Discrete Mathematics

Lecture 9

Finding the Optimum

Contents

1. Finding the **best** tree

- Borůvka(1926)-Kruskal(1956)'s Algorithm
- Jarník(1930)-Prim(1957)'s Algorithm (9.2.6)

2. The **traveling salesman problem**

- Tree Shortcut Algorithm

9.1 Finding the Best Tree

连通网问题：

假设要在城市之间建立通讯联络网，则连通*n*个城市只需要修建*n*-1条线路，如何在最节省经费的前提下建立这个通讯网？

最小生成树

设G=(*V*，*E*)是一连通图，G的每一条边e有权c(e)，G的生成树*T*的权c(*T*)就是*T*的边的权和。

定义　在图G所有生成树中，树权最小的那棵树称为G的最小生成树。

前问题等价于：

构造网的一棵最小生成树，即：在*m*条带权的边中选取*n*-1条边（不含圈），使"权值之和"为最小。

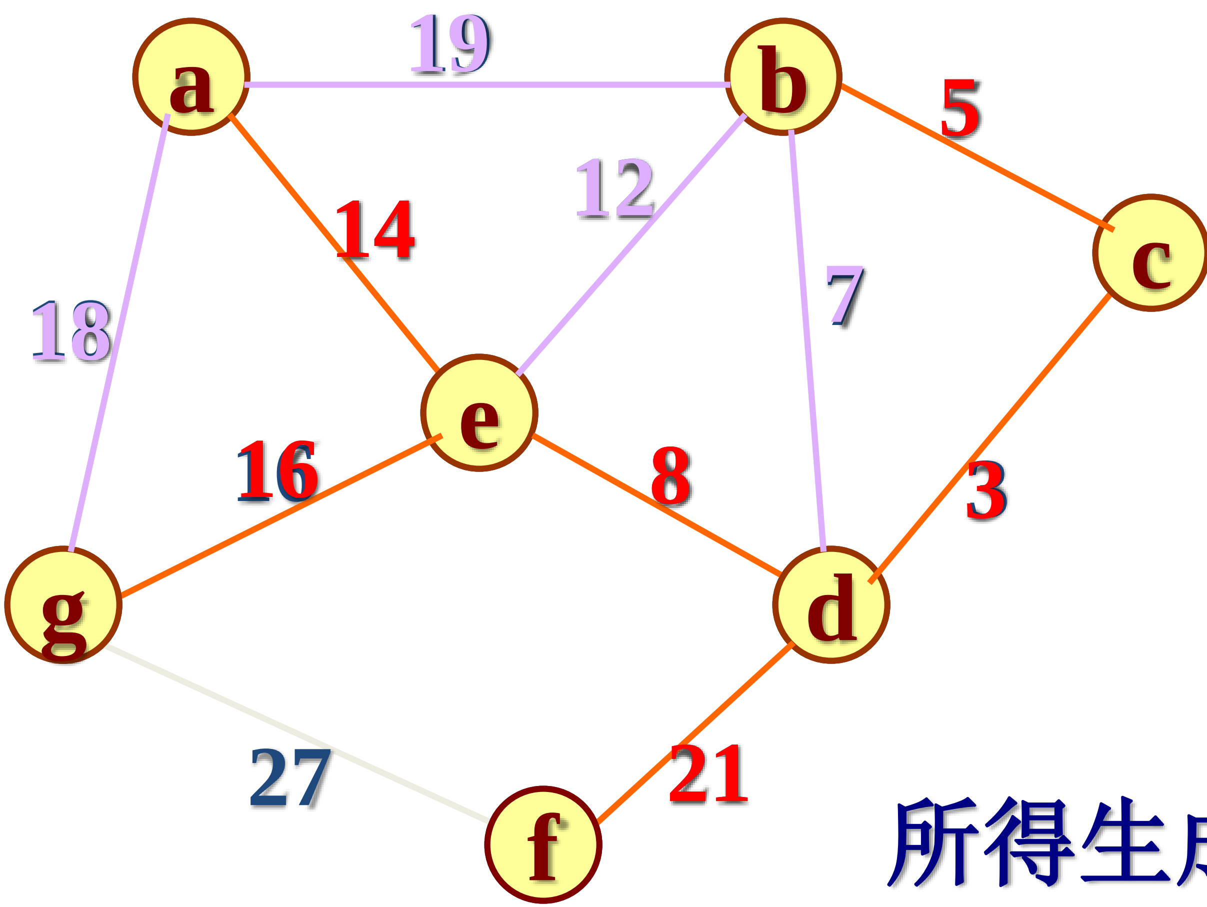
Borůvka(1926)-Kruskal(1956)'s Alg.

Jarník(1930)-Prim(1957)'s Algorithm

Borůvka-Kruskal算法的基本思想

- 考虑问题的出发点: 为使生成树上边的权值之和达到最小，则应使生成树中每一条边的权值尽可能地小。
- 具体做法: 先从权值最小的边开始，若它的添加不使图中产生圈，则在图上加上这条边，如此重复，直至加上n-1条边为止。

例如:



所得生成树权值和

=3+5+8+14+16+21=67.

Borůvka-Kruskal算法(避圈法):

a) 在G中选取最小权的边,记作e_1,置i=1。

b) 当$i=n-1$时结束，否则转c)。

c) 设已选择边为$e_1,e_2,\ldots,e_i$，此时无圈。在G中选取不同于这i条边的边e_{i+1}，该边使得$\{e_1，\ldots，e_{i+1}\}$生成的子图中无圈，并且e_{i+1}是满足该条件中权最小的一条边。

d) 置$i:=i+1$，转b)。

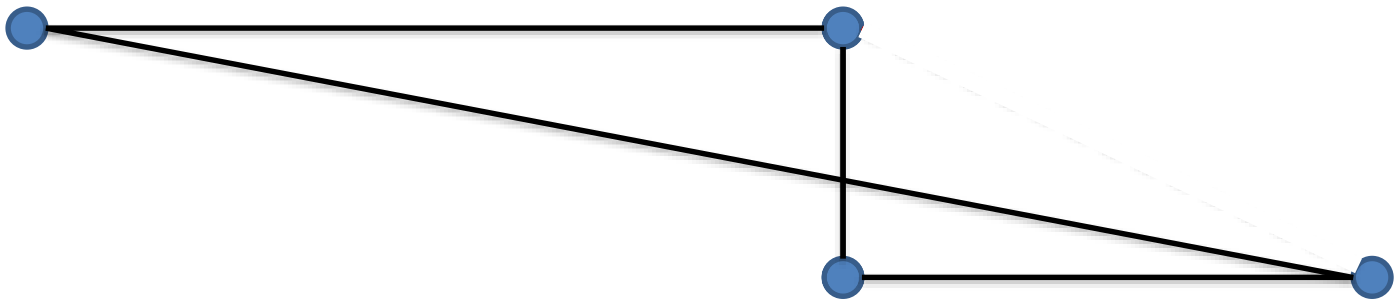
Is a Borůvka-Kruskal tree the best?

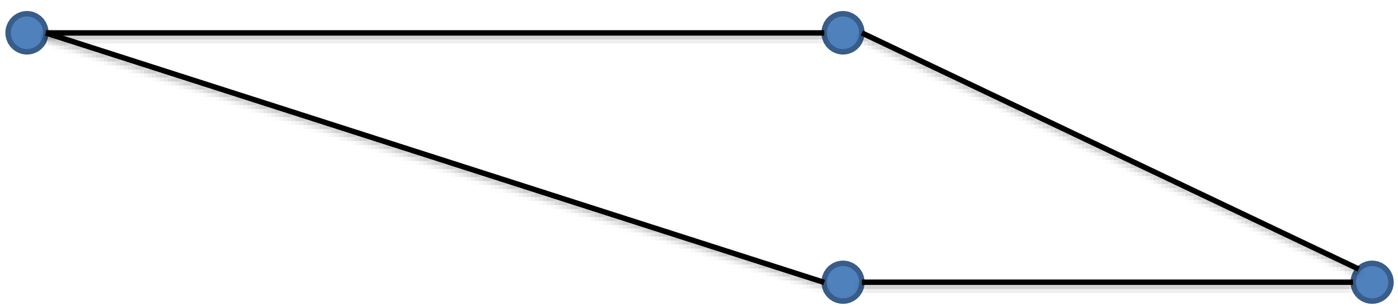
Yes, it is; but need a proof. (need not?)

Consider the Traveling Salesman Problem.

- We have n towns in the plane.
- The cost of connecting any two of them is proportional to their distance.

Aim: to find a Hamilton cycle with cost as small as possible.





Theorem 1 A B-K tree is the best.

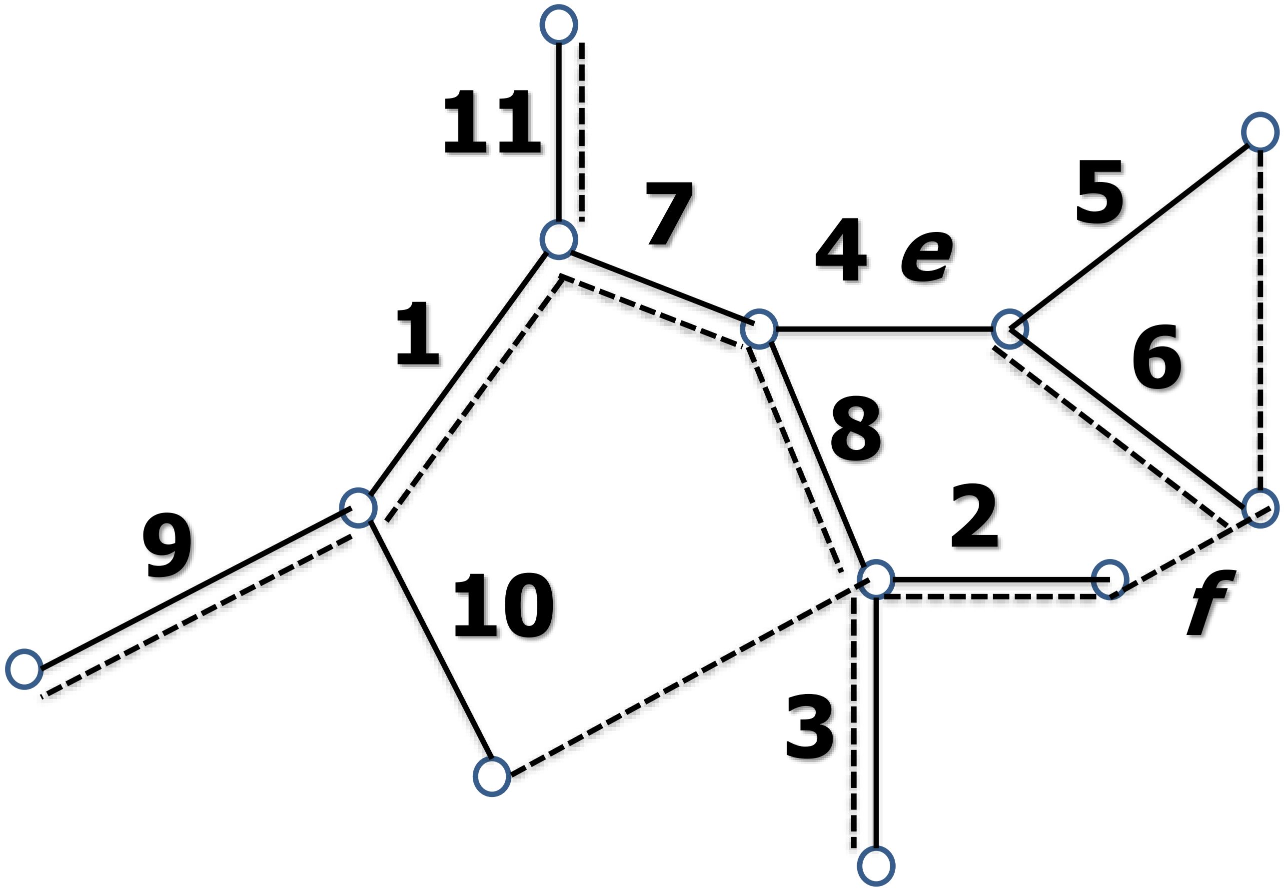
Proof idea.

If F is a Borůvka-Kruskal tree and T is another spanning tree, then

$$c(F) \le \cdots \le c(H) \le c(T),$$

where H is a spanning tree obtained from T and comes close to F.

Theorem 1 A B-K tree is the best.



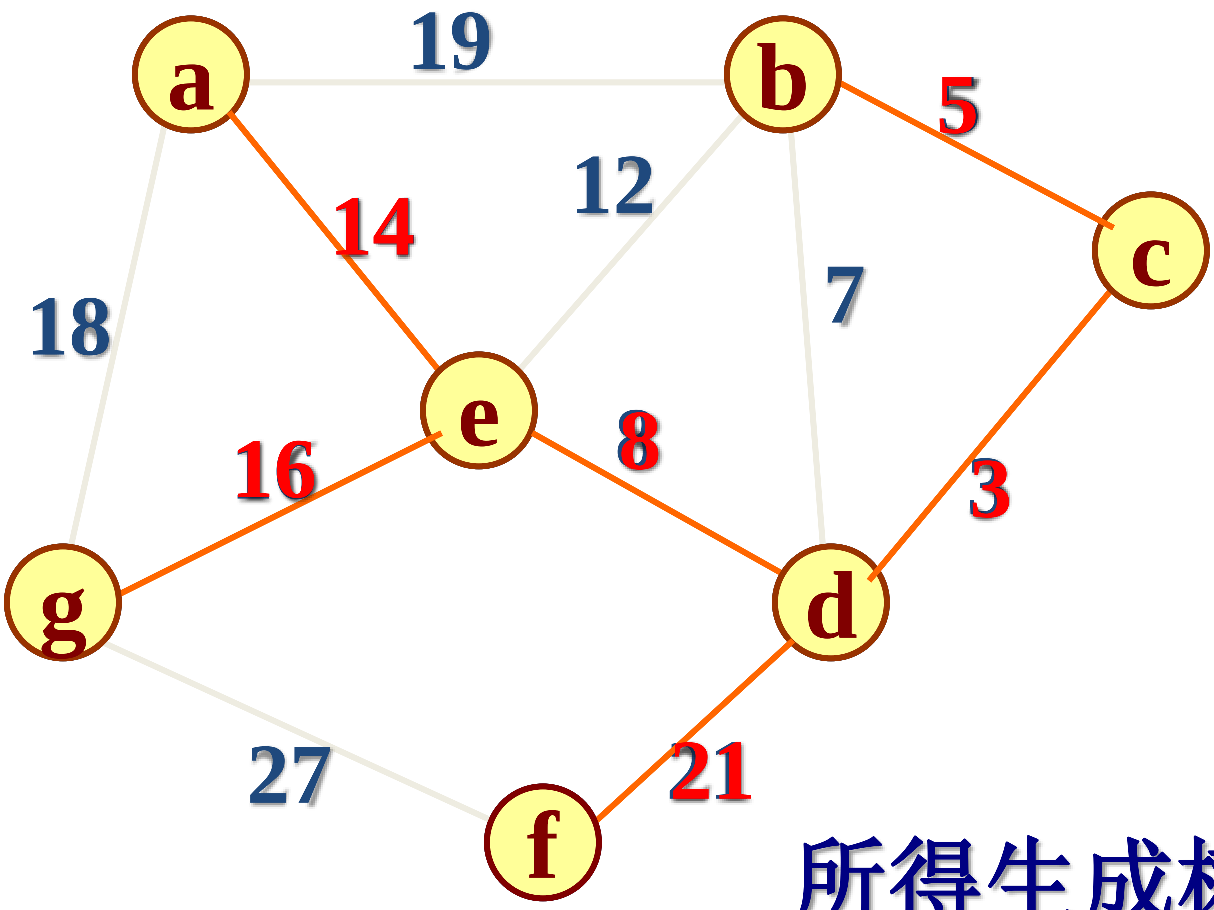
— *F* (a B-K tree), --- *T* (a tree)

If $H := T+e-f$, then $c(H) \le c(T)$.

Jarník-Prim算法的基本思想

- 取图中任意一个顶点*v*作为生成树的根，之后往树上添加新的顶点*u*让树生长。
- 在添加的顶点*u*和已经在树上的顶点*v*之间存在一条边，并且该边的权值在所有连接顶点*u*和*v*之间的边中取值最小。
- 之后继续往树上添加顶点，直至生成树上含有*n*个顶点为止。

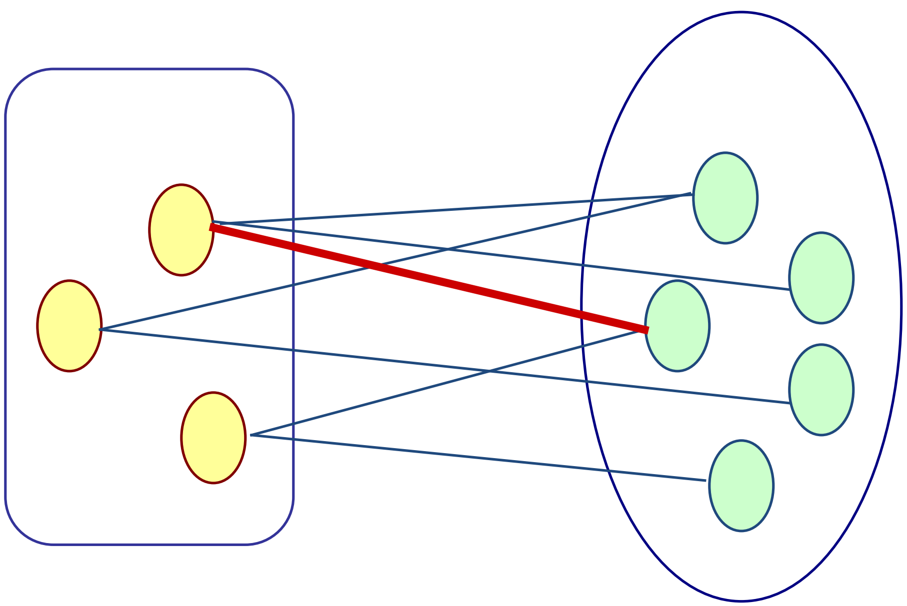
例如:



所得生成树权值和

=14+8+3+5+16+21=67.

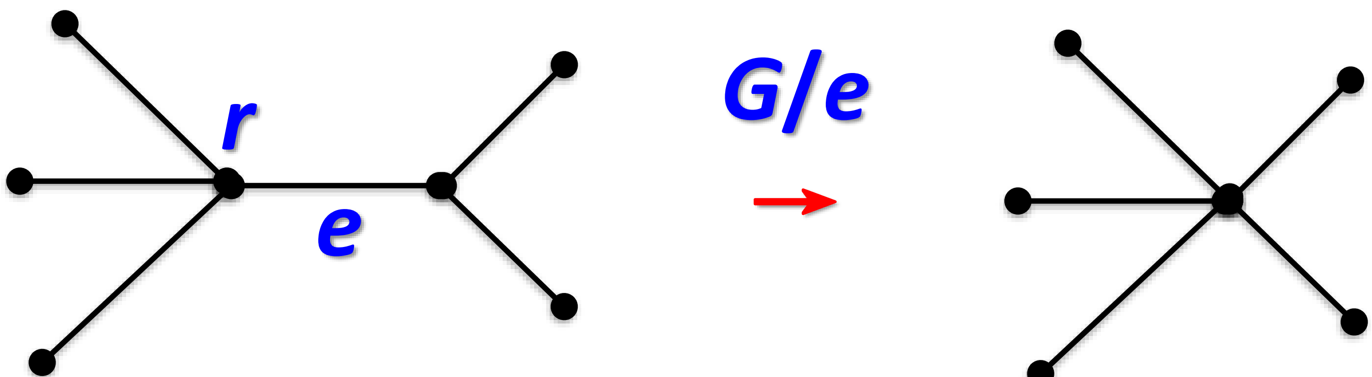
在生成树的构造过程中，图中n个顶点分属两个集合：已落在生成树上的顶点集U和尚未落在生成树上的顶点集$V \backslash U$，则应在所有连接U中顶点和$V \backslash U$中顶点的边中选取权值最小的边。



Theorem 2 A J-P tree is the best.

Proof idea. Use induction on $|V|$.

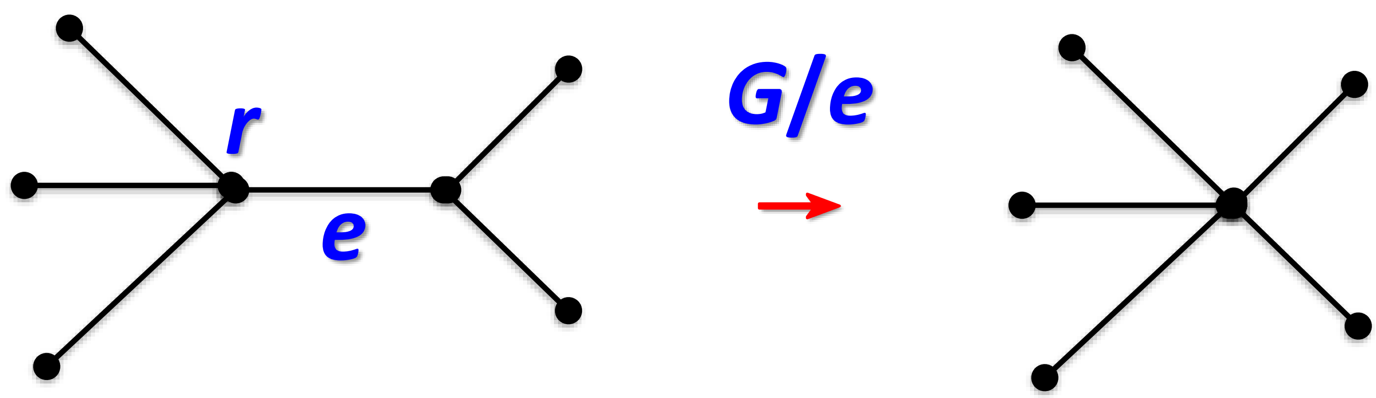
- Let T be a Jarník-Prim tree with root r.
- Let e be the **first** edge of T s.t. $c(e) \leq c(f)$ for all edges f incident with r.



Theorem 2 A J-P tree is the best.

Proof idea. Use induction on $|V|$.

- Is T/e a J-P tree of G/e?
- **Yes**, it is.
- Is $T=T/e+e$ the best?
- **Unnecessary**, then when is it best?



Theorem 2 A J-P tree is the best.

Proof idea. Use induction on $|V|$.

- Is T/e a J-P tree of G/e?
- **Yes**, it is.
- Is $T=T/e+e$ the best?
- **Unnecessary**, then when is it best?
- If an optimal tree $T^*\ni e$ in G, then T^*/e is also optimal in G/e (Why?), and

$$c(T)=c(e)+c(T/e)=c(e)+c(T^*/e)=c(T^*).$$

Theorem 2 A J-P tree is the best.

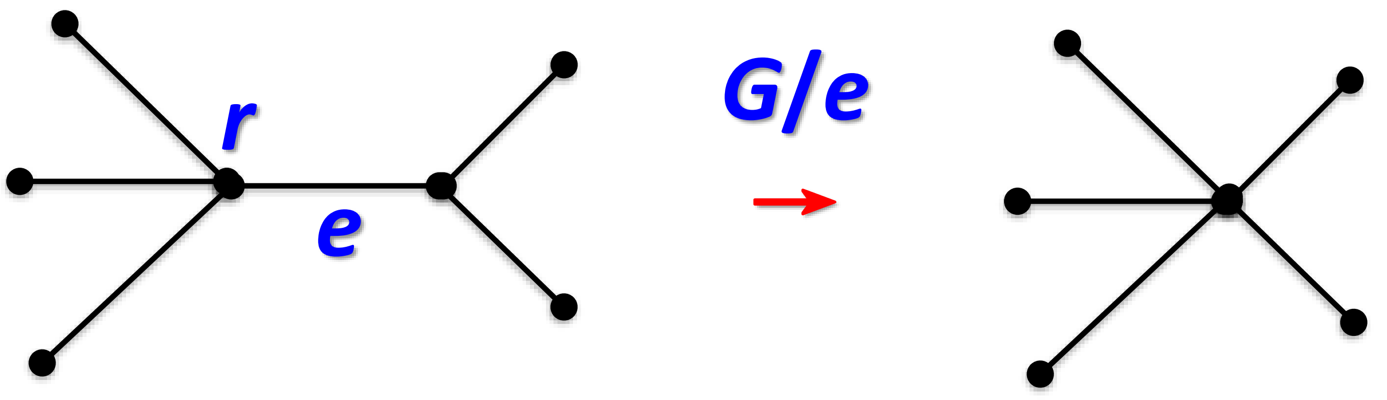
Proof. Use induction on $|V|$.

- Let T be a Jarník-Prim tree with root r.
- Let e be the **first** edge of T s.t. $c(e) \leq c(f)$ for all edges f incident with r.

Claim. **Some optimal tree includes** e.

- If T^* is an optimal tree but $e \notin T^*$, then T^*+e contains a cycle C.
- Let f be the other edge of C incident with r. Clearly, $c(e) \leq c(f)$.

- Then $T' := T^* + e - f$ is a spanning tree and
- $c(T') = c(T^*) + c(e) - c(f) \le c(T^*)$.
- So some optimal tree includes e.
- Then by induction, T/e is an optimal Jarník-Prim tree of G/e, and so is T of G.



9.2 The Traveling Salesman Problem

- We have *n* towns in the plane.
- The cost of connecting any two is given.

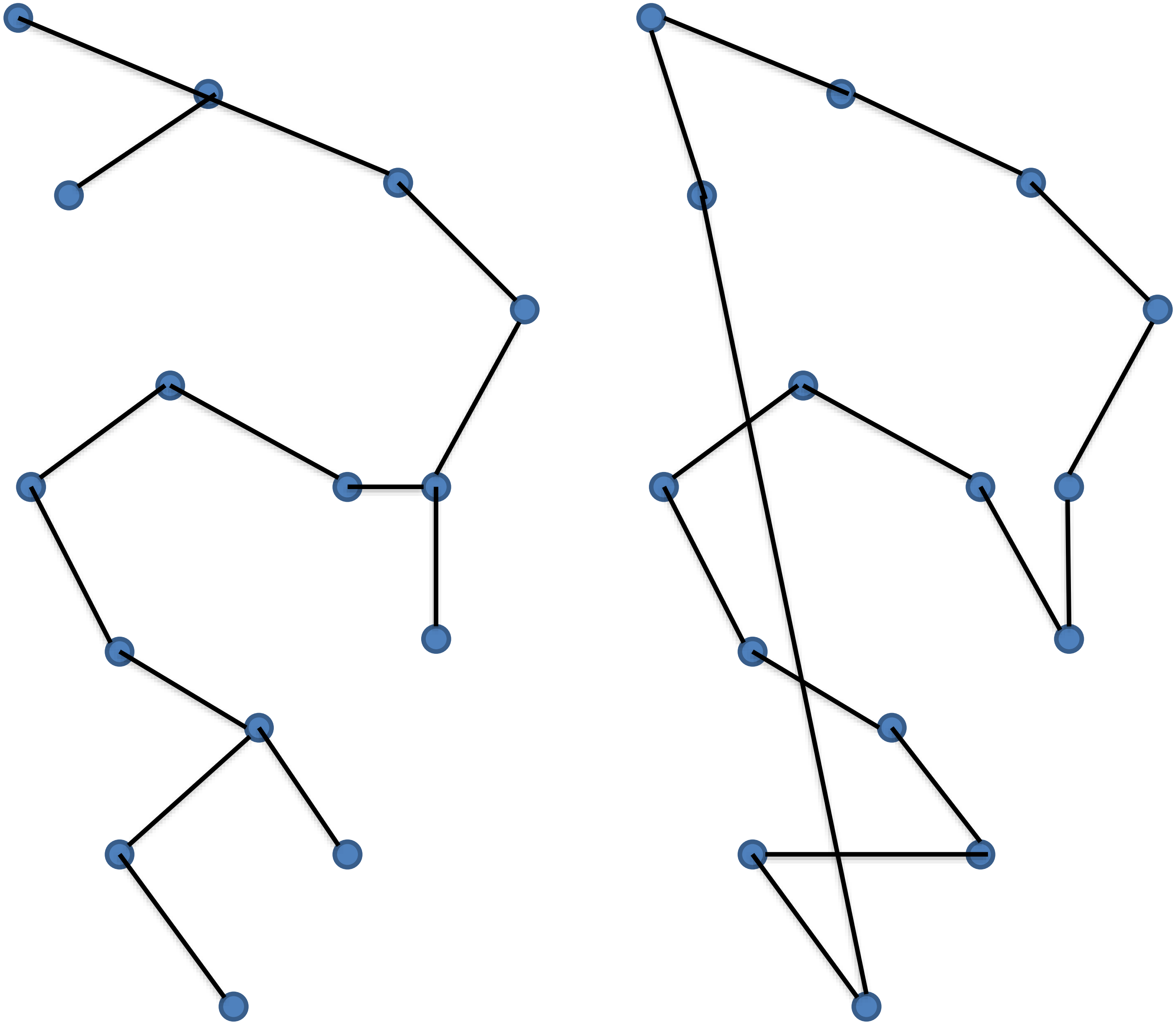
Aim: to find a Hamilton cycle with cost as small as possible.

The problem of whether a given graph is Hamiltonian can be reduced to the TSP.

- Define the cost of linking two nodes:
  1 for adjacent nodes, 2 otherwise.

Tree Shortcut Algorithm

1. Find a cheapest spanning tree
2. Get a closed walk around the tree
3. Make shortcuts: If the walk takes from i to j to k, and j has been seen, the walk can directly go from i to k.
4. Doing (3) as long as we can ends up with a Hamilton cycle.



Theorem 9.2.1 If the costs in a Traveling Salesman Problem satisfy the triangle inequality $c(ij)+c(jk) \geq c(ik)$, then the Tree Shortcut Algorithm finds a tour that costs less than **twice** as much as the optimum tour.

Proof. The optimum cost: c^*,

- The cost of the cheapest tree: $c(T)$,
- The cost of the walk: $c(walk)$,
- The cost of our tour: $c(tour)$,

$$c(tour) \le c(walk) = 2c(T) \le 2c^*.$$