Graph Algorithm II

Instructor: Shizhe Zhou

Course Code:00125401

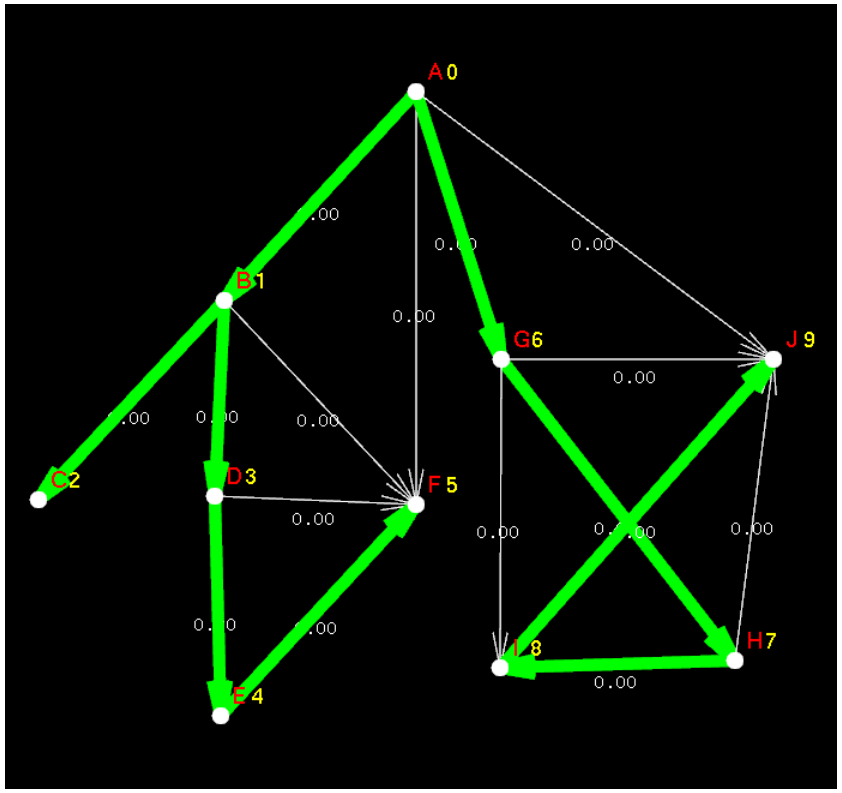
Scan the graph

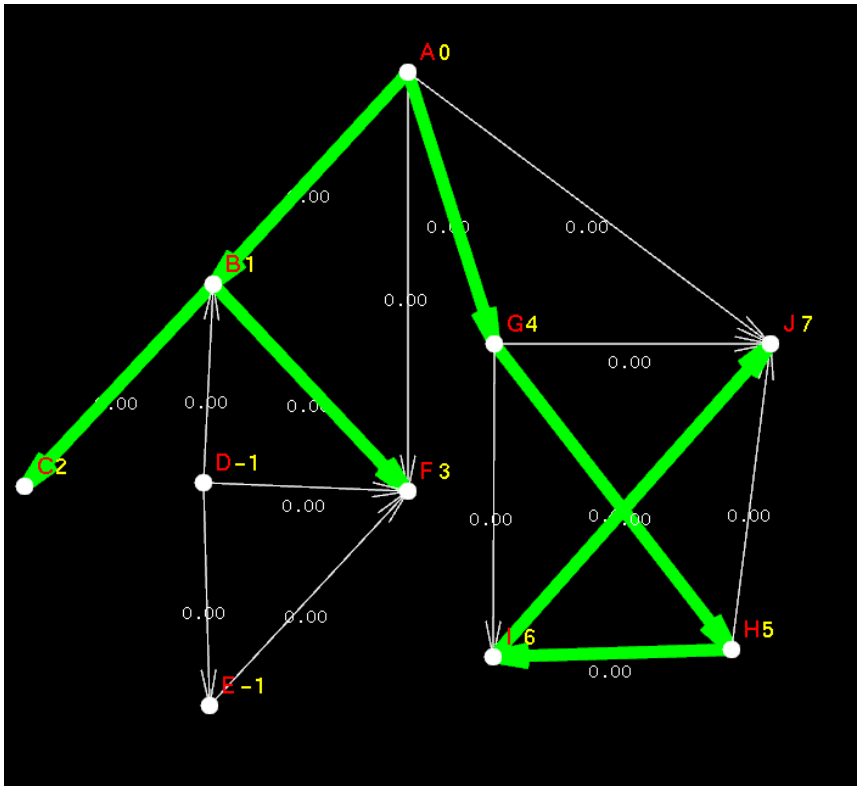
- Traverse method:
  - Depth-first search.
  - Breadth-first search.

DFS on directed graph

- Categorize all edges into 4 classes:

  Tree edge, forward/backward edge, cross edge.





Flip e(B,D), the DFS starting from a can no longer cover the graph.

Find cycles in directed graph

“一个有向图含有环的数量可能是#V的指数级别”.

- DFS维护的stack表示的就是从w到v的一条有向路径. 如果到了v点处发现边(v,w),那就说明发现了一个环.
- Code: 需要为节点增加两个变量：

1. bool on_the_path; //用于表示是否在该节点对应的分支中寻找下一个cycle(对v来说递归入栈时为true，出栈时为false)
2. Node* parent; //if当前节点v由w找到, 则parent=w;

Find cycles in undirected graph

"一个无向图含有环的数量可能是#V的指数级别".

- DFS维护的stack表示的就是从w到v的一条有向路径. 如果到了v点处发现边(v,w),并且w!=v.parent,那就说明发现了一个环.
- Code: 需要为节点增加两个变量：

1. bool on_the_path; //用于表示是否在该节点对应的分支中寻找下一个cycle(对v来说递归入栈时为true，出栈时为false)
2. Node* parent; //if当前节点v由w找到, 则parent=w;

Scan the graph

- Traverse method:
  - Depth-first search.
  - Breadth-first search.

Breadth-first Search

- 连通无向图的BFS从任意一个节点开始，都能遍历所有的节点.

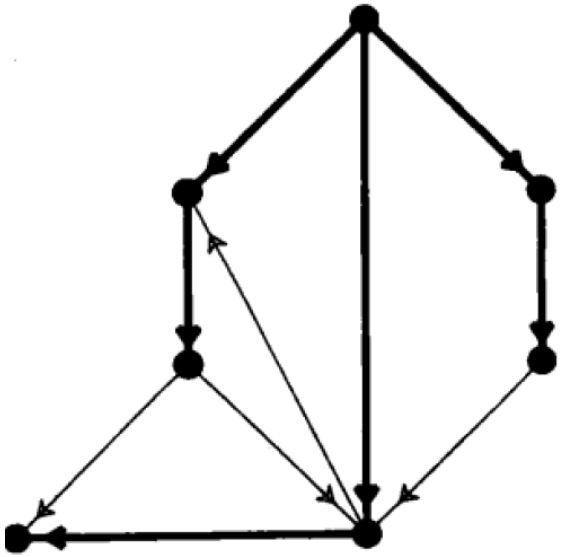
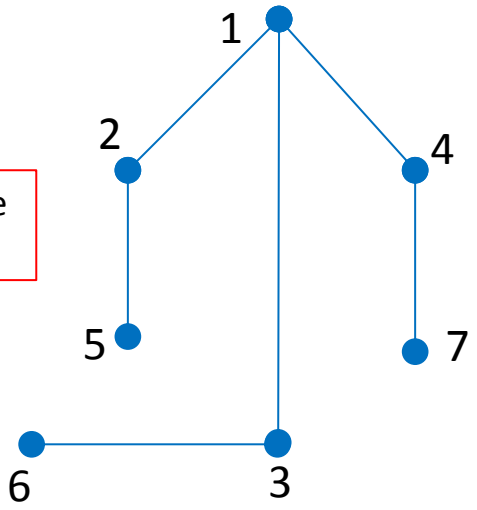


图 7.12　一棵有向图的 BFS 树

Keep the edges pointing to the next marked(visited) vertices



Visited:

1	1	0	0	....					

BFS tree

算法 *Breadth_First_Search* (*G*, *v*)

输入： $G=(V, E)$（一个无向连通图）和 v（G 中的一个节点）

输出： 与应用相关

```
begin
    mark v ;
    put v in a queue {先进先出}
    while the queue is not empty do
        remove the first vertex w from the queue ;
        perform preWORK on w ;
          {preWORK 依赖于 BFS 的具体应用}
        for all edges (w, x) such that x is unmarked do
            mark x ;
            add (w, x) to the tree T ;
            put x in the queue
end
```

Use a queue

图 7.13　广度优先搜索算法

BFS-tree

□ 引理 7.5

如果边 (u, w) 属于一棵 BFS 树，其中 u 是 w 的父母，则在具有导向 w 的边的顶点中，u 具有最小的 BFS 数。

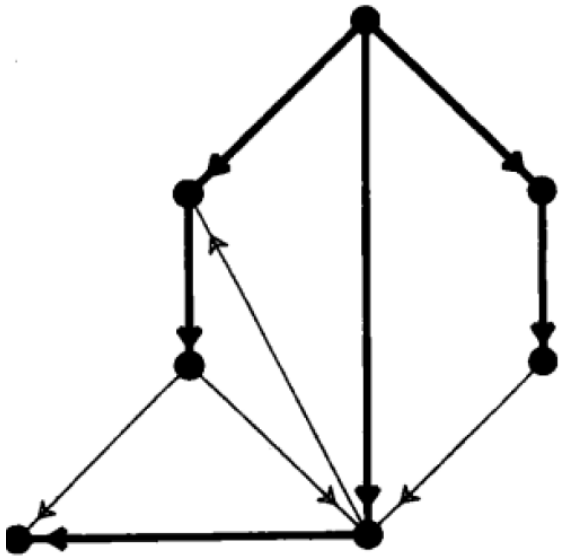


图 7.12　一棵有向图的 BFS 树

暂时不考虑带权图

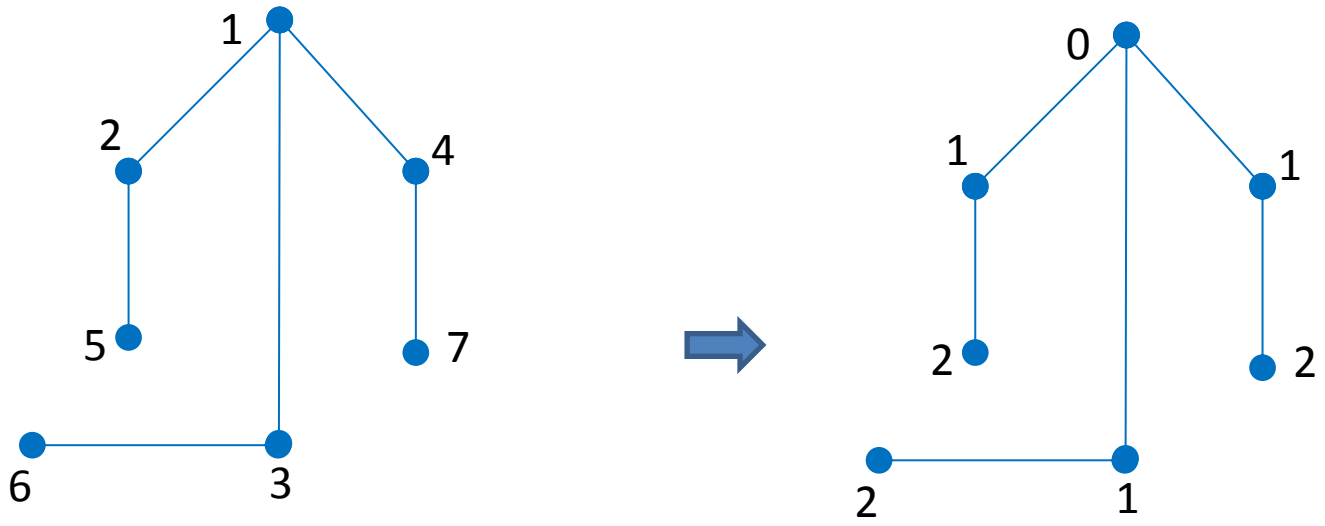
□ 引理 7.6

对于每一个顶点 w，在 T 中从根到 w 的路径是在 G 中从根到 w 的最短路径。

BFS-tree

□ 引理 7.7

如果(v, w)是 E 中的一条边且不属于 T，则它连接的两个顶点的层数至多相差 1。



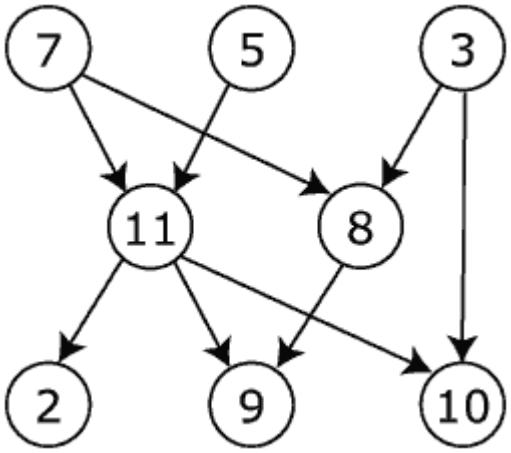
顶点 w 的层数是在树中从根到 w 的路径长度。BFS 是逐层对图进行遍历。

拓扑排序

□ **引理 7.8**

一个有向非循环图总有一个入度为 0 的顶点。

证明：如果所有的顶点都有一个正数的入度，则我们能够“向后”遍历这个图并永不停止。由于只有有限个顶点，所以必然进入一个循环，但这在一个非循环图中是不可能的。（使用同样的讨论可知，存在一个出度为 0 的顶点。） □



- 7, 5, 3, 11, 8, 2, 9, 10 (visual left-to-right, top-to-bottom)
- 3, 5, 7, 8, 11, 2, 9, 10 (smallest-numbered available vertex first)
- 3, 7, 8, 5, 11, 10, 2, 9
- 5, 7, 3, 8, 11, 10, 9, 2 (fewest edges first)
- 7, 5, 11, 3, 10, 8, 9, 2 (largest-numbered available vertex first)
- 7, 5, 11, 2, 3, 8, 9, 10

Multiple solution

Topological sorting

```
算法 Topological_Sorting (G)
输入：G=(V, E)（一个有向无回路图）
输出：Label 域的值指出图 G 的一个拓扑序
begin
    Initialize v.Indegree for all vertices ;{例如通过 DFS 算法}
    G_label := 0 ;
    for i := 1 to n do
        if v_i.Indegree = 0 then put v_i in Queue ;
    repeat
        remove vertex v from Queue ;
        G_label := G_label + 1 ;
        v.label := G_label ;
        for all edges (v, w) do
            w.Indegree := w.Indegree – 1 ;
            if w.Indegree = 0 then put w in Queue ;
    until Queue is empty
end
```

G_label is the order!!

Single source shortest path

- Book

- Code:

  http://staff.ustc.edu.cn/~szhou/course/algorithmFundamentals/graphalgo.zip