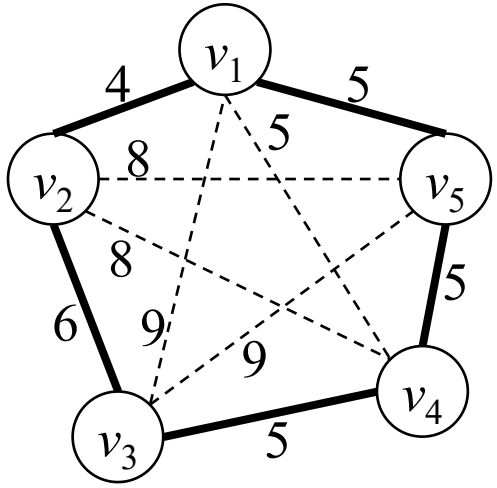
第6章 赋权图

程龚
南京大学 计算机学院
gcheng@nju.edu.cn
http://ws.nju.edu.cn/~gcheng

本章内容

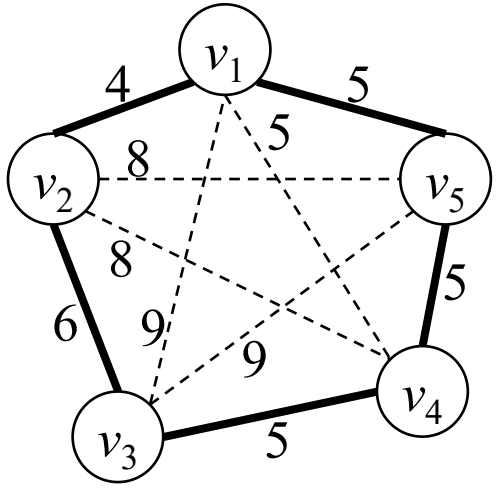
- 第6.1节 赋权图和距离
- 第6.2节 最小生成树
- 第6.3节 赋权欧拉图
- 第6.4节 赋权哈密尔顿图
  - 第6.4.1节 理论
  - **第6.4.2节 算法**

如何找出最短哈密尔顿圈?



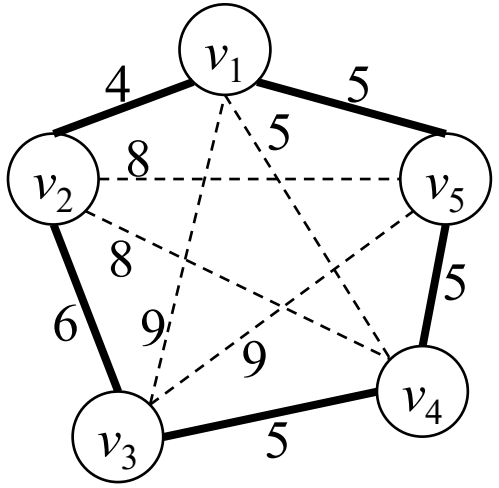
如何找出最短哈密尔顿圈？

- 哈密尔顿圈的存在性的判定问题的复杂度属于NPC，而该问题可归约为Δ-TSP
- 因此，Δ-TSP是一个**NP难**的优化问题，不存在多项式时间算法（除非P=NP）



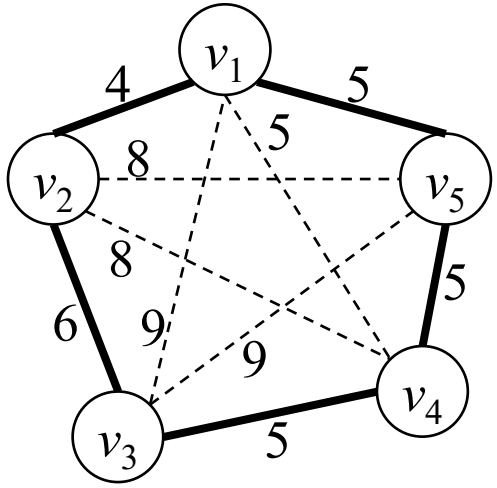
如何找出最短哈密尔顿圈？

- 哈密尔顿圈的存在性的判定问题的复杂度属于NPC，而该问题可归约为Δ-TSP
- 因此，Δ-TSP是一个NP难的优化问题，不存在多项式时间算法（除非P=NP）
- 通过算法找出一个较短（但未必最短）的哈密尔顿圈，其与最短哈密尔顿圈的长度的差异具有可以证明的界，这样的多项式时间算法称作**近似算法**



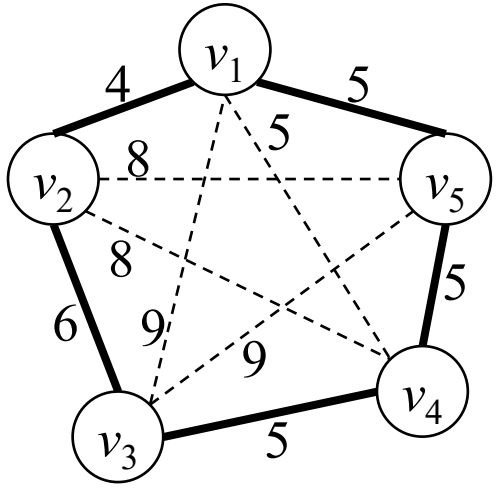
如何找出最短哈密尔顿圈?

1. 最近邻点算法
2. 克里斯托菲德斯-谢尔久科夫算法



如何找出最短哈密尔顿圈?

1. **最近邻点算法**
2. 克里斯托菲德斯-谢尔久科夫算法



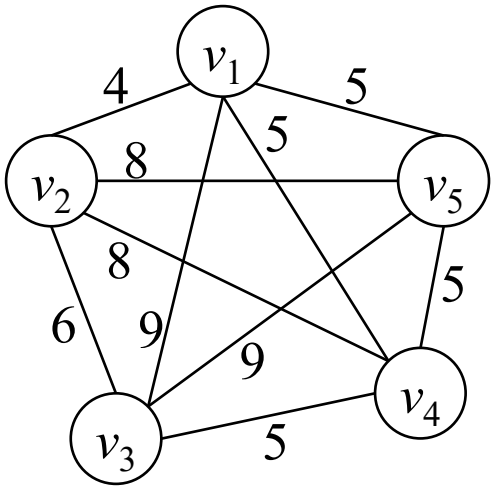
最近邻点算法

- 基本思路：逐步构造哈密尔顿圈，每步将当前路延长一条边，并尽可能选择权小的边。

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```



最近邻点算法

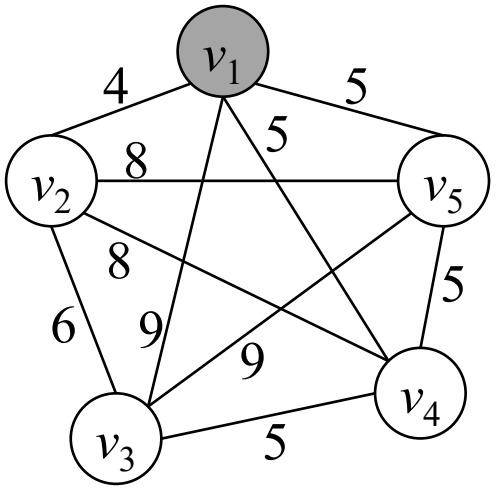
- 从顶点集V中任意一个顶点s出发：
  - 顶点u表示当前到达的顶点
  - 集合U表示经过的顶点的集合

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```

灰色顶点：当前到达的顶点



最近邻点算法

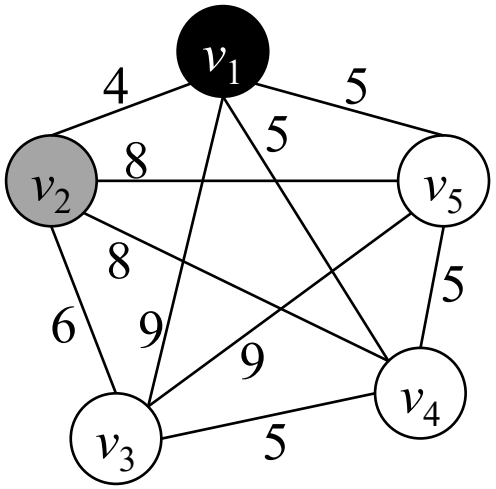
- 每轮while循环到达u的一个邻点v，作为下轮while循环的u，直至经过所有顶点：

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```

灰色顶点：当前到达的顶点
黑色顶点：经过的顶点



最近邻点算法

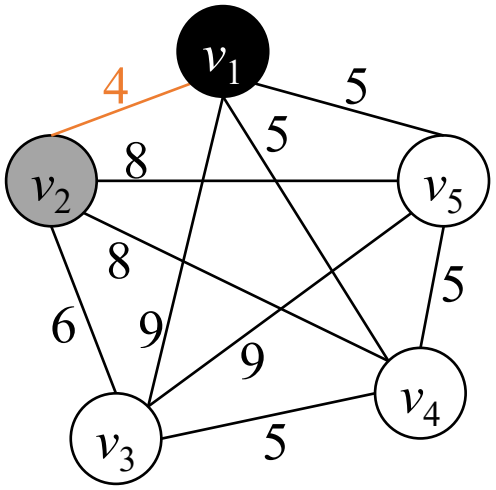
- 每轮while循环到达u的一个邻点v，作为下轮while循环的u，直至经过所有顶点：
  - v是未经过的顶点中，和u间的边权最小的，这是一种贪心策略

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $s \leftarrow V$ 中任意一个顶点;
2 $u \leftarrow s$;
3 $U \leftarrow \{u\}$;
4 输出 (u);
5 **while** $U \neq V$ **do**
6 $\quad u \leftarrow \arg\min_{v \in V \setminus U} w(u, v)$;
7 $\quad U \leftarrow U \cup \{u\}$;
8 $\quad$ 输出 (u);
9 输出 (s);

灰色顶点：当前到达的顶点
黑色顶点：经过的顶点



最近邻点算法

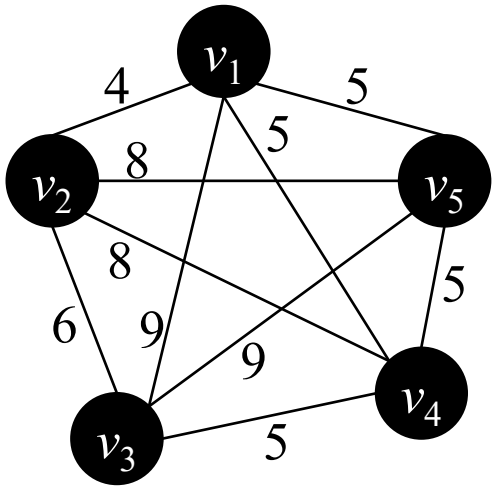
- 算法运行过程中，按序输出经过的顶点，最后再次输出出发点，形成一个较短哈密尔顿圈。

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```

灰色顶点：当前到达的顶点
黑色顶点：经过的顶点



最近邻点算法

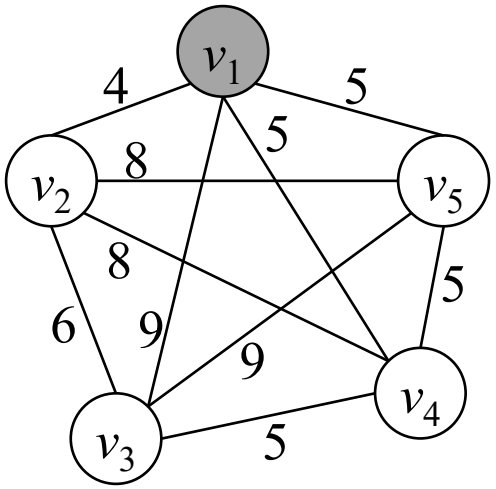
- 例如：从v_1出发，向U中增加v_1，输出v_1

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```

灰色顶点：当前到达的顶点
黑色顶点：经过的顶点



最近邻点算法

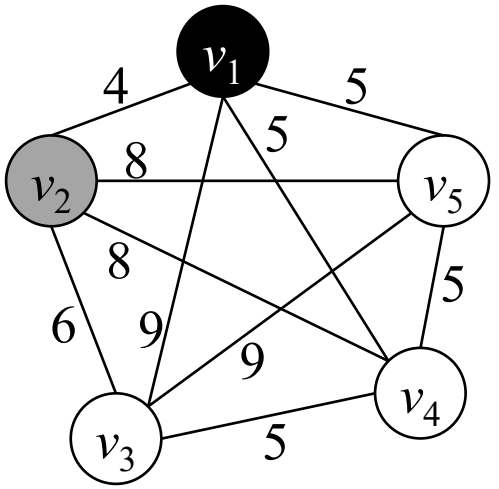
- 选择v_2
- 向U中增加v_2
- 输出v_2

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```

灰色顶点：当前到达的顶点
黑色顶点：经过的顶点



最近邻点算法

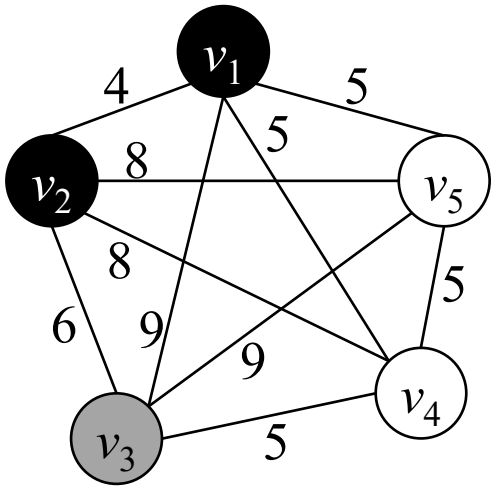
- 选择v_3
- 向U中增加v_3
- 输出v_3

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```

灰色顶点：当前到达的顶点
黑色顶点：经过的顶点



最近邻点算法

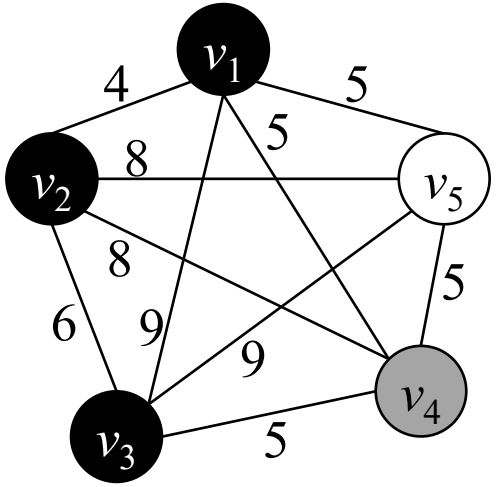
- 选择v_4
- 向U中增加v_4
- 输出v_4

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```

灰色顶点：当前到达的顶点
黑色顶点：经过的顶点



最近邻点算法

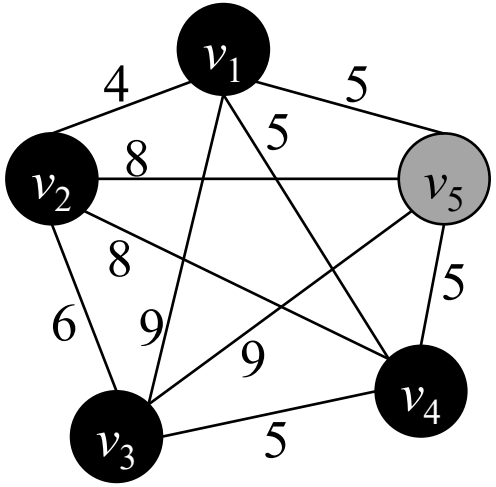
- 选择v_5
- 向U中增加v_5
- 输出v_5

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```

灰色顶点：当前到达的顶点
黑色顶点：经过的顶点



最近邻点算法

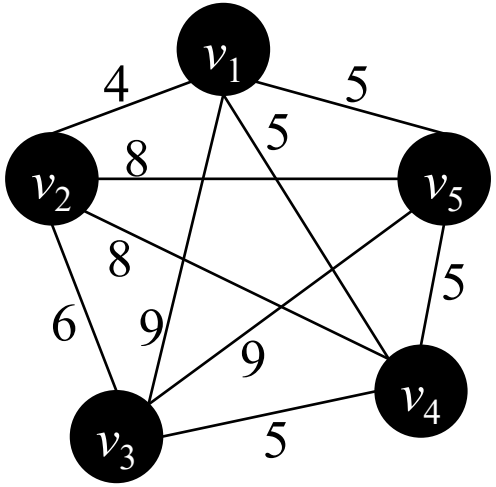
- $U = V$，输出出发点v_1

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

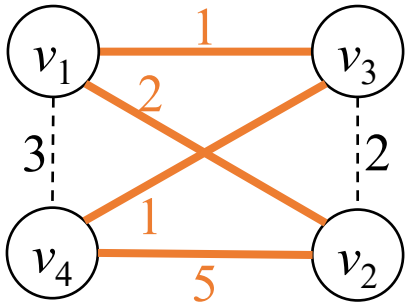
```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```

灰色顶点：当前到达的顶点
黑色顶点：经过的顶点

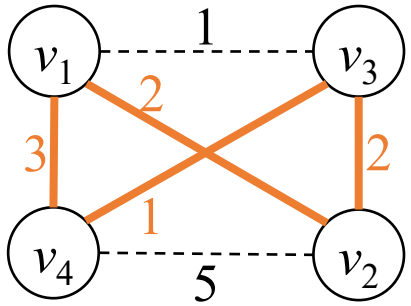


近似比

- 算法近似的准确度，称作**近似比**
  - 对于最小化目标函数的优化问题，近似比是算法输出的解与最优解的目标函数的比值
  - 对于最大化目标函数的优化问题，近似比是最优解与算法输出的解的目标函数的比值
- 因此，近似比越小表示近似越准确，近似比的最小值为1
- 例如：
  - 较短哈密尔顿圈的长度为9
  - 最短哈密尔顿圈的长度为8
  - 近似比为9/8



算法输出的解
（较短哈密尔顿圈）



最优解
（最短哈密尔顿圈）

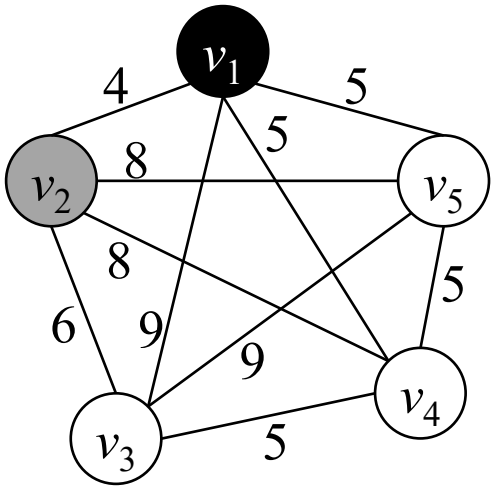
定理6.5

- 对于阶为n的赋权图，若赋权函数满足三角不等式，则较短哈密尔顿圈的长度不超过最短哈密尔顿圈的长度的$\frac{1}{2}\lceil \log n \rceil + \frac{1}{2}$倍。

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

1 $s \leftarrow V$ 中任意一个顶点;
2 $u \leftarrow s$;
3 $U \leftarrow \{u\}$;
4 输出 (u);
5 **while** $U \neq V$ **do**
6 $\quad u \leftarrow \arg\min_{v \in V \setminus U} w(u, v)$;
7 $\quad U \leftarrow U \cup \{u\}$;
8 $\quad$ 输出 (u);
9 输出 (s);



最近邻点算法

- 时间复杂度：$O(n^2)$
  - 循环的轮数：$O(n)$
  - 每轮循环找顶点u：$O(n)$

算法 6.5: 最近邻点算法

输入: 边权为非负实数的完全赋权图 $G = \langle V, E, w \rangle$

```
1 s ← V 中任意一个顶点;
2 u ← s;
3 U ← {u};
4 输出 (u);
5 while U ≠ V do
6 |   u ← arg min_{v∈V\U} w(u, v);
7 |   U ← U ∪ {u};
8 |   输出 (u);
9 输出 (s);
```

接下来进入其它算法部分