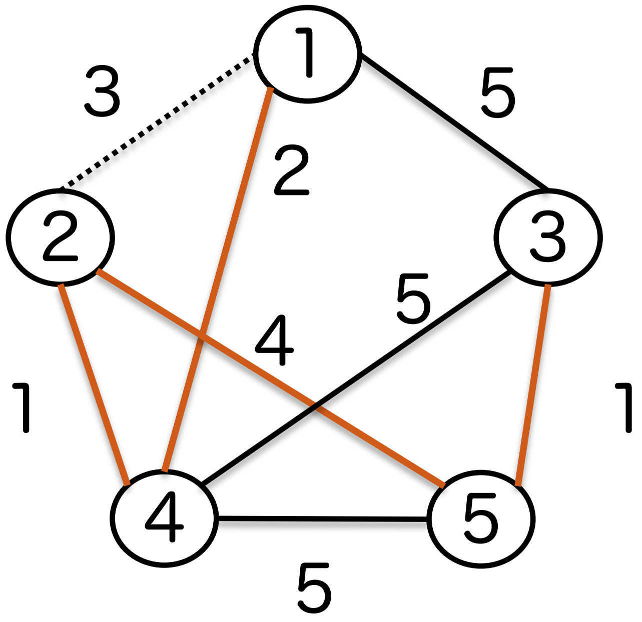


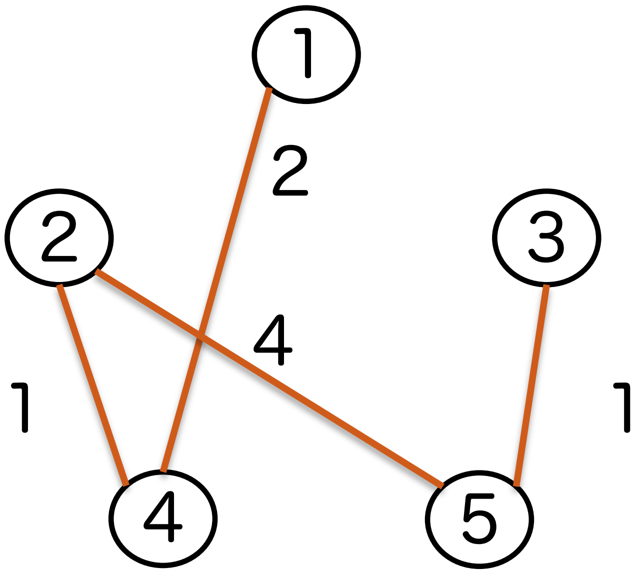
An example of Algorithm's procedure

Now, we have adopted with $n - 1\ (= 5 - 1 = 4)$ edges.

The algorithm stops and outputs T.

Indeed, we can see that T is a minimum spanning tree.





The running time of the algorithm

Let #vertices and #edges be n and m, respectively.

1. We set T to be an empty graph. $\rightarrow O(1)$
2. Until the number edges in T is $n-1$, we repeat the following operations.

   A) Pick up an edge e having the minimum weight.

   $\rightarrow$ the edges need to be sorted by weight. It takes $O(m\log_2 m)$.

   - If the edge e and edges in T make a cycle, we don't adopt it as an edge of T.

     $\rightarrow$ Checking whether e makes a cycle is $O(m)+O(n\log_2 n)$ time by using the union-find.

   - Otherwise, we adopt e as an edge of T. $\rightarrow O(1)$

3. Output a tree T with $n-1$ edges. $\rightarrow O(n)$

The total time is $O(m\log_2 m)$ and $m \leq n^2 \rightarrow O(m\log_2 n)$ time.

Summary

We study on fundamentals of approximation algorithms that attack toward NP-complete problems.

- 2-approximation algorithm for Metric TSP.
- Kruskal's algorithm for Minimum Spanning Trees.

The data is bigger and bigger.

- $O(n^3)$ time algorithms may be useless.
- It is important to find a "good" solution as fast as possible
- It is more pleasure if the worst-case complexity is ensured.