图论

清华大学 徐明宽

基础概念

- 图
  - 节点
  - 边
  - 路径/回路
  - 环
  - 链
  - 度数
  - 补图
  - 子图/导出子图
- 简单图/多重图
  - 重边
  - 自环
- 有向图/无向图
  - 有向边/无向边
  - 混合图
- 完全图
  - 团
  - 有向完全图
  - 竞赛图
- 连通性
  - 连通图/连通分量
  - 双联通图/双连通分量
  - 强连通图/强连通分量
  - 割点/桥
  - 割
- 网络流
  - 最大流/最小割
  - 最小割树
- 树
  - 根
  - 叶子
  - 最近公共祖先
  - 生成树
  - 有向生成树（树形图）
  - 森林
  - DFS树/森林
  - 仙人掌/沙漠
- 二分图
  - 染色
  - 匹配
  - 独立集
- 平面图
  - 对偶图
- 弦图
  - 弦
  - 区间图
- （k-）正则图
- 线图
- 完美图
  - 色数
- ……

目录

- 最短路
- 生成树、连通分量
- 二分图、匹配与覆盖
- 综合练习

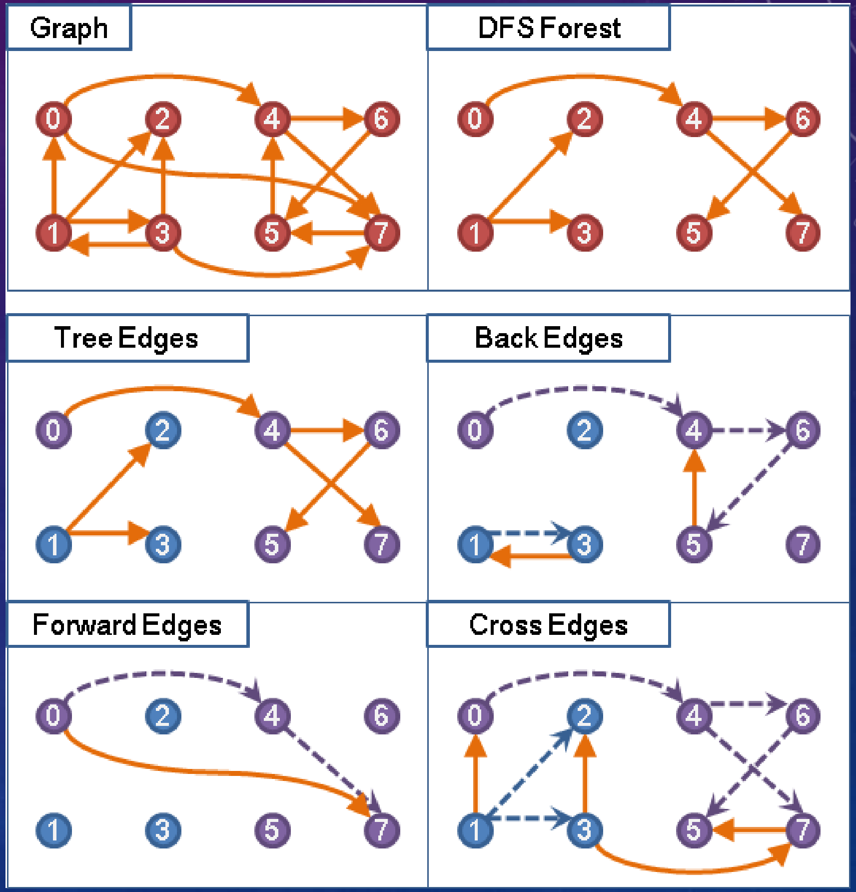
PART1 最短路

- DFS树/DFS森林
- 单源最短路
- 差分约束系统
- 多源最短路
- 传递闭包
- 半径/直径
- 绝对中心

DFS树/DFS森林

- Tree Edge指树边
- Back Edge指连向祖先的边
- Forward Edge指连向子孙的边
- Cross Edge指连向树其他分支或其他树的边

- 在无向图中只存在Tree Edge和Back Edge
- 即非树边都是连向祖先的边



单源最短路：BELLMAN-FORD算法

- 更新n次dist数组
- O(nm)
- 可以用来找负环

单源最短路：SPFA算法

- 队列优化的Bellman-Ford算法
- O(nm)
- 可以用来找负环

单源最短路：DIJKSTRA算法

- 每次用dist最小的点去更新
- 不适用于有负权的图
- 暴力：O(n^2 + m)
- 堆优化：O(m log n)
- 可以用Fibonacci堆优化到O(n log n + m)
- 注意稠密图暴力可能快于堆优化

单源最短路：BFS算法

- 边权都是0/1
- 像普通的bfs一样维护队列，在队头/队尾插入
- O(m)

单源最短路的应用：差分约束系统

- 根据最短路有不等式dist(v) <= dist(u) + w(u, v) （u到v的边）
- 而且dist(v)是满足条件的最大值

- 对于一些s(v) <= s(u) + w(u, v)的限制，可以类比最短路建图
- 求一个差的最大值：跑最短路
- 求一个差的最小值：跑最长路
- 有正环的最长简单路是NP-Hard的。有负环的最短简单路也是NP-Hard的。

单源最短路的应用：差分约束系统

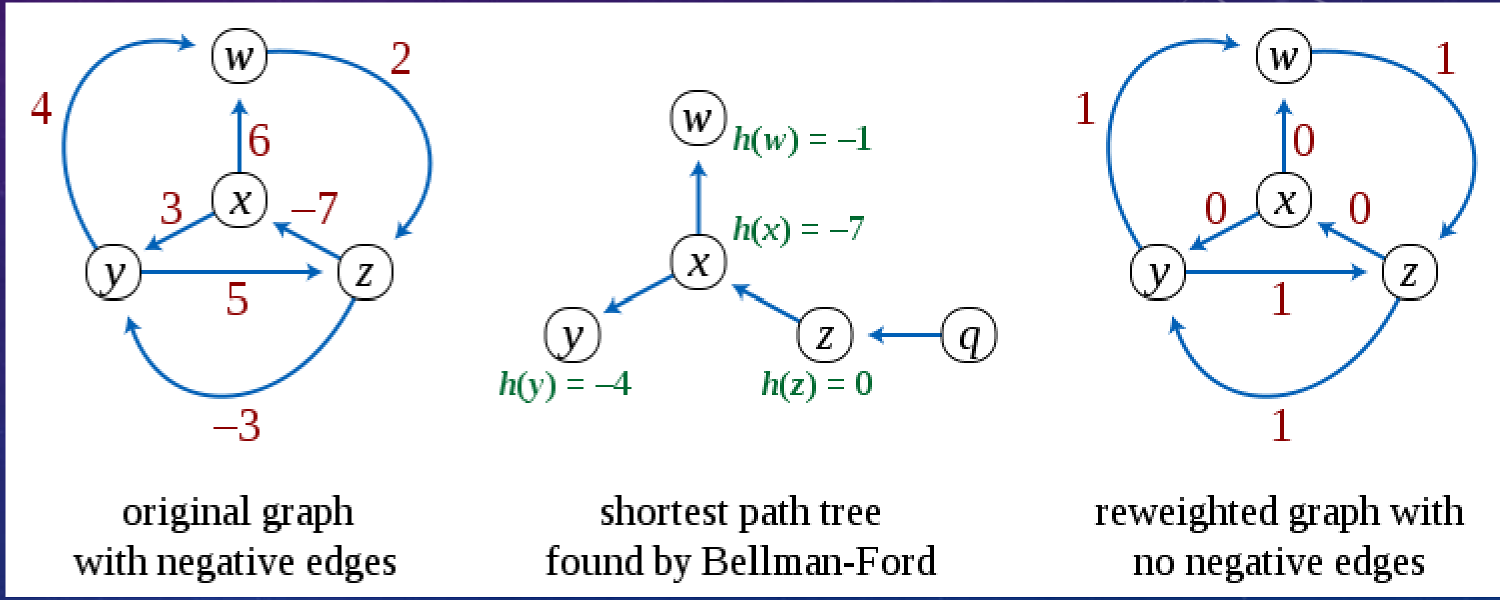
- 最长路：将每条边取反，跑最短路
- 判断解唯一：最小值与最大值相等
- 不求最小值/最大值，只判断有没有解：找负环

多源最短路：FLOYD算法

- 先枚举中间点，再枚举两个端点，更新dist数组
- O(n^3)
- 无负权且源点数量较少时可以做多遍堆优化Dijkstra算法，时间复杂度O(km log n)，其中k为源点数量

多源最短路：JOHNSON算法

- 有负权时，建虚拟源点，向每个点连0边，跑Bellman-Ford，得到最短路数组h
- 将每条边w(u, v)加上h(u) – h(v)
- 移除虚拟源点，跑堆优化Dijkstra
- O(nm + km log n)，其中k为源点数量



找一个环的FLOYD算法

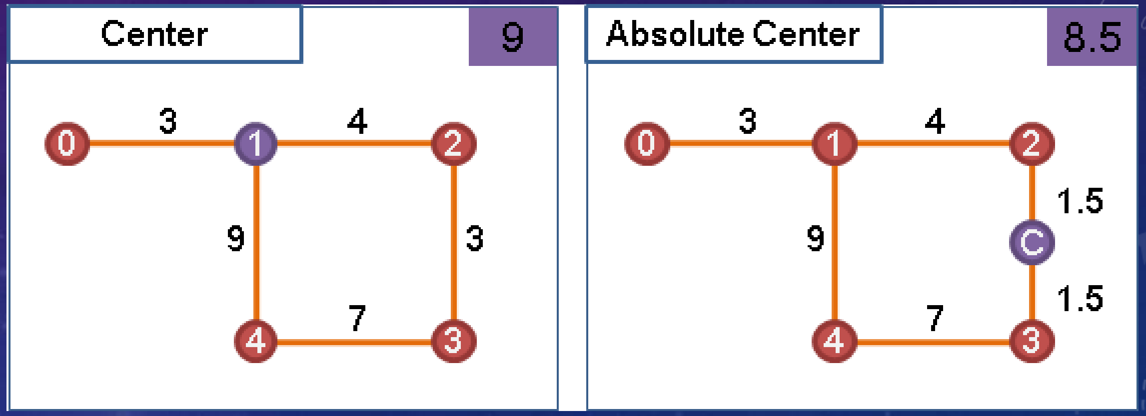
- 有向图，每个点的出度为1
- 维护两个指针，一个每次走一步，一个每次走两步
- O(n)时间，O(1)空间
- 应用于Pollard rho算法等

多源最短路的应用：传递闭包

- 一个有向图，问两点之间有没有路径。
- 将Floyd算法中的加法换成或运算即可。
- 还可用bitset优化成O(n^3/w)。

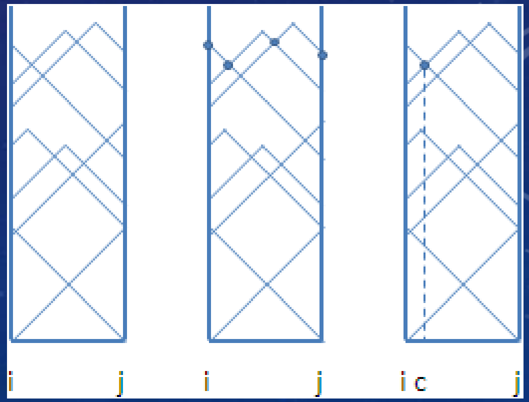
多源最短路的应用：无向正权图的半径/直径

- u的偏心距： ecc(u) = max dist(u,v)
- 直径： d = max ecc(u)
- 半径： r = min ecc(u) (d ≠ 2r)
- 中心： arg min ecc(u) (要求定义在点上)
- 时间复杂度与全源最短路相同。
  - Floyd： O(n^3)
  - 堆优化Dijkstra： O(nm log n)
- 绝对中心：(可以定义在边上)

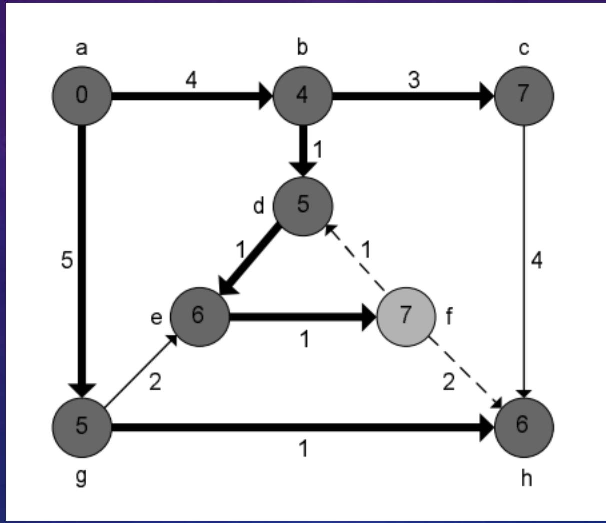


绝对中心

- 固定一条边(u, v)，考虑上面的点p的偏心距。
- 假设第三个点是z, dist(p, u) = x
- 那么对应的折线为 min(x + dist(u, z), w(u, v) – x + dist(v, z))。
- 那么偏心距为n条折线的最大值形成的折线。
- 按左端点排序维护一下。
- 时间复杂度O(nm log n)



最短路径树/最短路径图



故乡的梦 (BZOJ 2725)

- 给一个n个点m条边的无向图，每条边有一个边权。
- 给出两个点S和T作为起点和终点。
- 询问每条边删去之后这两个点的最短路(询问之间独立)。
- n, m <= 2e5

SOLUTION

- 首先求出一条S到T的最短路P，如果不是最短路上的边删了肯定不影响答案。
- 接着证明去掉一条边(u, v)之后最短路一定是这样存在一条边(x, y)，然后最短路径是S->x->y->T,并且S->x, y->T都是原图中的最短路。（考虑从S出发不在最短路径图上的第一条边）
- 考虑S的最短路径图和T的最短路径图。
- 于是只要考虑一条边(x, y)，然后求出S到x的最短路和这条最短路最早什么时候离开P，记作x’，同理求出y’（最晚什么时候离开P）。
- 于是(x, y)这条边可以更新x’到y’之间的答案。

安全路径 (BZOJ 1576)

- 给一个n个点m条边的无向图，每条边有一个边权。
- 给出一个点S作为起点，保证最短路径树唯一。
- 询问每个点不经过最短路径树上连向父亲的那一条边的最短路。
- n, m <= 2e5

SOLUTION

- 对于一个点T，最短路一定为从S到一个节点v，然后经过边(v, p)，其中p是T的子孙，v不是T的子孙，然后从p走到T，其中答案为dist(v) + w(v, p) + dist(p) - dist(T)。
- 将每个点T的答案都加上dist(T)，枚举非树边(v, p)，然后更新p到LCA(v, p)这段路径为dist(v) + w(v, p) + dist(p)。
- 可以使用树链剖分/可并堆/线段树维护。

实时路况 (计蒜之道 2016 复赛)

- 给出一个n个点的有向图，记d(u, v, w)为从u到达v不经过点w的最短路，如果不存在则为-1。
- 求

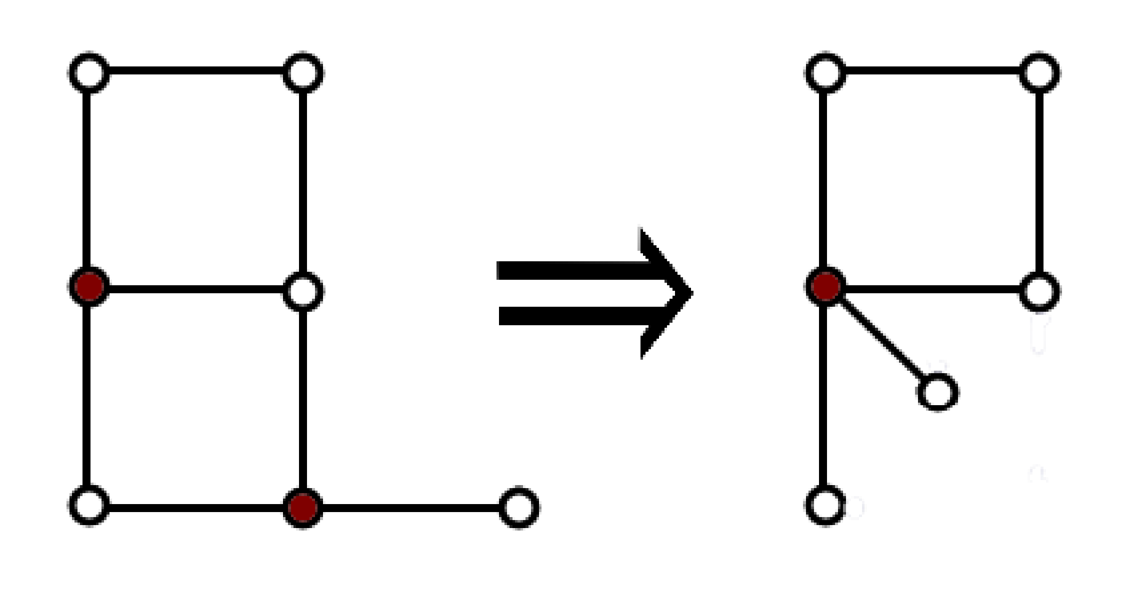
$$P = \sum_{1 \le x,y,z \le n, x \neq y, y \neq z} d(x, y, z)$$

- n <= 300

SOLUTION

- 使用分治，考虑Floyd算法本来就是一个增量的过程。
- 记solve(l, r)表示加入除了l到r之间的点的APSP
- 先取mid = (l + r) / 2，然后将l到mid的点加入，递归solve(mid + 1,r)
- 同理递归solve(l, mid)
- 时间复杂度O(n^3 log n)

PLAYING ON GRAPH (VK 2015 ROUND 3 E)



- 有一个n个点m条边的无向图，每次可以选一对没有边相连的点缩起来。
- 问最后形成的最长链的长度，如果不能形成链输出-1。
- n <= 1e3, m <= 1e5

SOLUTION

- 首先一个三元环一定是无解的，而对于一个奇环，经过缩点操作之后还是会剩下一个奇环。
- 所以存在奇环就一定无解。
- 对于一个二分图，能得到的最长链为这个图的直径。
- 将所有连通块的直径相加即可。

PART2 生成树、连通分量

- 最小生成树
- 最小瓶颈生成树
- 最小树形图
- 拓扑排序
- 欧拉回路
- DAG最短路/最长路
- 强连通分量
- 边双连通分量
- 点双连通分量
- 割点和桥
- 2-SAT
- LCA的Tarjan算法

最小生成树：PRIM算法

- 初始树上只有任意一点
- 每次找与当前树上点连边权值最小的点，连到树上
- 暴力：$O(n^2 + m)$
- 堆优化：$O(m \log n)$
- 可以用Fibonacci堆优化到$O(n \log n + m)$
- 注意稠密图暴力可能快于堆优化

最小生成树：KRUSKAL算法

- 将边按权值排序，依次加入，用并查集维护
- $O(m \log m) = O(m \log n)$

最小生成树：BORŮVKA算法

- 若干轮：
- 每个点找最近的点，将这条边加入生成树
- 将所有树边连接的两个点缩成一个点
- O(m log n)

最小瓶颈生成树

- 求一棵生成树，使得树上最大边权值最小。
- 最小生成树一定是最小瓶颈生成树。为什么？
- 比O(m log n)更快？

最小瓶颈生成树

- 线性第k大数思想
- 首先随机一个边权w。
- 然后将不超过这个边权的边加入，遍历这张图。
- 如果图连通，那么瓶颈不超过w，于是只需考虑边权不超过w的边。
- 否则将这些连通点缩起来，考虑边权大于w的边。
- $T(m) = C \cdot m + \frac{1}{m}(\sum_{i=1}^{x} T(m-x) + \sum_{i=x+1}^{m} T(i)) \leq 4C \cdot m$
- 期望时间复杂度O(m)。

UPRTOFF (CF GYM 100020 J)

- 有n个点m条边，第i条边的价值为2^i。
- 每个点有限制表示这个点是黑点还是白点。
- 要从中选出一个价值最小的边集，使得黑点和奇数条边相连，白点和偶数条边相连。
- n, m <= 1e5

SOLUTION

- 如果有一个连通块里有奇数个黑点，那么一定不合法。
- 如果一个连通块里有偶数个黑点，那么一定存在它的一个子图合法。为什么？
- 所以只需让每个连通块里的黑点个数为偶数。
- 注意到边权为2^i，所以从后往前每条边能不选尽量不选。

SOLUTION

- 如果一条边不改变连通性，那么一定不会被选入。
- 所以选的边一定在最小生成树上。
- 于是可以首先计算出最小生成树，然后看每条边两端的黑点个数。

最小树形图：朱刘算法

- 给定一个有向图和一个根root，求一棵以root为根出发的有向生成树，使得边权和最小。

- 对于除了root以外的每个点p，选择权值最小的入边，将其权值记为incost[p]，这样如果选出来的n – 1条边是一棵树的话就做完了。很显然所求的答案至少为这n – 1条边的权值和，所以我们可以把这个和先计入答案。
- 否则这n – 1条边里有环，建立新图，对于每个环建立一个新点，对于每个不在环上的点也建立一个新点。对于原图中的每条边(u, v, w)，在新图中建立对应的边(u′, v′, w – incost[v])。如果这条边连到了一个“环”点上，那么它意味着打开原来的环上incost[v]那条边并替换成(u, v)这条边以形成树；如果连到了一个普通点上，那么它也意味着将原来incost[v]那条边替换成(u, v)的代价。
- 递归上述过程即可，时间复杂度O(nm)。

拓扑排序：KAHN算法

- 拓扑序：对于每条有向边(u, v)，u在v前面出现（偏序）
- 每次去掉图中入度为0的点
- O(n + m)
- 如果最后不为空集那么这个图不为DAG （directed acyclic graph，有向无环图） 。为什么？
- “拓扑排序”一词经常也代指Kahn算法

字典序最小的拓扑序

- 将拓扑排序（Kahn算法）的队列改为优先队列即可
- $O(n \log n + m)$

拓扑排序的应用：DAG线性单源最短路/最长路

- DAG（directed acyclic graph）：有向无环图
- 我们可以按照拓扑序依次填写从源点到该点的最短路/最长路，或从任意一点到该点的最长路。

航空管制 (NOI 2010)

- 有n个航班依次起飞，并且有m个限制:
  - 第一种限制，第i个飞机必须在c(i)的时刻前起飞。
  - 第二种限制，第i个飞机必须在第j个飞机之前起飞。
- 询问:
  - 一个可行的起飞方案。
  - 每个飞机最早的起飞时间。
- n <= 2e3, m <= 1e4

SOLUTION

- 倒过来变成每个飞机在某个时刻之后可以起飞。
- 第二问变成每个飞机最晚什么时候起飞。
- 直接用拓扑排序的做法即可。
- $O(n \log n + nm)$

拓扑排序：反向DFS算法

- 将所有边反向
- 每次找一个没访问过的点开始dfs
- 在退出dfs之前将该点加入答案（后序遍历）
- O(n + m)
- 退出dfs，意味着当前顶点没有指向其它顶点的边了

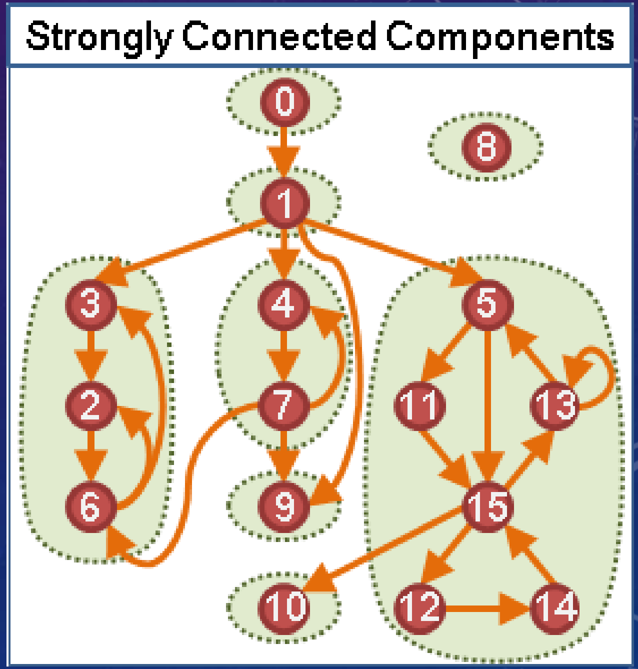
欧拉回路

- 欧拉回路：无向图中经过每条边恰好一次的回路。
- 如果图不连通或有任意一个点的度数为奇数，则不存在欧拉回路，否则一定存在。
- 从任意一点v开始dfs，求出后序遍历，将结果reverse就是答案。
- O(n + m)
- 这是因为dfs时只可能在回到点v时“卡住”，然后返回到某点再分叉后只可能回到该分叉点时才会“卡住”。

强连通分量：KOSARAJU算法

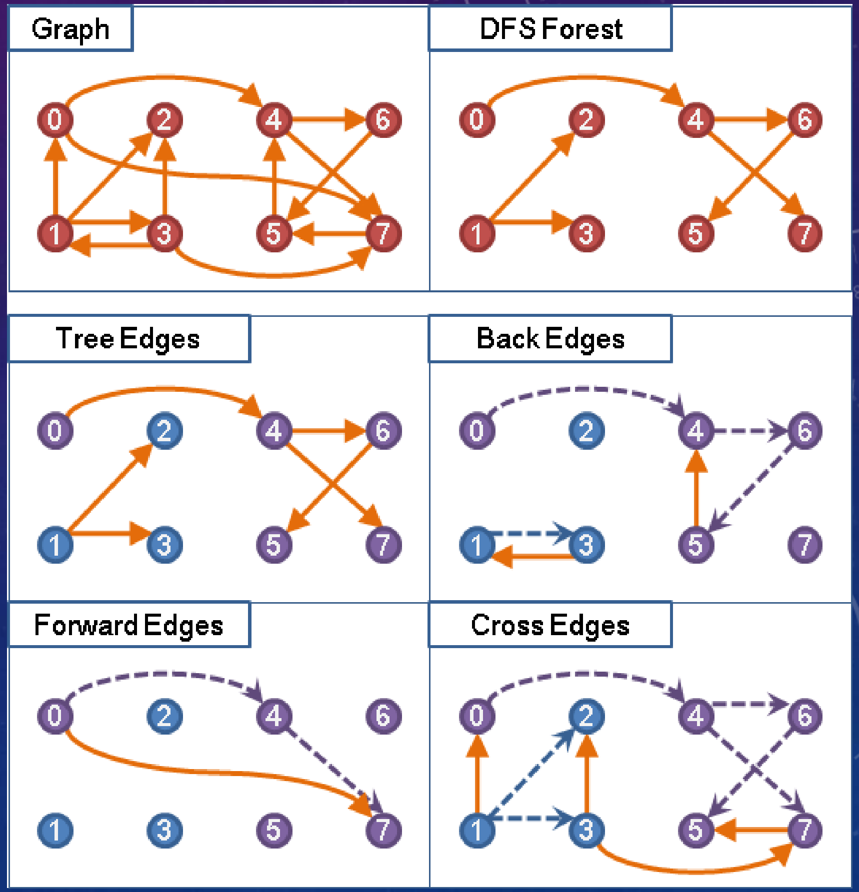
- 强连通分量：有向图中的任意两个点都互相可达的极大（导出）子图。
- 第一次dfs求出后序遍历L
- 将所有边反向
- 按L的倒序dfs每个点，每次dfs为一个强连通分量
- O(n + m)
- （常数比tarjan大而且后者很好写，所以没什么人用）

- 一个图将强联通分量缩起来将会形成一个DAG



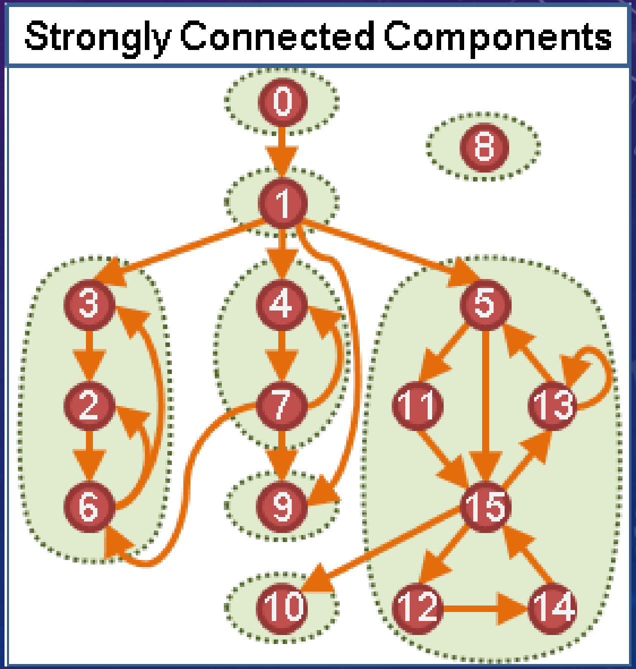
强连通分量：TARJAN算法

- 首先每个点根据dfs的时候访问的顺序进行标号，记作dfn。
- 然后每个点维护一个low值，即这个点通过Tree edge和至多一条连向当前强连通分量内部的非树边能访问到的dfn最小值。用是否在dfs栈中来区分是否在当前强连通分量内部。
- low[p] = min(low[p], low[son]);
- low[p] = min(low[p], dfn[others]);
- 其实 low[p] = min(low[p], low[others]); 也对
- 如果一个点的能访问到最早的点为这个点本身（low值等于dfn），就会形成一个新的强连通分量。
- O(n + m)



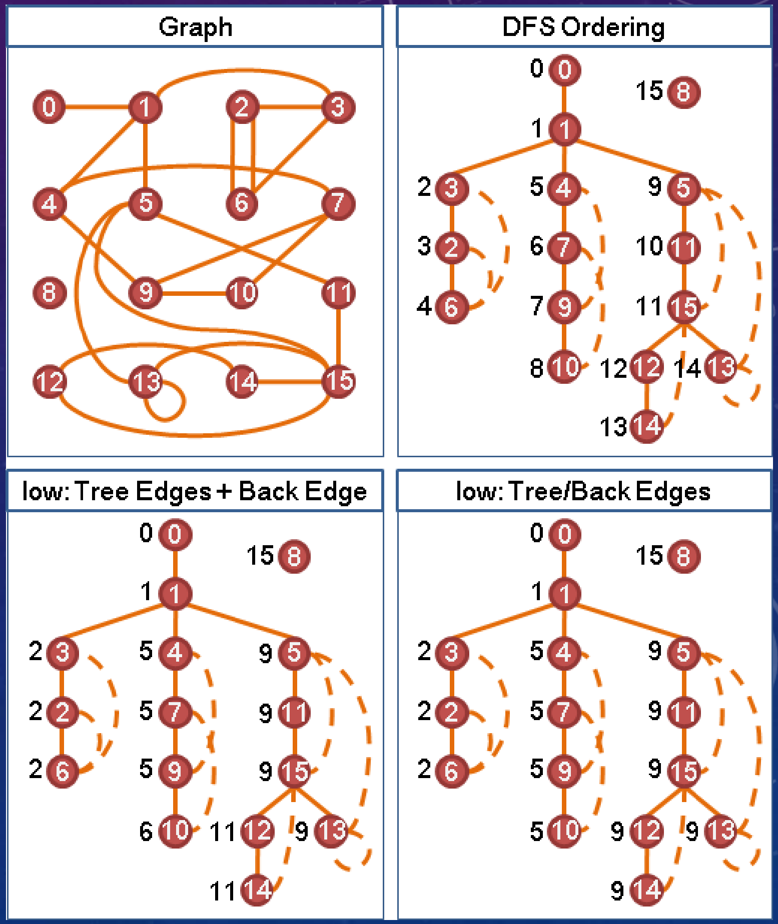
强连通分量：TARJAN算法

```
function strongconnect(v)
  // Set the depth index for v to the smallest unused index
  v.index := index
  v.lowlink := index
  index := index + 1
  S.push(v)
  v.onStack := true

  // Consider successors of v
  for each (v, w) in E do
    if (w.index is undefined) then
      // Successor w has not yet been visited; recurse on it
      strongconnect(w)
      v.lowlink  := min(v.lowlink, w.lowlink)
    else if (w.onStack) then
      // Successor w is in stack S and hence in the current SCC
      v.lowlink  := min(v.lowlink, w.index)
    end if
  end for

  // If v is a root node, pop the stack and generate an SCC
  if (v.lowlink = v.index) then
    start a new strongly connected component
    repeat
      w := S.pop()
      w.onStack := false
      add w to current strongly connected component
    while (w != v)
    output the current strongly connected component
  end if
end function
```



边双连通分量：TARJAN算法

- 点连通度：最小的点数使得删去之后图不连通
- 边连通度：最小的边数使得删去之后图不连通
- 如果一个图的点连通度大于等于2，那么是点双连通的，边双连通同理
- 双联通分量为图中的极大双联通子图



- 求边双连通分量即将强连通分量的代码增加【不用刚走过的边的反向边去更新low】的限制即可
- O(n + m)

点双连通分量：TARJAN算法

- 不能再偷懒只用low去更新low了（非儿子但instack时要用dfn去更新low）！
- low[son] == dfn[p]时会形成新的点双连通分量：栈顶直到son为止的所有点加上p这个点。
- 不要把p弹栈（也弹不了）
- O(n + m)
- 每个点可能属于多个点双连通分量

割点和桥

- 割点/桥就是删掉以后整个图不连通的点/边。
- 属于多个点双的点是割点，不属于任何一个边双的边是桥。
- O(n + m)。更好写的做法？
- 每条非树边对应着一个点到祖先的路径。
- 对于一条非树边只要把对应的边打上标记即可。
- 比如对于(u, v)这条非树边，只要在u点打上+1的标记，v点打上-1的标记。
- v到v的父亲的树边的覆盖次数为子树内所有标记的和。
- 割点同理(注意特判根节点和叶节点)。
- 桥还可以用并查集在线做。

强连通分量的应用：2-SAT

- 有一堆变量的二元限制，问是否有解。

$$(x_0 \vee x_2) \wedge (x_0 \vee \neg x_3) \wedge (x_1 \vee \neg x_3) \wedge (x_1 \vee \neg x_4) \wedge (x_2 \vee \neg x_4) \wedge (x_0 \vee \neg x_5) \wedge (x_1 \vee \neg x_5) \wedge (x_2 \vee \neg x_5) \wedge (x_3 \vee x_6) \wedge (x_4 \vee x_6) \wedge (x_5 \vee x_6).$$

强连通分量的应用：2-SAT

- 对每个变量x，建两个点：x = 0 和 x = 1
- 对每个限制，连边表示“推出”，如 x or y = 1 则连边：x = 0 -> y = 1，y = 0 -> x = 1
- 求强连通分量，如果 x = 0 <-> x = 1 则无解
- 强连通分量缩点，求反图拓扑序（逆拓扑序），依次贪心取，并把对立点打上不取的标记即可
- O(n)
- 其实Tarjan算法形成强连通分量的顺序就是逆拓扑序，所以不用再写一个拓扑排序了。

离线求LCA：TARJAN算法

- 先开一个并查集，记并查集父亲数组为Father
- 先将每个查询挂在查询的两个端点的链表上
- 进行dfs，在dfs退出之前将这个点标记为已访问过，然后看一端为这个点的查询，如果另一端也已访问过，则这个询问的答案为并查集查询另一端的结果
- 然后将这个点的Father数组改为这个点在树上的父亲
- 时间复杂度与并查集相同。由于LCA问题本身是线性的，这个特殊的并查集问题也能优化到线性，但那样可能常数较大。

PART3 二分图、匹配与覆盖

- 二分图判定
- 二分图最大匹配
- 二分图最大权匹配
- Hall定理
- Kőnig定理：二分图最小点覆盖、最大独立集、最小边覆盖
- DAG最小路径覆盖
- Dilworth定理：最长反链长度、最小反链覆盖

二分图判定

- bfs/dfs染色即可
- O(n + m)
- 树一定是二分图

CYCLE (HDU 5215)

- 你有一个n个点m条边的无向图，问是否存在一个长度为奇数/偶数的简单环。
- n <= 1e5, m <= 3e5

SOLUTION

- 判断是否存在奇环，只要看是不是二分图即可。
- 判断是否存在偶环，首先看每条非树边对应的环是不是偶环。
  - 如果存在那么就找到了偶环。
  - 否则考虑如果两个奇环相交，那么去除中间部分就会形成一个偶环。
  - 所以对于奇环的非树边只要暴力访问树边打上标记，如果已经有标记了就说明存在奇环。
- 时间复杂度O(n + m)

CYCLING CITY (CF 295 E)

- 你有一个n个点m条边的无向图。
- 问是否存在两个点，使得这两个点之间有三条简单路，并且这三条简单路没有公共点。
- n, m <= 2e5

SOLUTION

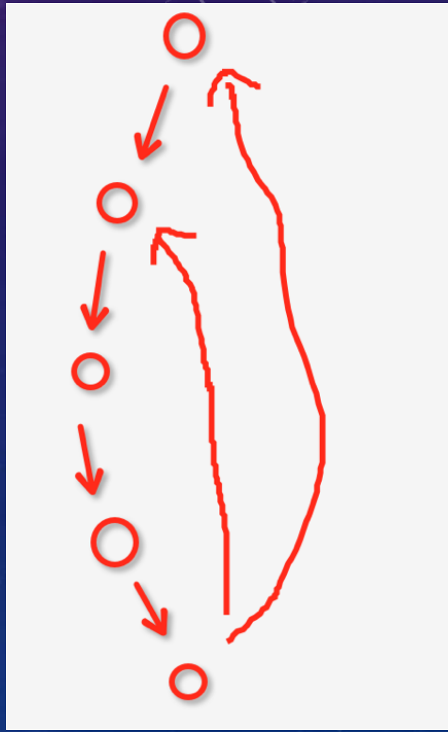
- 如果两条非树边对应的环有交，那么一定可以找到这样的两个点。
- 否则不存在。

FAIRY (CF 19 E)

- 你有一个n个点m条边的无向图。
- 对于每条边，判断删除之后是否为二分图。
- n, m <= 1e4

SOLUTION

- 首先如果一个图是二分图，每一条边删掉都是可行的。
- 一个图是二分图，那么就没有奇环。
- 所以要求删掉一条边使得去掉这条边之后没有奇环，换句话说也就是所有的奇环通过这条边。
- 于是先求出所有非树边对应的奇环的交。
- 可以删除的边只可能在这些边里面。



SOLUTION

- 同时可以发现，如果有一个偶环也覆盖了这条边，那么删除这条边之后这个偶环和原来的奇环会形成一个奇环。
- 所以这样的边是不可行的。
- 如果覆盖这条边的只有奇环，那么删除这条边之后剩下的图是二分图。
- 对于奇环，给上面的边打上+1的标记，对于偶环打上-1标记。
- 如果一条边的标记恰好等于奇环个数，那么就是可行的。
- 时间复杂度O(n + m)

二分图最大匹配

- 匈牙利算法（对左边每个点去找增广路）：O(nm)
- Dinic：O(m sqrt(n))

LIFE OF THE PARTY (ONTAK 2010)

- 一个二分图左边n个点，右边m个点，共k条边。
- 问哪些点一定要在最大匹配中。
- n, m <= 1e4, k <= 1e5

SOLUTION

- 首先求出最大匹配，下面考虑左边点的情况。
- 我们将匹配中的边从右往左连，不在匹配中的边从左往右连。
- 这个时候一条增广路成为一条连续的路径。
- 从每个左边未匹配的点开始遍历，如果被一个左边的点被访问到，说明存在一条增广路，也就是不一定在最大匹配中。
- 所有没有被访问到的点一定在最大匹配中。

二分图最大权匹配

- 可以归约到费用流
- 也可使用KM算法：O(n^4)或O(n^3)

HALL’S MARRIAGE THEOREM

- 对于一个二分图G = (X, Y ,E)，记S为X的一个子集，N(S)为所有S中所有点邻居的并集。
- 一个图有完备匹配当且仅当X的所有子集S都有|S|<=|N(S)|

- 推论：每个正则二分图都有完备匹配。
- 推论：每个k-正则二分图都能拆成k个完备匹配。

REVMATCHING (TCO 2015 1A HARD)

- 给定一个n个点的二分图，每条边有一个边权。
- 找到一个边权和最小的边集，使得删掉这个边集之后不存在完备匹配。
- n <= 20

SOLUTION

- 根据Hall定理，只要存在一个集合S，使得|N(S)|<|S|，则不存在完备匹配。
- 于是我们枚举S集合，然后贪心删除边集使得|N(S)|<|S|。

KŐNIG'S THEOREM

- 最小点覆盖 = 最大匹配 (与最大流最小割定理等价)
- 最大独立集 = 点数 - 最大匹配 (独立集为点覆盖的补集)
- 最小边覆盖 = 最大独立集 (独立集中每个点需要一条边去覆盖)

DAG最小路径覆盖

- 在一个DAG中，求若干个路径，使得每个点都出现在路径中，求路径数量的最小值。
- 如果要求路径不交（每个点恰出现在一条路径中）：
- 建立二分图，对于u -> v的边，看做二分图中的(u, v’)，然后答案为点数 - 最大匹配。
- 如果路径可以相交（每个点出现在至少一条路径中）：
- 先传递闭包再做即可。

- Dilworth 定理：最小路径覆盖（不交） = 最长反链长度
- Dilworth 定理的对偶定理：最小反链覆盖 = 最长路径长度
- 应用：求最长反链长度和最小反链覆盖。

拦截导弹（NOIP 1999）

- 有n个导弹，每个导弹有一个高度
- 现在有一套系统，可以拦截一个子序列，但每一发导弹不能超过上一发的高度
- 求一套系统可以拦截的导弹数量最大值，以及拦截所有导弹所需的系统数量最小值
- n <= 1000， 高度 <= 30000

SOLUTION

- 第一问是求最长不上升子序列长度
- 第二问是求最小不上升子序列划分
- 应用Dilworth定理，即为求最长上升子序列长度
- O(n log n)

SIŁOWNIA (PA 2015 5A)

- 体育馆里有k件不同的器材。
- 有n个人，第i个人希望在a(i), a(i) + 1, ..., b(i)之中选择一天，在这天使用p(i)这种器材，同一个器材一天最多只能被一个人使用。
- 如果一天体育馆没人锻炼，那么就关闭。
- 问最少需要开放多少天能满足所有人的需求，或者说明不可能。
- n <= 1e6, a, b, k, p <= 1e9

SOLUTION

- 首先考虑同一种任务，将每个任务和日期看成节点，于是这个问题变成了一个匹配。
- 回顾Hall定理，有完备匹配的充要条件是对于任意的任务集合S，都有|N(S)|>=|S|。
- N(S)也就是一堆任务对应的日期的并。
- 接着考虑怎么简化这个式子。
- 如果N(S)是不连续的，那么把它分成若干段S', S'', ...
- 如果每段有|N(S')|>=|S'|，那么它们的并集也满足条件。
- 所以只要考虑连续的一段l, r即可，因为要判断|N(S)|>=|S|,所以要求|S|尽量大，于是也就是要将l, r之间的任务全部选入。

SOLUTION

- 经过一些简化，有解的充要条件就是对于所有的l,r日期区间，体育馆开放的时间不少于在l,r之内的任务。
- 记$S(i)$表示前i天的开放总天数。
- 于是有$S(i + 1) >= S(i)$, $S(i + 1) <= S(i) + 1$这样的限制。
- 同时还有前面的$S(r) - S(l - 1) >= f(l, r, c)$这样的限制，其中$f(l, r)$为$l, r$之内的任务$c$的个数，每次$S(r)$取这样的最小值。
- 但是这样会违反$S(i + 1) <= S(i) + 1$的限制，因为如果r位置有很多新的区间加入，那么$S(i + 1)$会变大很多。

SOLUTION

- 解决方法: 对于两个任务[a, c], [b, c](其中a < b)，将第一个任务改为[a, c - 1]。
- 所以这样每次最多会加上1，不会违反S(i + 1) <= S(i) + 1的限制。
- 同时对于新加入的区间，只要用对应的S(l - 1) + f(l, r, c)更新即可。
- 用简单的数据结构维护f(l, r, c)和S(i)。
- 时间复杂度O(n log n)。

PART4 综合练习

- 一些题

BAJTMAN I OKRĄGŁY ROBIN (ONTAK 2015)

- 有n个强盗，每个强盗会在时刻li到时刻ri抢劫，会造成ci的损失。
- 在一个时刻，你可以选择抓一个强盗，强盗被抓住之后不会造成损失。
- 你要抓尽量多的强盗使得损失尽量小。
- n <= 5000

SOLUTION

- 按强盗从大到小排序，贪心选取每个强盗能不能抓。
- 贪心算法的正确性：考虑匈牙利算法，从大到小一个一个匹配，一个点一旦在匹配中，那么一直在匹配里面。
- 直接匈牙利是O(n^3)的。
- 判断一些强盗能不能抓完时，可以按左端点排序，使用优先队列维护右端点。
- O(n^2 log n)

WIDE SWAP (AGC ROUND 1 F)

- 给定一个n个元素的排列P,和k。
- 当两个元素i, j满足|P(i) - P(j)| = 1且j – i > k，那么i, j两个元素可以互换。
- 求出能通过交换得到的字典序最小的排列。
- n <= 5e5

SOLUTION

- 首先求出每个元素所在的位置pos(i)。
- 于是如果相邻两个元素的差超过k，那么可以交换。
- 通过交换使得1的位置（pos(i) = 1的i）尽量小，其次是2，以此类推。
- 如果两个元素的差不超过k，那么它们的前后顺序不会改变。
- 所以可以对于所有元素差不超过k的点前后关系建图，然后求字典序最小拓扑序即可。

SOLUTION

- 这样建图得到的边是O(n^2)的，但是我们发现每个点只需要向它前面比它大k以内的最近的点、以及比它小k以内的最近的点连边即可。
- 边数O(n)，连边可以线段树查询。
- 时间复杂度O(n log n)。

OGRÓD ZOOLOGICZNY (ONTAK 2015)

- 你有一个n个点m条边的平面图，每个点的相邻点按照极角序给出。
- 有q个询问，每个询问给出其中k个点，问图中仅保留这k个点(和这些点之间的边)形成的连通块区域(不包括无穷域)。
- $n <= 5e4$, $m <= 2e5$, $q <= 1e6$, $\sum k <= 5e6$

SOLUTION

- 首先注意到平面图的边数不超过3n - 6。
- 同时考虑生成树，域的个数为非树边个数。
- 于是我们只要统计每组询问有多少边即可。
- 每次删除图中度数不超过6的顶点，将图中的边定向。
- 于是查询不超过6k条边。
- 再算一下连通块个数即可。

补充题

- 给定一个连通无向图，问其中是否所有简单环都等长。
- n <= 5e5, m <= 1e6

谢谢大家

- 鸣谢杜老师提供大量例题