图论 day2 :: DFS 序和 *CC

哈尔滨工业大学　王翰坤

2020 年 8 月 12 日

再谈 DFS

- 二叉树的遍历顺序：前序、中序、后序

再谈 DFS

- 二叉树的遍历顺序：前序、中序、后序
- 类似地一般树的遍历顺序：pre 序、post 序

再谈 DFS

- 二叉树的遍历顺序：前序、中序、后序
- 类似地一般树的遍历顺序：pre 序、post 序
- 感性地说，打上某个点 u 的 pre 标记时，子树 u 是新鲜的、未知的；打上 u 的 post 标记时，子树 u 是熟悉的、已知的

再谈 DFS

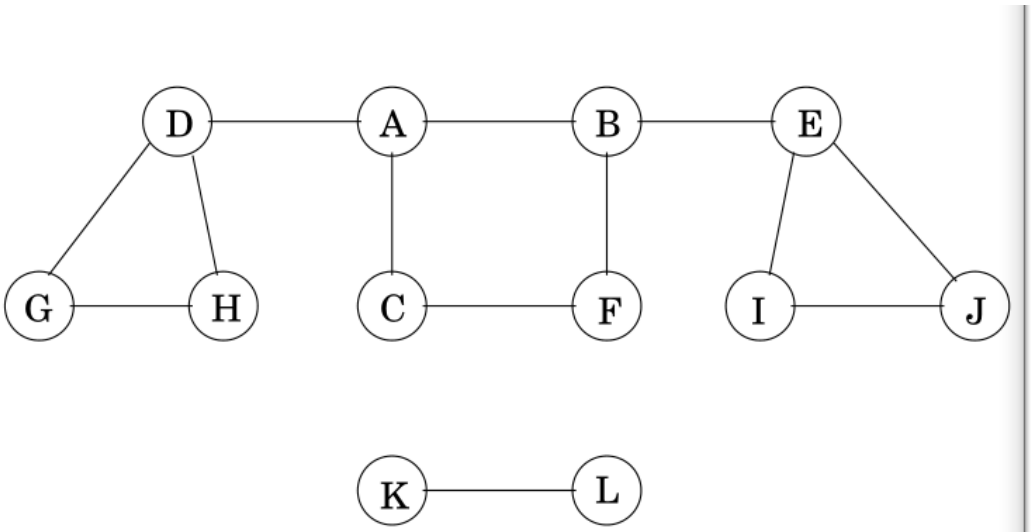
- 二叉树的遍历顺序：前序、中序、后序
- 类似地一般树的遍历顺序：pre 序、post 序
- 感性地说，打上某个点 u 的 pre 标记时，子树 u 是新鲜的、未知的；打上 u 的 post 标记时，子树 u 是熟悉的、已知的
- DFS 序

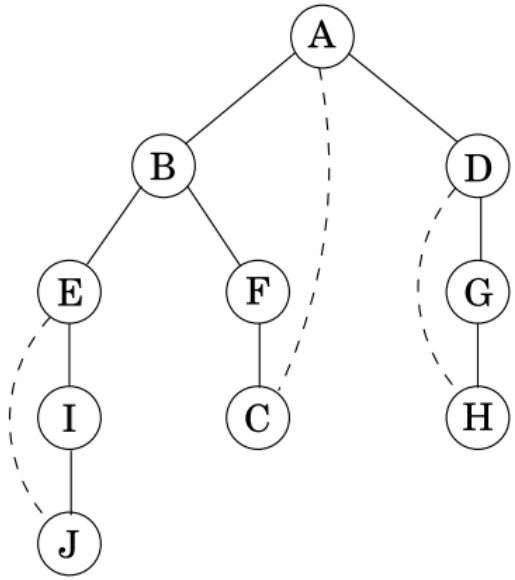
再谈 DFS

```
procedure previsit(v)
pre[v] = clock
clock = clock + 1
```

```
procedure postvisit(v)
post[v] = clock
clock = clock + 1
```

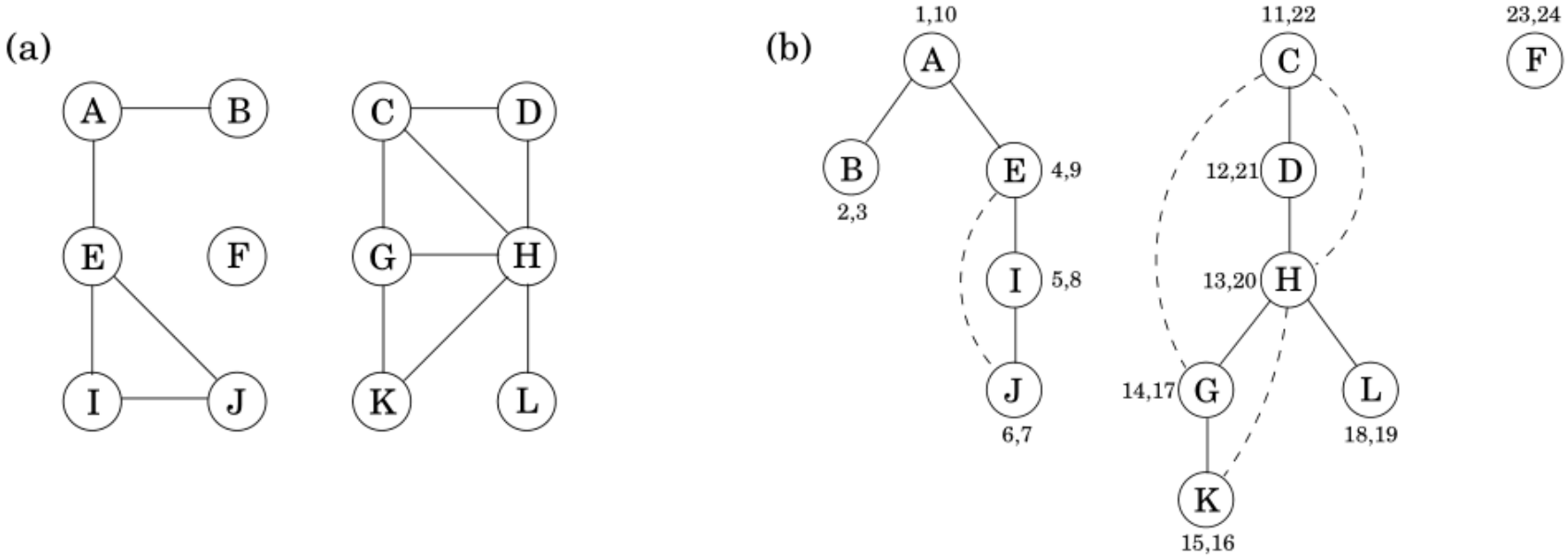
再谈 DFS





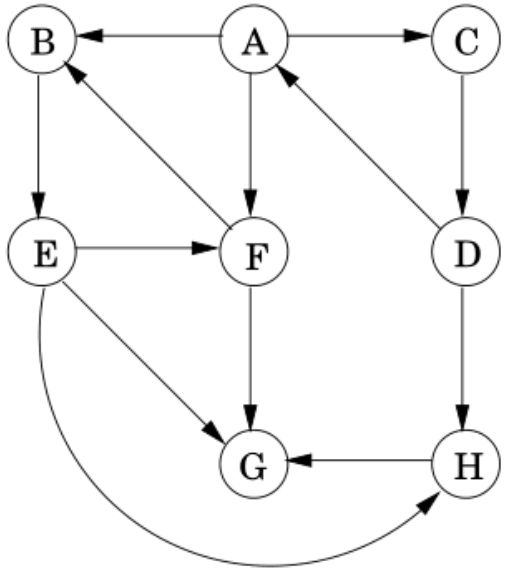
再谈 DFS

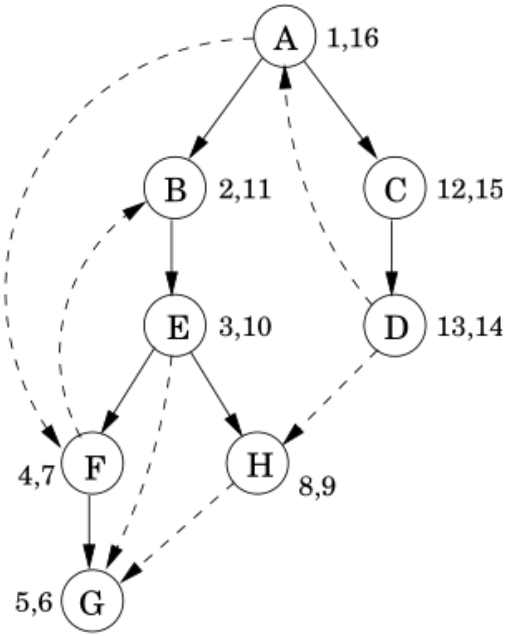
Figure 3.6 (a) A 12-node graph. (b) DFS search forest.



再谈 DFS

Figure 3.7 DFS on a directed graph.





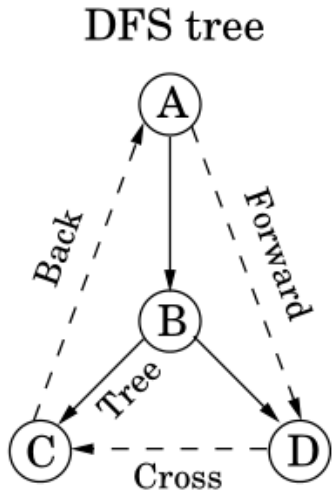
再谈 DFS

Tree edges are actually part of the DFS forest.

Forward edges lead from a node to a *nonchild* descendant in the DFS tree.

Back edges lead to an ancestor in the DFS tree.

Cross edges lead to neither descendant nor ancestor; they therefore lead to a node that has already been completely explored (that is, already postvisited).



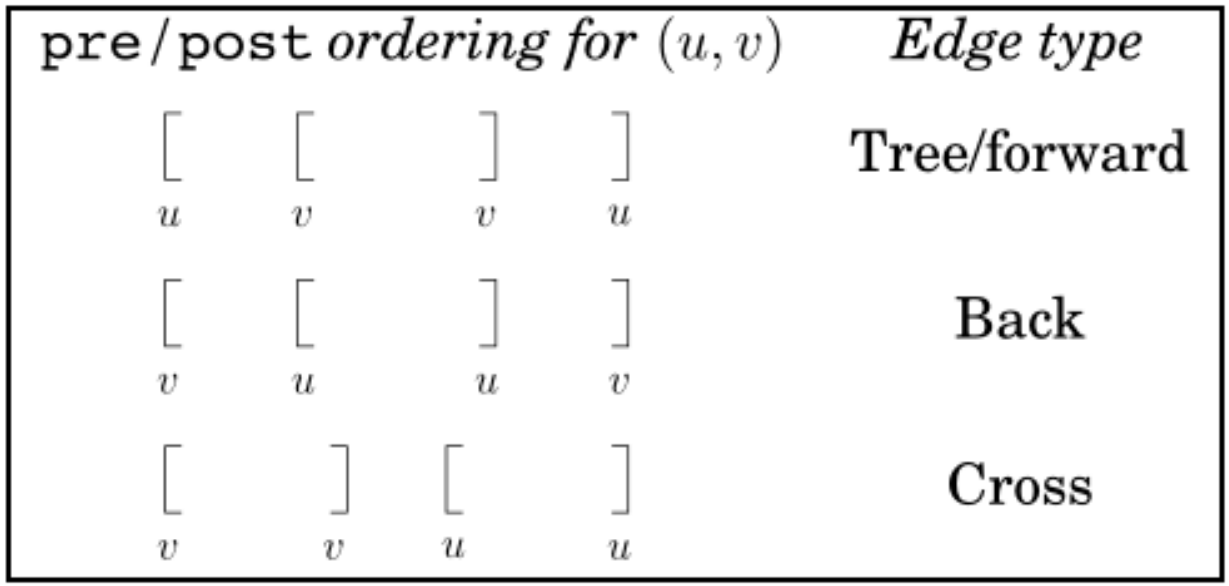
再谈 DFS

- 定理：对于任意一对节点 (u, v)，区间 $[pre_u, post_u]$ 和 $[pre_v, post_v]$ 要么没有交集，要么是包含关系
- 思考原因

再谈 DFS

- 定理：对于任意一对节点 (u, v)，区间 $[pre_u, post_u]$ 和 $[pre_v, post_v]$ 要么没有交集，要么是包含关系
- 思考原因
- DFS 序列中，一棵子树总是一个连续的区间
- 而两棵子树的关系：要么不相交，要么包含

再谈 DFS


pre/post ordering for (u, v)
Edge type
u v v u
Tree/forward
v u u v
Back
v v u u
Cross

再谈 DFS

- 用 DFS 序列求 LCA，请思考

再谈 DFS

- 用 DFS 序列求 LCA，请思考
- 神奇地转化为了区间最小值（RMQ）问题

例题 I

- 给出一棵 n 点的有根树，每个点都有一个权值 w_i。先要求选择不超过 K 个点，最大化权值和。此外，如果选了节点 i，就必须选它的父亲 fa_i（根节点除外）
- $n \leq 3000$

例题 I / 经典问题

- “树上背包”
- 把树上问题转化为熟悉的一维序列问题
- 又注意到，如果不选一个点 i，则其子树都不能被选；而子树在 DFS 序列上是连续的，容易处理
- 设 $d[i][j]$ 表示考虑到 DFS 序中第 i 个节点，已经选了 j 个时的最大权值和
- 状态转移：

$$f[i][j] = \begin{cases} \to f[i+1][j+1] & \text{选 } i, \\ \to f[i+size_i][j] & \text{不选 } i. \end{cases}$$

- 时间复杂度 $O(n^2)$

割顶和桥

- 对于一个无向图，若删除某个点后，连通分量的数目增加，则称其为“割顶”
- 对于连通图，删除割顶将使得它不再连通
- 类似地，可以类比到边，定义“桥”
- 考虑如何求割顶和桥

求割顶

- 暴力：删除-测试法，不够高效
- 考虑充分利用 DFS 树

DFS 树

- 一种用来描述图的 DFS 过程的树
- 为了后续叙述方便，给出一些概念
- 有向图的 DFS 树上的四种边：树边、前向边、回边、横切边
- 无向图的 DFS 树上，除了树边之外都是回边
- 定理：无向图的某个节点 u 是割顶，当且仅当 u 的某个子树 v 中不含有返回 u 的祖先（不含 u 本身）的回边。
- 特别地，根节点是割顶当且仅当它的子节点数量大于 1

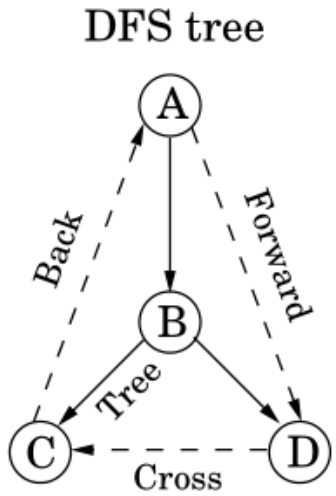
DFS 树

Tree edges are actually part of the DFS forest.

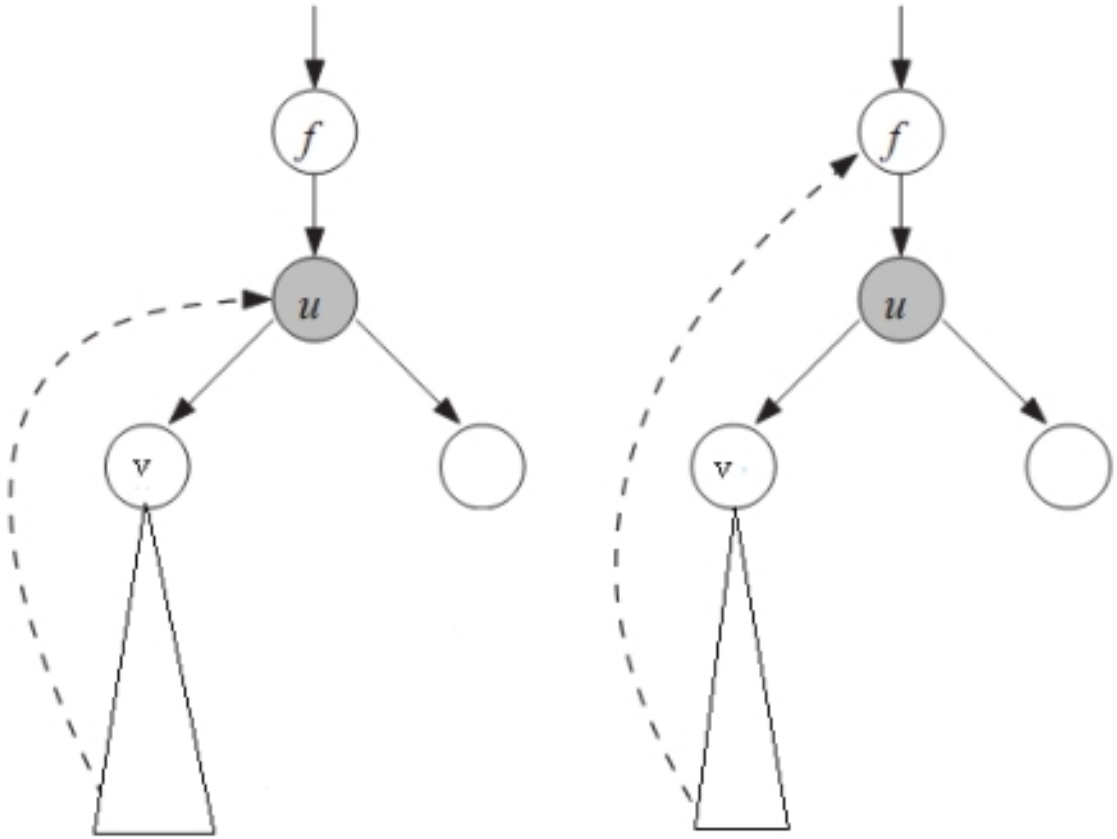
Forward edges lead from a node to a *nonchild* descendant in the DFS tree.

Back edges lead to an ancestor in the DFS tree.

Cross edges lead to neither descendant nor ancestor; they therefore lead to a node that has already been completely explored (that is, already postvisited).



DFS 树



DFS 树

- 记节点 u 及其后代能返回的最高祖先的 pre 值为 low_u

$$low[u] = \begin{cases} \min\{low[u], low[v]\} & (u, v) \text{ 为树边}, \\ \min\{low[u], dfn[v]\} & (u, v) \text{ 为回边}. \end{cases}$$

- 则上面定理的条件可表示为：u 存在一个子节点 v，使得 $low_v \geq pre_u$
- 问题转化为高效地求每个点的 low 值
- DFS 的过程中顺便求即可
- 尝试实现代码
- 思考桥的求法

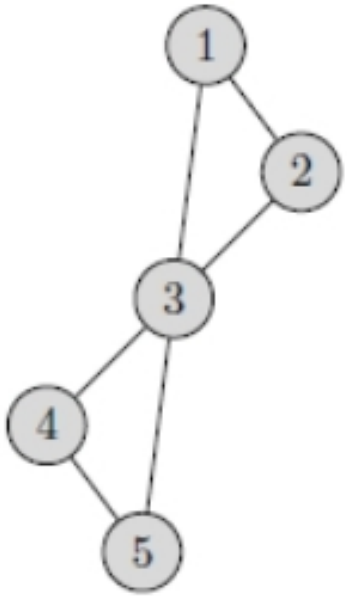
BCC 和 e-BCC

- 双连通分量（BCC）指的是，满足“任意两个点间都有至少两条「点不相同」路径”的极大子图

BCC 和 e-BCC

- 双连通分量（BCC）指的是，满足“任意两个点间都有至少两条「点不相同」路径”的极大子图
- “极大”?
- 类似地，定义“边-双连通分量（e-BCC)”，即「边不相同」

DFS 树



图中有一个 eBCC，两个 BCC

BCC 和 e-BCC

- 尝试研究 BCC 的性质（与割顶联系起来），以及 e-BCC 的性质（与桥联系起来）

BCC 和 e-BCC

- 尝试研究 BCC 的性质（与割顶联系起来），以及 e-BCC 的性质（与桥联系起来）
- 不难发现，两个 BCC 至多只有一个公共点，且它一定是割顶；不是割顶的点，一定属于某个 BCC
- 求出割顶就可以把 BCC 分离出来（BCC 相当于“不含割顶的极大子图”）
- 同理，两个 e-BCC 之间至多只有一条边相连，且它一定是桥；不是桥的边，一定属于某个 e-BCC
- 求出桥就可以把 e-BCC 分离出来（e-BCC 相当于“不含桥的极大子图”）

BCC 和 e-BCC

- 尝试实现 BCC 的分离。两个提示：
  - 一个割顶可能属于两个 BCC，不好标记，所以一般分离边
  - 可以用一个栈保存遍历到的边，一旦发现割顶就把栈顶的若干条属于同一个 BCC 的边弹出，实现分离

BCC 和 e-BCC

- 尝试实现 BCC 的分离。两个提示：
  - 一个割顶可能属于两个 BCC，不好标记，所以一般分离边
  - 可以用一个栈保存遍历到的边，一旦发现割顶就把栈顶的若干条属于同一个 BCC 的边弹出，实现分离
- eBCC 更加简单，请大家思考

BCC 和 e-BCC

- 尝试实现 BCC 的分离。两个提示：
  - 一个割顶可能属于两个 BCC，不好标记，所以一般分离边
  - 可以用一个栈保存遍历到的边，一旦发现割顶就把栈顶的若干条属于同一个 BCC 的边弹出，实现分离
- eBCC 更加简单，请大家思考
- 第一遍 DFS 求出桥，第二遍 DFS 分离即可

例题 II

- 定义一个节点的「WHX 值」为：去掉这个点之后图中连通分量的个数。给出一张 n 个点的无向连通图，输出 WHX 值前 m 大的节点
- $n \leq 5 \times 10^4$

例题 II / UVa 10765

Solution

- 不是割顶的点，「WHX 值」为 1；割顶的「WHX 值」为与之相邻的不同 BCC 数目

例题 III

- 在一个 n 个点的无向图上选择尽量少的点涂黑，使得任意删除一个点后，每个连通分量至少有一个黑点；并求出最优方案的数量
- $n \leq 5 \times 10^4$

例题 III / BZOJ 2730

Solution

- 把割顶涂黑是不划算的；在同一个 BCC 里涂两个点也是不划算的
- 每个 BCC 都涂一个，有必要吗?

例题 III / BZOJ 2730

Solution

- 把割顶涂黑是不划算的；在同一个 BCC 里涂两个点也是不划算的
- 每个 BCC 都涂一个，有必要吗?
- 一个 BCC 只有一个割顶时才需要涂，任选一个非割顶的点涂黑即可

例题 III / BZOJ 2730

Solution

- 把割顶涂黑是不划算的；在同一个 BCC 里涂两个点也是不划算的
- 每个 BCC 都涂一个，有必要吗?
- 一个 BCC 只有一个割顶时才需要涂，任选一个非割顶的点涂黑即可
- 特殊情形：整个图没有割顶
- 任涂两个点即可
- 方案数计算也就很简单了

强连通分量

- 强连通分量（SCC）一般是对于有向图而言
- “强连通”指两两相互可达，强连通分量就是指原图的一个极大强连通子图
- 显然，两个强连通分量既没有公共点，也没有公共边

强连通分量

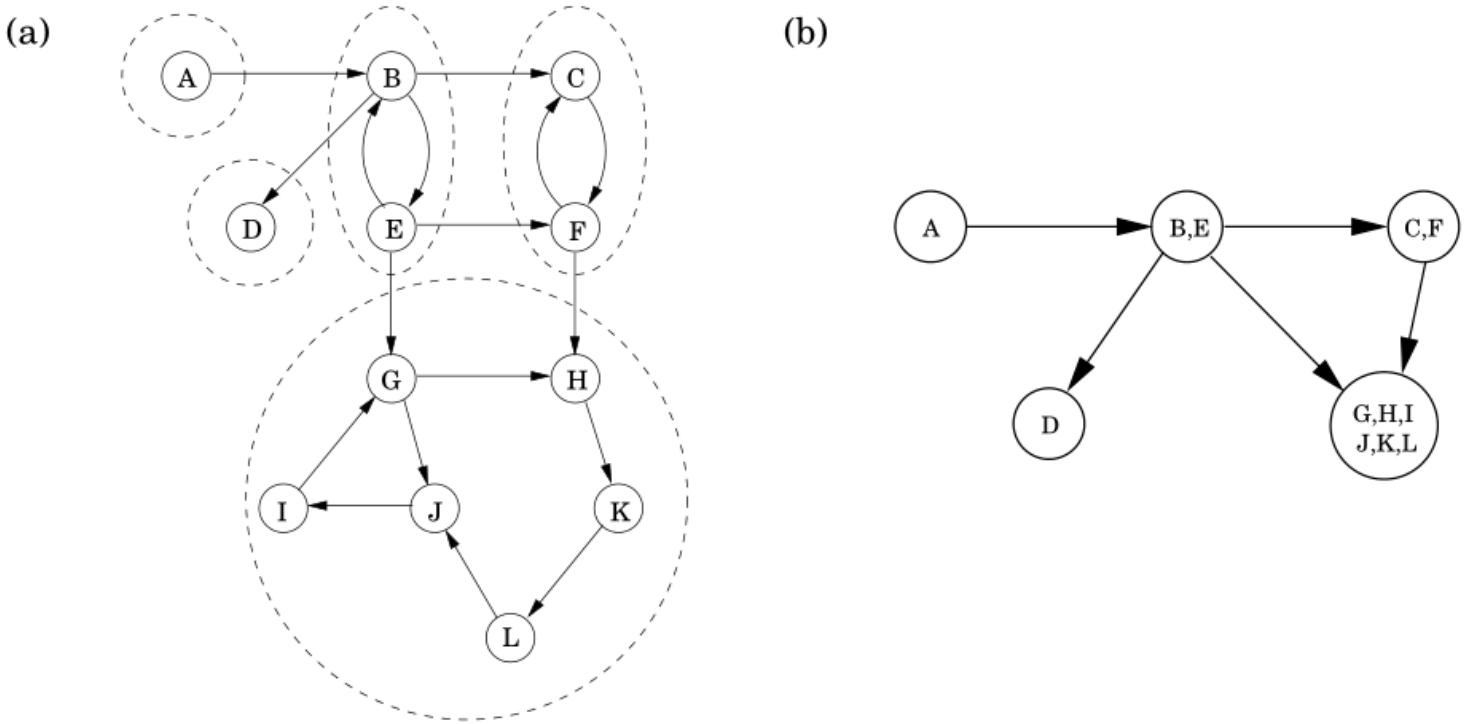
- 强连通分量（SCC）一般是对于有向图而言
- “强连通”指两两相互可达，强连通分量就是指原图的一个极大强连通子图
- 显然，两个强连通分量既没有公共点，也没有公共边
- 重要定理：所有有向图都是由其强连通分量构成的 DAG

强连通分量

- 强连通分量（SCC）一般是对于有向图而言
- “强连通”指两两相互可达，强连通分量就是指原图的一个极大强连通子图
- 显然，两个强连通分量既没有公共点，也没有公共边
- 重要定理：所有有向图都是由其强连通分量构成的 DAG
- 求强连通分量的算法主要有 Kosaraju 算法和 Tarjan 的算法
- 二者复杂度都是线性
- 前者实现相对简单，但其正确性稍难理解。后者更好理解且实际效率更高

强连通分量

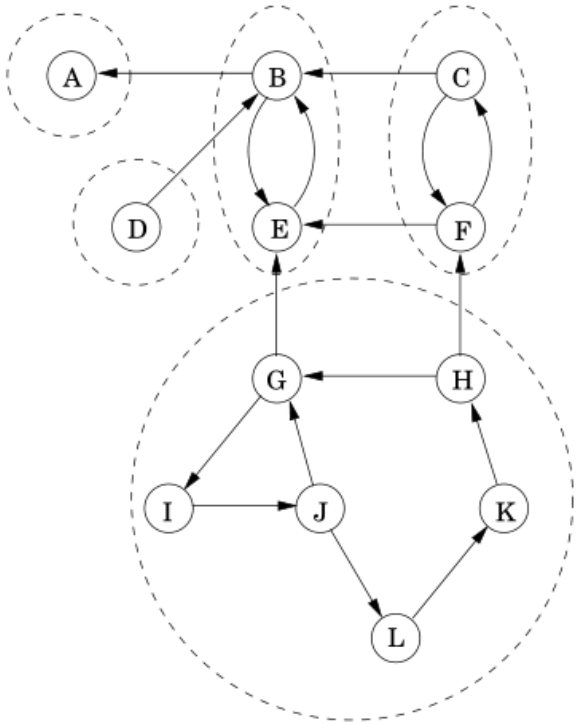
Figure 3.9 (a) A directed graph and its strongly connected components. (b) The meta-graph.

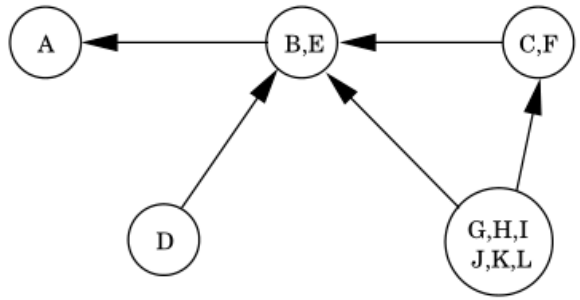


强连通分量

Kosaraju 算法

Figure 3.10 The *reverse* of the graph from Figure 3.9.





强连通分量

Kosaraju 算法

- Kosaraju 算法对原图 G 和它的逆图 G^T 先后进行 DFS
  - 对原图 G 进行 DFS，找出每个节点的 post 时间
  - 选择完成时间较大的节点开始，对逆图 G^T 搜索，能够到达的点构成一个强连通分量
- 正确性证明请参考相关书籍

强连通分量

Tarjan 算法

- Tarjan 的 SCC 算法仍然是在充分利用 DFS
- 设某强连通分量 C 中第一个被发现的点为 u，则 C 中其他点都是 u 的后代
- 我们希望在 u 访问完成时立刻输出 C，以此在同一棵 DFS 树中区分开所有 SCC
- 如何判断某个点 u 是否为一个 SCC 中第一个被发现的点?

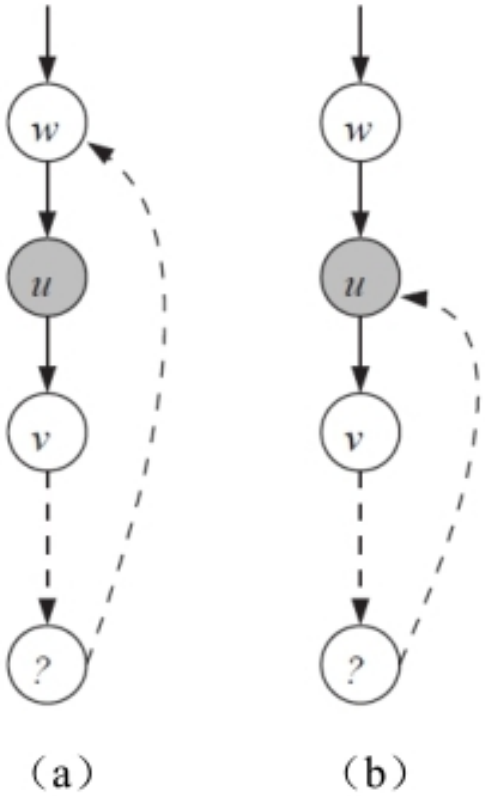
强连通分量

Tarjan 算法

- Tarjan 的 SCC 算法仍然是在充分利用 DFS
- 设某强连通分量 C 中第一个被发现的点为 u，则 C 中其他点都是 u 的后代
- 我们希望在 u 访问完成时立刻输出 C，以此在同一棵 DFS 树中区分开所有 SCC
- 如何判断某个点 u 是否为一个 SCC 中第一个被发现的点?
- 类似地定义 low 函数!
- 第一个被发现的点 u 就是满足 $pre[u] = low[u]$ 的点
- 因此，访问完某棵子树 u 之后，如果满足 $pre[u] = low[u]$，就弹出栈顶的节点直到弹出 u，标记为一个 SCC 即可

强连通分量

Tarjan 算法



例题 IV

- 给出一张 n 个点 m 条边的有向图，求一个最大的子节点集，使得子集中任意一对节点 (u, v) 要么可以从 u 到 v，要么可以从 v 到 u，要么双向可达
- $n \leq 1000, m \leq 50000$

例题 IV / UVa 11324

Solution

- 一个 SCC 中的点，要么都选，要么都不选
- 缩点后转化为 DAG 上最大权路径问题，DP 之即可

例题 V

- 有 n 个人召开圆桌会议，m 对人不能坐在相邻座位。一场会议必须有不少于 3 人的奇数人参加。问有哪些人不能参加任何一个会议
- $n \leq 1000, m \leq 10^6$

例题 V / UVa 1364

Solution

- 建图：一个人对应一个节点，把“能坐在相邻座位”的人连一条边
- 则一场可行的圆桌会议与一个简单奇圈[1]一一对应
- 问题转化为“求不在任何一个简单奇圈上的节点数目”

[1]一条不含重复边和重复节点的路被称做简单路；出发节点和终止节点相同的简单路被称为简单环

例题 V / UVa 1364

Solution

- 我们把“奇”和“圈”分开考虑

例题 V / UVa 1364

Solution

- 我们把“奇”和“圈”分开考虑
- 关于圈，定理：一个节点在一个圈中，等价于它在一个 BCC 中
  - 所有不在 BCC 上的点都将贡献给答案

例题 V / UVa 1364

Solution

- 我们把“奇”和“圈”分开考虑
- 关于圈，定理：一个节点在一个圈中，等价于它在一个 BCC 中
  - 所有不在 BCC 上的点都将贡献给答案
- 关于奇，二分图 等价于 不含奇圈的图
  - 所以，二分图 BCC 上的点将贡献给答案

例题 V / UVa 1364

Solution

- 我们把“奇”和“圈”分开考虑
- 关于圈，定理：一个节点在一个圈中，等价于它在一个 BCC 中
  - 所有不在 BCC 上的点都将贡献给答案
- 关于奇，二分图 等价于 不含奇圈的图
  - 所以，二分图 BCC 上的点将贡献给答案
- 那么，非二分图 BCC 上的点是否全都在奇圈上呢?

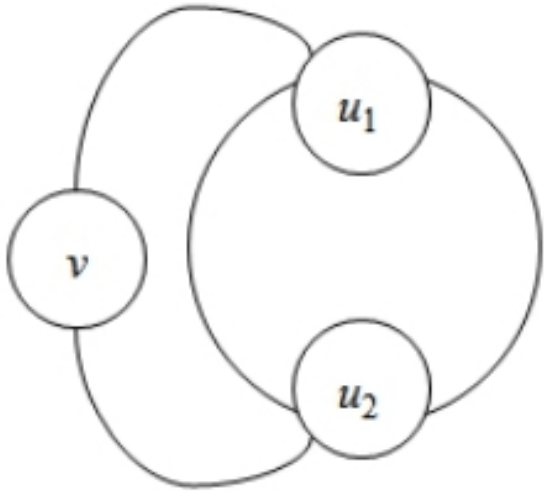
例题 V / UVa 1364

Solution

- 我们把“奇”和“圈”分开考虑
- 关于圈，定理：一个节点在一个圈中，等价于它在一个 BCC 中
  - 所有不在 BCC 上的点都将贡献给答案
- 关于奇，二分图 等价于 不含奇圈的图
  - 所以，二分图 BCC 上的点将贡献给答案
- 那么，非二分图 BCC 上的点是否全都在奇圈上呢?
- 画个图构造，易知的确全都在奇圈上：非二分图 BCC 对答案无贡献
- 问题得到解决

例题 V / UVa 1364

Solution



例题 VI

- 给出一张 n 个点 m 条边的无向连通图，问最少需要添加多少条边使得整张图成为边双连通的
- $3 \leq n \leq 1000, 2 \leq m \leq 1000$

例题 VI / POJ 3352

Solution

- eBCC 缩点变成树

例题 VI / POJ 3352

Solution

- eBCC 缩点变成树
- eBCC 本身无需处理，如何把添加最少的边把一棵树变成边双连通的呢?
- 一个显然的可行解是，把每个叶子都往根节点连一条边

例题 VI / POJ 3352

Solution

- eBCC 缩点变成树
- eBCC 本身无需处理，如何把添加最少的边把一棵树变成边双连通的呢?
- 一个显然的可行解是，把每个叶子都往根节点连一条边
- 一个更优的解是，把“相邻的叶子节点”之间连边即可
- 有没有特殊情况?

例题 VI / POJ 3352

Solution

- eBCC 缩点变成树
- eBCC 本身无需处理，如何把添加最少的边把一棵树变成边双连通的呢?
- 一个显然的可行解是，把每个叶子都往根节点连一条边
- 一个更优的解是，把“相邻的叶子节点”之间连边即可
- 有没有特殊情况?
- 若只有一个叶子节点，需要把它与叶子节点连边
- 综上，需要连边的点都是树上度数为 1 的点
- 设这样的点数目为 x，则答案为 $(x+1)/2$

例题 VII

- 给出一张 n 个点、m 条边的无向图（不一定连通）。现要求给所有的边定向，并添加最少的有向边，使整张图变成强连通的
- $n \leq 4 \times 10^5$，$m \leq 4 \times 10^5$

例题 VII / UVa 10972

Solution

- 定理：所有 BCC/eBCC 总存在一种边定向方式，使得其成为一个 SCC/eBCC
- 思考原因

例题 VII / UVa 10972

Solution

- 定理：所有 BCC/eBCC 总存在一种边定向方式，使得其成为一个 SCC/eBCC
- 思考原因
  - DFS 树上，回边箭头向上，树边箭头向下即可
- 问题变得和上一题几乎相同，eBCC 缩点后统计答案
- 考虑如何处理孤立点

例题 VII / UVa 10972

Solution

- 定理：所有 BCC/eBCC 总存在一种边定向方式，使得其成为一个 SCC/eBCC
- 思考原因
  - DFS 树上，回边箭头向上，树边箭头向下即可
- 问题变得和上一题几乎相同，eBCC 缩点后统计答案
- 考虑如何处理孤立点
- 设孤立点数目为 y，度数为 1 的点数目为 x，则答案为 $(x+2y+1)/2$
- 特殊情况?

例题 VII / UVa 10972

Solution

- 定理：所有 BCC/eBCC 总存在一种边定向方式，使得其成为一个 SCC/eBCC
- 思考原因
  - DFS 树上，回边箭头向上，树边箭头向下即可
- 问题变得和上一题几乎相同，eBCC 缩点后统计答案
- 考虑如何处理孤立点
- 设孤立点数目为 y，度数为 1 的点数目为 x，则答案为 $(x+2y+1)/2$
- 特殊情况?
  - 原图就是一个 eBCC，特判答案为 0

例题 VIII

- 给出一棵 n 个点的树，根节点（1 号点）是首都。
- 某年，首都爆发了传染病。为了控制疫情，不让其扩散到叶子节点，决定动用军队在一些节点建立检查点，使得从首都节点到叶子节点的每一条路径上都至少有一个检查点。叶子节点也可以建立检查点，但首都不可以。
- 一些节点中已经驻扎有军队（共 m 支），且一个节点可以驻扎多个军队。一支军队可以在有道路连接的节点间移动，并在除首都以外的任意一个节点建立检查点，且只能在一个城市建立检查点。不同的军队可以同时移动。
- 一支军队经过一条道路从一个城市移动到另一个城市所需要的时间等于道路的长度（单位为小时）
- 给出每支军队的原驻扎地点和各边的边权 w_i，问最少需要多少个小时才能控制疫情
- 对于 100% 的数据，$2 \leq m \leq n \leq 50000$ $0 < w_i < 10^9$

例题 VIII / NOIP2012 TG day2 T3

Solution

- 贪心：尽量往上走
- 最优化问题转化为判定性问题：二分答案 x，判断是否可以在 x 个小时之内控制疫情

例题 VIII / NOIP2012 TG day2 T3

Solution

- 贪心：尽量往上走
- 最优化问题转化为判定性问题：二分答案 x，判断是否可以在 x 个小时之内控制疫情
  - 对于 x 小时内走不到根节点的军队，走到哪是哪。处理出未被完全覆盖的儿子节点
  - 于是我们剩下一些“走到了根节点的军队”和一些“未被覆盖的儿子节点”
  - 根据每支军队的剩余时间和根节点到各未覆盖儿子的距离，贪心地分配
  - 注意，分配到原本经过的儿子节点的军队，不用再到根节点走一次

例题 IX

- GL 国有 n 个星球，还有 $n-1$ 条双向航道，每条航道建立在两个星球之间，这 $n-1$ 条航道连通了 GL 国的所有星球
- RXZ 掌管着一家物流公司，该公司有很多运输计划，每个运输计划形如：有一艘飞船需要从 u_i 号星球沿最快的宇航路径飞行到 v_i 号星球去。显然，飞船驶过一条航道是需要时间的，对于航道 j，任意飞船驶过它所花费的时间为 t_j，并且任意两艘飞船之间不会产生任何干扰
- 为了鼓励科技创新，GL 国国王同意 RXZ 的物流公司参与 GL 国的航道建设，即允许 RXZ 把某一条航道改造成虫洞，飞船驶过虫洞不消耗时间

例题 X

- 在虫洞的建设完成前 RXZ 的物流公司就预接了 m 个运输计划。在虫洞建设完成后，这 m 个运输计划会同时开始，所有飞船一起出发。当这 m 个运输计划都完成时，RXZ 的物流公司的阶段性工作就完成了
- 如果 RXZ 可以自由选择将哪一条航道改造成虫洞，试求出 RXZ 的物流公司完成阶段性工作所需要的最短时间是多少?

例题 XI

测试点编号	n=	m=	约定
1	100	1	
2		100	第 i 条航道连接 i 号星球与 i+1 号星球
3			
4	2000	1	
5	1000	1000	第 i 条航道连接 i 号星球与 i+1 号星球
6	2000	2000	
7	3000	3000	
8	1000	1000	
9	2000	2000	
10	3000	3000	
11	80000	1	
12	100000		
13	70000	70000	第 i 条航道连接 i 号星球与 i+1 号星球
14	80000	80000	
15	90000	90000	
16	100000	100000	
17	80000	80000	
18	90000	90000	
19	100000	100000	
20	300000	300000	
所有数据			$1 \le a_i, b_i, u_j, v_j \le n, 0 \le t_i \le 1000$

例题 XI / NOIP2015 day2 T3

Solution

- 一句话题意：一棵树，要求将一条边的权值变为 0，使得所有询问的两点间路径的最大长度最小

例题 XI / NOIP2015 day2 T3

Solution

- 一句话题意：一棵树，要求将一条边的权值变为 0，使得所有询问的两点间路径的最大长度最小
- “最大的最小”、“最小的最大”，我们常二分答案 x，转化为判定性问题
- 对于原本路径长就不超过 x 的路径，我们不用管
- 长度超过 x 的路径，改造的虫洞必须都要顾及到贪心地改造虫洞
- 即，虫洞必须在这些路径的交集上；更进一步，虫洞一定是交集上最大的边
- 问题转化为求树上一堆路径的交集上的最大边

例题 XI / NOIP2015 day2 T3

Solution

- 一句话题意：一棵树，要求将一条边的权值变为 0，使得所有询问的两点间路径的最大长度最小
- “最大的最小”、“最小的最大”，我们常二分答案 x，转化为判定性问题
- 对于原本路径长就不超过 x 的路径，我们不用管
- 长度超过 x 的路径，改造的虫洞必须都要顾及到贪心地改造虫洞
- 即，虫洞必须在这些路径的交集上；更进一步，虫洞一定是交集上最大的边
- 问题转化为求树上一堆路径的交集上的最大边
- “借教室”

EOF

- 谢谢大家 · ω ·