# 图论

wjh

最小生成树的应用

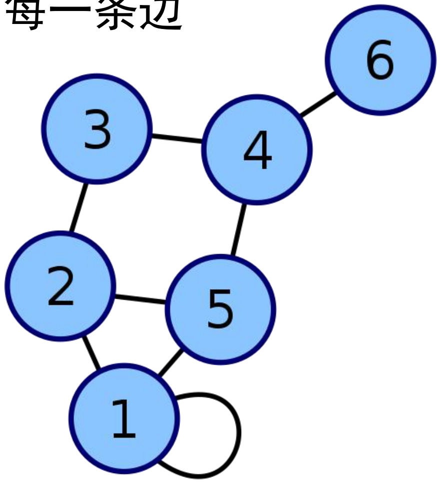
最短路的应用

有向图的强联通分量

无向图的双联通分量

# 图的概念

- 一张图G是一个二元组(V,E)，其中V称为顶点集，E称为边集
- 边集E的元素是二元组数对，用(x,y)表示，其中x,y ∈ V，代表有一条从x到y的边
- 有向图：边有方向，(x,y)和(y,x)不表示一条边
- 无向图：边无方向，(x,y)和(y,x)表示一条边，实际编程中常常认为每一条边都是两条有向边
- 重边：两点之间有多条边，有的题目会保证无重边
- 自环：边(x,x)，如右图点1处有一个自环，有的题目会保证无自环
- 边权：每条边有一个权值c，常常代表距离或者费用



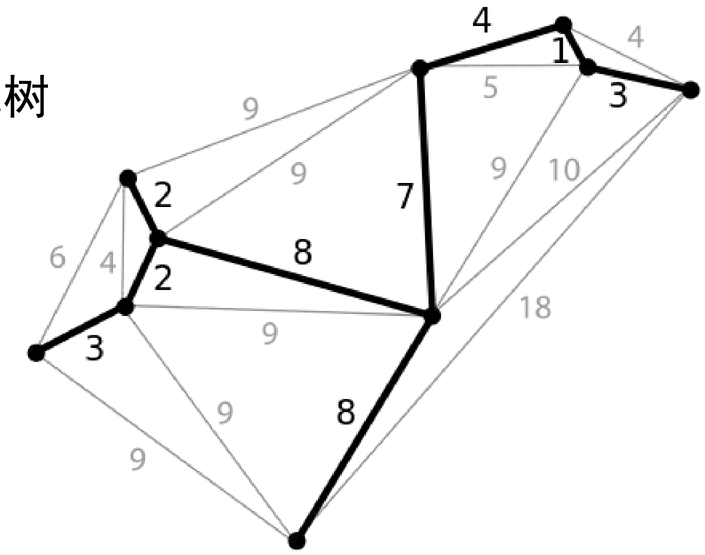

- OI中不需要拘泥于严格的定义，理解即可

# 名词解释

- 连通图：任意两点间存在路径
- 树:以下定义等价
  - 任意两点间只存在一条路径的图
  - 无环的连通图
  - n个点n-1条边的连通图
  - n个点n-1条边,无环的图
- 子图：图G'称作图G的子图如果V(G') ⊆ V(G)以及E(G') ⊆ E(G)。
  也就是从原图中选出一些点以及和一些边组成的新的图
- 生成子图：指满足条件V(G')=V(G)的G的子图G'。
  也就是选出所有的点和一些边组成的新的图，生成树则是指子图G'是一颗树

# 名词解释

- 最小生成树：对于带权图，权值和最小的生成树
- 最小瓶颈生成树：对于带权图，最大权值最小的生成树
- 最小生成树一定是最小瓶颈生成树



# 最小生成树问题

- Prim算法：
  - 从一个点开始建树，每次将连接树外和树里最小的边加入树
  - 稀疏图可以使用堆优化，类似Dijkstra。
  - 时间复杂度：$O(n^2)/O(mlogn)$
- Kruskal算法：
  - 最开始是n棵树的森林，每次选最小的边将两颗树连起来，使用并查集判断维护连通
  - 时间复杂度：$O(mlogn)$

- 后者好写一些

# 公路修建 P1265

- 给出平面上的n(≤5000)个点， 点与点之间可以连边，距离为欧几里得距离，求最小生成树。

- 如果直接建图，边数$m=O(n^2)$，此时堆优化的Prim效率低于朴素实现。
- 不需要显式保存邻接矩阵，每次枚举时计算距离即可

# [CSP-SJX2019]网格图 P5687

给定一个 $n \times m$ 的网格图，行从 $1 \sim n$ 编号，列从 $1 \sim m$ 编号，每个点可用它所在的行编号 $r$ 与所在的列编号 $c$ 表示为 $(r, c)$。

点 $(i, j)$ 与 $(i, j+1)$ 间连有一条权值为 $a_i$ 的边，其中 $1 \le i \le n, 1 \le j < m$。

点 $(i, j)$ 与 $(i+1, j)$ 间连有一条权值为 $b_j$ 的边，其中 $1 \le i < n, 1 \le j \le m$。

# 货车运输 P1967

- A 国有 n(≤10,000) 座城市，编号从 1 到 n，城市之间有 m(≤50,000) 条双向道路。每一条道路对车辆都有重量限制，简称限重。

- 现在有 q(≤30,000) 辆货车在运输货物， 司机们想知道每辆车在不超过车辆限重的情况下，最多能运多重的货物。

# 货车运输(NOIP2013) P1967

- 考虑两点间司机的移动路径，一定是最小权值最大的路径
- 看起来很像二分，但是多组询问

- 两点间路径唯一：树的定义之一
- 所以司机会走的边构成了一棵树
- 类似最小瓶颈生成树，本题走的边都在最大生成树上
- 然后问题转化为了多组询问，树上两点间路径最小值
- LCA时倍增维护即可

- 这道题思考方式是典型的，图->最小生成树->树上算法

# （严格）次小生成树 P4180

- 边权和大于最小生成树的，最小的一颗生成树

- 枚举没有被使用的边，这条边两个端点在树中有一条路径，而所枚举的边一定大于等于路径上的最大边。
- 这条边加入树后出现了一个环，现在需要从环上原来的路径里删去一条边。
- 删掉路径上的最大边，如果相等的话删去次大边，这样树的权值增加了。

# （严格）次小生成树 P4180

- 树中加入了这条边之后，目标仍是最小化边权和，考虑贪心的Kruskal算法。
- 最开始时加入了我们枚举的这条边，则Kruskal在执行过程中最开始加的一部分边不会受到影响。
- 但当原算法某一步加入MST上的边时，在次小生成树中原边两端已经联通，则会被跳过。
- 在接下来的算法流程中，连通性和MST的算法时一样，所以后面的边也不会受到影响。

- 结论：次小生成树和最小生成树只有一条边不同。

# 题目建模

- 如果只考最小生成树的话模型会比较简单，只要找到题目中哪些元素对应点，哪些费用对应边建图就可以了，可能用到拆点拆边之类的技巧
- 比较难的题目则是求最小生成树之后再做树上操作，最常见的是LCA/树剖后维护信息，但也可能需要树型DP，计数之类的算法。这种题目中求最小生成树是第一步，考虑到图上是否有多余信息，就会发现有用的信息构成了生成树。

# 最短路问题

- 求从s到t权值和最小的路径

- Floyd算法：
  - 多源最短路，求出所有点对的最短路长度
  - 时间复杂度：$O(n^3)$
- Dijkstra算法：
  - 单源最短路，求出某个点s到所有点的最短路长度
  - 时间复杂度：$O(n^2)$/$O(m\log n)$
  - 无法处理负权
- SPFA算法，即队列优化的Bellman-Ford算法：
  - 单源最短路，求出某个点s到所有点的最短路长度
  - 时间复杂度：声称为$O(m)$，最坏$O(nm)$，容易卡到最坏
  - 可以处理负权边，可以判断负权环

# 单源最短路

- 维护一个dis[MAXN]数组，dis[i]代表s到i的最短路径长度
- dis[s]=0,其他为INF

- 松弛操作：通过某条路径更新dis[v]的值
  - if (dis[v] > dis[u] + e.dist) dis[v] = dis[u] + e.dist
  - 尝试使用s到u的最短路加上边(u,v)的长度来更新s到v的最短路

# SPFA

- Bellman-Ford：对整张图进行n-1轮松弛，每次枚举每条边进行松弛。
- 第i轮松弛保证了边数i以内的最短路被更新，最后一定能得出最优解。
- SPFA：在上述过程中避免无意义的松弛
- 只有成功的松弛操作才会对终点产生影响，所以使用队列维护等待松弛的点，每次取出一个点进行松弛，对于所有松弛成功的点加入队列

- 判负环：加个入队计数器，某个点松弛了第n次，说明有边数为n的最短路，则最短路上有负环。
- 模板 P3385

# 灾后重建 P1119

## 题目背景

$B$地区在地震过后，所有村庄都造成了一定的损毁，而这场地震却没对公路造成什么影响。但是在村庄重建好之前，所有与未重建完成的村庄的公路均无法通车。换句话说，只有连接着两个重建完成的村庄的公路才能通车，只能到达重建完成的村庄。

## 题目描述

给出$B$地区的村庄数$N$，村庄编号从0到$N-1$，和所有$M$条公路的长度，公路是双向的。并给出第$i$个村庄重建完成的时间$t_i$，你可以认为是同时开始重建并在第$t_i$天重建完成，并且在当天即可通车。若$t_i$为0则说明地震未对此地区造成损坏，一开始就可以通车。之后有$Q$个询问$(x, y, t)$，对于每个询问你要回答在第$t$天，从村庄$x$到村庄

对于100%的数据，有$N \le 200$，$M \le N \times (N-1)/2$，$Q \le 50000$，所有输入数据涉及整数均不超过100000。

# Floyd算法

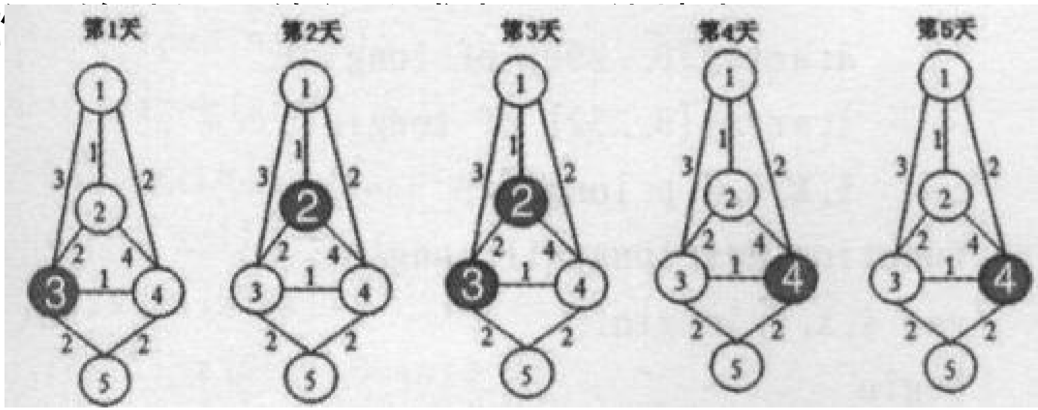
- 设d[i][j][k]为从i到j，仅通过编号为1-k的中间节点的最短路径距离
- d[i][j][k]=min(d[i][j][k-1], d[i][k][k-1] + d[k][j][k-1])
- 初始值d[i][j][0]为两点之间边权值，未连通为INF
- 从1到n枚举k，然后枚举(i, j)

- 为了方便可以不开第三维，在原地迭代，也就是我们的邻接矩阵上三重循环的floyd模板。

- 将询问离线处理，按时间顺序排列，floyd处理即可。

# 物流运输 P1772

- 物流公司要把一批货物从码头 A 运到码头 B。由于货物量比较大，需要 n 天才能运完。货物运输过程中一般要转停好几个码头。
- 物流公司通常会设计一条固定的运输路线，以便对整个运输过程实施严格的管理和跟踪。由于各种因素的存在，有的时候某个码头会无法装卸货物。这时候就必须修改运输路线，让货物能够按时到达目的地。
- 但是修改路线是—件十分麻烦的事情，会带来额外的成本（每次k）。因此物流公司希望能够订一个 n 天的



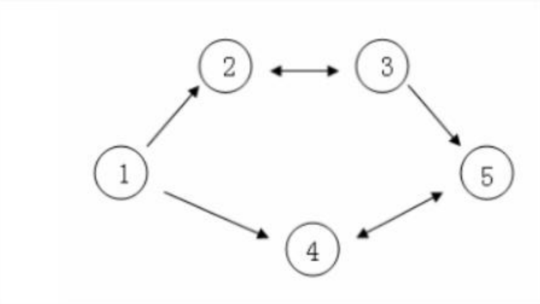

- $N \le 100$, $m \le 20$

# 物流运输 P1772

- 你需要将n天划分为若干个时间段，每个时间段内只能走一直开放的码头
- 划分过程使用dp转移，dp[i]表示完成前i天运输的最小花费
- 预处理出从第i天到第j天的最短路，直接转移即可
- 预处理过程直接跑$n^2$遍最短路即可。

# NOIP2009 P1073 最优贸易

- n(<=100000)个城市之间用单向边或双向边相连，现在要从1号点走到n号点，路径可以有回路
- 每个城市有一个水晶球售价，现在要在旅途中买卖一次，求最大的收益




假设 1 $n$ 号城市的水晶球价格分别为 $4, 3, 5, 6, 1$。

阿龙可以选择如下一条线路：1->2->3->5，并在 2号城市以3 的价格买入水晶球，在 3号城市以5的价格卖出水晶球，赚取的旅费数为 2。

阿龙也可以选择如下一条线路1->4->5->4->5，并在第1次到达5 号城市时以 1的价格买入水晶球，在第 2 次到达4 号城市时以6 的价格卖出水晶球，赚取的旅费数为5。

# 思路1

- 有费用的路径问题可以考虑最短路
- 分为三个阶段，起点到购买点、购买点到售卖点、售卖点到终点
- 使用分层图思想，将原图复制为三份对应三个阶段
- 从第1层图到第2层图对应购买，对于每个点从第1层到第2层连边，权值为在这个点购买的费用
- 从第2层图到第3层图对应售卖，同理连边，权值为负的费用
- 由于移动不需要费用，三层图内部的边权为0

- 之后求出来的最短路，层内部的对应移动，跨层的对应购买或者售卖操作

# 逛公园 P3953

- 策策同学特别喜欢逛公园。公园可以看成一张N(≤100,000)个点M(≤200,000)条边构成的有向图，且没有 自环和重边。其中1号点是公园的入口，N号点是公园的出口，每条边有一个非负权值， 代表策策经过这条边所要花的时间。
- 策策每天都会去逛公园，他总是从1号点进去，从N号点出来。
- 策策喜欢新鲜的事物，它不希望有两天逛公园的路线完全一样，同时策策还是一个 特别热爱学习的好孩子，它不希望每天在逛公园这件事上花费太多的时间。如果1号点 到N号点的最短路长为d，那么策策只会喜欢长度不超过d+K的路线。(K≤50)
- 策策同学想知道总共有多少条满足条件的路线，你能帮帮它吗?
- 为避免输出过大，答案对P取模。
- 如果有无穷多条合法的路线，请输出-1。

# 逛公园 P3953

- K比较小，考虑最短路并且记录K
- 因此建立K层分层图，代表最短路上额外花费了K时间
- 连边方式：首先跑一遍最短路，一条边的额外花费时间是w-(d[v]-d[u])，代表这条边是上升几层。
- 从起点到终点的一条路径（此时不必考虑权值）代表了最终的一条路径，其中每条边的额外花费时间相当于上升的层数。

- 然后考虑-1情况，是路径上出现了0权环。
- 如果一条边，起点到u的距离+w+v到终点的距离超过了d+K，则这条边不会被经过，可以不加入图。
- 这样子也不会加入路径外的0权，也就是所有0权均会被经过。
- 可以直接拓扑排序DAG DP计算方案数，而出现环一定是路径上的0权环，输出-1

# Legacy CF786B

- n个点m条边（≤100,000）
- 边有三种类型
  - –一点到一点
  - –一点到[l,r]内任意一个点
  - –[l,r]内任意一个点到一点
- 求s起点的单源最短路

# 区间最短路

- u -> [l,r]，长度为w
- 朴素建图：u到[l,r]的每个点都连长为w边
- 劣化建图：一个结点代表[l,r]区间，点向[l,r]内的点连0边，然后u向点连长为w边

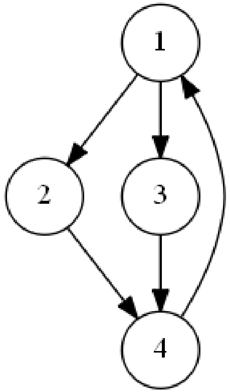
# 区间最短路

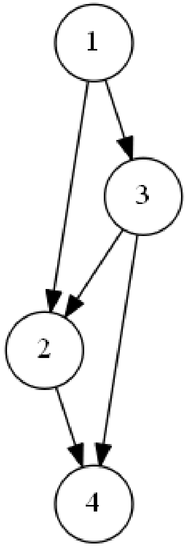
- 优化建图：对区间的分治划分可以使用线段树或者倍增（类似ST表）
- 每个结点代表一个区间，连向左右两个子区间。
- 用对应的方式选择出[l,r]区间代表的一些结点，如果用线段树式则是log个，用倍增式则是2个
- 然后u到这些点连边

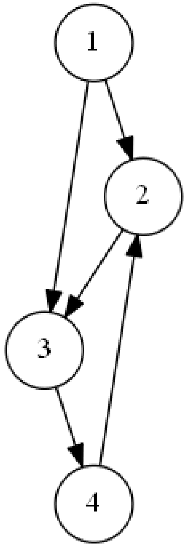
- 线段树：O(n)个点，O(mlogn)个额外边
- 倍增：O(nlogn)个点，O(m)个额外边

# 名词解释

- 强连通：有向图中，两个顶点至少存在一条路径
- 强连通图：每两个顶点都强连通的有向图
- 强连通分量（Strongly Connected Components）：有向图的极大强连通子图



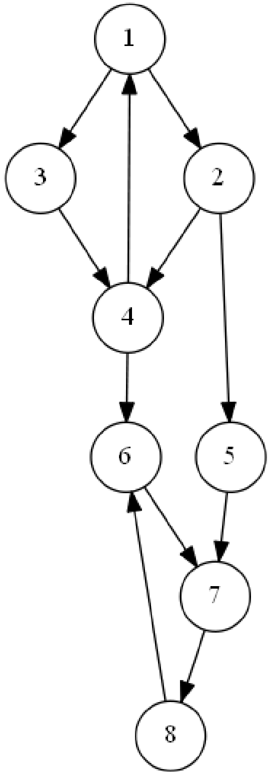



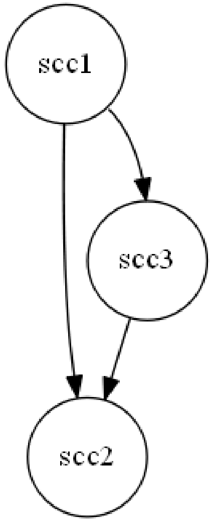



# 问题模型

- 对于一些存在依赖关系的模型，若其建图是一个DAG，则可以直接通过拓扑排序解决，但若其中有环则需要特殊处理
- 对于有环的问题，会出现一些互相依赖的关系，这些关系组成了一个强连通分量，根据题目要求的性质，对于这个强连通分量可以将其缩为一个点
- 将所有强连通分量缩成点后即可在DAG上求解

- 建模方式和DAG很相似，建出图不是DAG就先跑一边SCC缩点即可


1
3
2
4
6
5
7
8


scc1
scc3
scc2

# 有向图边的类型

- 使用DFS从任意节点遍历有向图时，可以得到DFS树
- 每一条边和DFS树的关系，可以分为以下四种

–树枝边：DFS树上的边，即指向未访问过节点的边

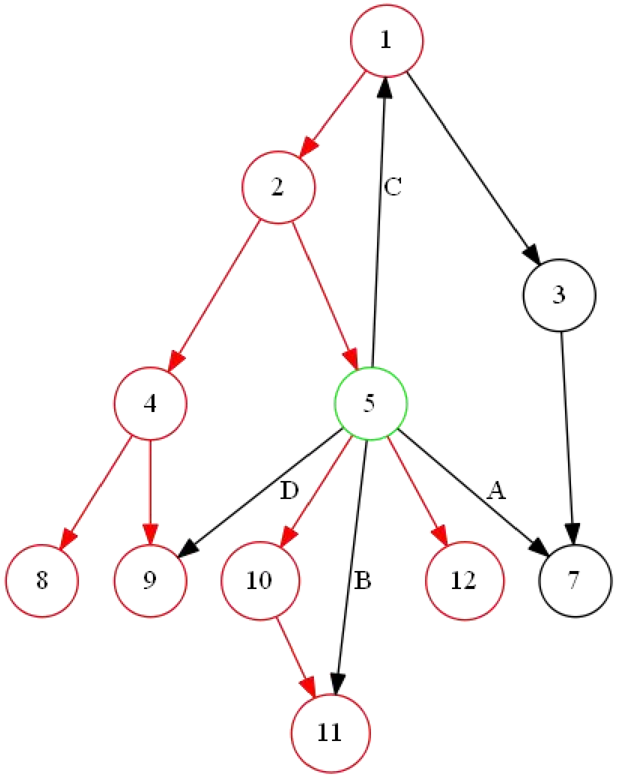
–前向边：指向DFS树中子树中节点的边

–后向边：指向DFS树中父亲的边

–横叉边：其他边，即指向DFS树中非子树的边

- 下面考虑如何判断这四种边

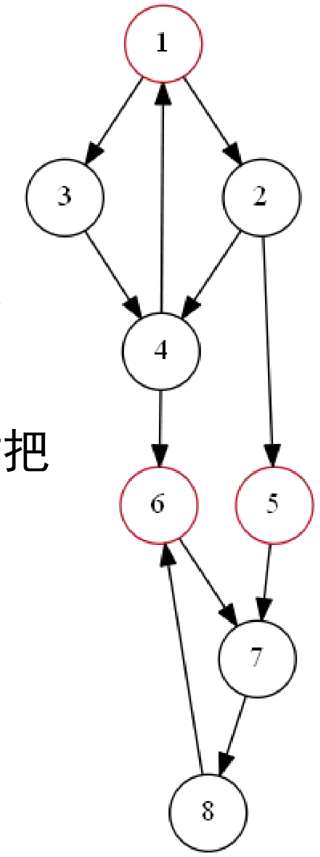
# 判定方式




- 红色标记为DFS树
- 记录dfs序
- A边终点未访问过，为树枝边
- B边终点已被访问过，且dfn[v]>dfn[u]，说明在子树中，说明为前向边
- C边终点已被访问过且不在子树中，终点在栈中则为后向边
- D边终点已被访问过且不在子树中且已经出栈，为横叉边

# Tarjan

- 在一个有向有环图上DFS，找出每一个强连通分量
- 考虑每一个强连通分量高度最低（离根近）的那个点
- 这些点将DFS树分割成了许多个子树，每个子树中的点组成了一个强连通分量
- 分割的方法是在dfs同时另外维护一个栈存放节点，离开分割点时把分割点往下的部分全部取出来就是一个强连通分量。
- 现在需要找到这些分割点



# low[]

- 维护一个数组low，low[u]代表点u所能到达子树中的，深度最小的点祖先的dfs序编号
- 初始low[u]=dfn[u]
- 对于边(u,v)

–若为树枝边，则用low[v]更新

–若为后向边，则用dfn[v]更新

–若为前向边，因为指向的点的信息已经通过树枝边传递过来，所以无需更新

–若为横叉边，则指向另一个强连通分量，无需更新

- 当low[u] == dfn[u]时，就是一个分割点

# 模板 P3387

## 题目背景

缩点+DP

## 题目描述

给定一个n个点m条边有向图，每个点有一个权值，求一条路径，使路径经过的点权值之和最大。你只需要求出这个权值和。

允许多次经过一条边或者一个点，但是，重复经过的点，权值只计算一次。

# 无向图的双连通性

- 对于无向图，定义双连通性

–点双连通：删去任何一个点仍然连通

–边双连通：删去任何一条边仍然连通

- 另一种定义是，任何两点之间至少存在两条不经过相同中间点（边）的路径

- 如果不满足双连通性

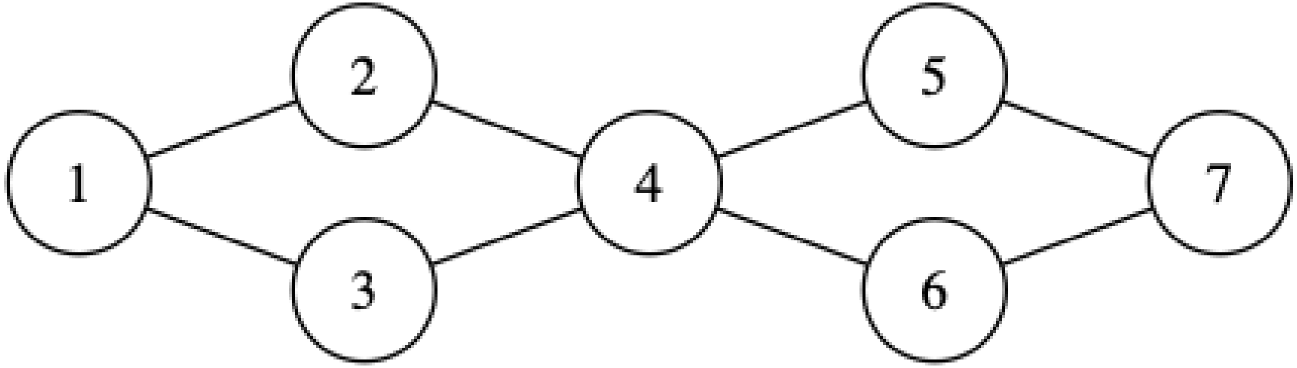
–割点（割顶）：删去后原图不连通的顶点集合

–割边（桥）：删去后原图不连通的边集合

- 满足点（边）双连通性的极大子图称为点（边）双连通分量

# 双连通性之间的关系

- 一张图的点双连通分量之间可能有公共点，边双连通分量之间不可能有公共点
- 点双连通性不满足传递性，边双连通性满足传递性
- 点双连通分量一定是边双连通的（无相同点的两条路径一定无相同边）



# 无向图的dfs树

- 前向边：指向DFS树中子树中节点的边
- 横叉边：其他边，即指向DFS树中非子树的边
- 假如存在这种边(u, v)，在v点被dfs访问时，(v, u)一定会被先枚举，则这条边应该是后向边/树枝边。

- 同样可以维护dfn[]与low[]数组，含义与有向图的dfs相同

# 割点和割边

- 割边(u, v)删去后变为两个连通块，v无法到达u前面的点，即

–low[v]>dfn[u]

- 割点u删去后会有至少一个子树中的点无法到达u前面的点，即

–存在至少一条树枝边(u, v) low[v]>=dfn[u]

–对于根结点需要特别判断，只要有多于一条树枝边则为割点。

# 双连通分量

- 边双连通分量：删去所有割边，每个连通块都是双连通分量
- 可以求出割边之后直接跑不走割边的bfs，也可以像强连通一样tarjan时维护一个栈记录未分配双连通分量的点，并在离开dfs遇到割边时弹栈处理

- 点双连通分量：由于一个点可以属于多个点双连通分量，所以并不好维护点的栈
- 改为维护存边的栈，同样在离开dfs遇到割点时，取出所有边和相邻的点，作为一个点双连通分量即可

# 模板 P3388

## 题目背景

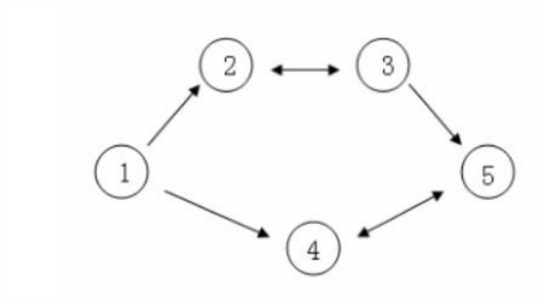
割点

## 题目描述

给出一个$n$个点，$m$条边的无向图，求图的割点。

# NOIP2009 P1073 最优贸易

- n(<=100000)个城市之间用单向边或双向边相连，现在要从1号点走到n号点，路径可以有回路
- 每个城市有一个水晶球售价，现在要在旅途中买卖一次，求最大的收益




假设 1 $n$ 号城市的水晶球价格分别为 $4, 3, 5, 6, 1$。

阿龙可以选择如下一条线路：1->2->3->5，并在 2号城市以3 的价格买入水晶球，在 3号城市以5的价格卖出水晶球，赚取的旅费数为 2。

阿龙也可以选择如下一条线路1->4->5->4->5，并在第1次到达5 号城市时以 1的价格买入水晶球，在第 2 次到达4 号城市时以6 的价格卖出水晶球，赚取的旅费数为5。

# 思路2

- 路线可以任选，只要能到达终点即可
- 那么对于一个SCC，其中所有的点可以互相到达，可以缩成一个点，记录最大最小价格
- 之后问题转化为DAG上的问题，拓扑排序时保存路径上最小买入价，不断更新答案即可