本次课程提纲：图的基本概念

- 图的代数表示
- 最短路算法及其应用

图的代数表示

- 表示一个图主要有定义描述、图形表示和代数表示
  - 代数表示：邻接矩阵、关联矩阵
- 邻接矩阵
  - 设 G 为 n 阶图，$V=\{v_1,\cdots,v_n\}$，邻接矩阵 $A(G)=(a_{ij})$
  - $a_{ij}=v_i,v_j$ 间的边数
- 邻接矩阵的性质
  - 非负性与对称性
  - 同一图的不同形式的邻接矩阵是相似矩阵
  - 如果 G 为简单图，
    - $A(G)$ 为布尔矩阵
    - 行和 (列和) 等于对应顶点的度数
    - 矩阵元素总和为图的总度数

邻接矩阵与图连通性

连通的充分必要条件

G 连通的充分必要条件是 $A(G)$ 不与矩阵 $\begin{pmatrix} A_{11}, 0 \\ 0, A_{22} \end{pmatrix}$ 相似，非连通图的邻接矩阵一定能够写成准对角矩阵形式

证明

- 必要性
  - 若不然：设 A_{11} 对应的顶点是 $v_1, \cdots, v_k$, A_{22} 对应的顶点为 $v_{k+1}, \cdots, v_n$
  - 显然，v_i $(1 \leq i \leq k)$ 与 v_j $(k+1 \leq i \leq n)$ 不邻接，即 G 是非连通图。矛盾！
- 充分性
  - 若不然：设 G_1 与 G_2 是 G 的两个不连通的部分
  - 如果在写 G 的邻接矩阵时，先排 $V(G_1)$ 中点，再排 $V(G_2)$ 中点，则 G 的邻接矩阵形式必为 $\begin{pmatrix} A_{11}, 0 \\ 0, A_{22} \end{pmatrix}$

邻接矩阵与路径

定理

记 A^k 的元素为 $\{a_{ij}^k\}$，a_{ij}^k 为 v_i 到 v_j 长度为 k 的途径条数

证明：对 k 作数学归纳法证明

- 当 $k = 1$ 时，显然，假设 $k - 1$ 时结论成立，当为 k 时
  - $A^k = A^{k-1} \cdot A = (a_{i1}^{k-1} a_{1j} + \cdots + a_{in}^{k-1} a_{nj})_{n \times n}$
- 考察 v_i 到 v_j 长度为 k 的途径
  - 设 v_m 是 v_i 到 v_j 的途径上和 v_j 邻接的点，
  - 则 v_i 到 v_j 的经过 v_m 且长度为 k 的途径数目为 $a_{im}^{k-1} a_{mj}$
  - 故 v_i 到 v_j 长度为 k 的途径数目为 $a_{i1}^{k-1} a_{1j} + \cdots + a_{in}^{k-1} a_{nj}$

推论

对简单图，A^2 的元素 a_{ii}^2 是 v_i 的度数；A^3 的元素 a_{ii}^3 是含 v_i 的三角形个数的 2 倍

Dijkstra 最短路算法

- 问题：给定图 $G=(V,A)$，计算从 s 到其他每个节点的最短路

$d(i) \leftarrow \infty$ for all $i \in V$
$p(i) \leftarrow$ **null** for all $i \in V$
Unmark all $i \in V$
$d(s) \leftarrow 0$
while not all vertices are marked **do**
    Find unmarked $i \in V$ that minimizes $d(i)$ and mark i
    for j such that $(i,j) \in A$ **do**
        if $d(j) > d(i) + c(i,j)$ **then**
            $d(j) \leftarrow d(i) + c(i,j)$
            $p(j) \leftarrow i$

Dijkstra's algorithm for the shortest path problem.

- 算法复杂度
    - $O(n^2)$：$n(n-1)/2$ 次比较

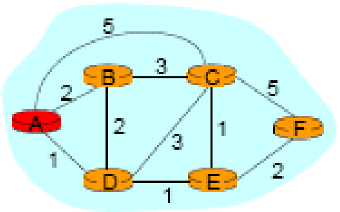
Dijkstra 算法完整伪码

```
Initialization:
  S = {A}
  for all nodes v
    if v adjacent to A
        then D(v) = c(A,v)
    else D(v) = ∞

Loop
  find w not in S such that D(w) is a minimum
  add w to S
  update D(v) for all v adjacent to w and not in S:
      D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in S
```

Dijkstra 算法例子

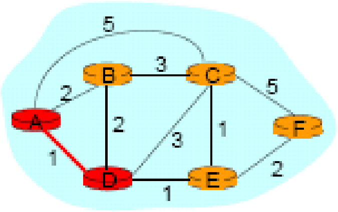
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→ 0	A	2,A	5,A	1,A	∞	∞
1						
2						
3						
4						
5						




```
Initialization:
  S = {A};
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v);
      else D(v) = ∞;
...
```

Dijkstra 算法例子

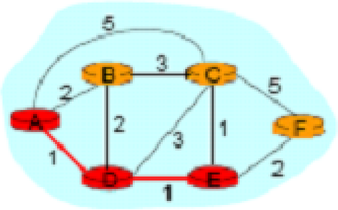
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2						
3						
4						
5						




```
...
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
  update D(v) for all v adjacent
  to w and not in S:
    D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Dijkstra 算法例子

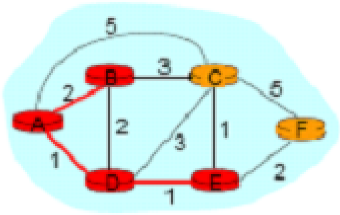
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3						
4						
5						




```
...
Loop
   find w not in S s.t. D(w) is a minimum;
   add w to S;
  update D(v) for all v adjacent
   to w and not in S:
      D(v) = min( D(v), D(w) + c(w,v) );
  until all nodes in S;
```

Dijkstra 算法例子

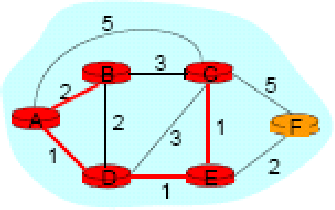
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4						
5						




```
...
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
  update D(v) for all v adjacent
   to w and not in S:
     D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

Dijkstra 算法例子

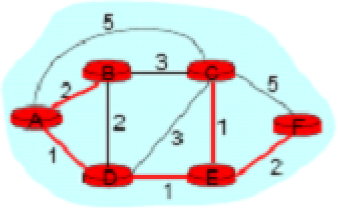
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5						




```
...
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
  update D(v) for all v adjacent
  to w and not in S:
    D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

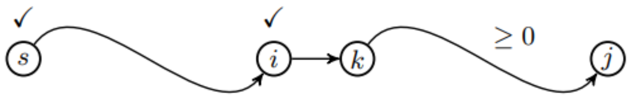
Dijkstra 算法例子

Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5	ADEBCF					




```
...
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
  update D(v) for all v adjacent
   to w and not in S:
     D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

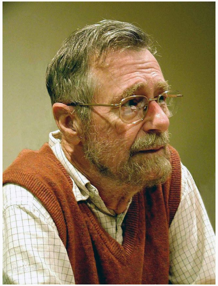
Dijkstra 算法正确性证明：数学归纳 + 反证法



- 假设图中 j 是本次被标号的点，s 到 j 最短路记为 P：如果 $d(j) > L(P)$
  - s 到 i 是 P 上已有标号的节点，而 k 未标号
  - k, j 重合的情况：根据算法，$d(j) \leq L(P)$，矛盾！
  - $k \neq j$：注意到 $L(P) < d(j)$，根据算法，k 应在 j 前被标号，矛盾！

Dijkstra

- Edsger Wybe Dijkstra，1930 年 5 月 11 日-2002 年 8 月 6 日
- 当代最伟大的计算机科学家之一（第一位非美英图灵奖得主，1972）



Dijkstra 主要成就

- 最短路径算法
- 提出 goto 有害论
- 提出信号量和 PV 原语
- 解决了哲学家聚餐问题
- 银行家算法的创造者
- 第一个 Algol 60 编译器的设计者和实现者
- THE 操作系统的设计者和开发者

Bellman-Ford 最短路算法

- 问题：给定图 $G = (V, A)$，计算从 s 到其他每个节点的最短路

$d(i) \leftarrow \infty$ for all $i \in V$
$p(i) \leftarrow$ **null** for all $i \in V$
$d(s) \leftarrow 0$
for $k \leftarrow 1$ to $n-1$ **do**
    for all $(i, j) \in A$ **do**
        if $d(j) > d(i) + c(i, j)$ **then**
            $d(j) \leftarrow d(i) + c(i, j)$
            $p(j) \leftarrow i$

The Bellman-Ford algorithm for the shortest path problem.

- 算法复杂度：$O(mn)$

Bellman-Ford 算法正确性证明

- 任何 $d(i)$ 都对应从 s 到 i 的一条路径
  - 由数学归纳法易证
- k 次循环后，$d(i) \le$ 不超过 k 跳的 $s-i$ 路径的最短距离
  - 由数学归纳法 + 反证法易证
- 所以 Bellman-Ford 算法返回最短路径

Bellman-Ford 算法改进

如果 $d(i)$ 上一次循环没有改动，则不需要执行判断 $d(j) > d(i) + c(i, j)$

for j such that $(i, j) \in A$ **do**
  if $d(j) > d(i) + c(i, j)$ **then**
    $d(j) \leftarrow d(i) + c(i, j)$
    $p(j) \leftarrow i$
    if not $q.contains?(j)$ **then**
      $q.add(j)$

Procedure QScan(i, q)

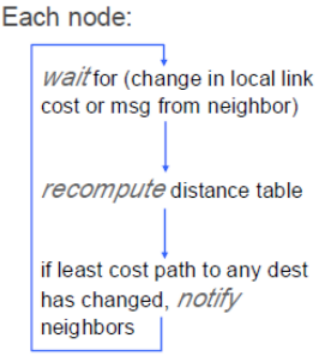
$d(i) \leftarrow \infty$ for all $i \in V$
$p(i) \leftarrow$ **null** for all $i \in V$
$d(s) \leftarrow 0$
$q \leftarrow new\ queue()$
$q.add(s)$
while not $q.empty?$ **do**
  `QScan`(q.remove(), q);

The Bellman-Ford algorithm using queues.

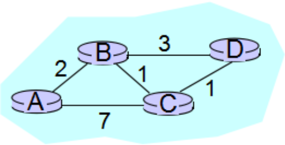
基于 Bellman-Ford 算法的路由协议

```
Initialization:
  for all neighbors V  do
    if V adjacent to A
      D(A, V) = c(A, V);
  else
      D(A, V) = ∞;

loop:
  wait (link cost update or update message)
  if (c(A,V) changes by d)
     for all destinations Y through V do
        D(A,Y) =  D(A,Y) + d
  else if (update D(V, Y) received from V)
     for all destinations Y do
        if (destination Y through V)
          D(A,Y) = D(A,V) + D(V, Y);
        else
          D(A, Y) = min(D(A, Y), D(A, V) + D(V, Y));
  if (there is a new minimum for destination Y)
    send D(A, Y) to all neighbors
forever
```



Bellman-Ford 路由协议例子



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	∞	-

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

```
Initialization:
  for all neighbors V  do
    if V adjacent to A
      D(A, V) = c(A,V);
  else
      D(A, V) = ∞;
…
```

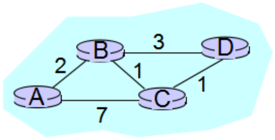
Node C

Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Bellman-Ford 路由协议例子



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	∞	-

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

(D(C,A), D(C,B), D(C,D))

Node C

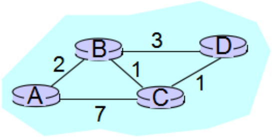
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
...
loop:
...
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A, Y) = D(A, V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Bellman-Ford 路由协议例子



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	8	C

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

D(A, D) = min(D(A, D), D(A, C) + D(C,D))
= min(∞ , 7 + 1) = 8

(D(C,A), D(C,B), D(C,D))

Node C

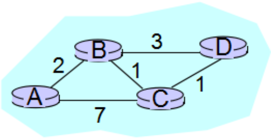
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
...
loop:
...
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A, Y) = D(A, V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Bellman-Ford 路由协议例子



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	8	C

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

```
...
loop:
...
  else if (update D(V, Y) received from V)
    for all destinations Y do
      if (destination Y through V)
        D(A, Y) = D(A, V) + D(V, Y);
      else
        D(A, Y) = min(D(A, Y),
                      D(A, V) + D(V, Y));
  if (there is a new minimum for dest. Y)
    send D(A, Y) to all neighbors
forever
```

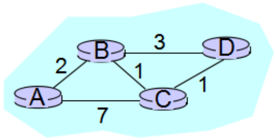
Node C

Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Bellman-Ford 路由协议例子



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	5	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

D(A,D) = min(D(A,D), D(A,B) + D(B,D))
= min(8, 2 + 3) = 5

D(A,C) = min(D(A,C), D(A,B) + D(B,C))
= min(7, 2 + 1) = 3

```
…
loop:
…
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A, Y) = D(A, V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

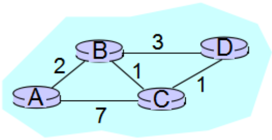
Node C

Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Bellman-Ford 路由协议例子




```
...
loop:
...
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A,Y) = D(A,V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	5	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

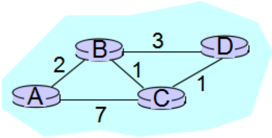
Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	5	B
B	3	B
C	1	C

Bellman-Ford 路由协议例子




```
...
loop:
...
  else if (update D(V, Y) received from V)
    for all destinations Y do
      if (destination Y through V)
        D(A, Y) = D(A, V) + D(V, Y);
      else
        D(A, Y) = min(D(A, Y),
                      D(A, V) + D(V, Y));
  if (there is a new minimum for dest. Y)
    send D(A, Y) to all neighbors
forever
```

Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	4	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

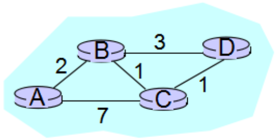
Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	4	B
B	3	B
C	1	C

Bellman-Ford 路由协议例子



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	4	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

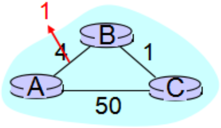
Node D

Dest.	Cost	NextHop
A	4	B
B	3	B
C	1	C

```
...
loop:
...
else if (update D(V, Y) received from V)
  for all destinations Y do
    if (destination Y through V)
      D(A, Y) = D(A, V) + D(V, Y);
    else
      D(A, Y) = min(D(A, Y),
                    D(A, V) + D(V, Y));
if (there is a new minimum for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Bellman-Ford 路由协议例子

```
loop:
  wait (link cost update or update message)
  if (c(A,V) changes by d)
     for all destinations Y through V do
        D(A,Y) =  D(A,Y) + d
   else if (update D(V, Y) received from V)
     for all destinations Y do
        if (destination Y through V)
          D(A,Y) = D(A,V) + D(V, Y);
        else
           D(A, Y) = min(D(A, Y), D(A, V) + D(V, Y));
   if (there is a new minimum for destination Y)
     send D(A, Y) to all neighbors
forever
```



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	2	B
B	1	B

D	C	N
A	2	B
B	1	B

time

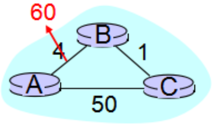
Link cost changes here

Algorithm terminates

"good news travels fast"

Bellman-Ford 路由协议例子

```
loop:
  wait (link cost update or update message)
  if (c(A,V) changes by d)
     for all destinations Y through V do
        D(A,Y) = D(A,Y) + d
   else if (update D(V, Y) received from V)
     for all destinations Y do
        if (destination Y through V)
          D(A,Y) = D(A,V) + D(V, Y);
        else
           D(A, Y) = min(D(A, Y), D(A, V) + D(V, Y));
   if (there is a new minimum for destination Y)
     send D(A, Y) to all neighbors
forever
```



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	8	C
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	7	B
B	1	B

D	C	N
A	2	B
B	1	B

...

"bad news travels slowly"

Link cost changes here; meanwhile, B also receives C's routing table. time

最短路算法的应用

- 某两人有一只 8 升的酒壶装满了酒，还有两只空壶，分别为 5 升和 3 升，最少的操作次数能均分酒?

最短路算法的应用

- 某两人有一只 8 升的酒壶装满了酒，还有两只空壶，分别为 5 升和 3 升，最少的操作次数能均分酒?
- 设 x_1, x_2, x_3 分别表示 $8, 5, 3$ 升酒壶中的酒量。(x_1, x_2, x_3) 的组合共 24 种
    - $x_1 + x_2 + x_3 \le 8$, $x_1 \le 8$, $x_2 \le 5$, $x_3 \le 3$
- 每种组合用一个点表示，两点连线，当且仅当可通过倒酒的方式相互变换
- 各边赋权为 1，问题转化为在该图中求 $(8, 0, 0)$ 到 $(4, 4, 0)$ 的最短路
    - $(8, 0, 0) \to (3, 5, 0) \to (3, 2, 3) \to (6, 2, 0) \to (6, 0, 2) \to (1, 5, 2) \to (1, 4, 3) \to (4, 4, 0)$

最短路算法的应用

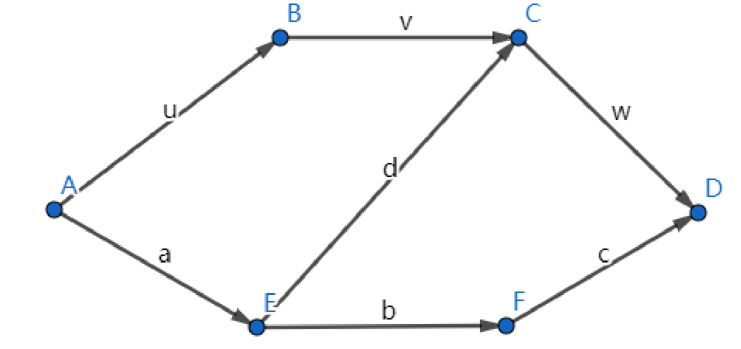
- 在一河岸有狼，羊和卷心菜。摆渡人要将它们渡过河去，由于船太小，每次只能载一样东西。由于狼羊，羊卷心菜不能单独相处。摆渡人至少要多少次才能将其渡过河?

最短路算法的应用

- 在一河岸有狼，羊和卷心菜。摆渡人要将它们渡过河去，由于船太小，每次只能载一样东西。由于狼羊，羊卷心菜不能单独相处。摆渡人至少要多少次才能将其渡过河?
- 人，狼，羊，菜所有组合形式有 16 中
  - 但是以下 6 种组合不能允许出现：狼羊菜，羊菜，狼羊，人，人狼，人菜
- 岸上只能允许出现 10 种组合
  - 人狼羊菜，人狼羊，人狼菜，人羊，空，菜，羊，狼，狼菜，人羊菜
- 每种情况用点表示。两点连线，当且仅当两种情况可用载人 (或加一物) 的渡船相互转变，每条边赋权为 1
- 问题转化为求由顶点“人狼羊菜”到顶点“空”的一条最短路
  - 人狼羊菜–> 狼菜–> 人狼菜–> 狼–> 人狼羊–> 羊–> 人羊–> 空
  - 人狼羊菜–> 狼菜–> 人狼菜–> 菜–> 人羊菜–> 羊–> 人羊–> 空

多条最短路

如何求两条总长度最短的点不交路



课后练习与思考题

- 设计算法，求出图的最小环的长度