# 图论考点解析

- 梳理考题，观察分析各个题目所涉及知识点
- 整理考点，从命题人的角度来看的各个考点
- 布置习题，下一次课讲结合重点的习题讲解

# 历年真题 (NOI)

- 2016 D1P2 网格 (双连通分量+离散化)
- 2015 D2P3 小园丁与老司机 (数据结构+网络流)
- 2014 D1P2 魔法森林 (LCT维护最小生成树)
- 2013 D2P3 快餐店 (环套外向树dp)
- 2012 D2P2 美食节 (费用流动态加点)
- 2011 D2P3 兔兔与蛋蛋 (匹配)

# 历年真题 (ZJOI)

- 2016 D1P2 旅行者 (分治+最短路)
- 2016 D2P3 电阻网络 (点双连通分量建树+dp)
- 2015 D2P2 醉熏熏的幻想乡 (网络流)
- 2013 D1P1 防守战线 (对偶费用流)
- 2012 D1P2 旅游 (树直径)
- 2012 D2P1 灾难 (伪-dominator tree)
- 2011 D1P3 最小割 (全局最小割 Gomory-Hu tree)
- 2011 D2P2 营救皮卡丘 (上下界费用流)

# 历年真题 (WC)

- 2016 挑战NPC
- 2015 K小割
- 2013 平面图
- 2012 记忆中的水杉树
- 2011 Xor

# 分析

- 一般来说在6题的比赛中会出现一道到两道的图论题。
- 网络流占了很大的一块，但是最近的热度比原来有下降。
- 图论题和其他的考点会经常结合在一起。
- 比如数据结构，可以做到同时对选手的代码能力和图论水平的考察。
- 比如dp，考察选手们在一个特殊的图或根据图的结构进行dp。
- 剩下的杂题需要大家慢慢积累了。

# 考(tao)点(lu)分(zong)析(jie)

# DFS

- 需要掌握手工栈模拟DFS，不过现在有些比赛也开始开栈空间了。
- 太普遍了>_<讲一些常见的套路。
- 套路1: 构造的时候选一棵DFS树忽略其他边进行构造。
- 给一个图，每个点有个标号0/1,选择一个边的子集，使得每个点的度数的奇偶性与给定的标号相同。
- 常用在证明和分析中，也有要求显式地构造。

- 套路2: 对于一个无向图求出一棵DFS树，然后每条非树边对应了一个环。
- Xor (每条路径是若干个环xor起来, 而非树边对应着一个环)
- 仙人掌判定 (看每条非树边是不是至多被覆盖了一次)
- balabala。。。还有好多好多题。。。

# BFS

- 做最短路。
- 注意在BFS图中，边只可能在相邻的两层和同一层之间连，并且每个点至少和上一层的一个点相连。(计数)

# 强连通/双连通分量

- 远古套路0: 强连通缩点(手工栈)之后然后在DAG上做一些事情(比如跑dp)。
- 套路1: 边(点)双连通分量缩起来形成一棵树，然后在树上搞事情。
- (比如加最少边让这个图变成双连通)
- 套路2: 动态维护边双连通分量树。(一般可以离线先把树建出来转化到dfs序上, 否则需要lct/虚树)
- 一般来说在OI比赛中会结合数据结构考，如果这样考图论模型一般比较明显。

# 欧拉回路

- 需要掌握正确的有向图/无向图的欧拉回路/路径求法(uoj模板题)。
- 套路1: 构造(构造一个序列或者能将一个无向图定向使得入度和出度差不超过1)。
- 平面上有n个点，讲这些点红蓝染色使得每行每列红蓝点个数的差不超过1。
- 生成2^n的01串(这个串头尾相连)，使得所有长度为n的01串都出现过。
- 套路2: 根据充要条件处理度数(有向图只要出度=入度，无向图只要连通且度数为偶数)。
- 无向图加最少的边使得整个图成为欧拉图。
- 混合图欧拉欧拉回路。
- 更一般的无向图定向需要使用网络流。

# 欧拉回路

- 一般来说欧拉回路是很容易被忽略的考点。
- 但是首先欧拉图的性质很好，只和每个点的度数的奇偶性有关。
- 并且欧拉回路的构造方法不是很平凡，所以一般需要注意到了模型之后才能容易解决。
- 今年的CEOI和IOI中都出现了欧拉回路的考题。
- 当然要出一道欧拉回路的好题对出题人来说也是很挺大的挑战。

# 2-SAT

- 需要掌握2-SAT的做法以及输出方案的方法。
- 套路1: 建立辅助变量优化建图。
- 比如a=b or c那么可以连边a’->b’, a’->c’, b->a, c->a。但是要注意这个东西不是严格的or，因为当b=0,c=0,a=1的时候是合法的。
- 会结合一些数据结构的常见手段(线段树 ST表 前缀后缀)。
- 比如一些变量至多只有一个可以为1的建图方法。
- 这个技巧在最短路/网络流建模中都会被用到。

- 套路2: 对于变量取值，拆出一堆x<=1,x<=2,x<=3,...,这样的变量记作p1,p2,p3。
- 然后连p2->p1,p1’->p2’,...这样保证前面的取的是1，后面的取的是0。
- 然后再根据不等式之类的限制连边。
- 这个建图在最小割中也会用到。
- 套路3: 额外加一个限制，判断可不可行之类的。
- 相当于在这个图中加了两条边，判断x与x’可不可达，转化到可达性问题。

- 一般来说2-SAT建模的难度和最小割比较类似(每个变量取0/1可以类比每个点取到S集还是T集)。但是最小割不能搞出p在T集推出q在S集这样的限制。
- 但是在2-SAT建模比较不常见，所以大家往往会忽视，以及在比赛中不容易想到。

# Dominator Tree

- dls。。。暂时不会。。。大家自行学习一下啊。。。
- 立个flag。。。感觉。。。这一年内不会考啊= =
- 等我学会了我就把这个flag收了。。。

# 最短路

- 基本算法大家应该都掌握的。
- 套路: 删点/边问最短路的。
- 删边最短路一般来说都是考虑最短路径树，然后考虑每条非树边当成替换边之类的。
- 删点可以使用分治解决。
- 边权小的话可以把堆换成桶解决。
- 有一些变种，比如边权和距离有关之类的，但是也可以用dijkstra的思想每次确定最小的解决。

- 有些奇怪的最短路题，比如某些参数的不知道，告诉你某些点之间的最短路的范围，然后求边权或者之类的。
- 有些题目是能通过一些精妙的分析解决。
- 这个时候。。。一般可以写成线性规划。。。直接使用simplex overkill一下就行了 = =

# K短路

- 虽然。。。dls。。。不会。。。。但是还是可能会考的= =
- 比如求balabala的第k优解之类的。。。
- 大家自行学习一下吧

# 差分约束系统

- 对于s(v)<=s(u)+w(u,v)的限制建图，然后求解。
- 一般来说根据题意把变量和限制写出来，如果每个限制都只某两项的差有关，可以使用差分约束解决。
- 考虑前缀和也是常见的手段。

# 最小生成树

- 掌握Prim/Kruskal/Boruvka算法。
- 套路1: 两点之间路径的边/点权最大值最小。
- 套路2: 动态维护最小生成树(加边/删边)。
- 动态图中可以把每条边的删除时间当成边权维护最大生成树。
- 套路3: 完全图，每条边的边权为w(u) op w(v)。其中op为某个运算符，比如and or xor gcd之类的。
- 一般来说这样的题目可以使用Prim或者Boruvka算法，然后使用数据结构维护当前点集到其他点的最短边，或者去掉其中没有用的边。

# 最小树形图

- 需要掌握朱刘算法及输出方案。
- 虽然最小树形图的题目不多，但是还是能出一些有趣的题的。
- 不过感觉考的可能性也不是很大。

# Steiner Tree

- 掌握一下O(3^kn+2^kmlog m)的方法。
- 以备不(tu)时(ran)之(kao)需(dao)。

# 网络流

- 最好能掌握多种费用流算法，以及掌握有负环/上下界的费用流。
- 网络流题有比较多的套路，时间太少不再展开。
- 但是由于现在选手们对网络流的套路比较熟悉，要出一个新的套路是比较困难的。
- 所以最近的比赛中网络流出现的渐渐变少，并且网络流一般和别的东西一起考。
- 几个比较难的套路:
- 套路0: 有负环/上下界的最大/最小/费用/循环流。
- 因为很多选手可能激动地建出了图发现不会处理这些东西然后狗带了。

- 套路1: 建立线性规划的式子然后对偶成为费用流。
- 这种题目一般直接simplex不会比对偶然后费用流慢多少，而且simplex还能输出方案。背一个带初始解的simplex能使你生活更加轻松。
- 套路2: 建立一堆辅助变量变成等式，然后通过一些高明的技巧使得每个东西一正一负出现在这些等式里，然后根据流量平衡见图。
- 这虽然是比较老的套路了，但是还是很困难，所以碰到这样的题目，还是直接simplex吧。
- 一般来说这些LP-based的网络流题都是费用流。因为最大流的对偶是最小割，处理的方法比较套路。而费用流的运行速度和simplex也差不了多少，所以个人还是比较推荐用无脑simplex流。

# 匹配

- 二分图最大匹配: Hungarian/Hopcroft-Karp/Dinic
- 二分图最大权匹配: KM/费用流
- 一般图最大匹配: 带花树
- Hall定理 Konig定理 Dilworth定理

- 远古套路0: 根据Konig定理，用最大匹配求最小覆盖最大独立集之类的。
- 如果点带权也可以用网络流解决。
- 套路1: 将在匹配中的边从右往左连，不在匹配中的边从右往左连，那么一条增广路就变成从一个未匹配点到匹配点的完整的路径。
- 用这个方法可以求每个点/边是不是一定/可能/不可能在最大匹配里。

- 套路2: Hall定理!
- 因为Hall定理的描述中有一个子集，所以很多情况下是不能直接使用的。
- 它的优点是可以把整个图转化成若干个子集，而每个子集的约束都是很简单的。
- 在有些和匹配相关的题目中没有多项式算法，考虑Hall定理只需要考虑2^n个子集就能知道是否有完备匹配，而不需要考虑n!种匹配方法。
- 当连边有一些特殊规律的时候我们也可以把需要检验的集合降到多项式级别。
- 比如左边每个点连的点为一个区间，这个时候只要检验O(n^2)个区间即可。

- 套路3: 匹配中的拟阵结构。
- 对于X中的点集S和T，如果S中的点和T中的点都可以匹配，且|S|>|T|，那么一定能在S中找到一个不在T中的点u，然后T∪{u}也是可以完全匹配的。
- 所以要找一个匹配使得匹配中点点权最大，可以使用贪心解决。
- 根据这个性质可以在线性时间内完成修改一个点点权，维护最大匹配。
- 一定是和一个点交换，所以只要找到能增广的权值最大/最小的点即可。
- 套路4: 如果有一堆s(u)+s(v)>=w(u,v)这样的限制可以转化成KM算法求顶标。

- 因为匹配比网络流来得简单，所以可以挖掘一些匹配本身更深的性质。
- 我觉得还是能挖出一些点出一些比较难的题目。
- 而一般图匹配建模一般比较困难(因为题目很少不能形成一些套路, 并且一般图匹配不能基于LP，所以一般都很non-trivial)。

# 仙人掌

- 需要掌握基本的仙人掌处理和dp。
- 特殊情况为环+外向树。
- 仙人掌上的数据结构。。。大家自行学习一下吧。。。

# 平面图

- 平面图可以结合计算几何来考。
- 比如给你一个平面图让你把它建出来，还能结合点定位之类的计算几何内容。
- 同时它又有比较好的性质，最重要的是对偶。
- 割可以对偶成一条路径。
- 对对偶图建生成树原图中一个环相当于在对偶图中切掉了若干树边。
- 特殊情况为网格图，可以在上面考虑分治之类的。

# 弦图

- 总共见到的弦图题。。。不超过4道。= =