两个冷门图论算法

匡正非 黄志翱

长沙市雅礼中学

February 8, 2014

自我介绍

匡正非

- 2012年接触OI
- NOIP2013 满分
- NOI2013 金牌，进入国家集训队
- CODEFORCES 红名爷

黄志翱

- NOIP2011 二等奖
- HNOI2012 一等奖
- APIO2013 铜牌

某道NOIP模拟题

给定n个点，每次删除一条边，添加一条边，或者询问是否存在连接任意两点之间的路径。即动态维护图的联通性。

某道NOIP模拟题

给定n个点，每次删除一条边，添加一条边，或者询问是否存在连接任意两点之间的路径。即动态维护图的联通性。

看起来不像是NOIP模拟题吧。。

某道NOIP模拟题

给定n个点，每次删除一条边，添加一条边，或者询问是否存在连接任意两点之间的路径。即动态维护图的联通性。

看起来不像是NOIP模拟题吧。。

$NP + OI \triangleq NOIP$

某道NOIP模拟题

给定n个点，每次删除一条边，添加一条边，或者询问是否存在连接任意两点之间的路径。即动态维护图的联通性。

看起来不像是NOIP模拟题吧。。

$NP + OI \triangleq NOIP$

超过半数的人拿了满分。

为什么超过半数的同学拿了满分呢？

为什么超过半数的同学拿了满分呢？
因为n，m不大于1000。

为什么超过半数的同学拿了满分呢？
因为n，m不大于1000。
为何剩下的半数同学没拿满分?

为什么超过半数的同学拿了满分呢？

因为n，m不大于1000。

为何剩下的半数同学没拿满分?

lol

几种简单的情况

1. N很小

几种简单的情况

1. N很小——。。。

几种简单的情况

1. N很小——。。。
2. N比较小

几种简单的情况

1. N很小——。。。
2. N比较小——bitset

几种简单的情况

1. N很小——。。。
2. N比较小——bitset
3. 稀疏图

几种简单的情况

1. N很小——。。。
2. N比较小——bitset
3. 稀疏图——维护一棵生成树，暴力

几种简单的情况

1. N很小——。。。
2. N比较小——bitset
3. 稀疏图——维护一棵生成树，暴力
4. 只加边

几种简单的情况

1. N很小——。。。
2. N比较小——bitset
3. 稀疏图——维护一棵生成树，暴力
4. 只加边——并查集O(N)维护

几种简单的情况

1. N很小——。。。
2. N比较小——bitset
3. 稀疏图——维护一棵生成树，暴力
4. 只加边——并查集O(N)维护
5. 树

几种简单的情况

1. N很小——。。。
2. N比较小——bitset
3. 稀疏图——维护一棵生成树，暴力
4. 只加边——并查集O(N)维护
5. 树——动态树维护

进入正题

有没有比$O(N^2)$更优的普适性算法？

想要优化算法。。

找重复是关键（快去问法法塔）

想要优化算法。。

找重复是关键（快去问法法塔）

我们发现，加入或删除一条边时，整个图所受的影响很小。如果我们能把图分成一些部分分开维护的话

想要优化算法。。

找重复是关键（快去问法法塔）

我们发现，加入或删除一条边时，整个图所受的影响很小。如果我们能把图分成一些部分分开维护的话

重复状态会少很多。怎么分？维护什么？

维护什么？

因为要求的是两点之间是否联通，不由而然会联想到生成森林。

维护什么？

因为要求的是两点之间是否联通，不由而然会联想到生成森林。

树上的连通性问题很好解决

关键在于非树边的处理

一个想法——分治

既然问题关键在于非树边的处理，可不可以直接利用边将整个图分开？

考虑分治，问题的范围随着递归深度增加而减少

一个想法——分治

既然问题关键在于非树边的处理，可不可以直接利用边将整个图分开？

考虑分治，问题的范围随着递归深度增加而减少

给边标上号（即视为递归深度）

一个想法——分治

既然问题关键在于非树边的处理，可不可以直接利用边将整个图分开？

考虑分治，问题的范围随着递归深度增加而减少

给边标上号（即视为递归深度）

对于每种标号都求一遍生成森林

一个想法——分治

既然问题关键在于非树边的处理，可不可以直接利用边将整个图分开？

考虑分治，问题的范围随着递归深度增加而减少

给边标上号（即视为递归深度）

对于每种标号都求一遍生成森林

不同的标号对应的森林的联通块大小不同（即视为问题范围）

经过一些改进之后

新的算法出现了

经过一些改进之后

新的算法出现了

对每条边都记一个level(以下简称lev)，lev的取值范围为$0 \rightarrow logN$(类比递归深度)，每条边的lev只可能减小。

经过一些改进之后

新的算法出现了

对每条边都记一个level(以下简称lev)，lev的取值范围为$0 \to logN$(类比递归深度)，每条边的lev只可能减小。

令G_i表示边权$\leq i$的所有边组成的图，F_i表示G_i上的最小生成森林

经过一些改进之后

新的算法出现了

对每条边都记一个level(以下简称lev)，lev的取值范围为$0 \to logN$(类比递归深度)，每条边的lev只可能减小。

令G_i表示边权$\leq i$的所有边组成的图，F_i表示G_i上的最小生成森林

规定F_i是F_{i+1}的子集，且F_i中任一联通块的大小都不大于2^i

增加一条边$e = (u, v)$

直接将e的lev设为$logN$(类比递归深度为0)，然后尝试将此边插入F_{logN}

判断u,v是否联通

即为判断u,v是否在F_{logN}中联通

删除一条边$e = (u, v)$

删除一条边$e = (u, v)$

求出$x = lev_e$

删除一条边$e = (u, v)$

求出$x = lev_e$

设u所在的生成树为T_u，v同样

删除一条边$e = (u, v)$

求出$x = lev_e$

设u所在的生成树为T_u，v同样

我们不知道删掉e是否会使得u,v不再联通，所以要找出是否有另一条边g，使得$lev_g \geq x$，且g连接T_u, T_v

删除一条边$e = (u, v)$

求出$x = lev_e$

设u所在的生成树为T_u，v同样

我们不知道删掉e是否会使得u,v不再联通，所以要找出是否有另一条边g，使得$lev_g \geq x$，且g连接T_u, T_v

不妨设$|T_u| <= |T_v|$,则$|T_u| <= 2^{x-1}$

删除一条边$e = (u, v)$

求出$x = lev_e$

设u所在的生成树为T_u，v同样

我们不知道删掉e是否会使得u,v不再联通，所以要找出是否有另一条边g，使得$lev_g \geq x$，且g连接T_u, T_v

不妨设$|T_u| <= |T_v|$,则$|T_u| <= 2^{x-1}$

我们拿出所有$lev = x$且一个端点在T_u中的边（可能是非树边），然后做以下步骤

删除一条边$e = (u, v)$

删除一条边$e = (u, v)$

1. 如果这条边属于T_u，将其lev改为x-1。

删除一条边$e = (u, v)$

1. 如果这条边属于T_u，将其lev改为x-1。
2. 如果这条边是第一条连接T_u, T_v的边，将其加入$F_x, F_{x+1} \ldots F_{logN}$。

删除一条边$e=(u,v)$

1. 如果这条边属于T_u，将其lev改为x-1。
2. 如果这条边是第一条连接T_u, T_v的边，将其加入$F_x, F_{x+1} \dots F_{logN}$。
3. 否则这条边的另一个端点必然在T_u中（为什么），如果还未出现情况2，则将其lev改为x-1，不需要改变任何生成树（再想想为什么）。

删除一条边$e = (u, v)$

1. 如果这条边属于T_u，将其lev改为x-1。
2. 如果这条边是第一条连接T_u, T_v的边，将其加入$F_x, F_{x+1} \dots F_{logN}$。
3. 否则这条边的另一个端点必然在T_u中（为什么），如果还未出现情况2，则将其lev改为x-1，不需要改变任何生成树（再想想为什么）。

如果所有边都不满足情况2，那么$x = x + 1$,然后再做一遍。

删除一条边$e = (u, v)$

1. 如果这条边属于T_u，将其lev改为x-1。
2. 如果这条边是第一条连接T_u, T_v的边，将其加入$F_x, F_{x+1} \dots F_{logN}$。
3. 否则这条边的另一个端点必然在T_u中（为什么），如果还未出现情况2，则将其lev改为x-1，不需要改变任何生成树（再想想为什么）。

如果所有边都不满足情况2，那么$x = x + 1$,然后再做一遍。

可以发现情况1和2,3可以分开处理，然后对于后者，一旦遇到情况2立即停止扫描。这样我们就能保证每次最多只会扫到一条不需要改变权值的边了。

删除一条边$e = (u, v)$

删除一条边$e = (u, v)$

这样做完后，在F_{x-1}中有可能出现一个新的联通块T_u，但因为$|T_u| \le 2^{x-1}$，所以这样并无大碍。

删除一条边$e = (u, v)$

这样做完后，在F_{x-1}中有可能出现一个新的联通块T_u，但因为$|T_u| \leq 2^{x-1}$，所以这样并无大碍。

可以发现，没有边的权值会≤ 0(显然)，所以每条边最多被减$O(logN)$次权值。

删除一条边$e = (u, v)$

这样做完后，在F_{x-1}中有可能出现一个新的联通块T_u，但因为$|T_u| \leq 2^{x-1}$，所以这样并无大碍。

可以发现，没有边的权值会≤ 0(显然)，所以每条边最多被减$O(logN)$次权值。

剩下的就是一些具体操作的问题：

询问一条边的权值，连接两棵树，分离两棵树，询问两点是否在同一生成树上，以及快速找出所有$lev = x$且一个端点在T_u中的边。

具体的实现

记录一条边的lev可以用map（CP的战争）

具体的实现

记录一条边的lev可以用map（CP的战争）

维护生成树大小和根（用来判断是否在同一生成树上），且支持插入删除的结构——欧拉遍历树

具体的实现

记录一条边的lev可以用map（CP的战争）

维护生成树大小和根（用来判断是否在同一生成树上），且支持插入删除的结构——欧拉遍历树

于是乎差不多了

时空复杂度

空间复杂度什么的先抛一边。

时空复杂度

空间复杂度什么的先抛一边。

显然一条边只会被减$O(logN)$次权值，再算上Splay的$O(NlogN)$，这里的复杂度是$O(Nlog^2N)$的，也是算法瓶颈所在。

时空复杂度

空间复杂度什么的先抛一边。

显然一条边只会被减$O(logN)$次权值，再算上Splay的$O(NlogN)$，这里的复杂度是$O(Nlog^2N)$的，也是算法瓶颈所在。

其他什么的都不会达到$O(Nlog^2N)$(除非你用了一些set)。

时空复杂度

空间复杂度什么的先抛一边。

显然一条边只会被减$O(logN)$次权值，再算上Splay的$O(NlogN)$，这里的复杂度是$O(Nlog^2N)$的，也是算法瓶颈所在。

其他什么的都不会达到$O(Nlog^2N)$(除非你用了一些set)。

所以总共是$O(Nlog^2N)$。

等等！

欧拉遍历树是个什么东西？

Euler-tour Tree

大家都学过Euler-tour吧（我有特别的遍历技巧）（很像DFS序）

Euler-tour Tree

大家都学过Euler-tour吧（我有特别的遍历技巧）（很像DFS序）

但是DFS序要求有根，Euler-tour不要求。

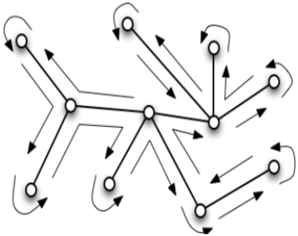
Euler-tour Tree

大家都学过Euler-tour吧（我有特别的遍历技巧）（很像DFS序）

但是DFS序要求有根，Euler-tour不要求。

为了帮助理解动手画(截)一个

- Euler Tour of T:

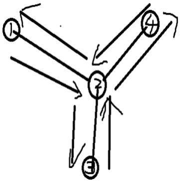


Euler-tour Tree

而Euler-tour Tree就是一棵用来维护Euler-tour经过的点的序列的树。注意这个序列的本质是一个环，我们需要指定一个起点：

Euler-tour Tree

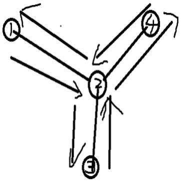
而Euler-tour Tree就是一棵用来维护Euler-tour经过的点的序列的树。注意这个序列的本质是一个环，我们需要指定一个起点：



这里如果把点1作为起点，然后按逆时针遍历的话，那么Euler-tour即为1-2-3-2-4-2

Euler-tour Tree

而Euler-tour Tree就是一棵用来维护Euler-tour经过的点的序列的树。注意这个序列的本质是一个环，我们需要指定一个起点：



这里如果把点1作为起点，然后按逆时针遍历的话，那么Euler-tour即为1-2-3-2-4-2

有趣的是每个序列上的点实际都代表了一条由无向边拆成的有向边。一个点可能出现一次，但是一条有向边只会被代表一次。

Euler-tour Tree

因为该算法没有单独对生成树的子树的操作，不需要考虑父子伦理的问题，直接记录Euler-tour Tree就能完成所需操作了。

Euler-tour Tree

因为该算法没有单独对生成树的子树的操作，不需要考虑父子伦理的问题，直接记录Euler-tour Tree就能完成所需操作了。

具体实现用Splay即可（据说B树可以做到更优）。

一些例题

一些例题

1. WC2005 dface：网格图，删掉一条边，询问两点的连通性

一些例题

1. WC2005 dface：网格图，删掉一条边，询问两点的连通性
2. AHOI2013 某题：删掉四条边，询问图的连通性

一些例题

1. WC2005 dface：网格图，删掉一条边，询问两点的连通性
2. AHOI2013 某题：删掉四条边，询问图的连通性
3. 某个不知来源的测试题：给定一个图，每次询问删除一个点或一条边后，某两点是否联通

一些例题

1. WC2005 dface：网格图，删掉一条边，询问两点的连通性
2. AHOI2013 某题：删掉四条边，询问图的连通性
3. 某个不知来源的测试题：给定一个图，每次询问删除一个点或一条边后，某两点是否联通

其实都可以用之前的做法在保证正确性的基础上TLE。

代码复杂度？

曾经有个神犇写了600+行最后放弃了。

代码复杂度？

曾经有个神犇写了600+行最后放弃了。
本逗比抱着试一试（享受被虐）的心态去写了写

代码复杂度？

曾经有个神犇写了600+行最后放弃了。

本逗比抱着试一试（享受被虐）的心态去写了写

没想到真写出来了

代码复杂度？

曾经有个神犇写了600+行最后放弃了。

本逗比抱着试一试（享受被虐）的心态去写了写

没想到真写出来了

调了大半个寒假（过年业余活动），虽然到现在还有很多$O(N^2)$的脚手架，但是主体部分已经正确了。

代码复杂度？

曾经有个神犇写了600+行最后放弃了。

本逗比抱着试一试（享受被虐）的心态去写了写

没想到真写出来了

调了大半个寒假（过年业余活动），虽然到现在还有很多$O(N^2)$的脚手架，但是主体部分已经正确了。

虽然花费了很多时间，但回想起来还是挺爽的

代码复杂度？

曾经有个神犇写了600+行最后放弃了。

本逗比抱着试一试（享受被虐）的心态去写了写

没想到真写出来了

调了大半个寒假（过年业余活动），虽然到现在还有很多$O(N^2)$的脚手架，但是主体部分已经正确了。

虽然花费了很多时间，但回想起来还是挺爽的

今天我们有幸将这份代码请到了现场。。。。

如果您对以上的内容有所疑问，或者是觉得代码好长好长好长好长的

如果您对以上的内容有所疑问，或者是觉得代码好长好长好长好长的

我给大家讲个恐怖故事吧

如果您对以上的内容有所疑问，或者是觉得代码好长好长好长的

我给大家讲个恐怖故事吧

HZA要讲算法了。。。。

————分割线————

四个经典问题

四个经典问题

- 给定一个二分图，求最大匹配

四个经典问题

- 给定一个二分图，求最大匹配
- 给定一个图，求最大匹配

四个经典问题

- 给定一个二分图，求最大匹配
- 给定一个图，求最大匹配
- 给定一个带权二分图，求最大权匹配

四个经典问题

- 给定一个二分图，求最大匹配
- 给定一个图，求最大匹配
- 给定一个带权二分图，求最大权匹配
- 给定一个带权图，求最大权匹配

一些定义

令无向图$G=(V,E)$，V为点集，E为边集。

$M(M\in E)$为图G的匹配，当且仅当M中任意两条边没有公共端点。

一个图为二分图，当且仅当这个图可以分成两部分，而不存在一条边连接同一部分的两点。

若给每条边一个边权，则最大权匹配即是找一个边权和最大的匹配。

增广路算法

增广路算法

若P是图G中一条连通两个未匹配顶点的路径，并且属于M的边和不属于M的边在P上交替出现，则称P为相对于M的一条增广路径。

算法轮廓

算法轮廓

- 置M为空

算法轮廓

- 置M为空
- 找出一条增广路径P，通过异或操作获得更大的匹配M’代替M

算法轮廓

- 置M为空
- 找出一条增广路径P，通过异或操作获得更大的匹配M’代替M
- 重复操作2直到找不出增广路径为止

如何寻找增广路？

二分图最大匹配

二分图最大匹配

二分图不会出现奇环。从X部一个未匹配点开始进行dfs。

二分图最大匹配

二分图不会出现奇环。从X部一个未匹配点开始进行dfs。

1. 设当前点为u，枚举从u出发到达的每个点v。

二分图最大匹配

二分图不会出现奇环。从X部一个未匹配点开始进行dfs。

1. 设当前点为u，枚举从u出发到达的每个点v。
2. 若v未匹配，则令u,v匹配。

二分图最大匹配

二分图不会出现奇环。从X部一个未匹配点开始进行dfs。

1. 设当前点为u，枚举从u出发到达的每个点v。
2. 若v未匹配，则令u,v匹配。
3. 否则从v的匹配点开始dfs，寻找增广路。

二分图最大匹配

二分图不会出现奇环。从X部一个未匹配点开始进行dfs。

1. 设当前点为u，枚举从u出发到达的每个点v。
2. 若v未匹配，则令u,v匹配。
3. 否则从v的匹配点开始dfs，寻找增广路。

显然，在一次寻找增广路的算法中，X部的每个点最多只会遍历一遍。时间复杂度是O($n+m$)的。

任意图呢？

任意图呢？

在寻找增广路的时候自然会对点奇偶标号。

任意图呢？

在寻找增广路的时候自然会对点奇偶标号。
对图的奇偶标号不同，增广路的情况不同。

任意图呢？

在寻找增广路的时候自然会对点奇偶标号。
对图的奇偶标号不同，增广路的情况不同。
——平衡规划！

带花树算法

1. 依旧是从一个偶点u出发寻找增广路，并对点按照奇偶标号

带花树算法

1. 依旧是从一个偶点u出发寻找增广路，并对点按照奇偶标号
2. 枚举从u出发到达的每个点v

带花树算法

1. 依旧是从一个偶点u出发寻找增广路，并对点按照奇偶标号
2. 枚举从u出发到达的每个点v
3. 若点v未访问，且未匹配，则找到了增广路

带花树算法

1. 依旧是从一个偶点u出发寻找增广路，并对点按照奇偶标号
2. 枚举从u出发到达的每个点v
3. 若点v未访问，且未匹配，则找到了增广路
4. 若点v未访问，但点v与v’匹配，则将v标记为奇点，v’标记为偶点，从v’开始寻找增广路

带花树算法

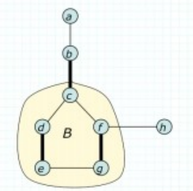
1. 依旧是从一个偶点u出发寻找增广路，并对点按照奇偶标号
2. 枚举从u出发到达的每个点v
3. 若点v未访问，且未匹配，则找到了增广路
4. 若点v未访问，但点v与v’匹配，则将v标记为奇点，v’标记为偶点，从v’开始寻找增广路
5. 若点v已访问，且v为奇点，则忽略

带花树算法

1. 依旧是从一个偶点u出发寻找增广路，并对点按照奇偶标号
2. 枚举从u出发到达的每个点v
3. 若点v未访问，且未匹配，则找到了增广路
4. 若点v未访问，但点v与v’匹配，则将v标记为奇点，v’标记为偶点，从v’开始寻找增广路
5. 若点v已访问，且v为奇点，则忽略
6. 若点v已访问，且点v为偶点？出现奇环

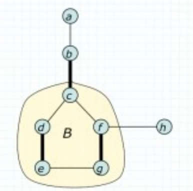
带花树算法

可以发现一个奇环可以等价为一个点。



带花树算法

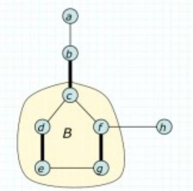
可以发现一个奇环可以等价为一个点。



将奇环缩成一朵花！并在新的图中寻找增广路。

带花树算法

可以发现一个奇环可以等价为一个点。



将奇环缩成一朵花！并在新的图中寻找增广路。

寻找增广路的过程会形成一棵树。

带权呢？

将算法修改为：每次寻找权值最大的增广路。

带权呢？

将算法修改为：每次寻找权值最大的增广路。

时间复杂度？

二分图最大权匹配

将一条匹配边看成流，一边连S，一边连T。最大费用流。

二分图最大权匹配

将一条匹配边看成流，一边连S，一边连T。最大费用流。

是不是显得太弱了呢？

KM算法

Theorem

二分图G和匹配M，每个点有个非负整数点权z_u，若

1. $z_e \geq w(e)$ *for* $e \in E \quad (z_e = z_u + z_v)$
2. $z_e = w(e)$ *for* $e \in M$
3. $z_u = 0$ 如果点u没有被匹配。

那么M是图G的最大权匹配。其中$w(e) = z(e)$的边称为*equality edges*。

KM算法

Theorem

二分图G和匹配M，每个点有个非负整数点权z_u，若

1. $z_e \geq w(e)$ *for* $e \in E \quad (z_e = z_u + z_v)$
2. $z_e = w(e)$ *for* $e \in M$
3. $z_u = 0$ 如果点u没有被匹配。

那么M是图G的最大权匹配。其中$w(e) = z(e)$的边称为*equality edges*。

通过修改标号，不断将边加入相等子图中，寻找最大匹配。

证明？

任意图呢？

任意图最大权匹配

可以仿照二分图得到如下定理：

Theorem

图G和匹配M，每个奇数点集有个非负整数点权z_B。对于边$e=(u,v)$，有$z_e=\sum_{B|e\in BorB\in\{u,v\}} z_B$。

1. $z_e \geq w(e)$ *for* $e \in E$
2. $z_e = w(e)$ *for* $e \in M$
3. $z_B = 0$ B是一个点，且点B没有被匹配，或者B是集合，且B中匹配边个数小于$\frac{|B|-1}{2}$。

那么M是图G的最大权匹配。其中$w(e)=z(e)$的边称为*equality edges*。

同样是不断调整标号，在相等子图中寻找增广路。

同样是不断调整标号，在相等子图中寻找增广路。

显然我们不可能为$2^{|V|-1}$个奇数子集维护标号，我们只维护Z_B不为0的集合——即匹配边数恰好达到$\frac{|B|-1}{2}$的点集。

同样是不断调整标号，在相等子图中寻找增广路。

显然我们不可能为$2^{|V|-1}$个奇数子集维护标号，我们只维护Z_B不为0的集合——即匹配边数恰好达到$\frac{|B|-1}{2}$的点集。

考虑带花树算法：这不就是花么？

同样是不断调整标号，在相等子图中寻找增广路。

显然我们不可能为$2^{|V|-1}$个奇数子集维护标号，我们只维护Z_B不为0的集合——即匹配边数恰好达到$\frac{|B|-1}{2}$的点集。

考虑带花树算法：这不就是花么？

为每朵花也维护一个权值。

算法轮廓

算法轮廓

1. 对所有点和所有花维护Z值，最开始$M = \emptyset, Z_u = 0.5 * maxWeight$

算法轮廓

1. 对所有点和所有花维护Z值，最开始$M = \emptyset, Z_u = 0.5 * maxWeight$
2. 如果已经得到最大权匹配，停止算法，否则在equality edges中寻找增广路

算法轮廓

1. 对所有点和所有花维护Z值，最开始$M = \emptyset, Z_u = 0.5 * maxWeight$
2. 如果已经得到最大权匹配，停止算法，否则在equality edges中寻找增广路
3. 若找不到增广路，修改z值

算法轮廓

1. 对所有点和所有花维护Z值，最开始$M = \emptyset, Z_u = 0.5 * maxWeight$
2. 如果已经得到最大权匹配，停止算法，否则在equality edges中寻找增广路
3. 若找不到增广路，修改z值
4. 若找到增广路，则增广，修改M，并展开z值为0的花

继续使用带花树

继续使用带花树

对带花树上的花和点进行奇偶标号。

BFS

一旦一个点变成偶点，将其插入队列。

BFS

一旦一个点变成偶点，将其插入队列。
注意：队列中只存放节点，而不会存入花。

花有奇偶性，且存在嵌套关系。

具体实现

具体实现

对于每朵花维护：

具体实现

对于每朵花维护：

1. base节点

具体实现

对于每朵花维护：

1. base节点
2. 儿子（子花或者节p点）

具体实现

对于每朵花维护：

1. base节点
2. 儿子（子花或者节p点）
3. 父亲

具体实现

对于每朵花维护：

1. base节点
2. 儿子（子花或者节p点）
3. 父亲
4. 儿子之间的连边

具体实现

对于每朵花维护：

1. base节点
2. 儿子（子花或者节p点）
3. 父亲
4. 儿子之间的连边
5. 权值

具体实现

具体实现

对于个节点维护：

具体实现

对于个节点维护：

1. 最外层的花

具体实现

对于个节点维护：

1. 最外层的花
2. 匹配边（若存在）

具体实现

对于个节点维护：

1. 最外层的花
2. 匹配边（若存在）
3. 权值

label和labelEdge

在一次增广中，对于所有节点维护标号label和在增广树与父亲的连边labelEdge。

label和labelEdge

在一次增广中，对于所有节点维护标号label和在增广树与父亲的连边labelEdge。

对于label为偶数的点，labelEdge要么为空，要么为其匹配边。

label和labelEdge

在一次增广中，对于所有节点维护标号label和在增广树与父亲的连边labelEdge。

对于label为偶数的点，labelEdge要么为空，要么为其匹配边。

建议写一个函数对label和labelEdge同时配置。

缩花

缩花

若相等子图中存在连接两个偶点的边，且两个点在同一棵树中，则必须缩花。

缩花

若相等子图中存在连接两个偶点的边，且两个点在同一棵树中，则必须缩花。

暴力寻找LCA，将LCA设置为新的花的base，然后将花中所有节点丢进队列，将花的label设为偶数。

花的展开

花的展开

只有权值为0的花拆会被拆开。

花的展开

只有权值为0的花拆会被拆开。

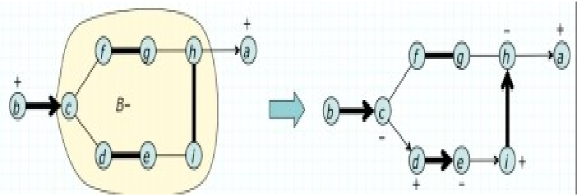
偶花直接拆开，重新标记其中所有节点的最外层点，其余不变。

奇花的展开

奇花的展开

找到这朵花的labelEdge，找到那条边连向的儿子，根据其与base的距离决定按照顺时针还是逆时针，展开成链。

剩余部分的label设为空。（如果这个点可以与带花树相连，最好添加进去）



增广

增广

若两个相邻的偶点u, v不在同一棵树上，则进行增广。

增广

若两个相邻的偶点u, v不在同一棵树上，则进行增广。

分别从u,v点沿着labelEdge走，依次将所有未匹配边修改为匹配边，最后将u和v匹配。

奇花的增广

奇花必须单独处理。why?

奇花的增广

奇花必须单独处理。why？

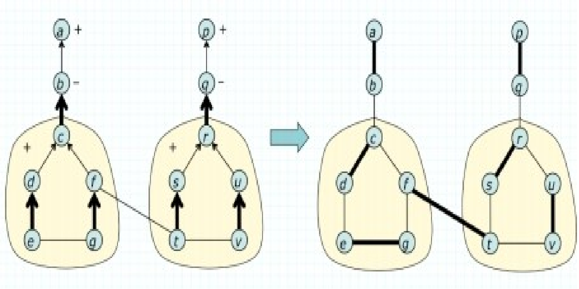
设点v与花外某点匹配，和展开类似，根据与base的距离决定顺逆时针，将其与base之间的所有边异或。

奇花的增广

奇花必须单独处理。why？

设点v与花外某点匹配，和展开类似，根据与base的距离决定顺逆时针，将其与base之间的所有边异或。

将点v设为base，如图：



标号修改

我们得求出最小的delta，使得相等子图中至少加入了一个边或者将一朵花展开。

1. $z_u- = delta$，若u为偶点，反之，$z_u+ = delta$
2. $z_B+ = 2 * delta$，若B为最顶层的偶花，反之，$z_B- = 2 * delta$

注意一个点的奇偶性为其最外层的花的奇偶性。

标号修改

我们得求出最小的delta，使得相等子图中至少加入了一个边或者将一朵花展开。

1. $z_u- = delta$，若u为偶点，反之，$z_u+ = delta$
2. $z_B+ = 2 * delta$，若B为最顶层的偶花，反之，$z_B- = 2 * delta$

注意一个点的奇偶性为其最外层的花的奇偶性。

容易证明，这样做不会将树上或花中以及匹配边从相等子图中删除。

delta的取值

delta的取值

令$e = (u, v)$

delta的取值

令$e = (u, v)$

1. $z_e - w(e)$，若u为偶点，v未访问

delta的取值

令$e = (u, v)$

1. $z_e - w(e)$，若u为偶点，v未访问
2. $\frac{z_e - w(e)}{2}$，若u,v为偶点，且u,v不在同一朵花中。

delta的取值

令$e = (u, v)$

1. $z_e - w(e)$，若u为偶点，v未访问
2. $\frac{z_e - w(e)}{2}$，若u,v为偶点，且u,v不在同一朵花中。
3. $z_B/2$，B是最顶层的奇花。

相等子图的维护

相等子图的维护

暴力$O(n^4)$

相等子图的维护

暴力$O(n^4)$

类似于KM，记录松弛变量$O(n^3)$，或者使用堆$O(nm\log n)$

最大权匹配在现实生活中的应用

最大权匹配在现实生活中的应用

1. 理性愉悦

最大权匹配在现实生活中的应用

1. 理性愉悦
2. 显得自己好厉害的样子

最大权匹配在现实生活中的应用

1. 理性愉悦
2. 显得自己好厉害的样子
3. 用作弱省省选难度的noip模拟题

最大权匹配在现实生活中的应用

1. 理性愉悦
2. 显得自己好厉害的样子
3. 用作弱省省选难度的noip模拟题
4. 用于解决二分图匹配已经不能满足的各种cp问题

无向图的中国邮递员问题

在一个连通的无向图中找到一最短的封闭路径，且此路径需通过所有边至少一次。

无向图的中国邮递员问题

在一个连通的无向图中找到一最短的封闭路径，且此路径需通过所有边至少一次。

如果原图中存在欧拉回路，答案显然。

无向图的中国邮递员问题

在一个连通的无向图中找到一最短的封闭路径，且此路径需通过所有边至少一次。

如果原图中存在欧拉回路，答案显然。

否则添加尽可能少的边使得这个图存在欧拉回路，即添加尽可能少的路径连接原图中度为奇数的点。

边权可以用floyd求出，问题转化为最小权匹配。

thusc 2013 理想国

problem

给定平面图，请求$\frac{m(m-1)}{3}$和最大割

最大割：求图的一个划分S,T，使得S和T之间的连边尽可能少。

thusc 2013 理想国

problem

给定平面图，请求$\frac{m(m-1)}{3}$和最大割

最大割：求图的一个划分S,T，使得S和T之间的连边尽可能少。

即删掉最小的边使得原图中不存在奇环（最小奇环覆盖）。

thusc 2013 理想国

考虑其对偶图，删去一条边相当于连接了其对偶图中对应的两个点，且不改变其度数和的奇偶性。

thusc 2013 理想国

考虑其对偶图，删去一条边相当于连接了其对偶图中对应的两个点，且不改变其度数和的奇偶性。

选择最少的边，使得所有奇数点被连接，且连通块中点的度数和为偶数。

thusc 2013 理想国

考虑其对偶图，删去一条边相当于连接了其对偶图中对应的两个点，且不改变其度数和的奇偶性。

选择最少的边，使得所有奇数点被连接，且连通块中点的度数和为偶数。

显然将奇点配对即为最优解（？），最大权匹配!

感谢

感谢

感谢在座各位耐心的倾听

感谢

感谢在座各位耐心的倾听

感谢ccf提供的机会

感谢

感谢在座各位耐心的倾听

感谢ccf提供的机会

感谢刘研绎神犇的指导

感谢

感谢在座各位耐心的倾听
感谢ccf提供的机会
感谢刘研绎神犇的指导
感谢vfleaking提供了最大权匹配的可运行的python版本

References

范浩强神犇，《无向图匹配的带花树算法》

Zvi Galil，Efficient Algorithms for Finding Maximal Matching in Graphs

演算法笔记，http://www.csie.ntnu.edu.tw/~u91029/Matching.htmlp

Joris van Rantwijk，http://jorisvr.nl/maximummatching.html

F. HADLOCK，FINDING A MAXIMUM CUT OF A PLANAR GRAPH IN POLYNOMIAL TIME

6.851: Advanced Data Structures L20