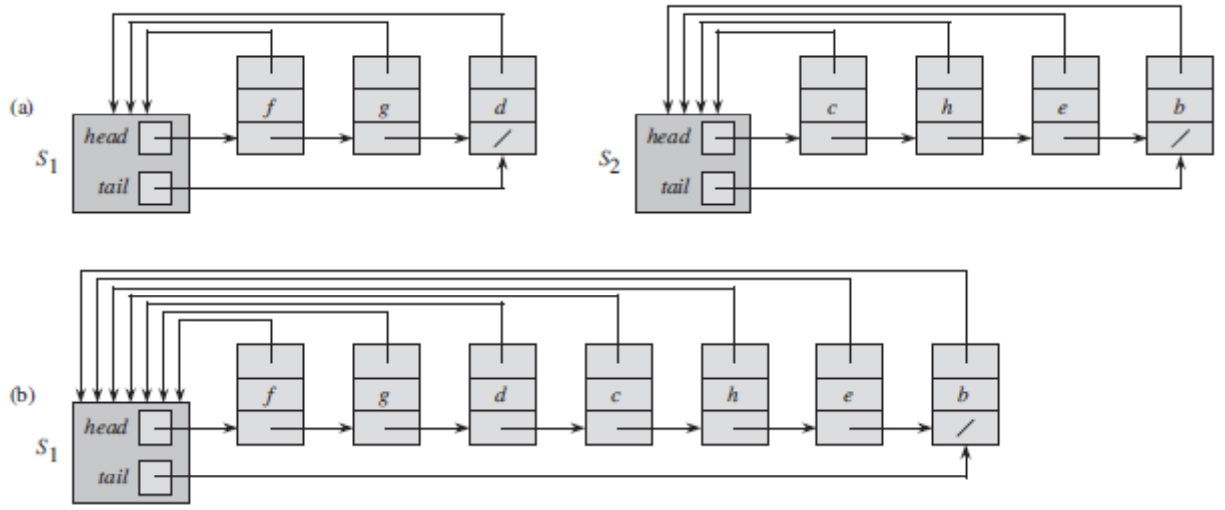
- 书面作业讲解
  - TC第21.1节练习2、3
  - TC第21.2节练习1、3、6
  - TC第21.3节练习1、2、3
  - TC第21章问题1
  - DW第1.1节练习4、5、10、14、16、31
  - DW第1.2节练习5、7、11、17、18、20、38、40
  - DW第1.3节练习1、10、14、15、18、61
  - DW第1.4节练习1、5、8、10、11、21

TC第21.1节练习2

- Two vertices are in the same connected component if and only if they are in the same set.
  - in the same connected component → in the same set
  - in the same set → in the same connected component

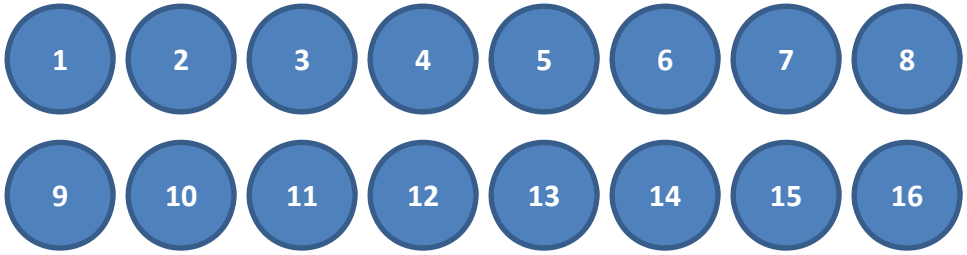
TC第21.2节练习1

- UNION之后，需要维护哪些指针？



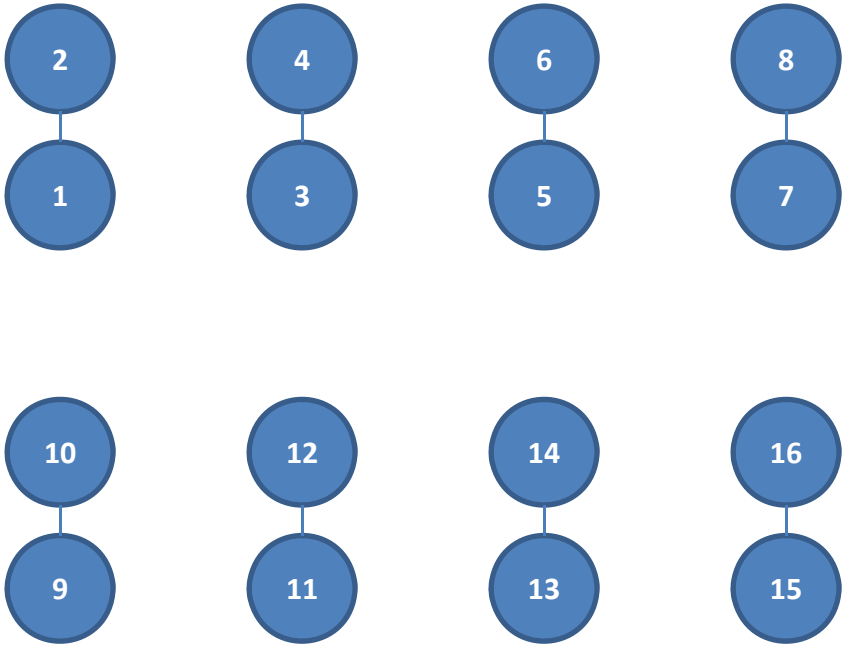
TC第21.3节练习1

- for i=1 to 16
  - MAKE-SET(x_i)



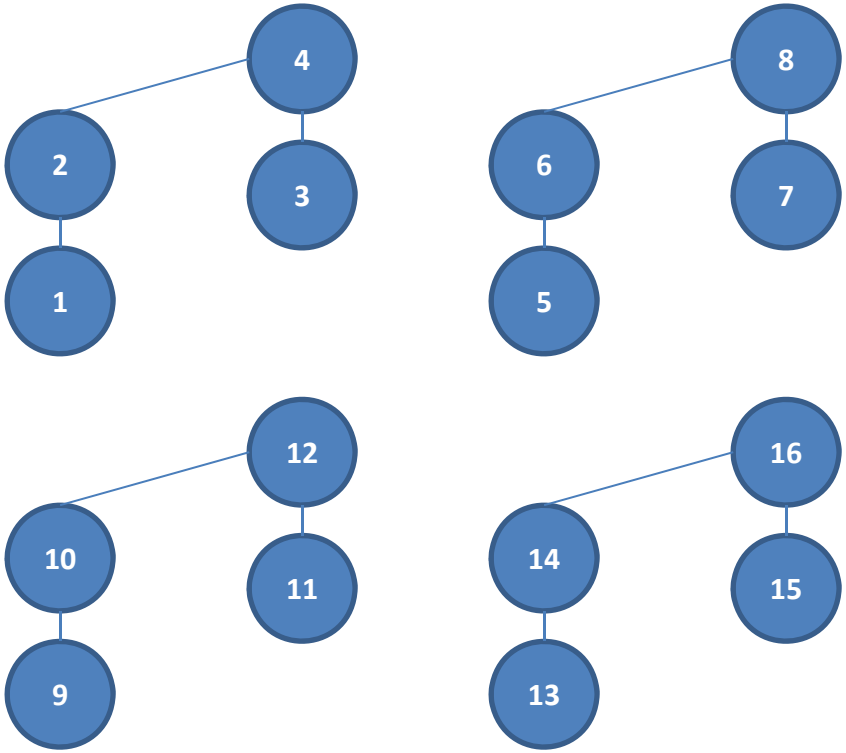
TC第21.3节练习1 (续)

- for i=1 to 16
  - MAKE-SET(x_i)
- for i=1 to 15 by 2
  - UNION(x_i, x_{i+1})



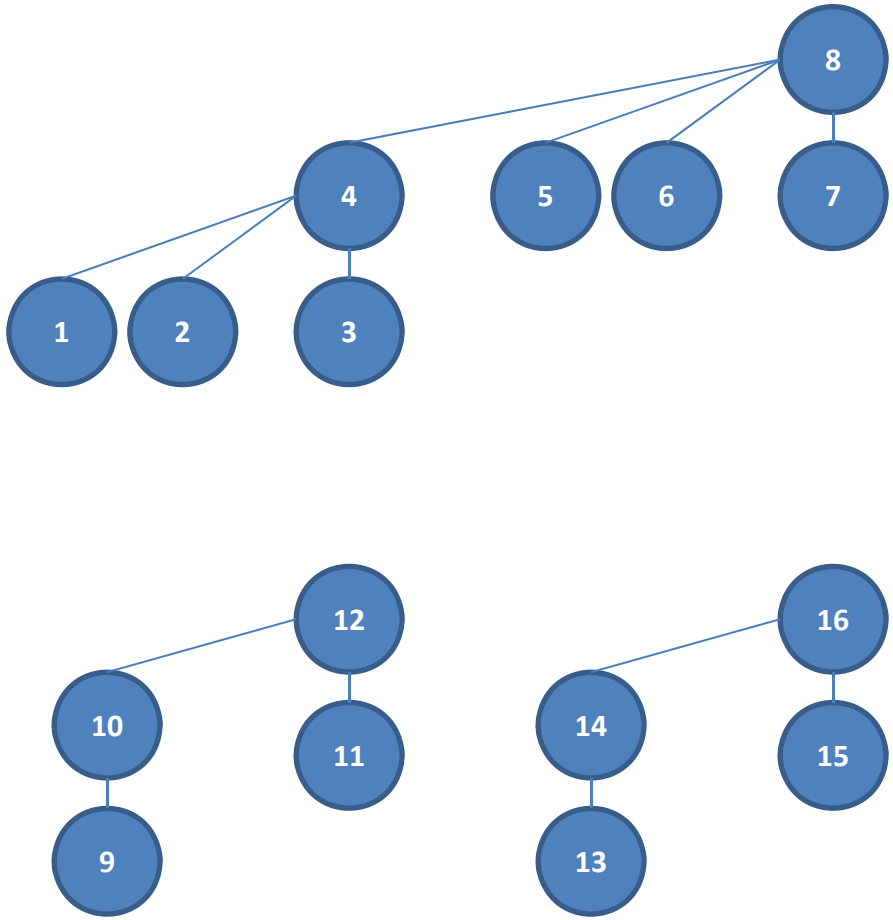
TC第21.3节练习1 (续)

- for i=1 to 16
  - MAKE-SET(x_i)
- for i=1 to 15 by 2
  - UNION(x_i, x_{i+1})
- for i=1 to 13 by 4
  - UNION(x_i, x_{i+2})



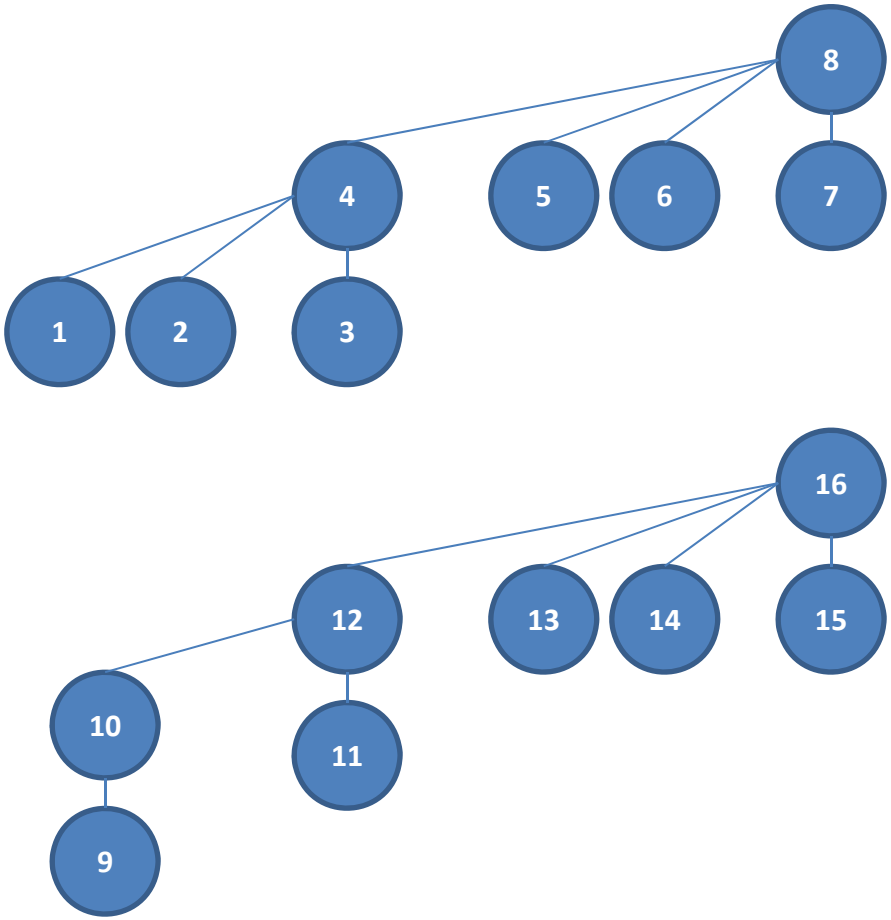
TC第21.3节练习1 (续)

- for i=1 to 16
  - MAKE-SET(x_i)
- for i=1 to 15 by 2
  - UNION(x_i, x_{i+1})
- for i=1 to 13 by 4
  - UNION(x_i, x_{i+2})
- UNION(x_1, x_5)



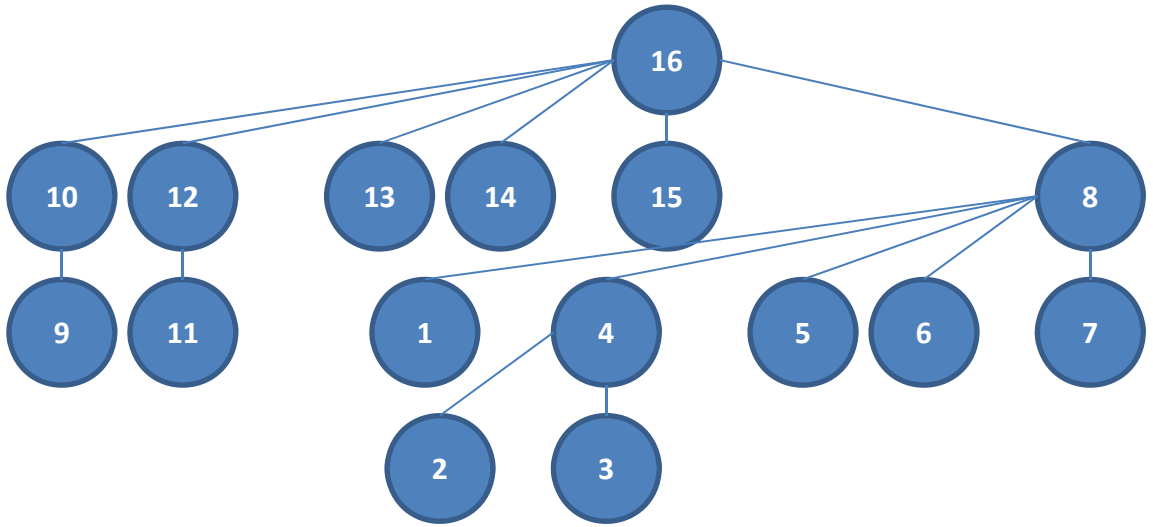
TC第21.3节练习1 (续)

- for i=1 to 16
  - MAKE-SET(x_i)
- for i=1 to 15 by 2
  - UNION(x_i, x_{i+1})
- for i=1 to 13 by 4
  - UNION(x_i, x_{i+2})
- UNION(x_1, x_5)
- UNION(x_{11}, x_{13})



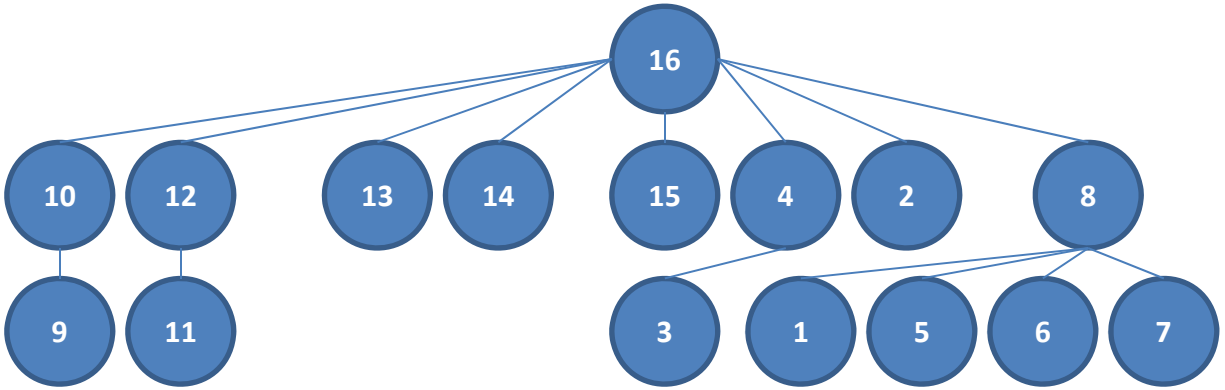
TC第21.3节练习1 (续)

- for i=1 to 16
  - MAKE-SET(x_i)
- for i=1 to 15 by 2
  - UNION(x_i, x_{i+1})
- for i=1 to 13 by 4
  - UNION(x_i, x_{i+2})
- UNION(x_1, x_5)
- UNION(x_{11}, x_{13})
- UNION(x_1, x_{10})



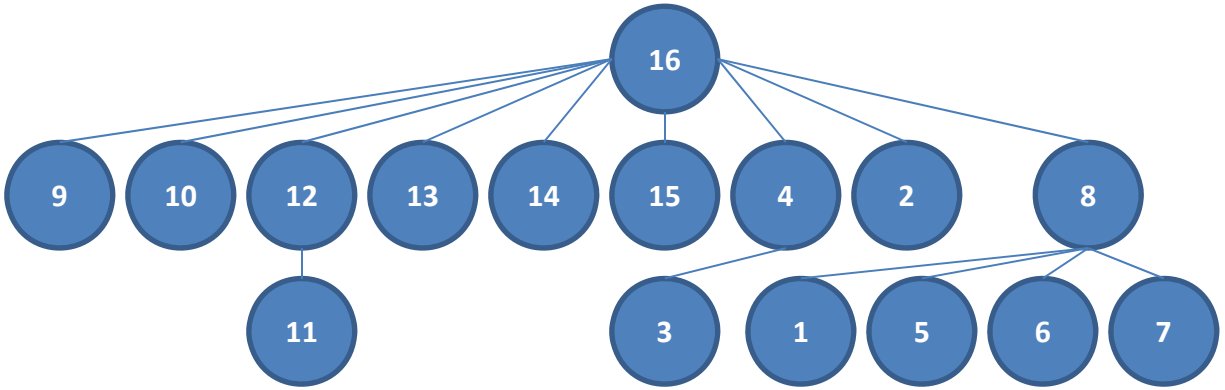
TC第21.3节练习1 (续)

- for i=1 to 16
  - MAKE-SET(x_i)
- for i=1 to 15 by 2
  - UNION(x_i, x_{i+1})
- for i=1 to 13 by 4
  - UNION(x_i, x_{i+2})
- UNION(x_1, x_5)
- UNION(x_{11}, x_{13})
- UNION(x_1, x_{10})
- FIND-SET(x_2)



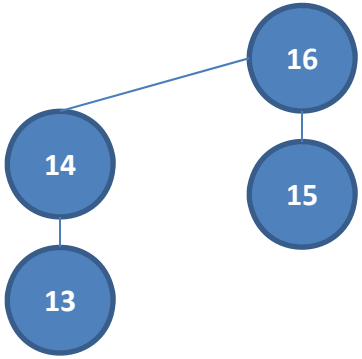
TC第21.3节练习1 (续)

- for i=1 to 16
  - MAKE-SET(x_i)
- for i=1 to 15 by 2
  - UNION(x_i, x_{i+1})
- for i=1 to 13 by 4
  - UNION(x_i, x_{i+2})
- UNION(x_1, x_5)
- UNION(x_{11}, x_{13})
- UNION(x_1, x_{10})
- FIND-SET(x_2)
- FIND-SET(x_9)



TC第21.3节练习2

- FIND-SET(x)
  - y=x;
  - while (y!=y.p) //先找根
    - y=y.p;
  - whie (x!=x.p) //再压缩路径
    - x.p=y;
    - x=x.p;
- 这段程序有什么问题？
- ...
    - temp=x.p;
    - x.p=y;
    - x=temp;

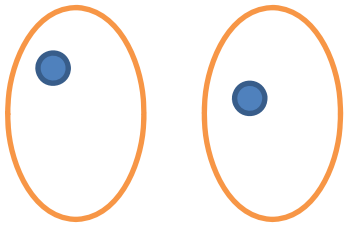


DW第1.1节练习4

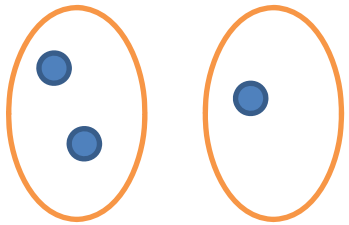
- 根据定义证明图同构包括哪些步骤？
  - 找顶点间的双射
  - 证明边的保持性

DW第1.1节练习10

- 不连通图的补图必连通。
  - 如果要分情况讨论，应当对要证的结论分情况，并讨论所有情况。
  - 证明：补图中任取两点……



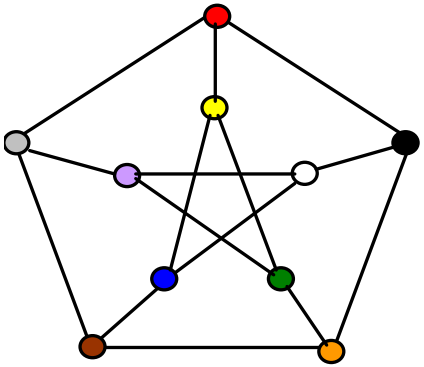
属于原图的不同连通分支

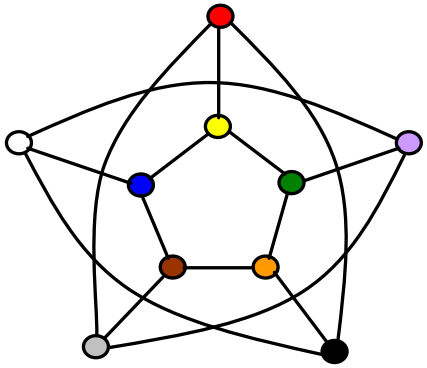


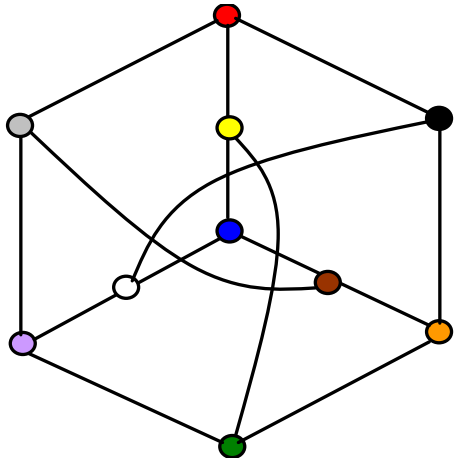
属于原图的同一连通分支

DW第1.1节练习16

- 你能想出哪些判断图的同构性的启发式方法？

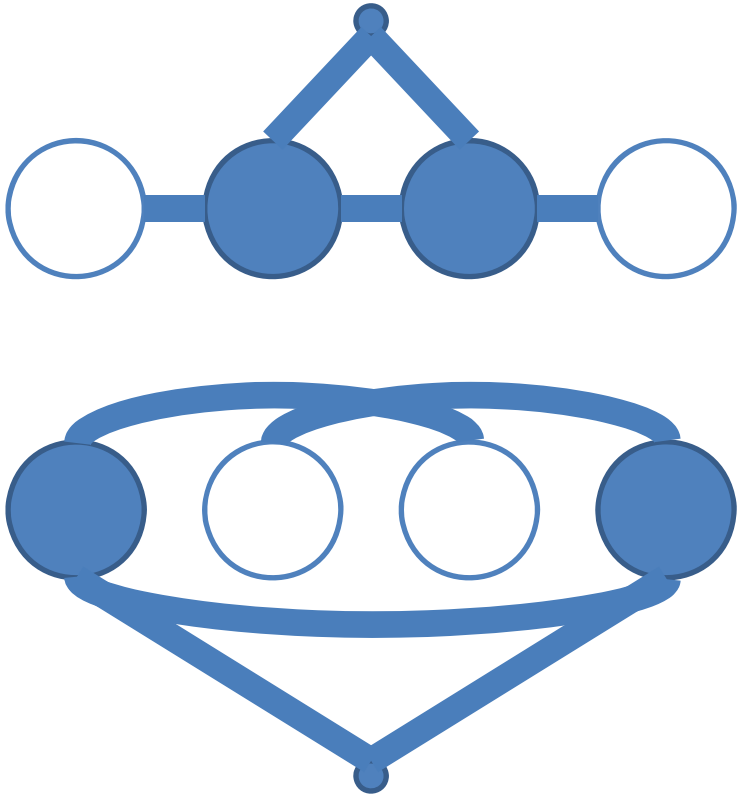






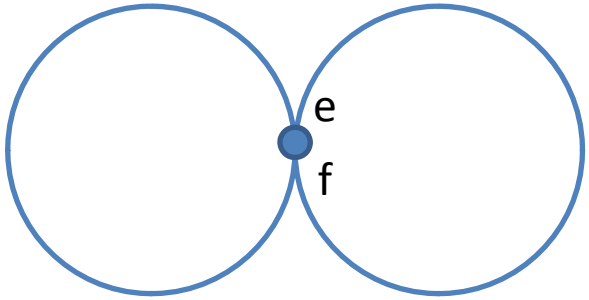
DW第1.1节练习31

- 必要性易证。
- 充分性：



DW第1.2节练习11

- 看清问题：能不能找到一个欧拉回路，其中e和f相继出现？



DW第1.2节练习18

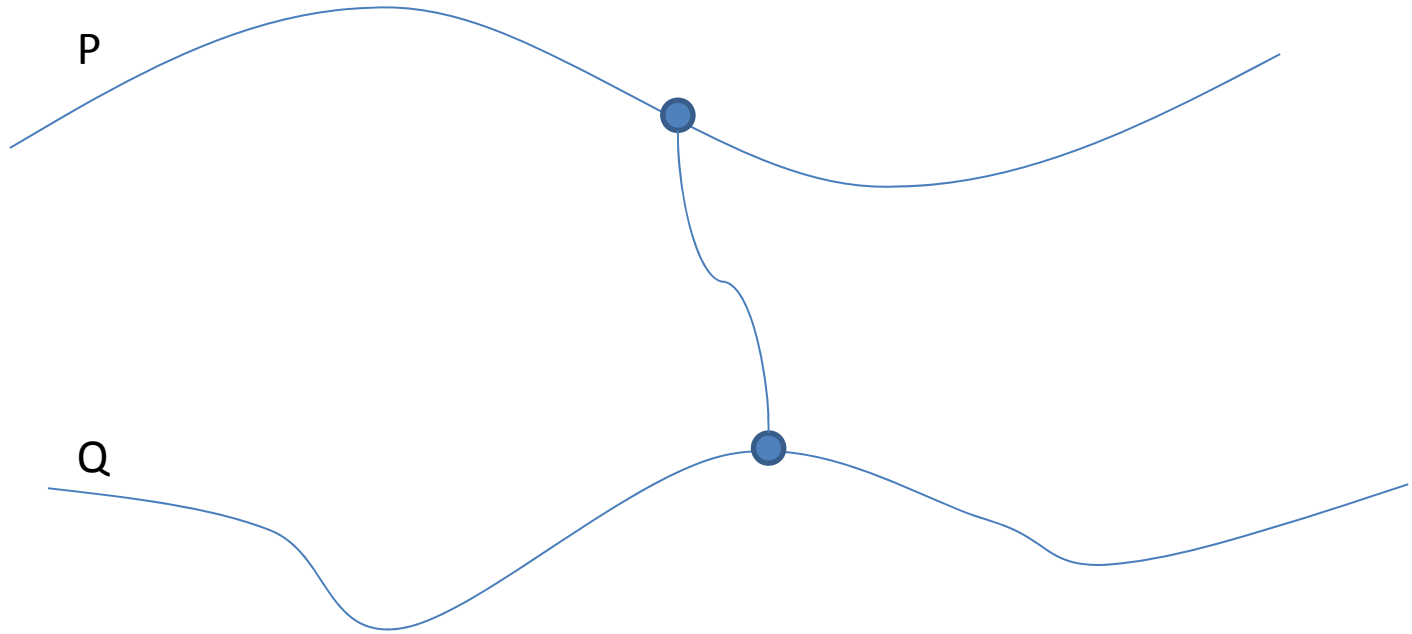
- 奇数个1的一组，偶数个1的一组
  - 证明至少有这两个组：互相不通
  - 证明至多有这两个组：内部皆通

DW第1.2节练习20

- v是G的割点 ⇒ G-v不连通 ⇒ 补集连通

DW第1.2节练习40

- 图论证明的表述要严谨：导致矛盾的更长的路如何构造？

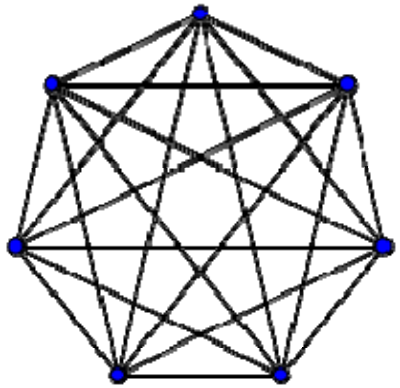


DW第1.4节练习5

- 看清题意：decomposes into cycles
  - 思路
    1. 利用最长路径证明存在一个圈。
    2. 去掉这个圈，剩余的图仍保持性质，可如此往复，最终分解为若干个圈。
  - 严格证明要使用数学归纳法
    - 假设边数<n的图满足这一性质，则边数=n时：找到一个圈，去掉，对剩余部分利用归纳假设。

DW第1.4节练习8

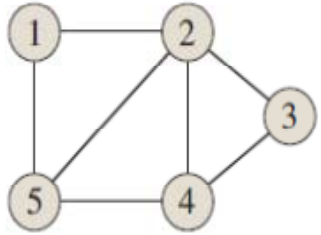
- 图论中使用数学归纳法证明，谨防陷入induction trap (P42)。
- 看清题意：if and only if。
- 充分性证明的一种思路：利用欧拉回路对K_n定向。

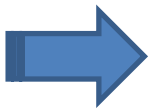


- 教材答疑和讨论
  - TC第22章
  - DW第2章
  - TC第23章

问题1：图的计算机表示

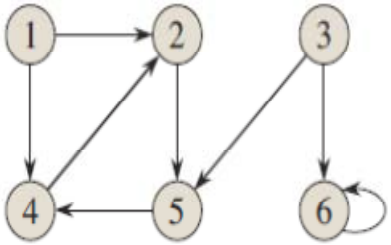
- 图的计算机表示 = 矩阵的计算机表示

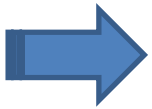




	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

重边怎么办？





	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

问题1：图的计算机表示 (续)

- 你能想出哪些方法在计算机中存储一个矩阵？

```
[ 10 20  0  0  0  0 ]
[  0 30  0 40  0  0 ]
[  0  0 50 60 70  0 ]
[  0  0  0  0  0 80 ]
```

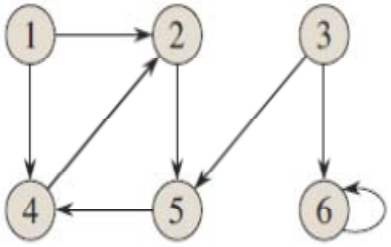
问题1：图的计算机表示 (续)

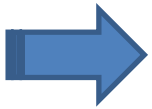
- Two-dimensional array
- 它有什么优缺点？
  - 时间
  - 空间

```
[ 10 20  0  0  0  0 ]
[  0 30  0 40  0  0 ]
[  0  0 50 60 70  0 ]
[  0  0  0  0  0 80 ]
```

问题1：图的计算机表示 (续)

- 实际问题中的图往往是稀疏的，对应的矩阵称作稀疏矩阵。
- 为什么稀疏矩阵的存储空间有可能缩小？

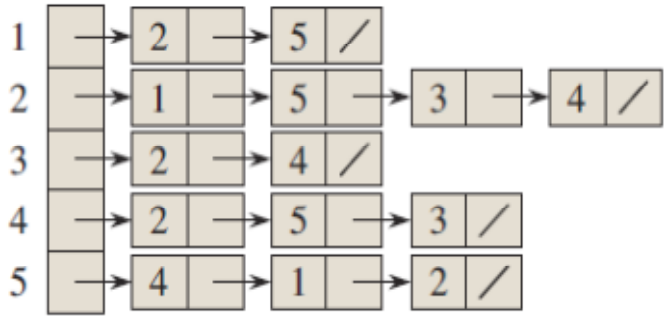




	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

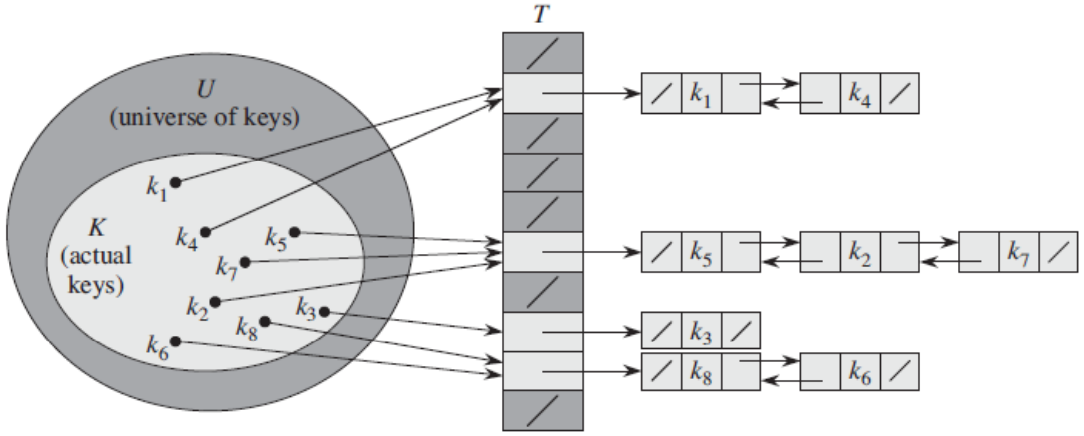
问题1：图的计算机表示 (续)

- List of lists
  - Stores one list per row, where each entry stores a column index and value.
  - Entries are kept sorted by column index.
- 它适合于什么样的应用？不适合于什么样的应用？
  - 随机访问
  - 向量/矩阵运算
  - 增删改



问题1：图的计算机表示 (续)

- Dictionary of keys
  - Represents non-zero values as a dictionary.
  - Maps (row, column)-tuples to values.
- 它适合于什么样的应用？不适合于什么样的应用？
  - 随机访问
  - 向量/矩阵运算
  - 增删改



问题1：图的计算机表示 (续)

- Coordinate list
  - Stores a list of (row, column, value) tuples.
  - Entries are sorted (by row index, then column index).
- 它适合于什么样的应用？不适合于什么样的应用？
  - 随机访问
  - 向量/矩阵运算
  - 增删改

```
{
{1, 2, 3},
{1, 4, 1},
{2, 3, 1},
{3, 1, 2},
{3, 4, 1},
{4, 1, 2}
}
```

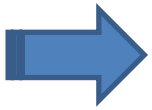
问题1：图的计算机表示 (续)

- 总体而言，这些表示方法都比较适合增删改，但不适合向量/矩阵运算。

问题1：图的计算机表示 (续)

- Yale format
  - A: array of non-zero element values.
  - IA: array of index of first nonzero element of row *i*.
  - JA: array of column index of each A element.
- 它适合于什么样的应用？不适合于什么样的应用？
  - 随机访问
  - 向量/矩阵运算
  - 增删改

```
[ 10 20  0  0  0  0 ]
[  0 30  0 40  0  0 ]
[  0  0 50 60 70  0 ]
[  0  0  0  0  0 80 ]
```



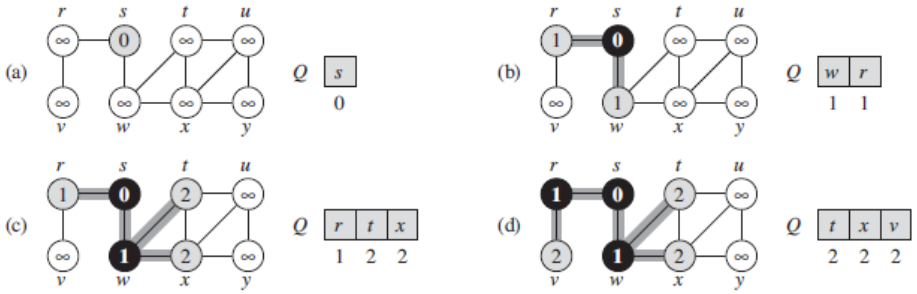
```
A  = [ 10 20 30 40 50 60 70 80 ]
IA = [ 0  2  4  7 8 ]
JA = [  0  1  1  3  2  3  4  5 ]
```

问题1：图的计算机表示 (续)

- 怎样才能同时做到
  - 高效的构建矩阵
  - 高效的向量/矩阵运算
- 结合使用
  1. Use dictionary of keys, list of lists, or coordinate list to construct the matrix.
  2. Once the matrix is constructed, it is typically converted to Yale format or similar formats for more efficient matrix operations.

问题2：图的搜索

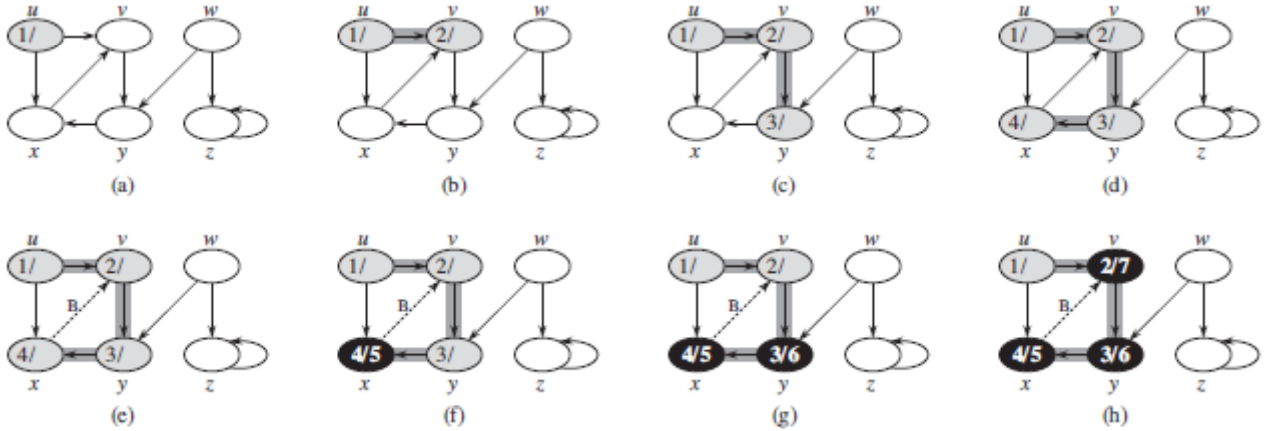
- 广度优先搜索 (BFS)
  - 顶点的三种颜色分别表示什么意思？
  - BFS-tree是如何构建的？为什么一定是树？




```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```

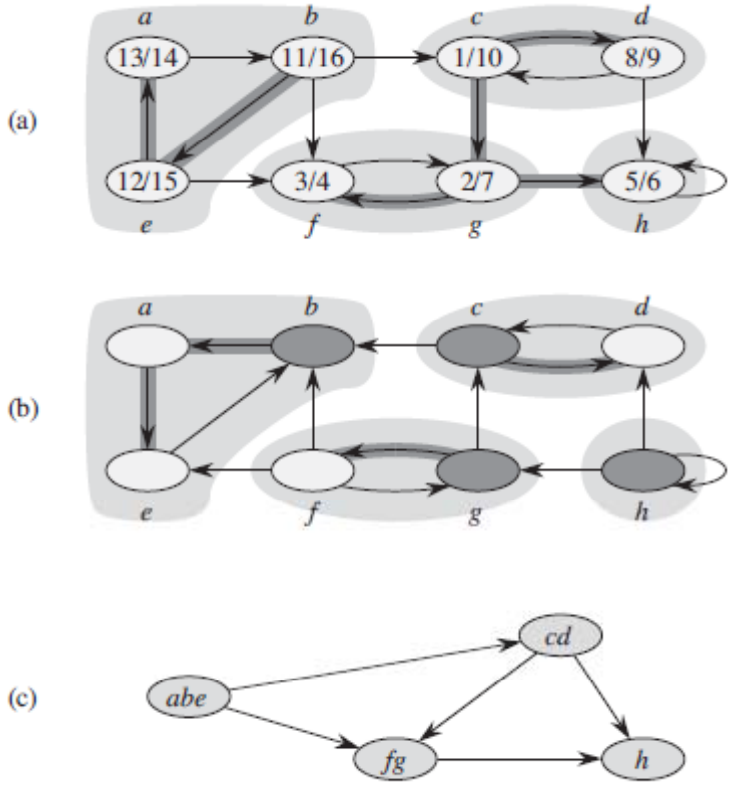
问题2：图的搜索 (续)

- 深度优先搜索 (DFS)
  - 如何理解子孙关系的三个充要条件？
  - v is a descendant of u in the depth-first forest if and only if
    - v is discovered during the time in which u is gray.
    - u.d < v.d < v.f < u.f.
    - At the time u.d, there is a path from u to v consisting entirely of white vertices.



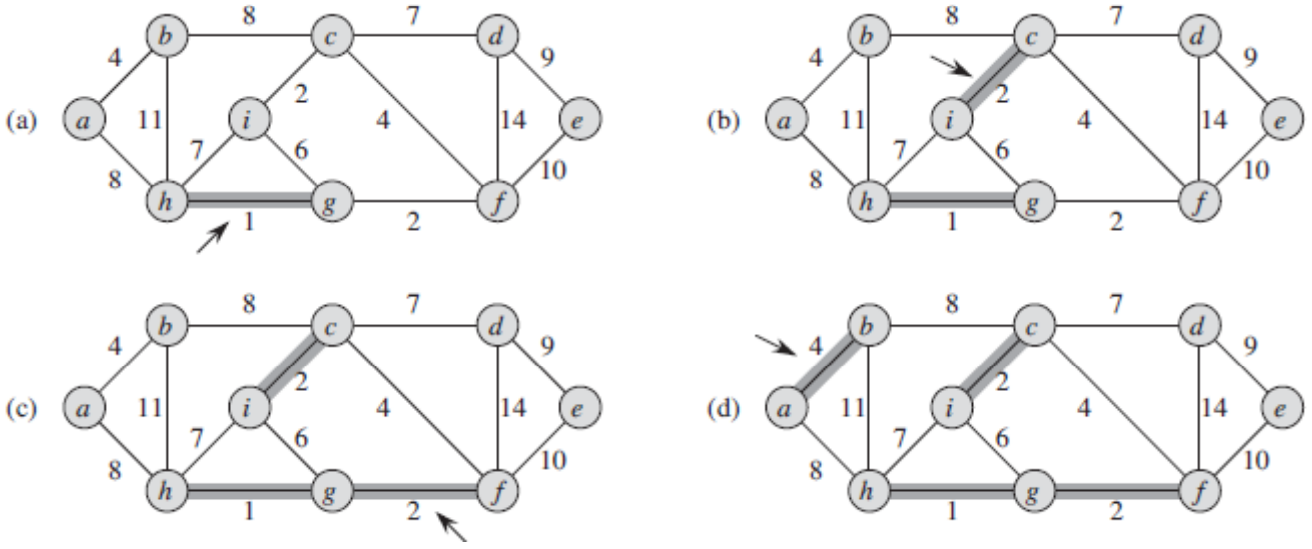
问题2：图的搜索 (续)

- 深度优先搜索 (DFS)
  - 你理解SCC算法的原理了吗？
    - We shall see that by considering vertices in the second depth-first search in decreasing order of the finishing times that were computed in the first depth-first search, we are, in essence, visiting the vertices of the component graph (each of which corresponds to a strongly connected component of G) in topologically sorted order.



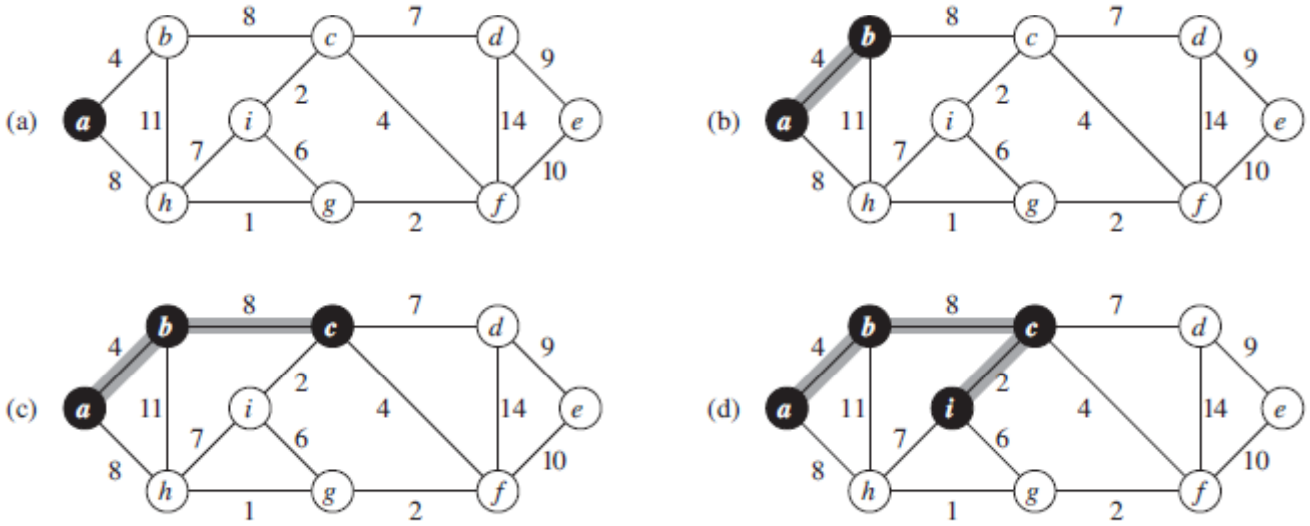
问题3：最小生成树

- 你能简述Kruskal算法的思想吗？
- 实现中最有技巧的步骤是什么？怎么实现？



问题3：最小生成树 (续)

- 你能简述Prim算法的思想吗？
- 实现中最有技巧的步骤是什么？怎么实现？



问题3：最小生成树 (续)

- 留给你们暑假思考
  - 找MST看起来是个挺难的问题，为什么用简单的贪婪算法就可以解决？
  - 还记得fractional knapsack problem吗？它也可以用贪婪算法找到最优解。想想看，这两个问题有什么共性？
  - 想不出来的话，去查一个概念：matroid (拟阵)。