MOPEC-2010-001图论习题课讲义

# 图论(5) 树(Tree)

魏恒峰 2011 年 6 月 11 日

# 1 Special Topics

## 1.1 树(Tree)

树,即连通无圈之图.

树有几个等价定义.这可能会让一些同学心生畏惧.实际上，一个概念有多个等价定义，往往是件好事情，因为这通常意味着我们对该概念的了解已经比较深入，可以从不同角度来阐释该概念。在解题或寻求新的定理时，我们可以先通过一种定义证明某图是树，然后就可以将其余的等价定义看作树的性质来使用.

在给出树的其它等价定义之前，我们先介绍树的一个性质.该性质虽然简单,但是却为关于树的归纳证明提供了基础.

**Lemma 1.1. 树的归纳性质**

1. $n \geq 2$的树至少有两个叶节点;
2. 对于$n \geq 2$的树,删除任一叶节点,所得图是大小为$n-1$的树.

*Proof.* 树的归纳性质

1. 极大路径法.任取某极大路径$u \rightsquigarrow v$,则两端点$u, v$即为叶节点(即,度数为1.因为无圈).
2. 设原树为$T$,删除叶节点$v(deg(v) = 1)$,得到新图$T'$.$T$为树,故无圈,删除一个顶点不会造成圈，故$T'$无圈;$deg(v) = 1$,故$T$中任意两点$x, y$之间的路径不经过$v$，故$x, y$在$T'$中仍然连通.

□

**Remark 1.2. 树的归纳性质**

引理*1.1*说明我们可以从大小为$n$的树通过添加叶节点来构造大小为$n+1$的树.在我们使用数学归纳法证明与树相关的定理时，该引理提供了一种从$n+1$推到$n$的方法,从而可以使用归纳假设.

**Theorem 1.3. 树的等价定义**

对于$n$阶图$G$，以下各点等价*(*都刻画了$n$阶树*):*

1. *G*连通,无圈*;*
2. *G*连通,边数$m = n - 1$*;*
3. *G*边数$m = n - 1$,无圈*;*
4. *G*无环,任意两点之间存在唯一路径.

下面只给出证明的原理和思路，具体证明过程从略.

*Proof.* **树的等价定义**

注意到,定理1.3中的前三点1,2,3实际上说明了: "连通","无圈"和"边数$m = n - 1$"这三个条件中,如果任知其中两个成立,则可以推出另一个条件成立.

**连通+无圈⇒ 边数$m = n - 1$:** 采用数学归纳法.对顶点数$n$做归纳.该数学归纳法即用到了引理1.1.

**连通+边数$m = n - 1$ ⇒ 无圈:** "破圈法".

**无圈+边数$m = n - 1$ ⇒ 连通:** 反证法.对每个连通分支的边数计数.

关于定理第4点的证明不再赘述. □

**Remark 1.4. 树的等价等价**

实际上,对定理第*2*点的证明过程*"*破圈法*"1.1*说明了每个连通图必有生成树*1.5(*构造性证明*).*下一接我们就将重点放在生成树上*1.2.*

求生成树的算法：设连通图 $G$ 的边集为 $\{e_1, \ldots, e_m\}$。

破圈法($G$)

```
1  T ← G
2  for i ← 1 to m
3       do if  e_i 在 T 的回路上
4            then T ← T − e_i
```

图 1: 求生成树的破圈法

我们用伪码来描述求生成树的破圈法*1(*引自宋方敏老师讲义*):*

与此相对应,还有一种求生成树的算法,称之为*"*避圈法*"2(*同样引自宋方敏老师讲义*)*

避圈法($G$)

```
1  T ← ∅
2  for i ← 1 to m
3       do if G[T ∪ {e_i}]
4            then T ← T ∪ {e_i}
5                 T ← G[T]
6
```

图 2: 求生成树的避圈法

我们不加证明地列出树的一些简单性质.它们都可以容易地从定理1.3的证明过程中得到.

**Corollary 1.5. 树的性质**

*1.* 树的每一条边都是割边.

*2.* 往树中添加一条边,将会形成唯一的圈.

*3.* 每一个连通图都有生成树.

## 1.2 生成树(Spanning Tree)

关于生成树,我们关心如下问题:

1. 任给连通图$G$，存在生成树吗? 此即存在性问题(existence).
2. 如果生成树存在,唯一吗? 如果不唯一, 有多少个? 此即计数问题(Counting).
3. 如果图$G$是带权图,最小生成树唯一吗?
4. 如何求最小生成树?

其中问题1已经有了肯定回答2,并且在1.1中给出了构造性的证明(也就是提供了一种算法).

在回答下面的问题之前，我们先来介绍一个与生成树对(pairs of spanning trees)相关的命题.该命题在许多和生成树相关尤其是与最小生成树相关的证明中都会用到.

我们以习题的形式给出该命题,见习题2.1.

### 1.2.1 生成树的计数问题(Counting)

下面我们考虑生成树的计数问题(问题2).

需要说明的是，在本计数问题中，我们考虑的是带标记的树.在带标记的情况下,路径$P_1 : a---b---c$(也是树)与路径$P_2 : a---c---b$是不同构的.当然$a---b-----c$与$c---b---a$是同构的.

我们先考虑一类特殊的图的生成树计数问题,也就是完全图$K_n$的生成树有多少?换言之,即大小为$n$的带标记的树有多少种?

**Example 1.6. 带标记的树计数**

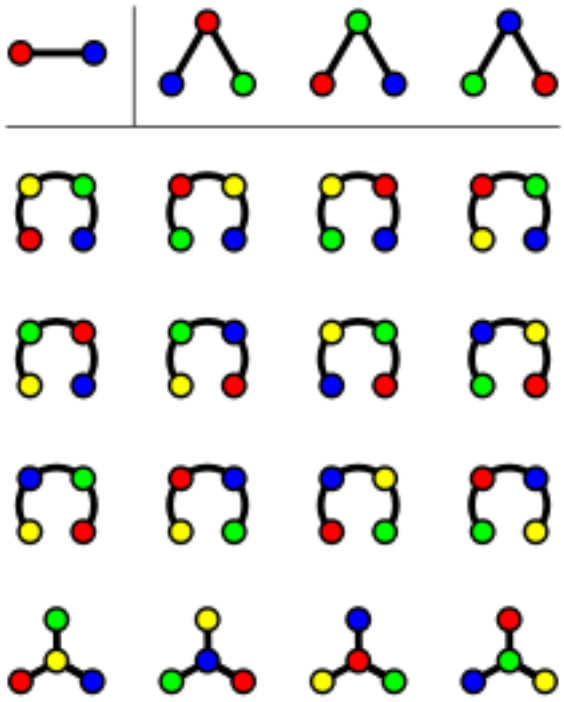

图 3: cayley's formula with small size

将大小为$n$的带标记的树的个数记为$T(n)$.对于较小的$n$，我们有*(*如图*3*所示*)*:

$T(1) = 1; T(2) = 1; T(3) = 3; T(4) = 16 = 4\ star + 12\ path; T(5) = 125$

Cayley 公式给出了一个简单的计数公式:

**Theorem 1.7.** *Cayley* 公式[1889]

大小为$n$的带标记的树的个数为$n^{n-2}$.

*Proof.* **带标记的树计数** 下面,我们假定该$n$个顶点分别标记为$1, 2, \cdots, n$,顶点的集合记为$[V]$.

本定理有不少证明.其中一些证明还很精彩.实际上,该定理及其证明已被收录在“Proofs from THE BOOK”4 一书中.所谓的“THE BOOK”，有“圣经”之意，也就是说这些证明如此精美,被认为是上帝所为.

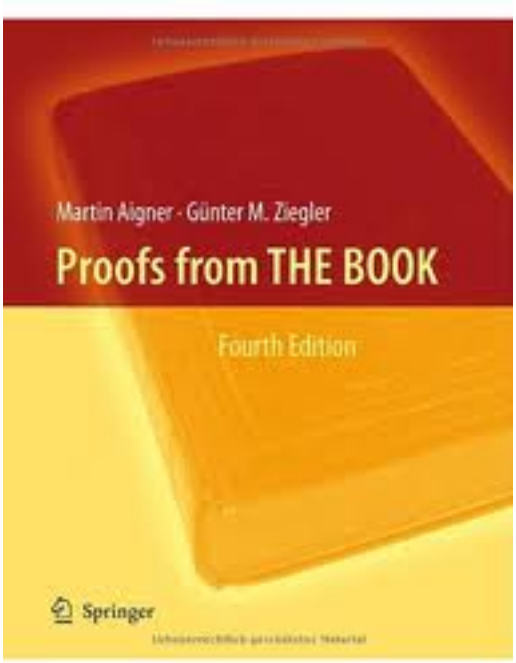


图 4: “Proofs from THE BOOK”

不过，我们介绍的该证明方法在“THE BOOK”4中被一笔带过,其理由是它在一般的图论书中即可见到.该证明的主要思想是对生成树进行编码,编码成长度

为$n-2$的数字串,其中每一位$i \in \{1,2,\cdots,n\}$.显然,后者(数字串)的个数为$n^{n-2}$.所以，问题的关键在于寻找一种编码方法,在生成树与数字串之间建立一一对应关系!

**Definition 1.8.** $Prüfer$ **编码** 顶点集合为$V$的标记树$T$的$Prüfer$编码记为$f(T) = (a_1, a_2, \cdots, a_{n-2})$.编码过程为*:* 在第$i$步,删除剩余的树中最小的叶节点$v$*(*顶点用数字标记*)*,$a_i$等于$v$的邻接顶点.

**Example 1.9.** $Prüfer$ **编码**

如图*5*所示的标记树,其$Prüfer$ 编码为$f = (744171)$.

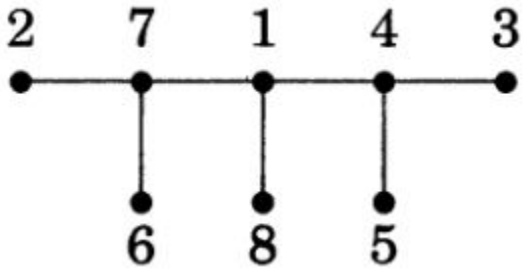


图 5: labeled spanning tree

要证明编码与带标记树之间的一一对应关系,即证明$f(T)$是带标记树集合与数字串集合之间的双射函数.根据编码方式,一个带标记树产生一个编码是显然的,也就是该编码函数是单射的;要证明其为满射,需要说明对于任意一个长度为$n-2$的数字串,都至少有一个带标记树与之对应,也就是说我们需要从编码逆推出带标记树.

关于为何该编码与带标记树是一一对应的具体细节，感兴趣的同学可到`http://en.wikipedia.org/wiki/Main_Page`Wiki网站 中查看$Prüfer$ sequence词条和Cayley's formula 词条. □

定理1.7解决了$K_n$的生成树的计数问题.那么对于更一般的图的生成树的计数问题呢? 一般图不像$K_n$那样具有很好的对称性,所以对计数会带来困难.但是,该问题也是有很高效的解决方法的.一种方法采用了递归的思想,这也是比较自然的;另一种方法就非常巧妙,它利用了线性代数的知识.我们介绍第一种方法.

**Example 1.10. 生成树的计数问题*(general)***

图*6*显示了我们在试图以某种系统的方式来计数生成树.

图 6: contains the diagonal edge or not

要想使用递归思想,就需要找到一种方法来将问题分解成若干更小规模的子问题,同时还要怎么通过合并这些小问题的解合来得到原问题的解. 这也就是"分而治之"的思想.在这里,很显然,我们针对某条边$e$来分解问题:图$G$的生成树个数$=$ 不包含边$e$的生成树的个数$+$ 包含边$e$的生成树的个数. 现在的问题是, "包含边$e$的生成树的个数该怎么求,这似乎并不是一个比原问题规模更小的子问题.但是,实际上,它确实是的,只是我们需要一步转换.

**Definition 1.11. 边的收缩*(Contraction)***

在图$G$中,将边$e = (u, v)$收缩是指,用单个顶点$w$替代$u, v$两点,同时将图$G$中与$u$或$v$相邻接的顶点*(*除$u, v$外*)*邻接到新顶点$w$上. 经过此操作得到的新图记为$G \cdot e$.

需要注意的是,在很多应用中,重边是不考虑的,这也是大家在图论的基本概念一节中就学习的定义,如图*7*所示. 但是,在这里,我们需要考虑重边.考虑了重边的边收缩操作如图*8*所示.

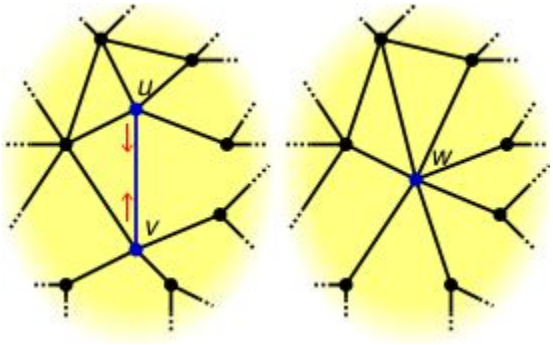


图 7: edge contraction(ignore multiple edges)

**Remark 1.12. 边的收缩*(Contraction)***


图 8: edge contraction(with multiple edges)

经过定义*1.11*所描述的边的收缩操作,$G \cdot e$比$G$少一条边!所以,我们得到了一个规模更小的子问题.

另外,定义*1.11*对重边的特殊规定是必要的，请思考为什么?你可以举出一个反例,在该例中,不考虑重边则会导致错误的计数结果.

虽然,我们通过定义保证了"含边$e$的生成树的个数"是一个比原问题规模更小(小1)的子问题.但是,定义本身并不能说明该种转化(将求"含边$e$的生成树的个数"问题规约为求"$G \cdot e$的生成树的个数"问题)是正确的.我们需要证明这一点.要想证明这一点, 即需要证明"含边$e$的生成树"与"$G \cdot e$的生成树"之间具有一一对应关系.我们不加证明地给出该定理.

**Theorem 1.13.** $G \cdot e$**的生成树**

*"含边$e$的生成树"与"$G \cdot e$的生成树"之间具有一一对应关系.*

有了上面的铺垫,我们很容易就可以得到计算一般图(general)的生成树的算法:

**Proposition 1.14. 一般图生成树计数**

一般图$G$的生成树记为$\tau(G)$,则有: 如果$e \in E(G)$且不是环(loop),

$$\tau(G) = \tau(G - e) + \tau(G \cdot e)$$

**Example 1.15. 一般图的生成树计数**

在本小节末尾,我们给出另一个更为简便的一般图的生成树的计数方法.正如一开始就提到的，该定理及其证明使用了线性代数的知识.定理的应用虽然很方便，但是要准确地描述该定理并给出证明却是一件比较困难的事情.我们

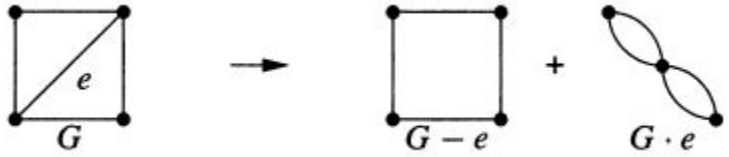


图 9: example for recursive formula

只能提一下它的名字:“Matrix Tree Theorem”.感兴趣的同学可以查看这里:http://en.wikipedia.org/wiki/Matrix_tree_theoremMatrix Tree Theorem, 或者大家在学习了线性代数的知识之后可以回过头来了解该定理,如果那时你还记得自己跟该定理曾经有过这样的约定.

### 1.2.2 最小生成树(Minimum Spanning Tree)

定理2已经告诉我们,任何连通图必含有生成树.既然如此,在带权图(边的权值为正整数)的情况下,也必定含有最小生成树(某整数序列必有最小值).

求最小生成树的算法同cayley公式的证明一样，也有多种.同学们在课堂上学习了Kruskal(Joseph Bernard Kruskal, Jr. (January 29, 1928 – September 19, 2010))算法,其余算法可以参见http://en.wikipedia.org/wiki/Minimum_spanning_treeMinimum Spanning Tree. Kruskal算法属于避圈法,同求生成树的避圈法2的思想是类似的.但是,由于这里需要求的是最小生成树, 实际上是个优化问题,所以,我们需要证明该算法得到的确实是最优的生成树.

Kruskal算法的基本步骤是：将边按照权值排序(升序),按次序考虑所有边,如果将该边加入生成树中不会产生圈,则添加;否则略过该边考虑下一条边.算法的伪码可描述如下11:(引自宋方敏老师讲义)

**Example 1.16.** ***Kruskal*算法**

*Kruskal*算法的一个实例如图所示.

**Theorem 1.17.** ***Kruskal*算法的正确性**

对于无向连通图$G$,*Kruskal*算法构造了其最小生成树.

*Proof.* **Kruskal算法的正确性**

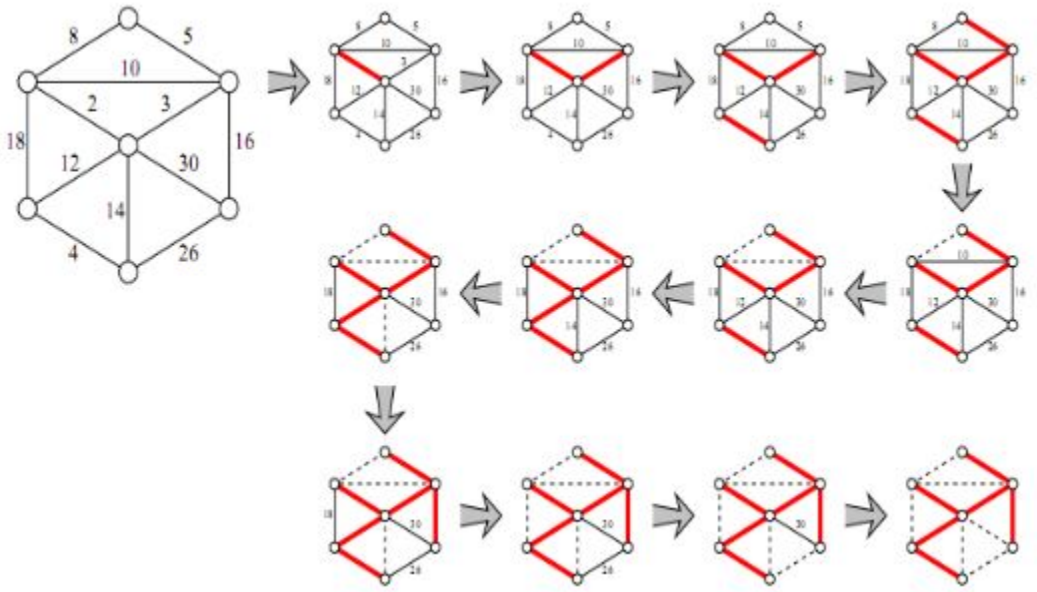


图 10: Kruskal's algorithm

KRUSKAL ALGORITHM($E$)

1 $T^* \leftarrow$ 空图
2 **for** $i \leftarrow 1$ **to** $n-1$
3 **do** $e_i \leftarrow e$ 其在 $E - \{e_1, \ldots, e_{i-1}\}$ 中极小且 $T^* + e_i$ 无回路
4 $T^* \leftarrow T^* + e_i$
5 return $T^*$

图 11: illustration of kruskal's algorithm

**Kruskal算法得到了树:** Kruskal是避圈法,故所得子图无圈;又若所得图不连通,则必有边可选(原图连通),也就是算法不会终止.故所得图连通.

**Kruskal算法得到了最小生成树:** 该命题的证明需要用到与生成树对相关的定理2.1.

由Kruskal算法得到的生成树记为$T$,图$G$的最小生成树记为$T'$.

- 如果$T = T'$或者$w(T) = w(T')$，则得证;
- 如果$T \neq T' \wedge w(T) \neq w(T')$.

  考察算法的执行过程,记$e$为第一条选择加入$T$而没有加入$T'$的边.$T' + e$构成圈,而$T$中无圈,故必存在$e' \in E(T') - E(T)$,满足$T' + e - e'$.

  根据Kruskal算法,在面临$e, e'$时,该算法选择了$e$，这说明$w(e) \leq w(e')$,则$w(T' + e - e') \leq w(T')$.如果$w(T' + e - e') < w(T')$,则导致矛盾;否则,重复上述讨论，由于$w(T) \neq w(T')$，必会导致矛盾.

□

# 2 Problem Set

**Problem 2.1. 生成树对*(pairs of spanning trees)***

如果$T, T'$是连通图$G$的两个不同的生成树,边$e \in E(T) - E(T')$,那么必存在边$e' \in E(T') - E(T)$满足$T - e + e'$是$G$的生成树.

**Solution 2.2. 生成树对*(pairs of spanning trees)***

见图例*12.*

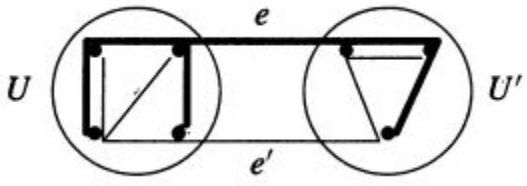


图 12: pairs of spanning trees

**Remark 2.3. 生成树对*(pairs of spanning trees)***

该习题实际上涉及到了图的一种转化操作.而转化操作在涉及优化*(*比如*Kruskal*算法的证明过程*1.2.2)*，寻求特殊结构等问题中往往至关重要.所以希望大家能理解该命题.记住并理解定理的一种方法就是领会它的证明过程，在图论中，大家可以记住定理及其证明过程中所使用的图例.

**********

**Problem 2.4. 生成树计数**

求$\tau(K_{2,3})$,即$K_{2,3}$的带标记的生成树的个数.

**********

**Problem 2.5. 最小生成树的唯一性**

请证明:如果无向连通图$G$的每条边的权值都不同，则最小生成树是唯一的.

# 3 Application and Extension

证明过程1.1 说明了连通图必有生成树,而且生成树不唯一(with high probability).习题2.1 告诉了我们生成树对所具有的非常重要的一个性质.现在,我们要通过一个游戏来说明生成树的应用.

先来定义“co-spanning trees”, 它是指彼此无公共边(edge-disjoint)的生成树.与图的连通度类似,如果一个连通图具有“co-spanning trees”, 则我们可以想象该图也具有比较好的连通性.

**Definition 3.1. 游戏*Bridg-it***

*“Bridg-it”* 是双人游戏.游戏的配置如图**??** 所示.为方便起见,我们称该配置图为 *“board”*.该 *“board”* 由四行五列黑色顶点与五行四列白色顶点交叉排列而成. 黑色顶点属于*player 1,* 白色顶点属于*player 2.* 玩家*1*的目标是每次连接相邻的两个黑色顶点,直到在最左边一列黑色顶点与最右边一列黑色顶点之间构造起一条路径; 玩家*2*的目标与之对称,每次连接相邻的两个白色顶点,直到在最上边一行的白色顶点与

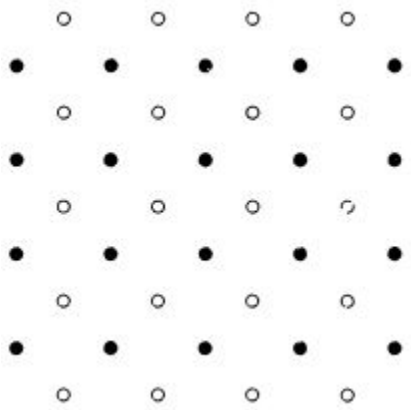

图 13: Graph game: Bridg-it

最下边一行的白色顶点之间构造起一条路径. 玩家*1*先走,即为先手.注意: 两玩家的连线不能交叉.所以该游戏是个有攻有防的竞赛类游戏.

**Remark 3.2.** ***游戏Bridg-it***

我们先从大局上分析该游戏:

1. 每一条从左到右的路径与每一条从上到下的路径都是交叉的.所以,两人不可能都赢得比赛.
2. 该游戏结束时,必有一方获胜.
3. 该游戏配置是对称的,而玩家*1*是先手,所以玩家*2*不可能有必胜策略.所以某玩家有必胜策略是指,不管对方怎么进行, 该玩家都可以通过遵循某种策略取得胜利.
4. 我们断言玩家*1*有必胜策略.即该游戏是先手必胜游戏.

通过对游戏的分析,我们已经得出玩家*1*有必胜策略的结论.现在,我们的目标就是为玩家*1*找到一种必胜的策略.

**Theorem 3.3.** ***游戏Bridg-it***

*"Bridg-it"*游戏中,玩家*1*有必胜策略.

*Proof.* **游戏Bridg-it**

我们显示地给出玩家1的必胜策略.

我们先为玩家1构造一幅可能的连接图,见图**??**.注意我们现在关注的是从左边到右边的通过黑色顶点的连接.

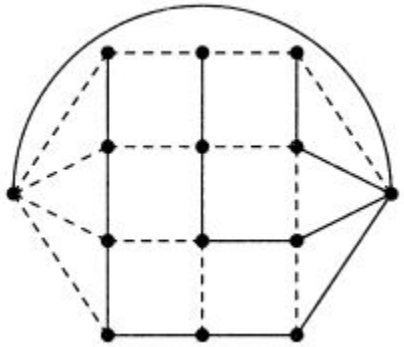

图 14: 玩家1的连接图

对该图**??**需要做几点解释:

1. 我们现在是在寻求玩家1的必胜策略,所以考虑黑色顶点的所有可能连接情况(包括虚线与实线).
2. 玩家1的目标是连接左右两栏,与具体顶点无关,所以左右两栏均可等价看作一个点,称为辅助点.
3. 连接辅助点,得到一条辅助线.
4. 虚线和实线分别构成了生成树.并且这两个生成树为“co-spanning trees”.

辅助线实际是不存在的,可以让玩家2先cut掉该边,这也就相当于是玩家1先走.玩家2每走一步,就等于cut掉图中的一条边,也就是删除某一个生成树中的一条边$e$,从而使该生成树断成两个连通分支,由习题2.1知,在另一个生成树中必存在一条边$e$可以连接这两个连通分支.我们让玩家1走这条边,也就是使得该边成为“double edge”.已走的边是不能被割断的,也就是说“double edge”是不可被cut的.走这条边的效果是,仍然存在两个无公共边的生成树.

在游戏过程中,玩家1总是可以维护两个无公共边的生成树. □