图

深度优先搜索：算法

10-E1

Keep it simple, stupid.

- K. Johnson

如右边天气阴沉多雨，我便往左边走；如遇上不宜骑马的地方，我就停下。如此这般行路……我身后是否还有什么东西可看？有，我便返回去，因为那也是我要走的路。

邓俊辉

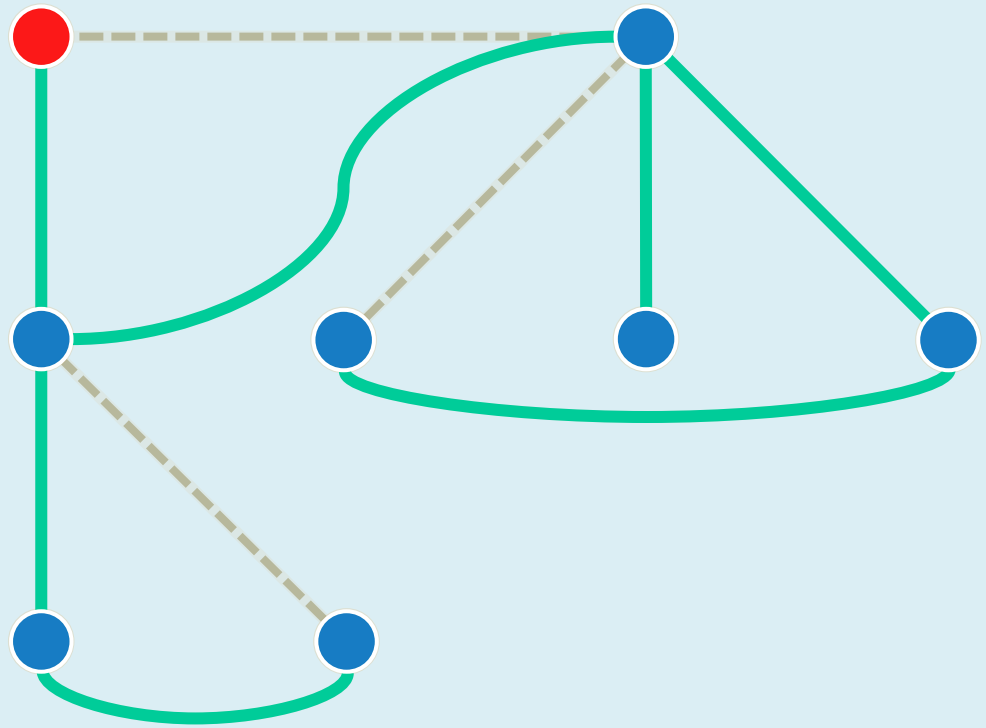
deng@tsinghua.edu.cn

Depth-First Search

- DFS(s) //始自顶点s的深度优先搜索
  - 访问顶点s
  - 若s尚有未被访问的邻居，则

    任取其一u，递归执行DFS(u)
  - 否则，返回
- 若此时尚有顶点未被访问

  任取这样的一个顶点作起始点
- 重复上述过程，直至所有顶点都被访问到
- 对树而言，等效于先序遍历：DFS也的确会构造出原图的一棵支撑树（DFS tree）



Graph::DFS() [1/2]

```
template <typename Tv, typename Te>

void Graph<Tv, Te>::DFS( Rank v, int & clock ) {

   dTime(v) = ++clock; status(v) = DISCOVERED; //发现当前顶点v

   for ( Rank u = firstNbr(v); -1 < u; u = nextNbr(v, u) ) //考察v的每一邻居u

      /* ... 视u的状态，分别处理 ... */

      /* ... 与BFS不同，含有递归 ... */

   status(v) = VISITED; fTime(v) = ++clock; //至此，当前顶点v方告访问完毕

}
```

Graph::DFS() [2/2]

```
for ( Rank u = firstNbr(v); -1 < u; u = nextNbr(v, u) ) //考查v的每一邻居u
   switch ( status(u) ) { //并视其状态分别处理
      case UNDISCOVERED: //u尚未发现，意味着支撑树可在此拓展
         type(v, u) = TREE; parent(u) = v; DFS( u, clock ); break; //递归
      case DISCOVERED: //u已被发现但尚未访问完毕，应属被后代指向的祖先
         type(v, u) = BACKWARD; break;
      default: //u已访问完毕（VISITED，有向图），则视承袭关系分为前向边或跨边
         type(v, u) = dTime(v) < dTime(u) ? FORWARD : CROSS; break;
   } //switch
```