由一道题目浅谈——

图论的基本思想及方法

湖南省长郡中学
任恺

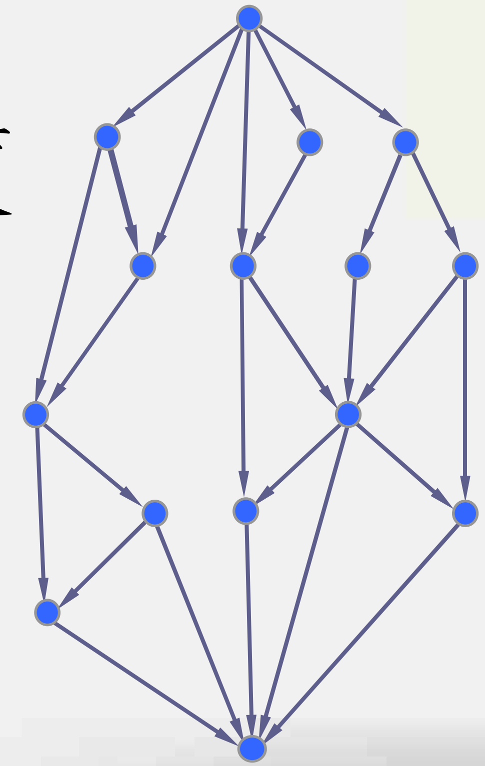
概述

- 信息学中的图论问题层出不穷，变化多端，惟有掌握其基本思想和方法，才能以不变应万变！
- 下面通过实例主要从两方面论述图论的基本思想：
- 一、合理选择图论模型
- 二、充分挖掘和利用图的性质

• 例题：滑雪者 (POI 2002)

• 雪山上有一个滑雪场。滑雪场由[illegible]站点和滑道组成。[illegible]最高点。每[illegible]一个最低点。[illegible]连接着两[illegible]可以[illegible]点到较低[illegible]

• 问题：[illegible]要检查滑道。每个工人[illegible]不能检查完所有的滑道吧？[illegible]路径上的滑道便都被检查。[illegible]工人[illegible]能滑行[illegible]的滑行路径可以有相同的滑道。



- 例题：滑雪者 **(POI 2002)**

Nar.in

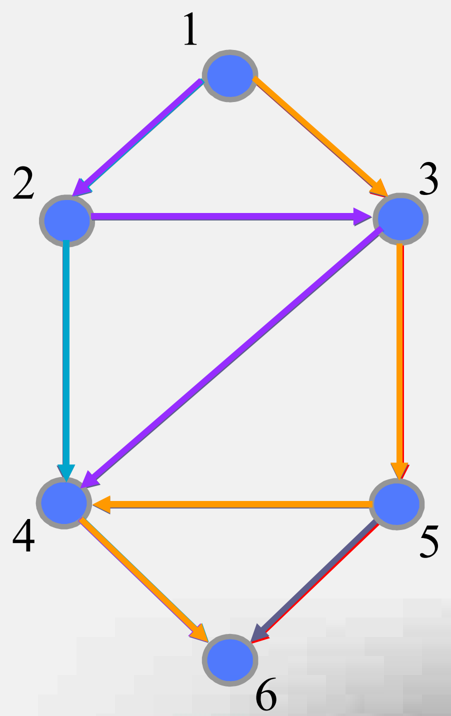
6
2 2 3
2 3 4
2 4 5
1 6
2 4 6

Nar.out

4

顶点个数 ***n*** **(1≤*n*≤5000)**

从左到右描述第 ***i*** 个顶点发出边的另一个端点

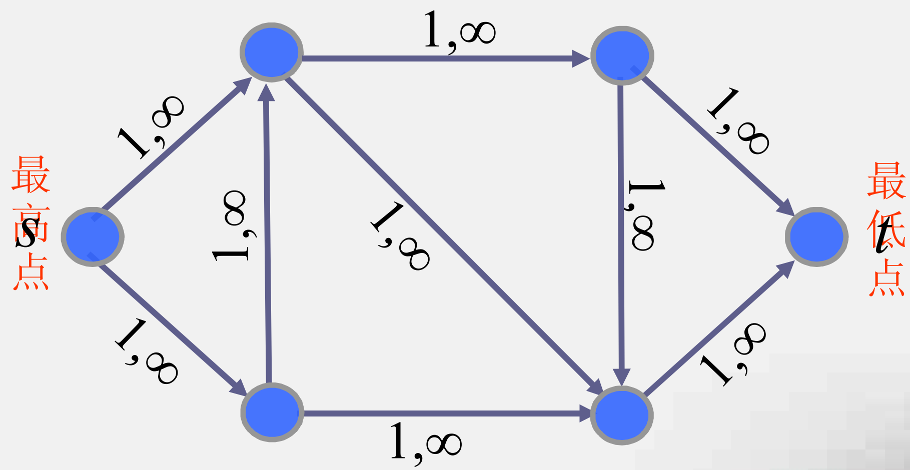


选择模型 (1)—— 网络流模型

分析模型，发现问题中有如下关键点：

很容易想到建立一个网络流模型

- 最高点 —— 网络的源点 *s*
- 最低点 —— 网络的汇点 *t*
- 每条滑道可以多次通过 —— 边上界 *u* 为 ∞
- 每条滑道必须被检查 —— 边下界 *l* 为 1
- 有向无环图



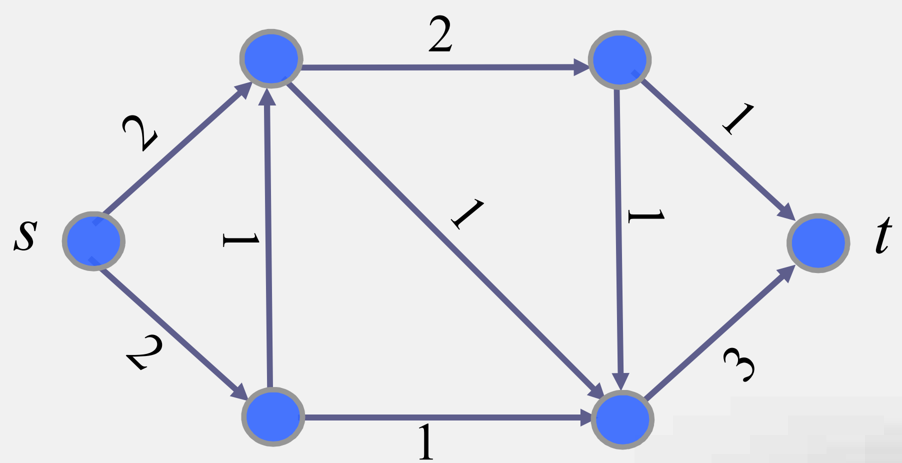
确定所求目标

建立模型后，可以确定要求的目标：

求图 ***G*** 中一个最小可行流，满足：

a) 每条边的流量大于等于下界

b) 从源点流出的总流量最小



求最小流的方法

如何求图 G 的最小流呢

求最小流可以分成两步：

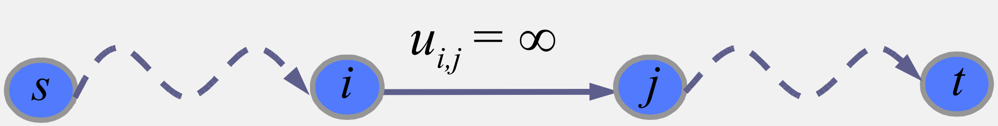
1）求出图 G 上的可行流 f

2）将可行流 f 转化成最小流

求可行流的方法

- 对于有上下界的网络，通常用构造附加网络的方法求可行流。
- 但是观察图 *G* 可以发现，边的上界都是无穷大，也就是说没有流量上限。

因此可以利用这个性质构造可行流

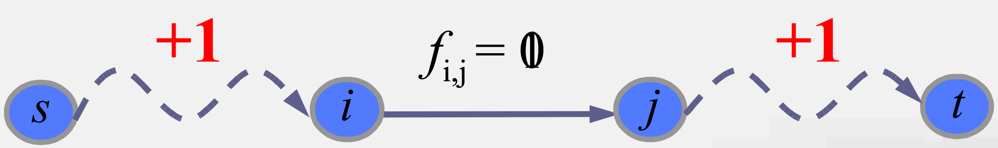


求可行流的方法

枚举每条流量为 0 的边，设为 (i, j)

任意找到一条从 s 到 i 的路径和一条从 j 到 t 的路径

那么 $s-i-j-t$ 便是一条可行的流，将这条路径上的边流量加 1， 便满足了边 (i, j) 的容量下界



求最小流

- 设用上法求出的可行流的总流量为 ***F***，这个可行流可能“过剩”。
- 因此要将多余的流从汇点“退回”到源点。

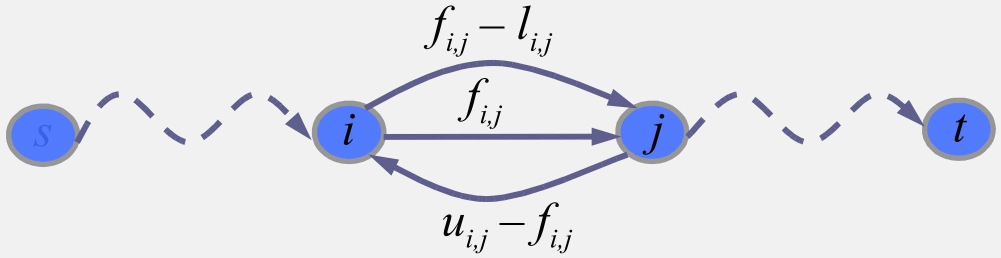
求最小流

回退“过剩”流量可以用如下方法：

•在新图中，原图 G 的边 (i, j) 拆成两条边：

边 (i, j), $u'_{i,j} = f_{i,j} - l_{i,j}$，$l'_{i,j} = 0$

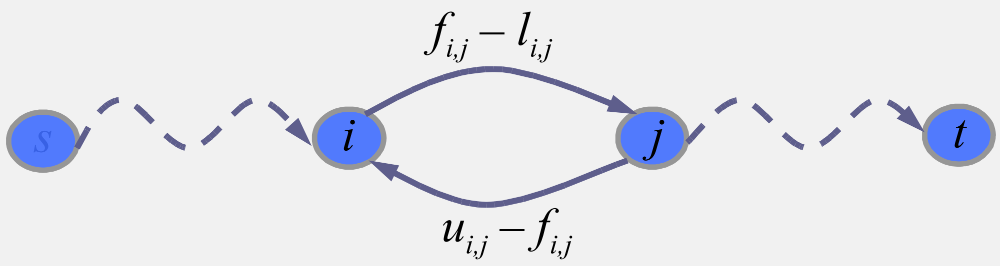
边 (j, i), $u'_{j,i} = u_{i,j} - f_{i,j}$，$l'_{j,i} = 0$



求最小流

•在新图中，从 t 到 s 求出一个最大流，令这个最大流的总流量为 F'。

•可得图 G 的最小流的流量为 $F - F'$。



算法一的复杂度

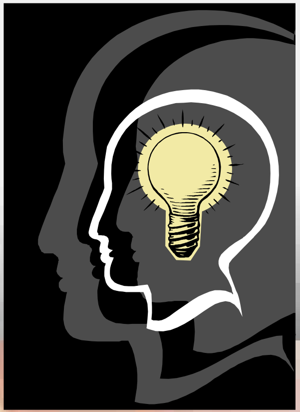
- 易知构造可行流的时间复杂度为 $O(nm)$
- 修改可行流所用的最大流算法时间复杂度为 $O(mC)$，其中 C 为增广的次数。
- 由于图 G 是平面图，所以边数是 $O(n)$ 级别。而由可行流构造方法可知，增广次数 C 也是 $O(n)$ 级别。

总的复杂度为 $O(n^2)$

选择模型 (2)—— 另辟蹊径的偏序集

- 算法一能够很迅速的解决原题数据。
- 但当 *n* 的范围扩大时，算法一便无能为力了。
- 是否存在效率更高的算法？

下面介绍的偏序集模型将更好的解决此问题。



偏序集的定义

偏序集是一个集合 P 和一个偏序关系 $\leqslant$ ，满足下列性质：

- 自反性：
  所有 $x \in P$，都有 $x \leq x$ 。
- 反对称性：
  所有 $x,y \in P$，若 $x \leq y$ 且 $y \leq x$，则 $x = y$。
- 传递性：
  所有 $x,y,z \in P$，若 $x \leq y$ 且 $y \leq z$，则 $x \leq z$。

偏序集的相关概念

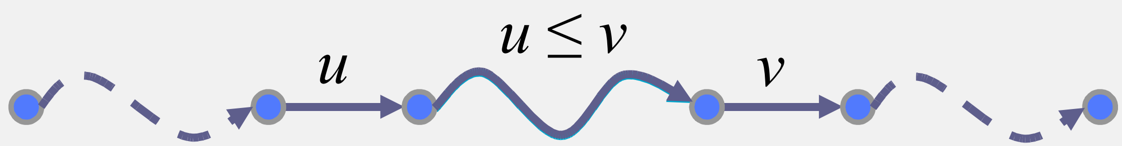
- 对于属于 P 的两个元素 x、y，若有 $x \leq y$ 或 $y \leq x$，则 x 和 y 被称作是可比的，否则被称为不可比的。
- 链：链是 P 的一个子集 C，在偏序关系≤下，它的每一对元素都是可比的。
- 反链：反链是 P 的一个子集 A，在偏序关系≤下，它的每一对元素都是不可比的。

问题的偏序集模型

图 G 可以定义成一个偏序集 E：

- E 中的元素是图 G 中的边；
- E 中的偏序关系：

  对于边 $u,v \in E$, $u \le v$ 当且仅当 $u = v$ 或图 G 中存在 v 到 u 的一条路径。



问题的偏序集模型

可以发现，图 G 中从最高点到最低点的路径对应了 E 的一个链。

因此，原问题可以重新用偏序集语言表述为 ：

将偏序集（ $E,\leq$ ）划分成最少的链，使得这些链的并集包含所有 E 中的元素。

目标的转化

- 直接计算链的最少个数

  —— 与网络流没有差别

- 唯有

  —— 继续转化目标

目标的转化

- 链和反链的计数满足下列关系：

Dilworth 定理 令 $(E, \leq)$ 是一个有限偏序集，并令 L_A 是 E 中最大反链的大小，S_C 是将 E 划分成最少的链的个数。在 E 中，有 $L_A = S_C$。

- 目标最终转化为：

求 E 中最长反链的大小

求最长的反链

由偏序集 ***E*** 的定义可以知道：

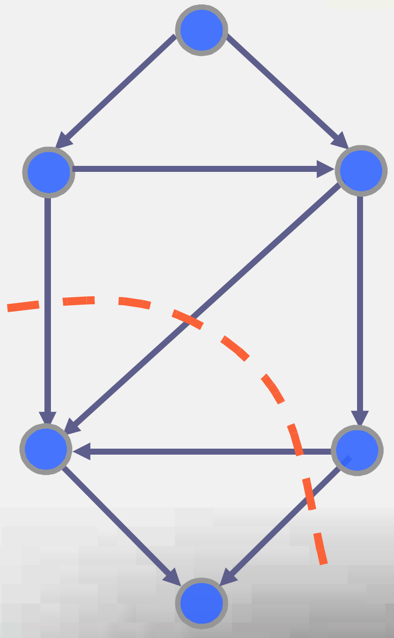
偏序集 ***E*** 中的反链对应着图 ***G*** 中的一些边，其中任意两条边之间都不能互达。

右图橙色线段便是样例的最长反链

这些线段满足如下性质：

如果用一条线将最长反链所对应的边从左到右连起来

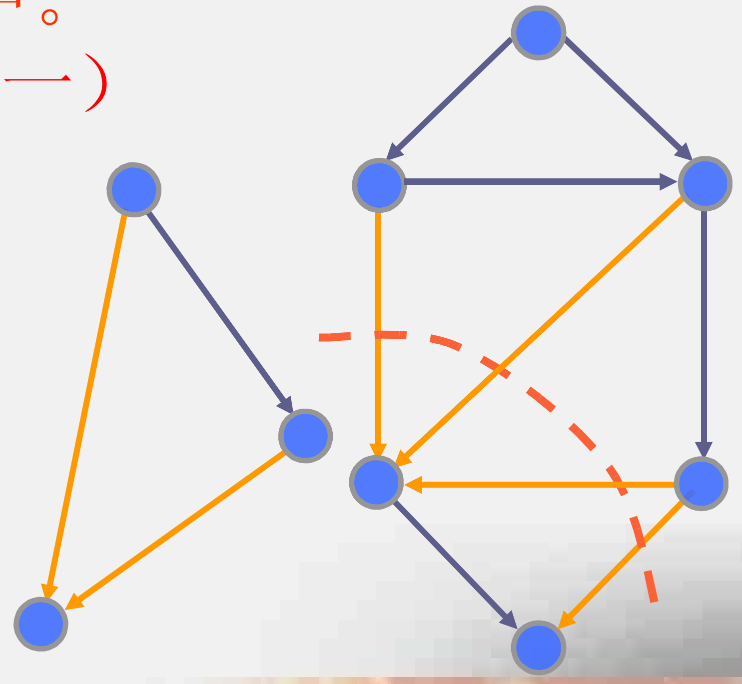
那么这条线不会与平面图中的其它边相交 ！



求最长的反链

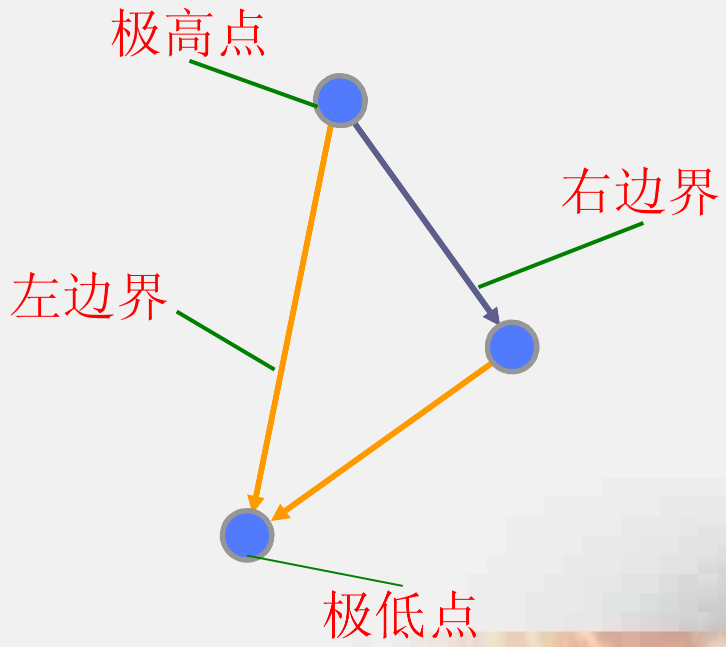
换句话说，将最长反链所对应的边从左到右排列好，相邻的两条边一定是在同一个域（闭曲面）中。

（结论一）



求最长的反链

所谓域，是指由从极高点到极低点的两条独立路径围成的一个曲面，在这个曲面里没有其他的点和边。



求最长的反链

由结论一可以用递推方法计算最长反链：

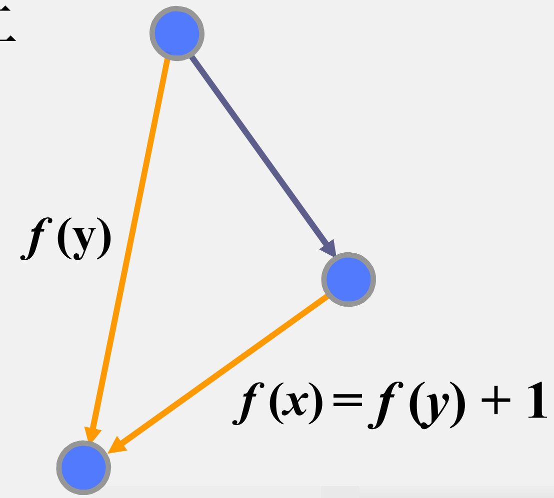
令 $f(x)$ 表示图 G 中在边 x 左边的平面区域中以 x 结尾的最长反链的长度。

设 x 在某个域 F 的右边界上，有递推式：

$$f(x) = \max\{f(y)\} + 1$$

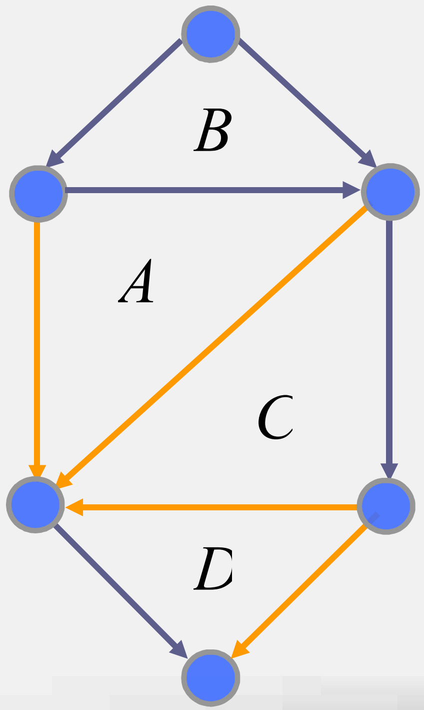
(y 属于 F 的左边界）

递推式 (1)



求最长的反链

因此只需要将所有的域求出来，然后按照一定的顺序，在每个域上运用递推式 **(1)** 求解每条边的 ***f*** 函数。



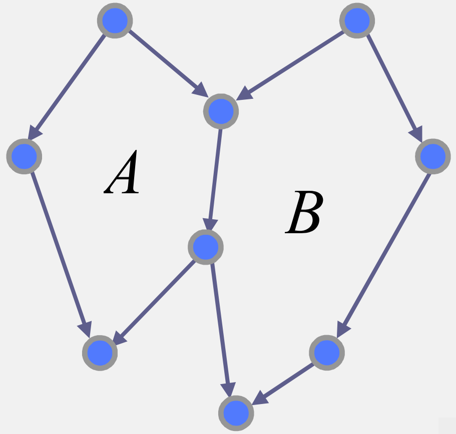
递推的顺序

一定的顺序　　如何确定递推的顺序呢？

一个域能够进行递推的前提条件

——它左边界上的边的 f 函数都已经求出

以此可以确定递推顺序：若域 B 左边界上的某条边在域 A 的右边界上，则 A 一定先于 B 进行递推。



$A \xrightarrow{\text{先于}} B$

算法的选择

找到了递推关系，接下来只需要选择合适的算法求出图 ***G*** 中所有的域来进行递推。

注意到，题目中的输入文件格式满足：

对于任意顶点，和它相邻的点已经从左到右排好序。

因此很容易想到一个方法，能够按照递推顺序找到所有的域！

DFS 深度优先遍历

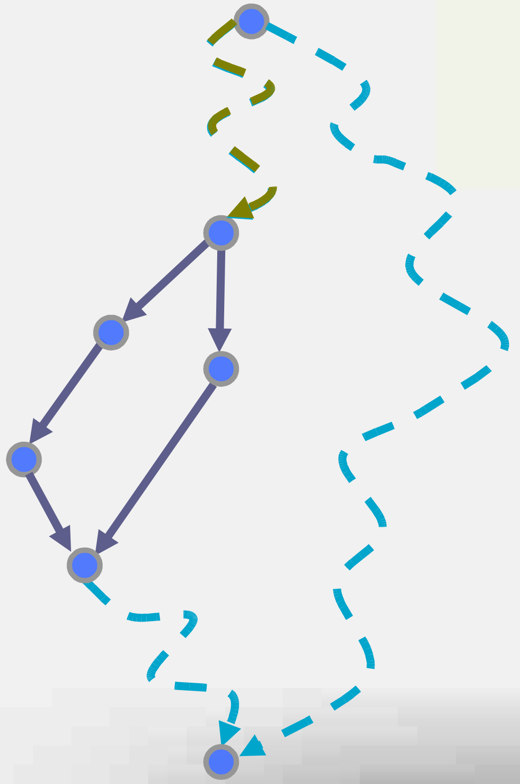
算法设计—— DFS

对图 *G* 进行深度优先遍历，图 *G* 中的顶点在遍历中有三种状态：

(1　一开始，所有点都处于未遍历的状态。

(2　当遍历到一个点，但没有检查完它发出的边时，标记这个点为未扩展完的状态。

(3　当一个点发出的边都被检查完时，这个点标记为扩展完毕状态。



算法设计—— DFS

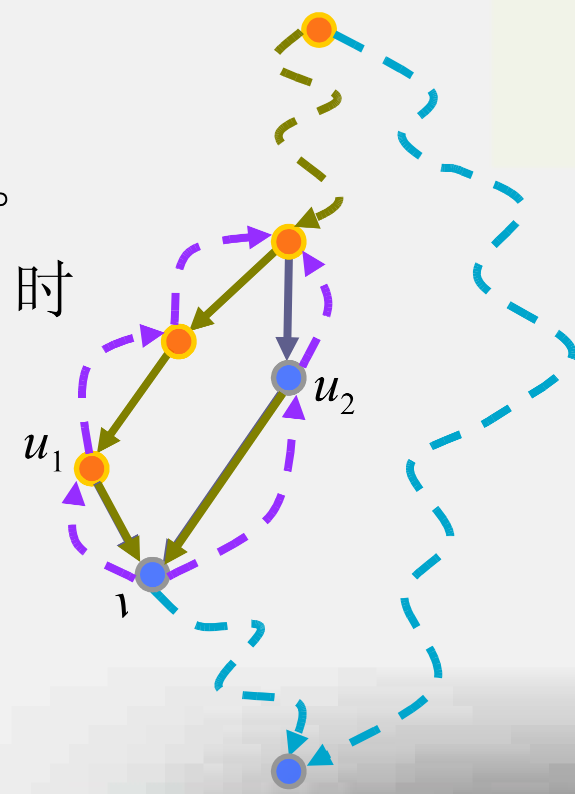
在遍历中用一个指针 pre[*v*] 记录 *v* 最新的前驱结点。

指针 pre 的更新方式如下：

当 u_1 扩展到 *v* 时 pre[*v*] = u_1。

后来检查 u_2 发出的边 (u_2, *v*) 时虽然 *v* 已经扩展完毕，但仍更新 pre[*v*]：

pre[*v*] = u_2。



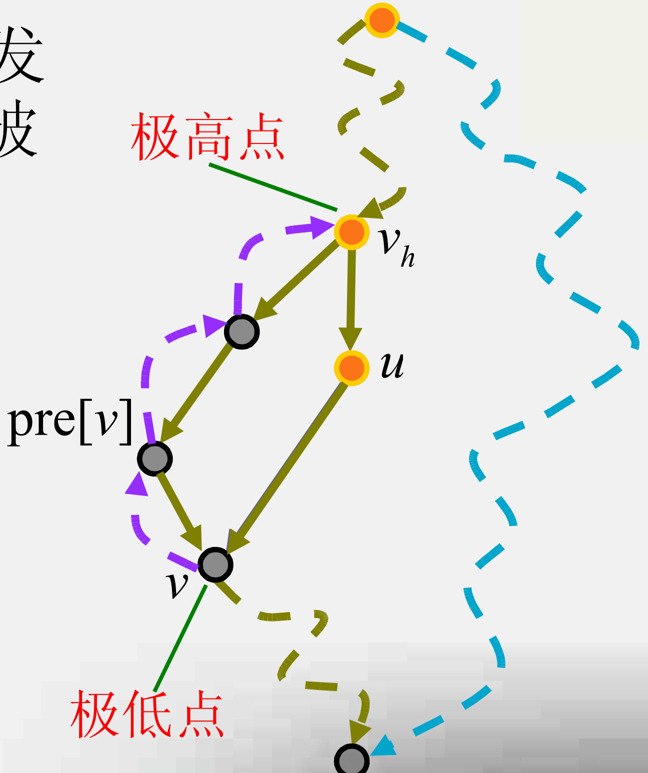
算法设计—— DFS

根据 **DFS** 中点的状态和指针 **pre** 就可以按如下方法确定图 ***G*** 中的域：

当检查点 ***u*** 的某条边时，发现边的另一个顶点 ***v*** 已经被扩展完毕。

可知，点 ***v*** 一定是边 **(*u*, *v*)** 所在域的极低点。

而 **pre[*v*]** 和 ***u*** 最近公共祖先点一定是域的极高点。

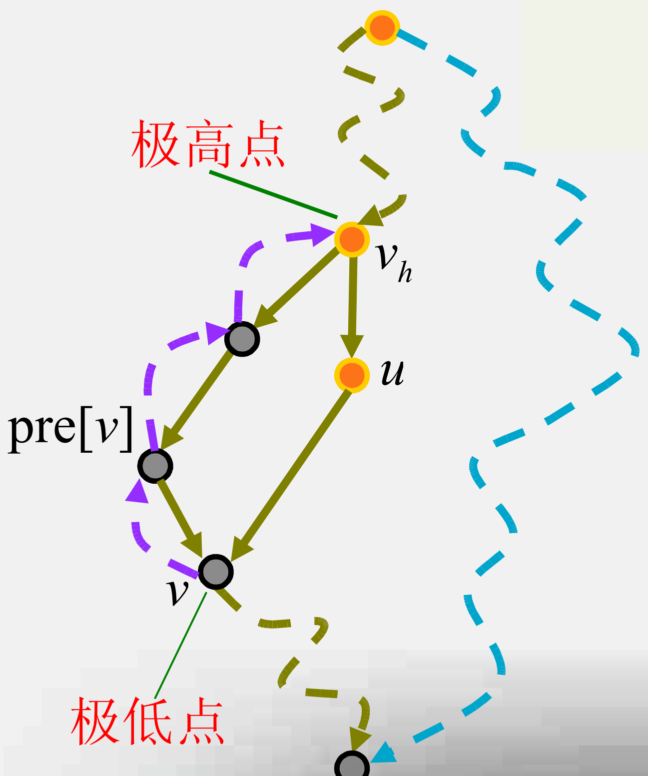


算法设计—— DFS

寻找 **pre[*v*]** 和 ***u*** 的最近公共祖先，只需要利用 **pre** 回溯寻找 ***v*** 的祖先，第一个未被扩展完毕的祖先便是域的极高点。

从 ***v*** 到 **pre[*v*]** 再回溯到 $\boldsymbol{v_h}$ 的路径便是域的左边界。

从极高点到 ***u*** 再到 ***v*** 的路径便是域的右边界。



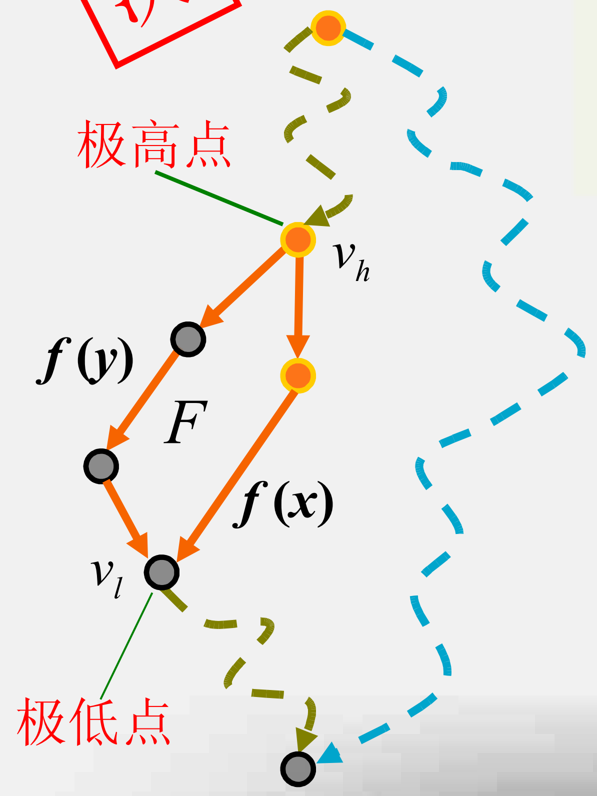
算法设计—— DFS

问题解决

找到域之后，域左边界上的边都被遍历过，***f*** 函数都已经求出。

$$\boldsymbol{f(x) = \max\{ f(y) \} + 1}$$

可见， **DFS** 寻找图 ***G*** 的域的同时，已经完成 ***f*** 函数的求解。



算法二的复杂度

- 对每一个点进行 **DFS** 遍历，复杂度为 $O(|E|)$ ；
- 回溯寻找每个域边界上的边，并且进行递推求解。由于是平面图，每条边最多属于两个不同域的边界，因此这一步的复杂度为 $O(|E|+|F|)$ 。
- 因为原题给出的图是平面图，根据欧拉定理，边数 $|E|$ 和域数 $|F|$ 都是 $O(n)$ 级别的。

总的复杂度为 $O(n)$

总结

- 算法一直接根据题目描述建立了网络流模型，体现了原题的网络有向无环图特性。

没有利用图 **G** 平面图的性质

算法一的效率不是最优

解法具有一般性，适用任何有向无环图

直接从定义下手的思考方式值得借鉴

总结

- 算法二很好的利用偏序集模型实现了问题目标的转变，从原来的网络流问题回归到问题本身的平面图上，完整的揭示了问题的本质。

两个算法思考历程的共同点

实际问题

设计相应的算法

寻找合适的图论模型

以图论模型为平台挖掘问题的性质

解决问题

结语

“模型”

图论基本思想的精华

解决图论问题的关键

建立模型

熟练掌握经典模型

勇于探索，大胆创新

挖掘利用模型性质

独具慧眼

一击中的

谢谢

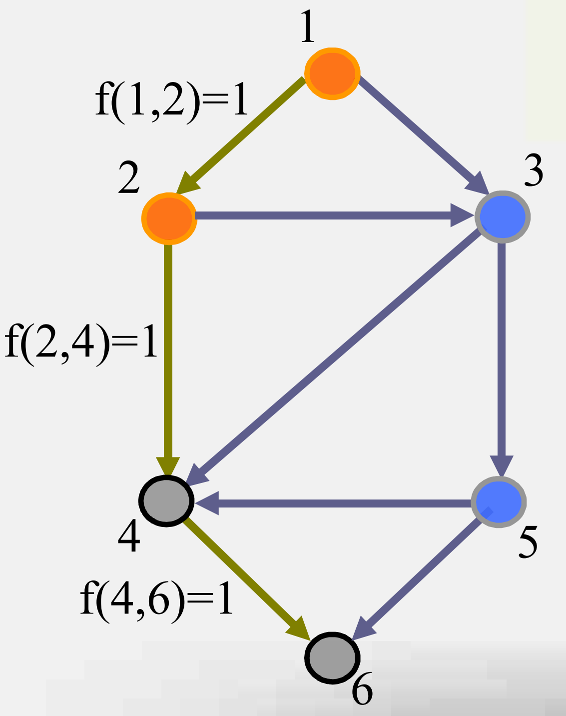
样例的模拟

下面在样例上模拟运行算法二，说明算法二是如何执行的：

从结点 1 开始遍历

一直深搜到结点 6

可知 (1,2), (2,4) 和 (4,6) 为最靠左的边，所以 f 值为 1

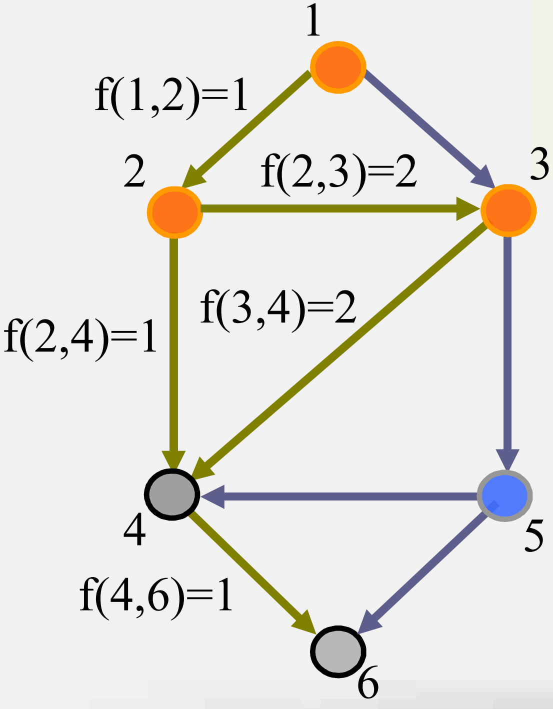


样例的模拟

回溯到 2，扩展结点
3
扩展结点 4，发现 4
已经被扩展。根据前
驱指针找到域 *A* 。

进行递推：
$f(2, 3) = f(2, 4) + 1 = 2$
$f(3, 4) = f(2, 4) + 1 = 2$

将 4 的前驱指针指向
3

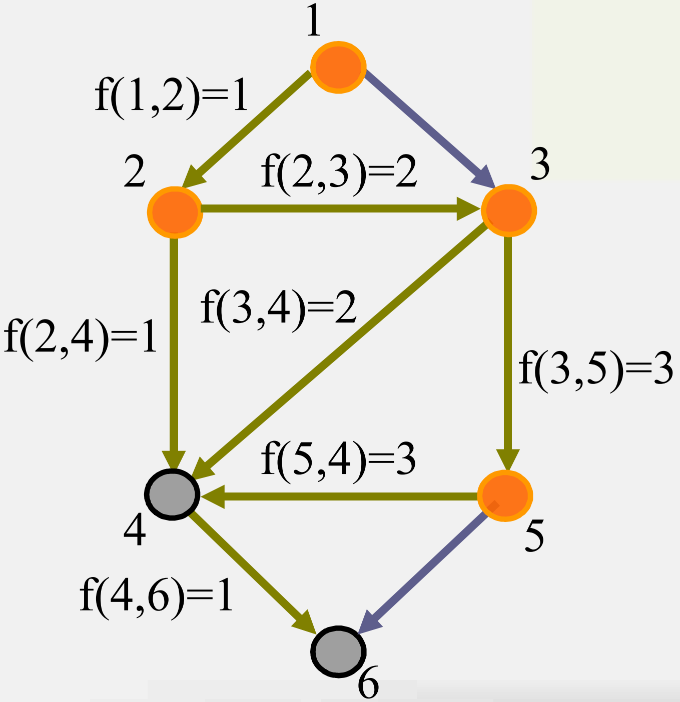


样例的模拟

回溯到 3， 扩展结点
5
扩展结点 4， 发现 4
已经被扩展。根据前
驱指针找到域 A 。

进行递推：
$f(3, 5) = f(3, 4) + 1 = 3$
$f(5, 4) = f(3, 4) + 1 = 3$

将 4 的前驱指针指向
5



样例的模拟

扩展结点 6， 发现 6 已经被扩展。根据前驱指针找到域 C。

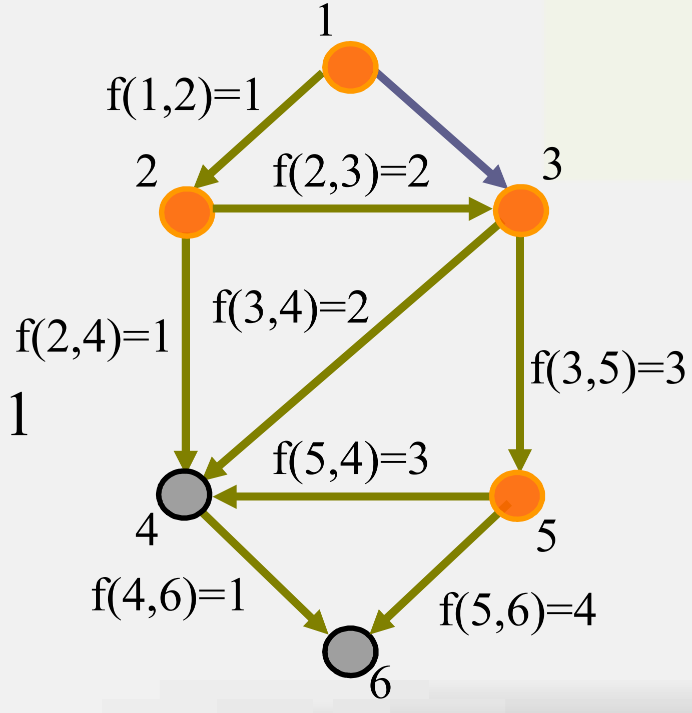
进行递推：

$f(5, 6)$

$= \max\{f(4, 5),\ f(4,6)\} + 1$

$= 4$

将 6 的前驱指针指向 5



样例的模拟

回溯到 1

扩展结点 3， 发现 3 已经被扩展。根据前驱指针找到域 D 。

进行递推：

$f(1,3)$

$= \max\{f(1,2), f(2,3)\} + 1$

$= 3$