浅谈图的边染色

中山纪念中学高嘉煊

前言

- 在这个课件中，我将会介绍关于图的边染色问题中的一些算法以及维津定理。
- 内容包括：
- Vizing's Theorem。
- 一般图中的边染色算法(Misra & Gries edge coloring algorithm)。
- 二分图中的边染色算法。
- 欢迎大家指正课件中的错误。

一些概念

- G=(V,E)表示一个图，其中V为G中的点集，E为G中的边集，如果没有特殊说明，将用n=|V|,m=|E|来表示点数和边数。（如果没有特殊说明，接下来本课件中讨论的图G=(V,E)我们都认为是无向图）
- G=(V,E)是正则图，当且仅当每个点的度数都相同，进一步的，如果图中每个点的度数都是d，那么这个图我们称为“d-正则图”
- G=(V,E)是二分图，当且仅当可以把V分成两个集合X和Y，使得X∩Y为空集，X∪Y=V且不存在(u,v)∈E，使得u,v同时属于X或同时属于Y
- G=(V,E)是d-正则二分图，当且仅当它是二分图且图中每个点的度数都是d。

图染色问题

- 图染色问题(Graph coloring)中最出名的有节点染色(Vertex coloring)以及边染色(edge coloring)。
- 点染色的定义为：在图G=(V,E)中，给每个节点x选择一种颜色col_x，使得对于所有(u,v)∈E，都有$col_u \neq col_v$，需要的最少颜色数称为G的色数，记为$\chi(G)$
- 值的一提的是，判断一个图的色数是NP-Complete的，但是在一些特殊的图上是有多项式算法来判定色数的（比如一个图是二分图的充要条件是它的色数不超过2）
- 边染色的定义为：在图G=(V,E)中，给每条边选择一种颜色，使得对于每个节点，与之相邻的边中没有两条边的颜色相同，给一个图边染色需要的最少颜色数为G的最小边染色数[1]。
- 接下来将介绍Vizing's Theorem和边染色中的一些算法。

"最小边染色数"的定义是笔者为了方便下面的描述而定义的。

Vizing's Theorem

- Vizing's Theorem的内容是：在一个简单无向图G=(V,E)中，记Δ为图中的最大度数，那么G的最小边染色数至少为Δ，至多为Δ+1
- （需要注意的是，Vizing's Theorem讨论的是没有重边以及自环的情况，当存在重边的时候，G的最小边染色数可以达到3Δ/2）
- 首先下界Δ很容易证明，因为每个节点相邻的边的颜色需要两两不同。
- 根据Vizing's Theorem，可以把图分成两类，第一类是最小边染色数为Δ的，第二类是最小边染色数为Δ+1的。
- 有一些类型的图，比如二分图，高阶平面图，它们属于第一类。
- 通过接下来的一般图的边染色的构造方法，我们将认识到Vizing's Theorem的上界是Δ+1

一般图边染色

Edge Coloring in Simple Undirected Graph

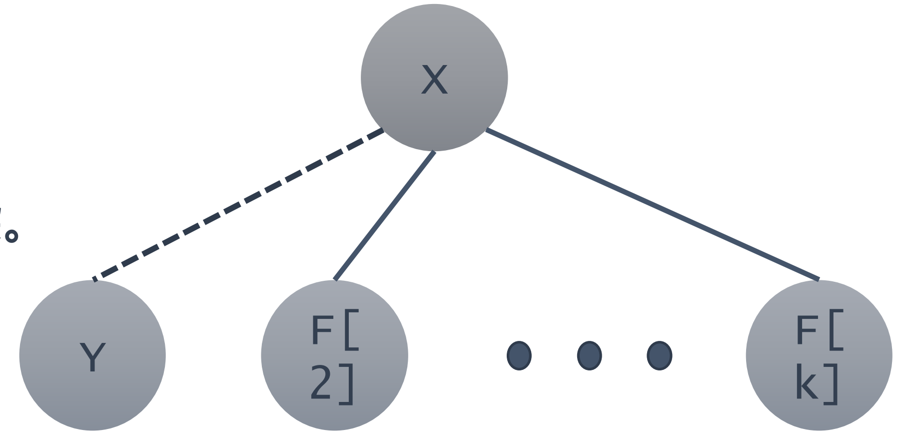
Misra & Gries Edge Coloring Algorithm

- 本算法会对简单无向图G=(V,E)给出一种使用至多Δ+1种颜色进行边染色的染色方案。
- 对于(u,v)∈E，记c(u,v)表示边(u,v)的颜色。
- 我们称一条边(u,v)的颜色x在节点u上是可用的，当且仅当对于所有(u,z)∈E且z≠v有：c(u,v)≠x
- 定义节点u的一个Fan[1]为一个序列F[1:k]，它满足：
  - F[1:k]是一个非空序列，它包含了u的一些不重复的相邻节点
  - (F[1],u)∈E(G)是还未被染色的。
  - 对于1≤i<k,c(F[i+1],u)在节点F[i]上是可用的。
- 一条“ab_x-路径”是指一条经过x的只包含颜色a和颜色b并且极长的路径（由于边染色的限制，这样的路径只有一条）

1、在此并没有对Fan进行翻译，直接沿用了原文。

Misra & Gries Edge Coloring Algorithm

算法流程

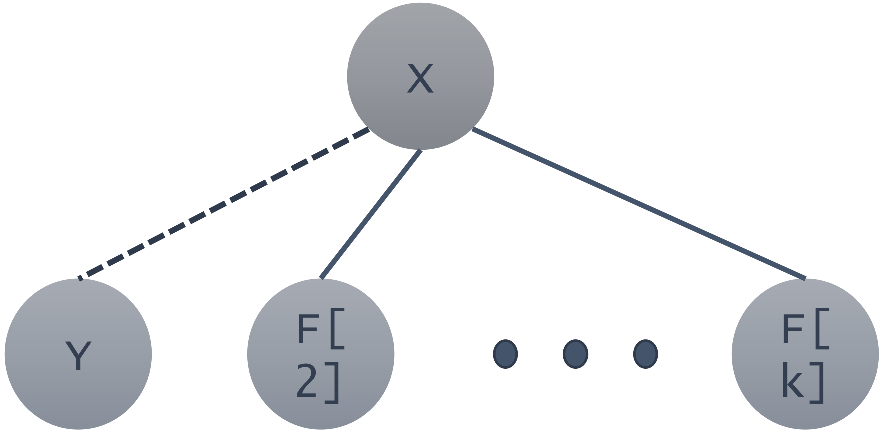


- 下面开始阐述这个算法。
- 一开始，所有的边都是未染色的，不妨考虑一条一条边的染。
- 假设现在要对(x,y)染色。
- 首先求出x的一个以y开头的极长的Fan F[1:k]
- 记a为一种在节点x上可用的颜色，b为一种在节点F[k]上可用的颜色（这样的颜色一定存在）
- 找出“ab_x-路径”，并且将路径中的a替换为b，将b替换为a（我们称为“翻转操作”）。
- 记节点w∈V(G)满足w∈F，F'=[F[1]..w]是一个Fan并且颜色b在节点w上是可用的。
- 然后对于1≤i<|F'|，将c(x,F'[i])变为c(x,F'[i+1])，并将c(x,w)变为b（我们称之为“旋转操作”）.

Misra & Gries Edge Coloring Algorithm

正确性证明

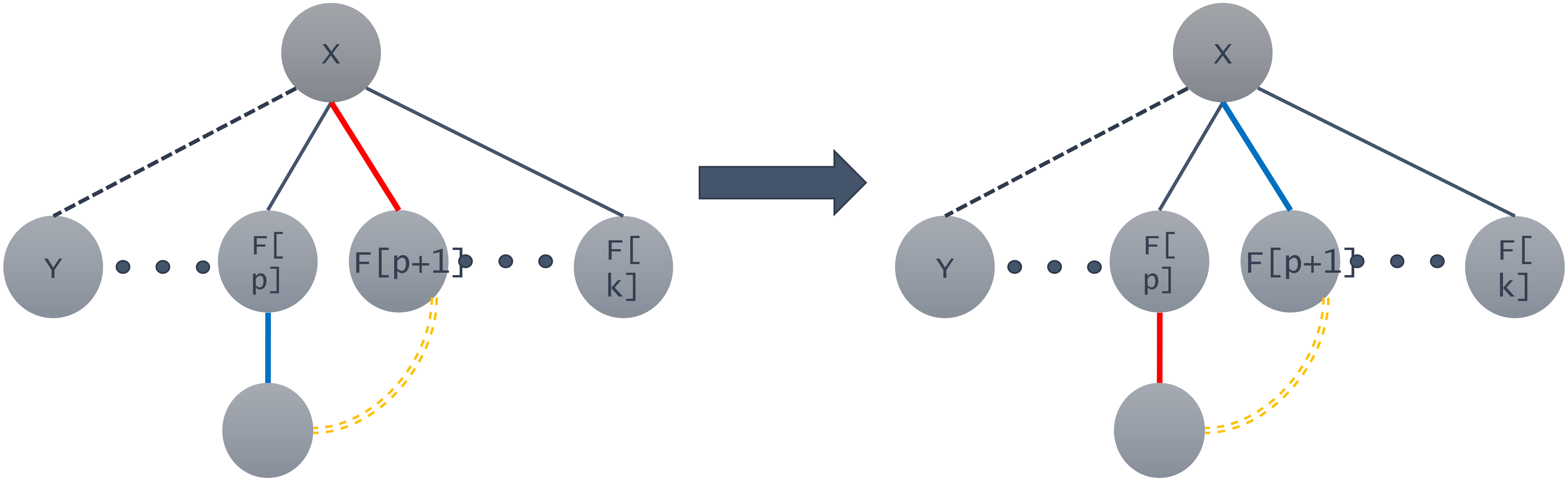
- 下面证明正确性。
- 在F[1:k]中不存在一条边(x,F[i])的颜色是b.
  - 这种情况下，ab$_x$-路径实际上只包含x这一个节点，即翻转操作不会对原来的图产生影响，那么只需要将w设为F[k]即可。（由于F[1:k]是极长的，所以一定是合法的）



Misra & Gries Edge Coloring Algorithm

正确性证明

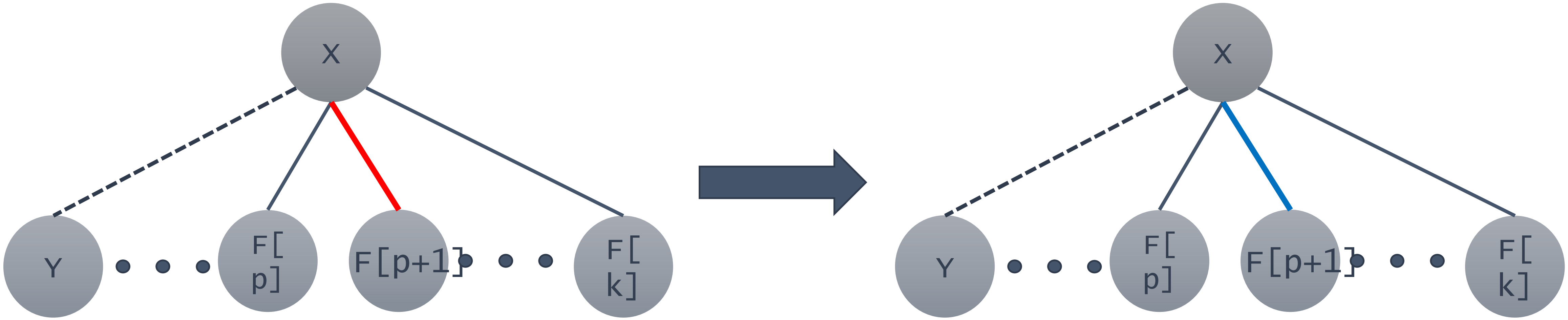
- 在F[1:k]中存在p(0<p<k)满足c(x,F[p+1])=b(那么b在F[p]上是可用的)
  - 若F[p]为ab_x-路径的一个端点，那么F[p]原本相邻的颜色为a的边会变成b，c(x,F[p+1])会从b变成a，将w设为F[k]即可（F[1:k]仍然是一个Fan）



Misra & Gries Edge Coloring Algorithm

正确性证明

- 在F[1:k]中存在p(0<p<k)满足c(x,F[p+1])=b(那么b在F[p]上是可用的)，
  - 若F[p]不是ab_x-路径的端点，操作只会影响(x,F[p+1])的颜色，b在F[p]上仍然是可用的，将w设为F[p]即可



Misra & Gries Edge Coloring Algorithm

正确性证明

- 通过上述构造算法，我们可以知道Vizing's Theorem的正确性。

Misra & Gries Edge Coloring Algorithm

时间复杂度分析

- 据论文以及wikipedia的说法，找一个极大的Fan，进行一次翻转操作以及一次旋转操作的时间复杂度都是O(n)的，所以总的时间复杂度是O(nm)的。
- （然而我并没有想到怎么在O(n)时间内找一个极大的Fan，论文中貌似也并没有提及如何O(n)找一个极大的Fan，如果大家知道正确的方法，欢迎与我交流
- （然而网络上的代码都并没有去找一个极大的Fan

Misra & Gries Edge Coloring Algorithm

更进一步

- 实际上并没有必要找极大的Fan.
- 这个算法里面要求Fan极大的作用：对于在F[k]上可用的颜色b，要么b在x上也是可用的，要么x相连的颜色b的边已经出现在F[1:k]中。
- 于是可以对要求“极大的Fan”进行修改，实际上要求的Fan，只需要满足对于F[k]，有一种在F[k]上可用的颜色b，满足b在x上也是可用的或存在0<i<k满足c(x,F[i])=b
- 对于每个点s维护一个to[s]表示在s上的一种可用的颜色。
- 然后加入(x,y)的时候，只需要沿着to[y]走即可得到要求的Fan
- 对于一次翻转操作，只有路径两端的节点的to会发生变化。
- 对于一次旋转操作，F'[1]的to可以暴力重新找，后面的to可以直接用原来与x相连的边的颜色。
- 这样改了之后，总的时间复杂度就是O(nm)的。

Misra & Gries Edge Coloring Algorithm

更进一步

- 考虑用权值线段树维护每个节点相邻的边中的颜色。
- 不妨将to[x]变为在节点x上可用的标号最小的颜色，这个可以在线段树上面查询得到。
- 考虑一次加边，沿着to走的得到的Fan。首先，得到Fan中除了最后一个节点没有两个节点的to是相同的；其次，由于to[x]是标号最小的颜色，那么就是说这个点的度数至少为to[x]-1。
- 综合上面两点，可以知道，得到的Fan的最大长度为O(√m)级别的（并且Fan的长度也远远不能达到O(√m)级别）。
- 出现的问题是如果仅仅这样做，在精心构造的数据下，虽然无法将Fan卡满，但是有可能将ab_x-路径卡得很长。
- 这样找Fan的复杂度可以变成O(m√m log Δ)

Misra & Gries Edge Coloring Algorithm

写在后面

- 至此，关于一般图的边染色问题的内容已经结束。
- 前面所说的确切算法是O(nm)的，根据Wikipedia，有一个$O(|E|\sqrt{|V|\log|V|})$还没有被公布。

二分图边染色

Edge Coloring in Bipartite Multigraph

二分图边染色

- 在二分图边染色问题中，我们讨论的是存在重边的情况。
- 记最大度数为Δ，那么二分图边染色的最小边染色数为Δ

一个O(nm)的算法

- 这个算法与一般图边染色的算法有一些相像之处，都是先找出两个点各自相连的边中没有的一种颜色，然后，从一个点出发，将这两种颜色的交叉路径翻转，然后就可以染色了。
- 由于是二分图，所以情况并没有一般图那么复杂。

更优秀的算法

- 在二分图边染色问题中，还有时间复杂度为O(m log n),O(m log n+n log n log Δ)...的算法
- 很多算法都是基于分治的，这些算法优化的基本都是在正则二分图中找完美匹配的时间复杂度。

转化

- 将二分图G=(V,E)转化为Δ-正则二分图。
- 首先，对于X中的点，不断地合并两个度数加起来不超过Δ的点，直到不可以合并为止；对于Y做同样的事情。（做完之后，X/Y中度数不超过Δ/2的都只有1个）
- 然后，给点数较小的那部分补上一些孤立点，使得X部和Y部的大小相同。
- 接着，在图中加入一些边，使得每个点的度数都为Δ.
- 那么，加入的点数至多为|V|，加入的边数至多为E（极限情况为，除了度数最小和最大的点之外，每个点度数都是Δ/2）
- 于是转化之后的图的规模与转化前的图的规模是一样的，只需要求出转化后的图的边染色，即可得到原图的边染色。
- 那么接下来讨论的二分图都是X/Y部点数相同的Δ-正则二分图.
- 注意，nΔ=2m，|X|=|Y|=n/2

正则二分图的边染色

- 在正则二分图中，一种边染色方案，实际上相当于将所有边分成Δ组完美匹配（每一种颜色的边都是一组完美匹配）。
- 在Δ=2的特殊情况下，可以通过求欧拉回路来求解。
- 对于一条回路，将回路中的边按照交替拆分成两部分，将所有回路都拆开即可得到
- 类似的，在$\Delta=2^t$的情况下，可以通过求若干次欧拉回路，每次将欧拉回路拆分成两部分，然后分开处理，这样的时间复杂度是$O(m \log \Delta)$。

分治

- 由上面的特殊情况的启发，容易想到可以用分治算法。
- 当Δ为偶数的时候，可以直接利用欧拉回路分成两个部分，求出第一部分的染色方案(最大度数为Δ/2)，记t为最大的t使得2^t小于等于Δ，然后在已经求出的Δ/2组最大匹配中，选出2^t-Δ/2种出来，将这些边放到第二部分，然后第二部分就变成了Δ'=2^t的情况，可以直接用上一页的方法做。
- 当Δ为奇数的时候，要做的就是求出图中的一个完美匹配，然后就可以变成偶数的情况了。
- 所以，时间复杂度是O(T+m log Δ)，其中T是求正则二分图完美匹配的时间复杂度。

正则二分图-完美匹配

O(mΔ)

- 首先，给每条边一个初始权值w(e)=1，那么每个点相邻的边的权值和为Δ。
- 然后，找到图中存在一个边权均非零的环，取一个点作为开头，使得排在奇数的边的边权和小于等于排在偶数的边的边权和，然后将所有排在奇数的边的边权减一，排在偶数的边的边权加一。
- 如此不断重复这个过程，最后一定会出现n/2条权值为Δ的边，这些边即为我们要求的完美匹配
- 证明？（考虑如果有一个点相邻边中有至少两条边的权值非0会出现什么情况。）
- 时间复杂度？
- 势能分析：设势能为$\sum_{e\in E}(w(e))^2$，那么一次操作之后，对于一个环C，他的势能的变化是：

$$\sum_{e\in Odd}(w(e)-1)^2-w(e)^2+\sum_{e\in Even}(w(e)+1)^2-w(e)^2=\left(\sum_{e\in Even}2w(e)+1\right)-\left(\sum_{e\in Odd}2w(e)-1\right)\geq |C|$$

- 时间复杂度为O(mΔ)

正则二分图-完美匹配

O(m log n)

- 记一个M表示我们要求的完美匹配，初始时M为$K_{n/2,n/2}$的任意一组完美匹配（我们称初始时M中的边为虚边）。
- 在G=(V,E)的基础上补上$2^t-\Delta$组M得到G'=(V,E')，那么G'就变成了度数是2的幂次的情况，同样利用欧拉回路，每次分成两部分之后，继续对虚边数量较少的那部分进行划分。最后得到一个匹配之后，将这个匹配作为新的M，继续执行这个过程，直到M中没有虚边。
- 假设M中一开始有k条虚边，那么在进行划分之前，G'中共有$(2^t-\Delta)k$条虚边，每次划分会减少一半，那么由于$2^t-\Delta<\Delta$那么得到的匹配中会有至多k/2条虚边，所以上面过程执行O(log n)次，每次的复杂度为O(m+m/2+m/4+...)=O(m)，这个算法的复杂度为O(m log n)

正则二分图-完美匹配

$O(m+n \log n \log \Delta)$

- 在$O(m\Delta)$的算法里面，可以将边权看成这条边的数量，那么这个算法相当于是在对图进行修改，修改完之后的图的一组完美匹配也是原图的一组完美匹配。
- 用类似的思路，考虑将图进行修改，可以把本质不同的边数的级别降下来。
- 初始所有边的边权都记为1.
- 枚举$i=0..[\log \Delta]$，不断地找出图中边权全是2^i的环，选择环上的一个点作为一个起点，将环上的边按照顺序奇偶，将排在奇数的边删掉，排在偶数的边的边权翻倍。
- 每次到下一层的时候边数至少除以2，所以找环的时间复杂度是$O(m+m/2+m/4..)=O(m)$
- 对于$i=0..[\log \Delta]$，由于边权是2^i的边中没有环，所以第i层边数是$O(n)$的，总边数为$O(n \log \Delta)$
- 同样用前面$O(m \log n)$算法的思路，时间复杂度做到$O(m+n \log n \log \Delta)$

More

- 二分图边染色问题中还有很多优秀的算法（比如O(m log Δ)），有兴趣的同学可以自行了解。
- ~~有(hen)兴(du)趣(liu)的同学可以考虑出题~~

参考资料

- [1]https://en.wikipedia.org/wiki/Edge_coloring
- [2]https://en.wikipedia.org/wiki/Vizing%27s_theorem
- [3]https://en.wikipedia.org/wiki/Misra_%26_Gries_edge_coloring_algorithm
- [4]R. Cole,J. Hopcroft. *On Edge Coloring Bipartite Graphs.* SIAM J. Comput. 11(1982), 540-546
- [5]R. Cole, K. Ost and S. Schirra. *Edge-coloring Bipartite Multigraphs in O(E log D) time*,Combinatorica 21 (2001),5-12
- [6]K. Makino,T. Takabatake and S. Fujishige, *A Simple Matching Algorithm for Regular Bipartite Graphs*, to appear

谢 谢 观 看

T H A N K   Y O U