- 作业讲解
  - TC第22.1节练习3、8
  - TC第22.2节练习3、4、5
  - TC第22.3节练习6、7、8、9、12
  - TC第22.4节练习2 、3
  - TC第22.5节练习5、7

TC第22.1节练习8

- 找到点的位置时间是O(1)，哈希链表的查找时间是O(α)，所以总共查找时间为O(1+ α)
- 哈希表的缺点：空间消耗太大
- 改进：BST
- 改进之后的缺点：时间消耗增加

TC第22.3节练习7

- 需要注意整个图不连通的情况

TC第22.4节练习2

- 从起始点开始，根据拓扑排序的结果向后搜索
- Funcition(G,u,v)
  - 除了u以外的点count为0，count[u] = 1
  - 根据拓扑排序结果，对每个节点n依次进行：
    - For all m in adj[n]
      count[m] += count[n]
  - return count[v]

- 不建议用DFS
  - 不易写对，写对了时间复杂度也很高

TC第22.4节练习3

- 如何证明算法的时间复杂度是O(|V|)？
  - 如果不存在回边，那么|E| = |V| - 1，DFS复杂度为O(|E|) = O(|V|)
  - 如果存在回边，那么在检测到回边的时候算法就终止了，之前检测过的子图必然是不存在回边的，那么算法时间复杂度也是O(|V|)

- 教材讨论
  - TC第23章

问题1：Generic method

- 什么样的边称作safe？
- safe边为什么一定存在？
- 如何证明这个算法的正确性？

```
GENERIC-MST(G, w)
A = Ø
while A does not form a spanning tree
    find an edge (u, v) that is safe for A
    A = A ∪ {(u, v)}
return A
```

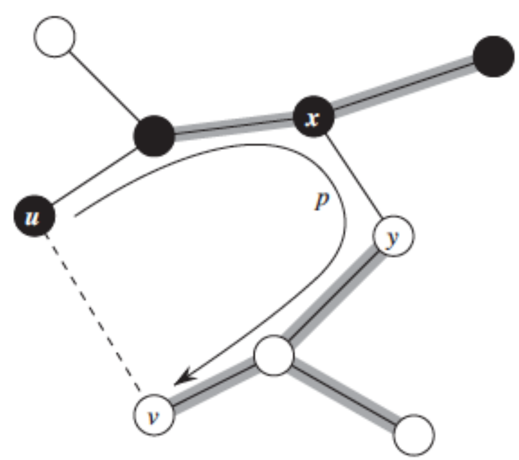
问题1：Generic method (续)

- 定理23.1的作用是什么？

Theorem 23.1

Let $G = (V, E)$ be a connected, undirected graph with a real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, let $(S, V - S)$ be any cut of G that respects A, and let (u, v) be a light edge crossing $(S, V - S)$. Then, edge (u, v) is safe for A.

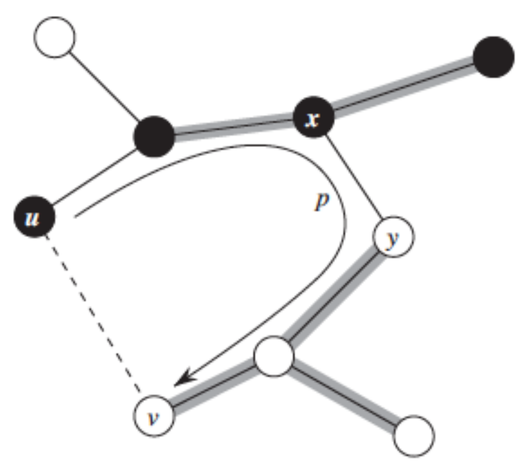
- 请结合这个图，简述定理的主要证明过程。



问题1：Generic method (续)

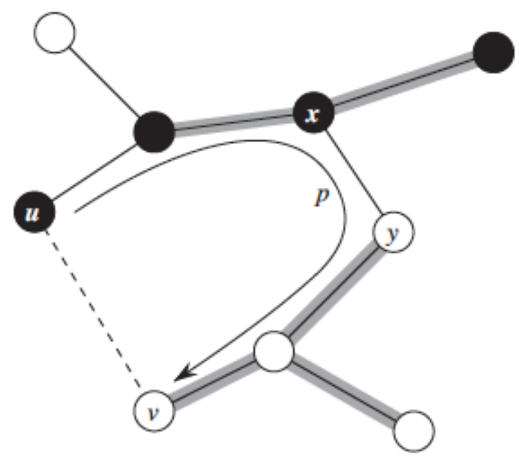
- Let (u, v) be a minimum-weight edge in a connected graph G. Show that (u, v) belongs to some minimum spanning tree of G.

- Show that if an edge (u, v) is contained in some minimum spanning tree, then it is a light edge crossing some cut of the graph.



问题1： Generic method (续)

- Show that a graph has a unique minimum spanning tree if, for every cut of the graph, there is a unique light edge crossing the cut. Show that the converse is not true by giving a counterexample.



问题2： Kruskal and Prim

- Kruskal和Prim分别如何选择safe边？

```
MST-KRUSKAL(G, w)
A = ∅
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

- 请简述Kruskal和Prim的实现方法

问题2： Kruskal and Prim (续)

- Suppose that all edge weights in a graph are integers in the range from 1 to $|V|$. How fast can you make Kruskal's algorithm run? What if the edge weights are integers in the range from 1 to W for some constant W?

```
MST-KRUSKAL(G, w)
A = ∅
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

问题2： Kruskal and Prim (续)

- For a sparse graph $G = (V, E)$, where $|E| = \Theta(V)$, is the implementation of Prim's algorithm with a Fibonacci heap asymptotically faster than the binary-heap implementation? What about for a dense graph, where $|E| = \Theta(V^2)$? How must the sizes $|E|$ and $|V|$ be related for the Fibonacci-heap implementation to be asymptotically faster than the binary-heap implementation?

```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```