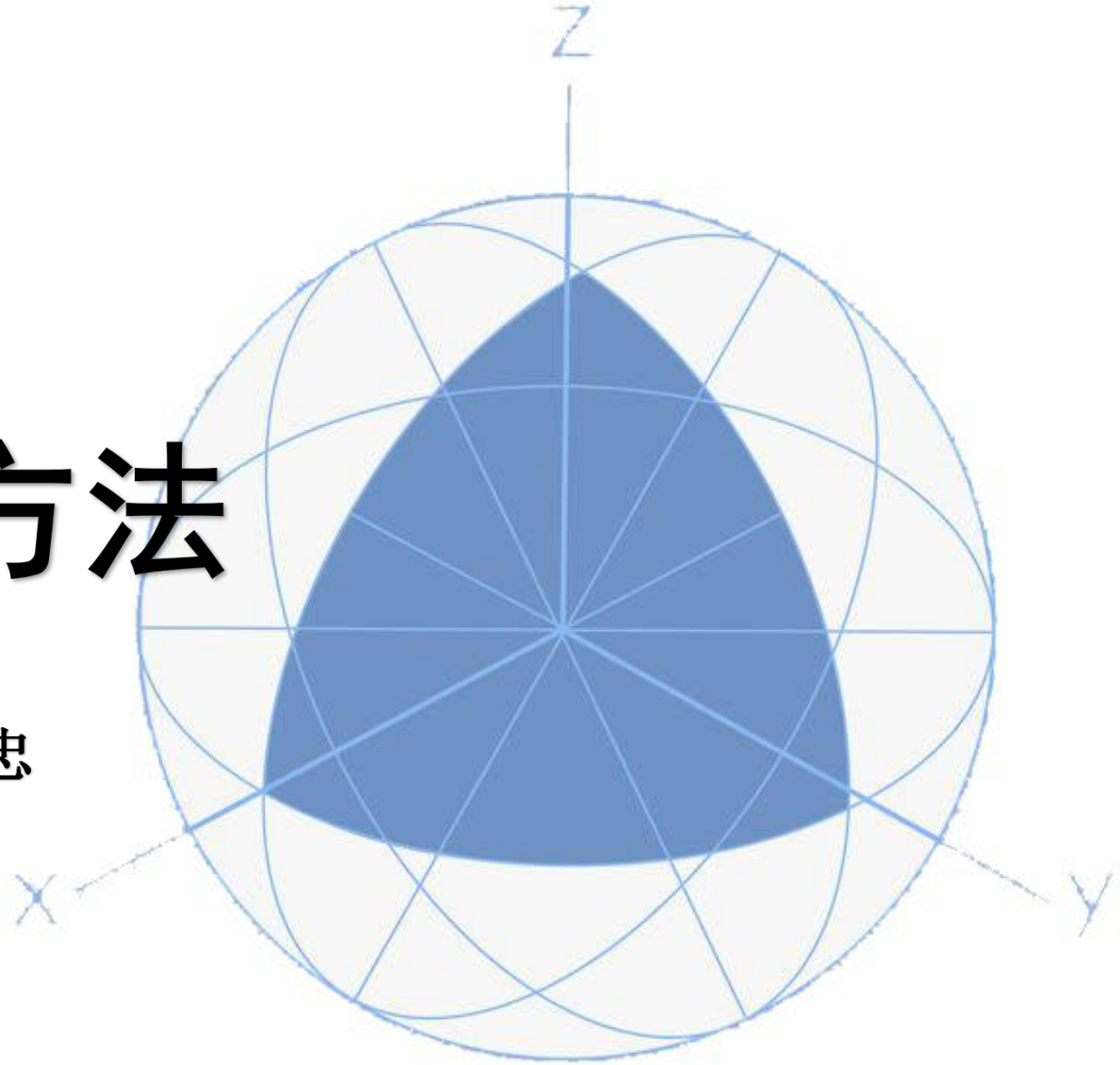
厦門大学
XIAMEN UNIVERSITY

图论方法

谭　忠



数学科学学院
SCHOLL OF MATHEMATICS SCIENCE

厦门大学
XIAMEN UNIVERSITY

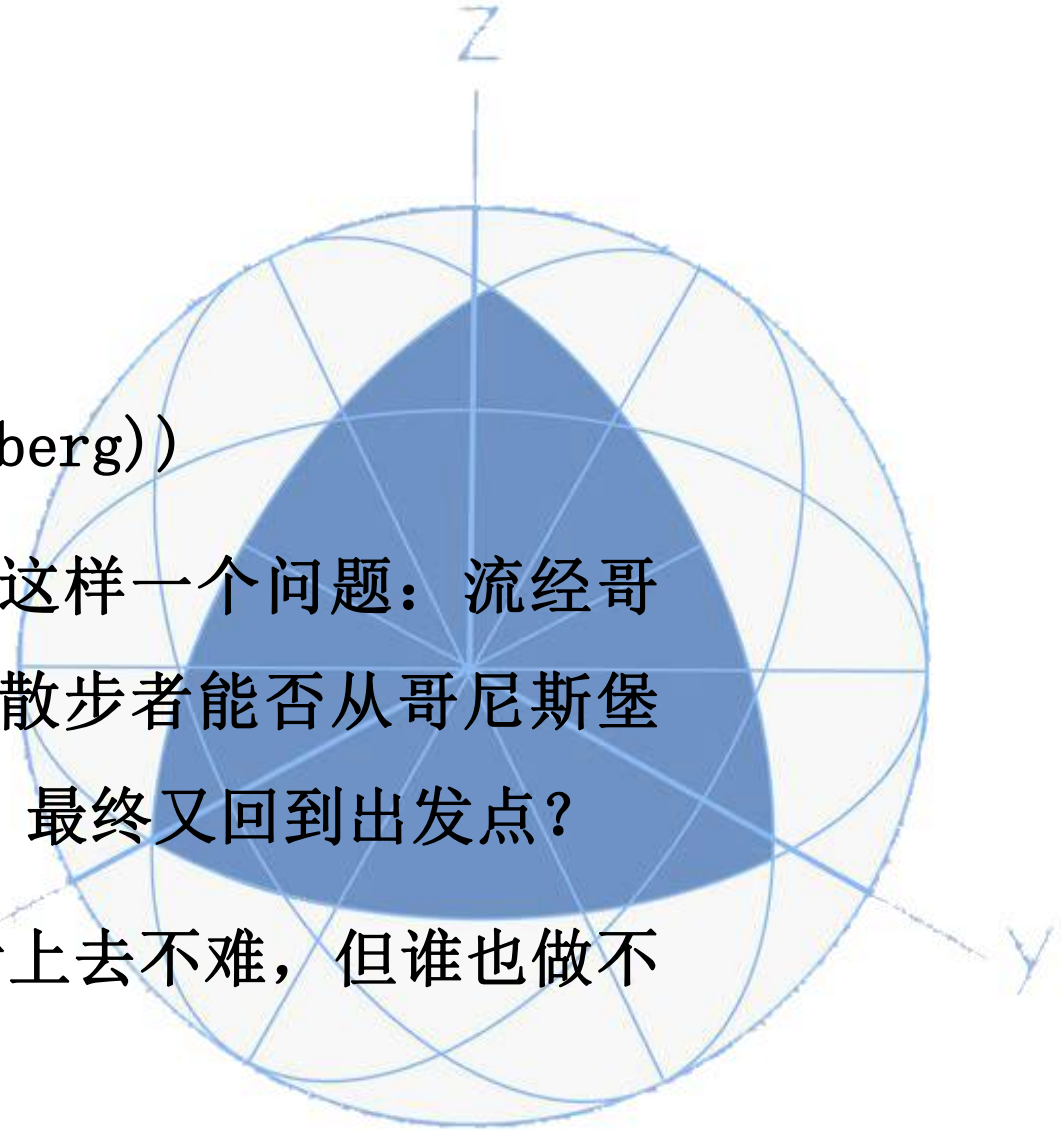
目录
CONTENTS

厦门大学
XIAMEN UNIVERSITY

Part 1

源头问题与当今应用

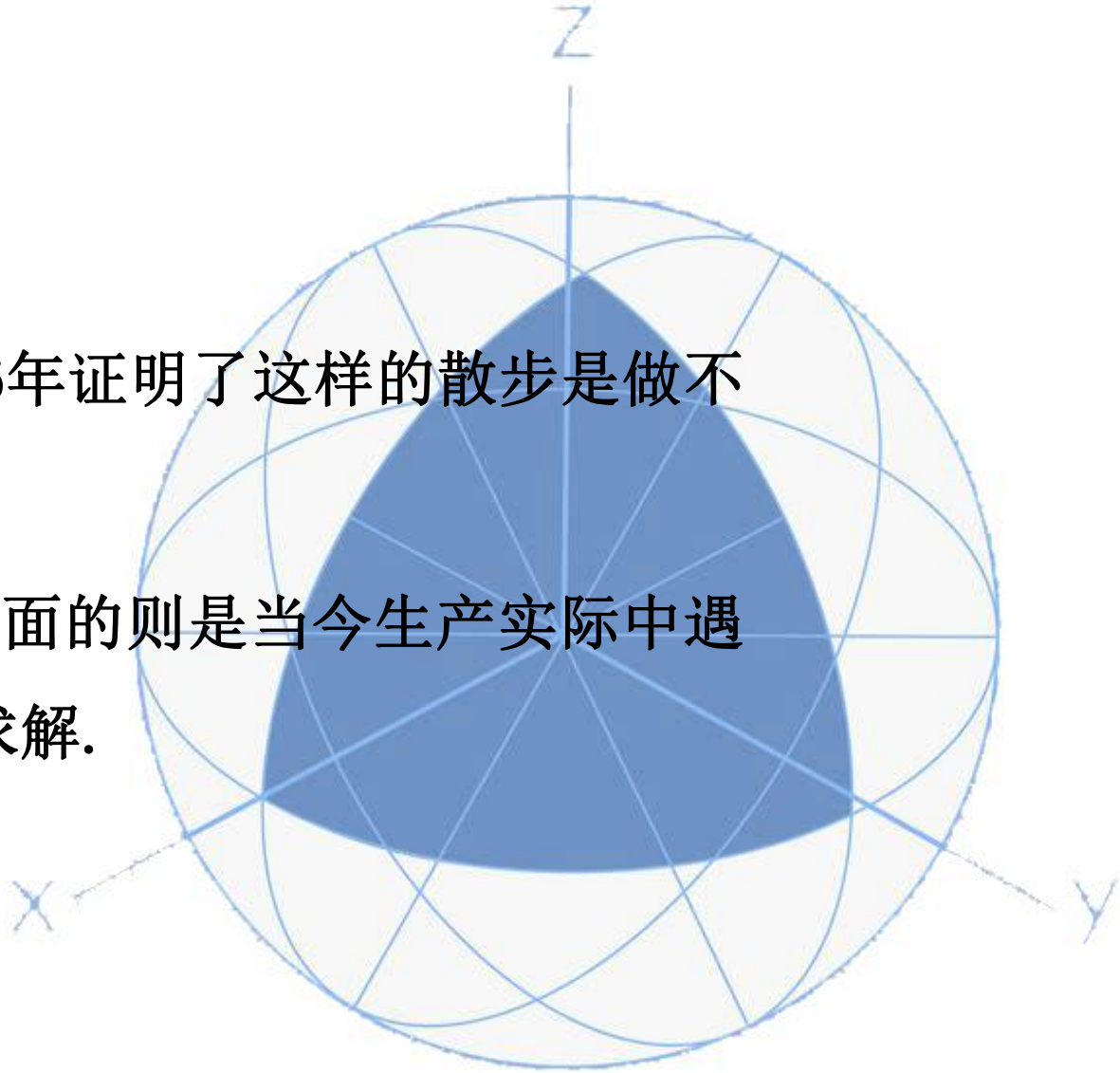
12.1 源头问题与当今应用



例12.1 （哥尼斯堡七桥问题(Konigsberg)）

18 世纪，在哥尼斯堡的居民热衷于这样一个问题：流经哥尼斯堡的普雷格尔河上有七座桥．一个散步者能否从哥尼斯堡某处出发走遍七座桥且每座桥只走一次，最终又回到出发点？

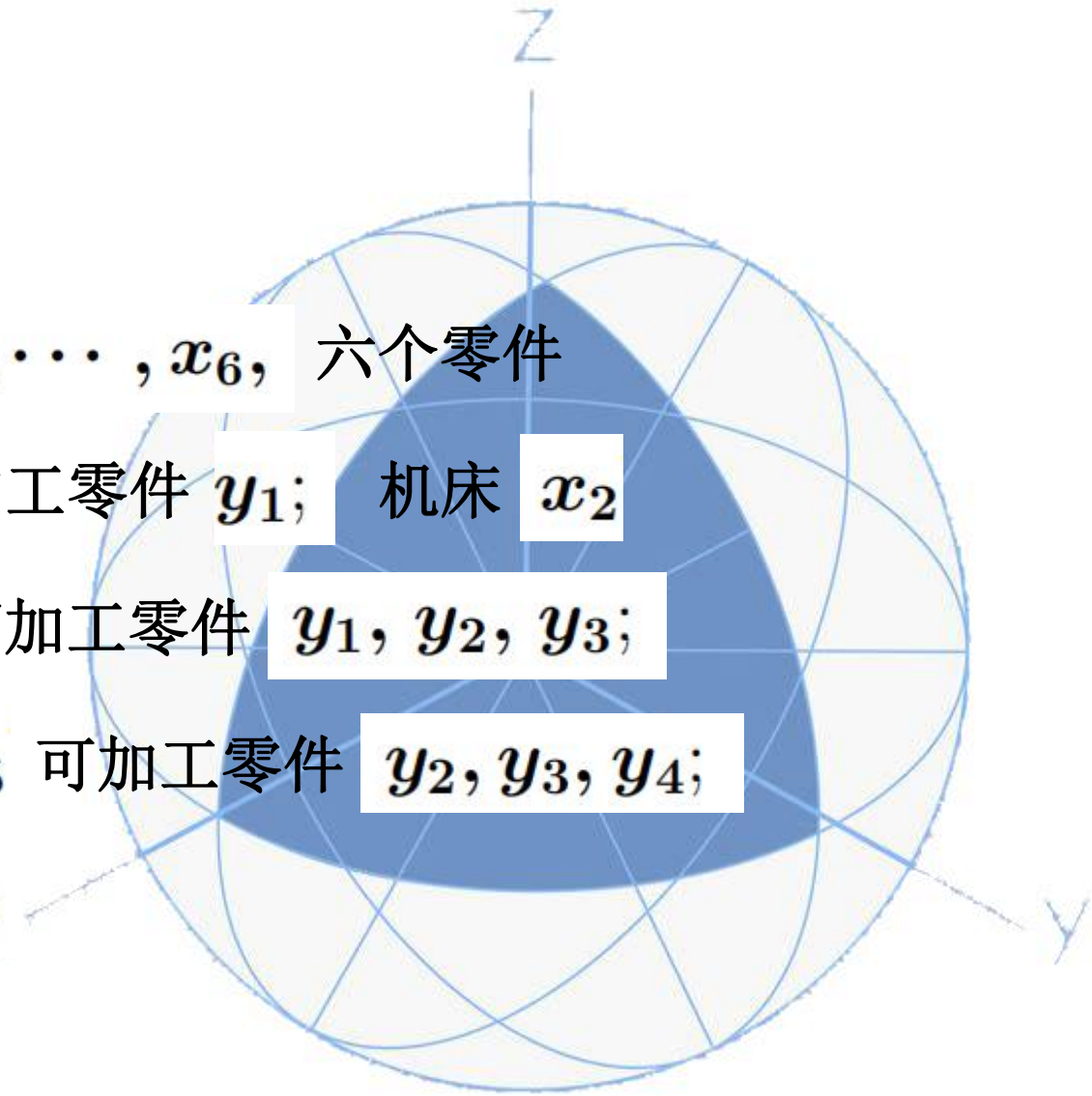
这就是著名的七桥问题．这个问题看上去不难，但谁也做不到．

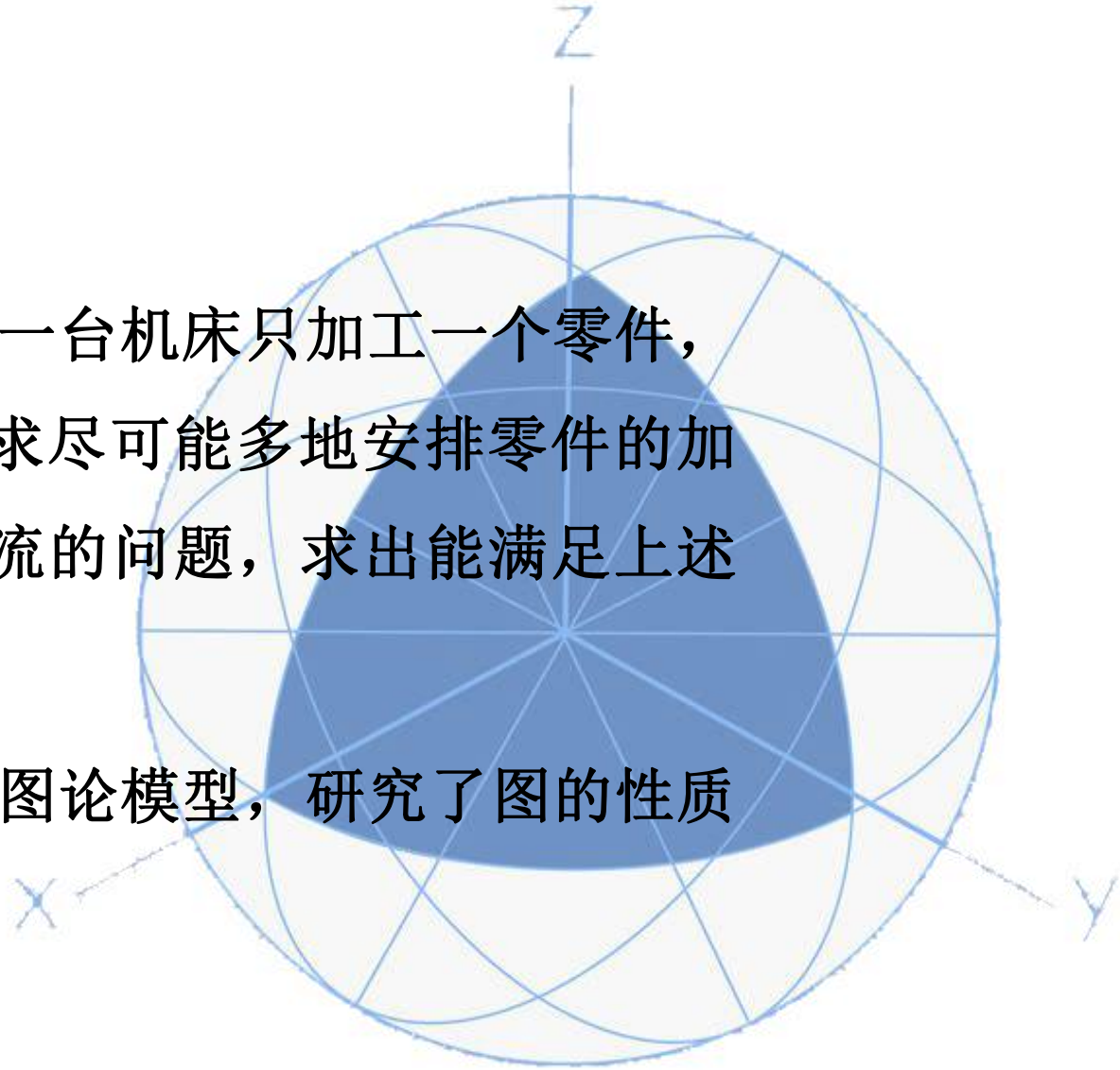


数学家Euler(1707-1783)在1736年证明了这样的散步是做不到的．那是为什么呢？

上面的例子是历史源头问题，下面的则是当今生产实际中遇到的问题，也可转化为图论的问题求解.

例12.2：已知有六台机床 $x_1, x_2, \cdots, x_6$，六个零件 $y_1, y_2, \cdots, y_6$. 机床 x_1 可加工零件 y_1；机床 x_2 可加工零件 y_1, y_2；机床 x_3 可加工零件 y_1, y_2, y_3；机床 x_4 可加工零件 y_2；机床 x_5 可加工零件 y_2, y_3, y_4；机床 x_6 可加工零件 y_2, y_5, y_6.





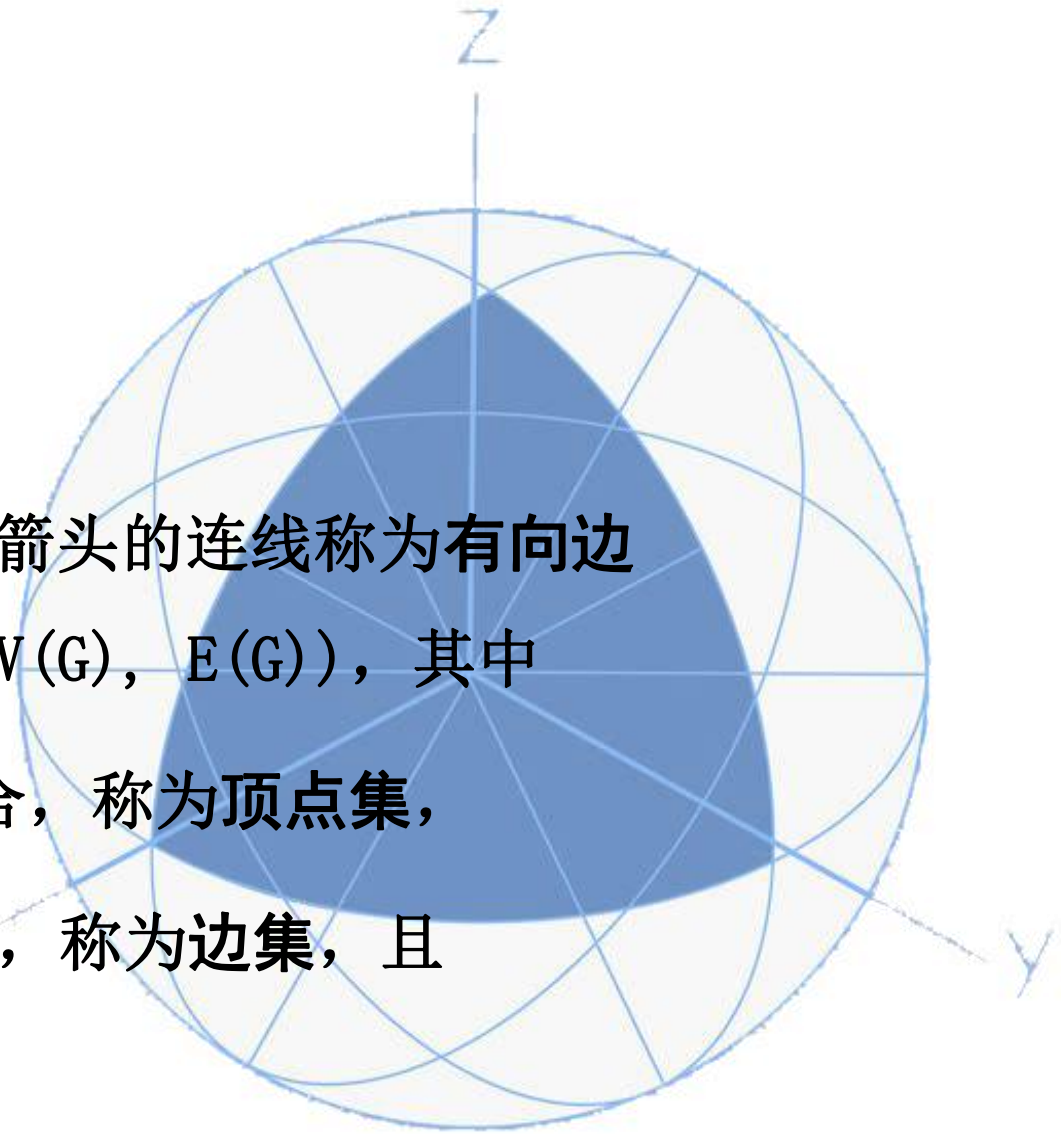
现在要求制定一个加工方案，使一台机床只加工一个零件，一个零件只在一台机床上加工，要求尽可能多地安排零件的加工. 试把这个问题化为求网络最大流的问题，求出能满足上述条件的加工方案.

为了解决这些问题，人们构建了图论模型，研究了图的性质和算法，建立了现代图论理论.

Part 2

图论思想与建模方法

12.2 图论思想与建模方法



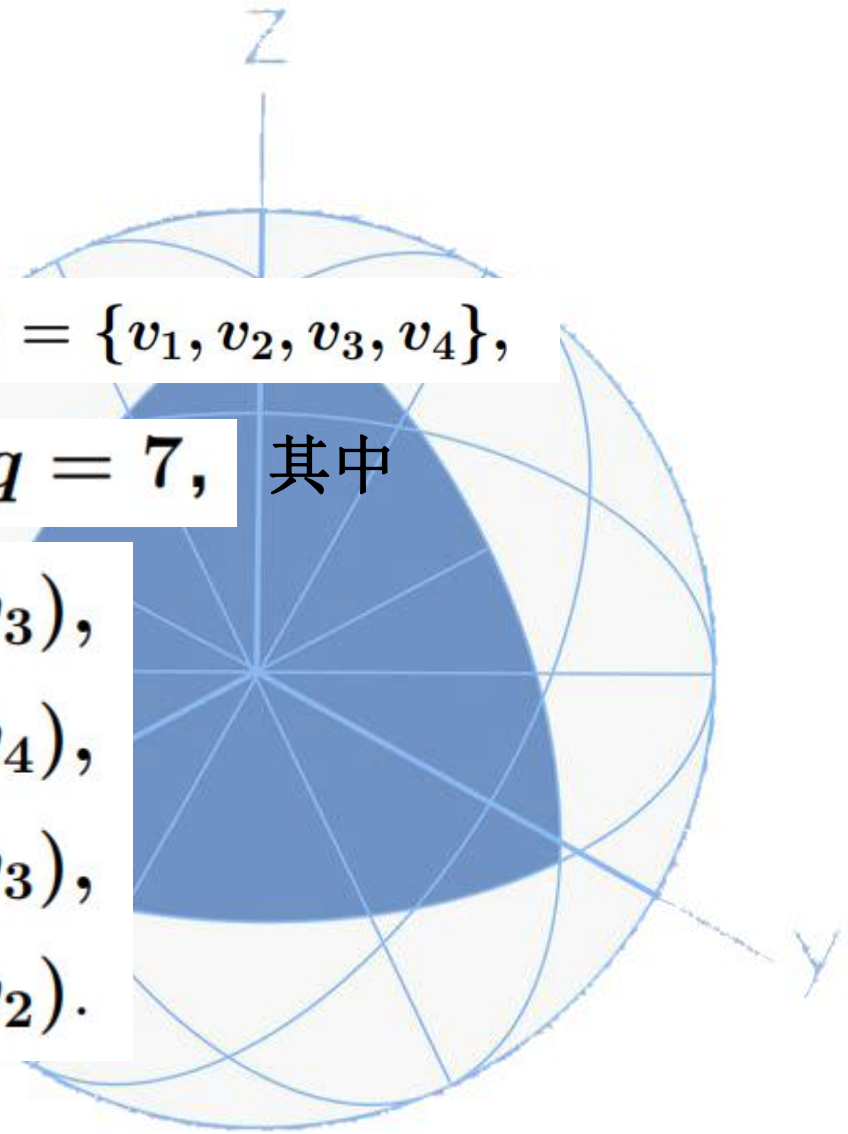
一、无向图和有向图

把两点间不带箭头的连线称为**边**，带箭头的连线称为**有向边**或**弧**. 一个无向图G是指一个有序二元组(V(G)，E(G))，其中

$V(G)=(v_1,v_2,\cdots,v_p)$，是p个点的集合，称为**顶点集**，

$E(G)=(e_1,e_2,\cdots,e_q)$ 是q条边的集合，称为**边集**，且

$e_k=(v_i,v_j)=(v_j,v_i).$



例12.3： $G=(V(G),E(G))$，这里 $V(G)=\{v_1,v_2,v_3,v_4\}$，

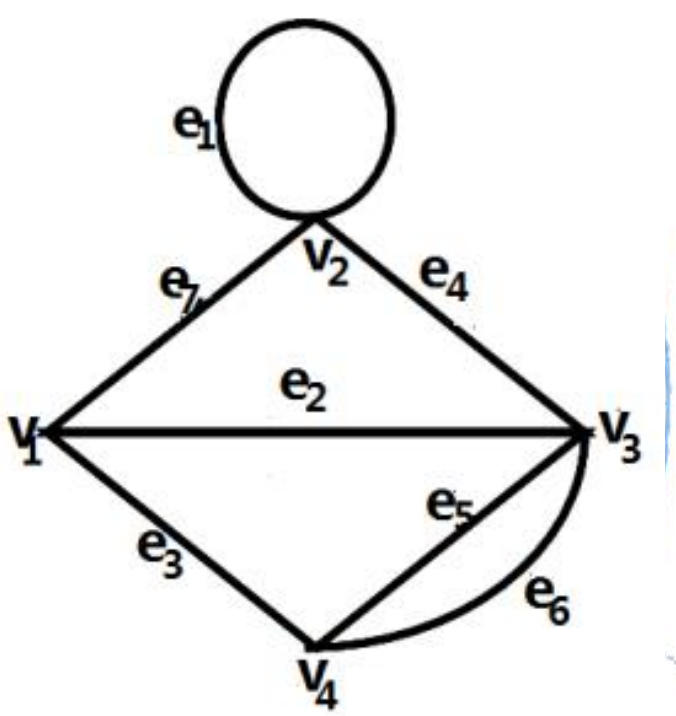
$p=4$，$E(G)=\{e_1,e_2,\cdots,e_7\}$，$q=7$，其中

$$e_1=(v_2,v_2),\quad e_2=(v_1,v_3),\quad e_3=(v_1,v_4),$$

$$e_4=(v_2,v_3),\quad e_5=(v_3,v_4),\quad e_6=(v_4,v_3),\quad e_7=(v_1,v_2).$$

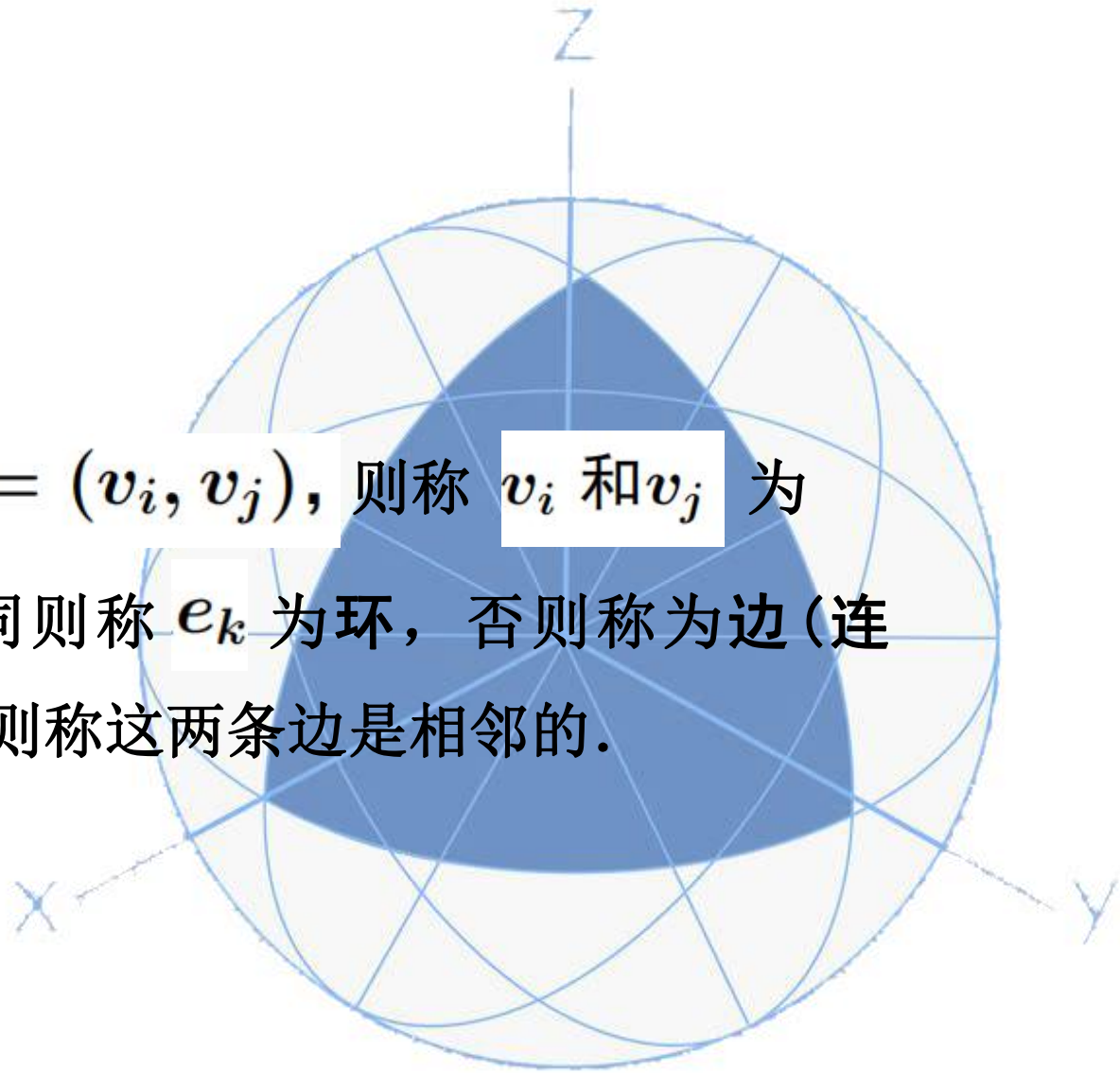
上面给出的是图的抽象定义．通常形象地用圆圈或黑点表示顶点，用连结顶点的曲线表示边，将图在平面上画出来．

在画图时，我们要求表示边的曲线除了从某个顶点开始和到某个顶点结束外中途不再经过任何顶点．如例12.3中抽象定义的图G为右图．



以下是有关图的几个概念.

(1)若记边 (v_i, v_j)为 e_k，即 $e_k = (v_i, v_j)$，则称 v_i 和v_j 为边 e_k 的两个顶点．若两顶点相同则称 e_k 为环，否则称为边(连杆)．若 e_i和e_j 只有一个公共顶点，则称这两条边是相邻的.

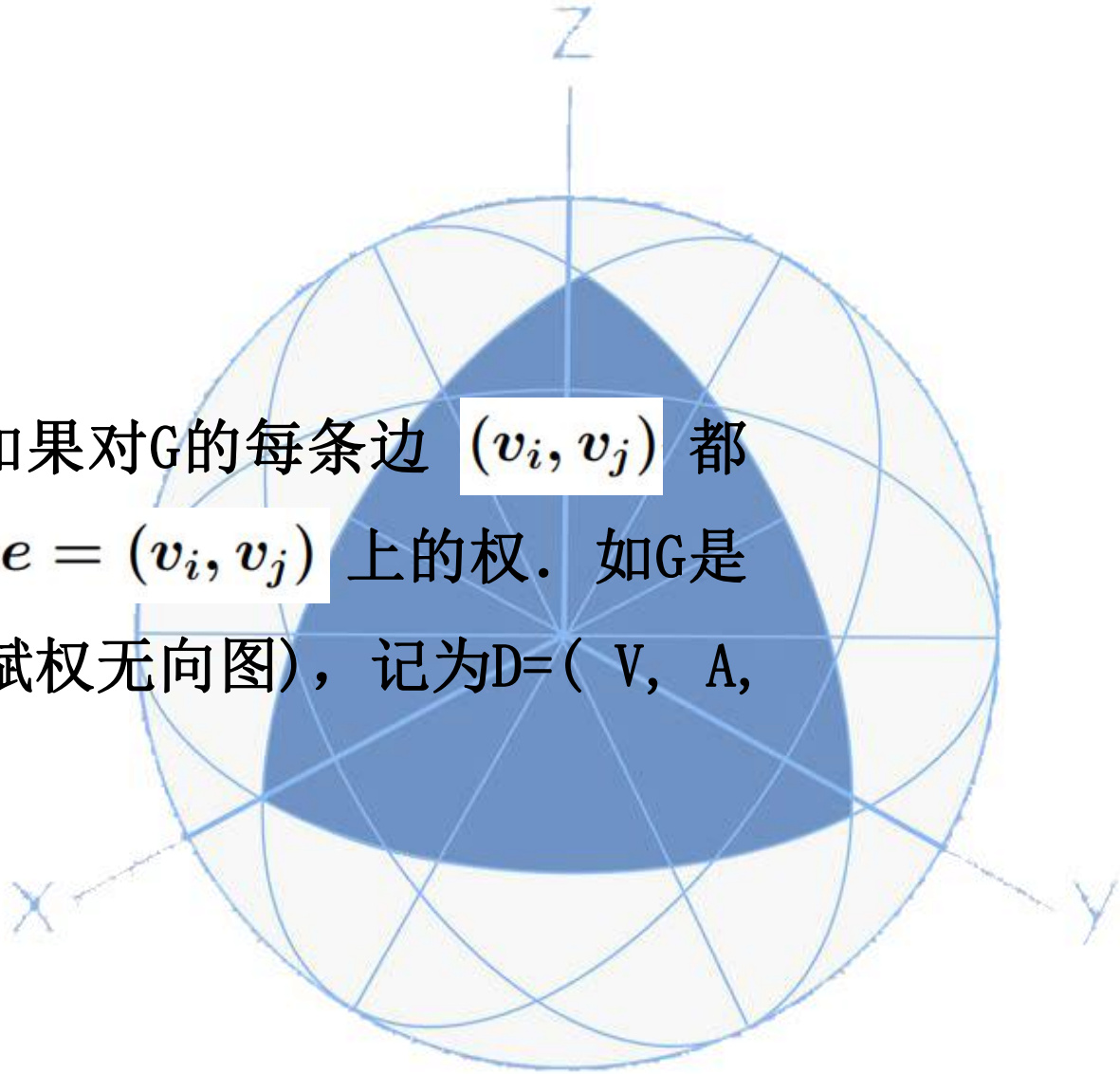


若两条边的两端点都相同，则称它们为**重边**. 既不含环，又不含重边的图称为**简单图**. 如果一个图是由点和弧所组成，则称此图为**有向图**，记为D=(V,A).其中 $V=\{v_1,v_2,\cdots,v_p\}$,

$A=\{e_1,e_2,\cdots,e_q\}$. $e_k=(v_i,v_j)\neq(v_j,v_i),v_i,v_j\in V$,

e_k就是以v_i为始点，v_j为终点的弧，v_i和v_j分别称为e_k的**尾**和**头**，用带箭头(由尾指向头)的曲线表示有向边.

(2)一个图G称为是一个**赋权图**，如果对G的每条边 (v_i, v_j) 都被赋予一个权函数 w_{ij}, w_{ij} 称为边 $e = (v_i, v_j)$ 上的权．如G是有向图(无向图)，称G为赋权有向图(赋权无向图)，记为D=(V, A, w)(G(V, E, w)).

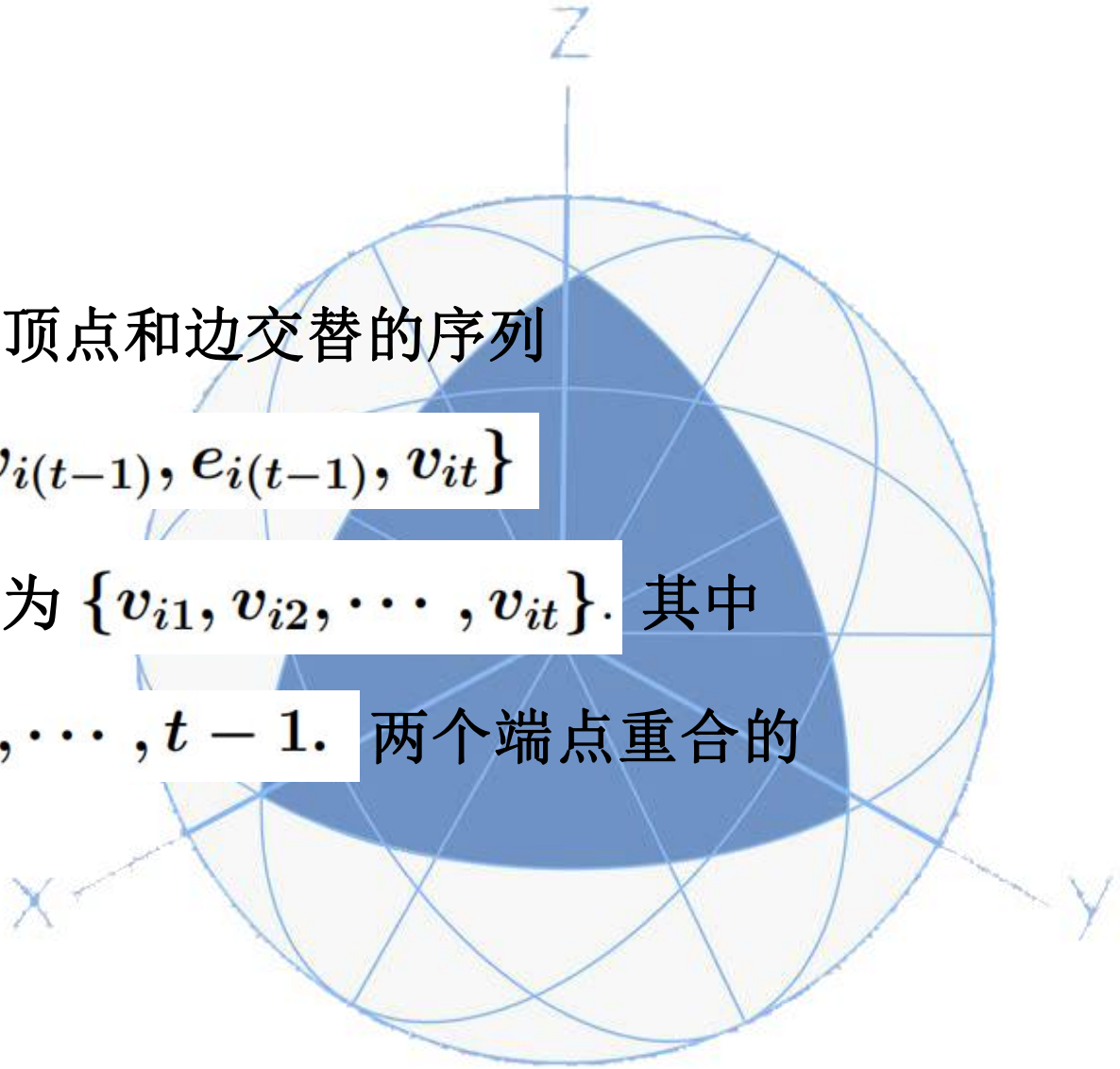


(3)对于无向图G=(V, E)，称某顶点和边交替的序列

$$\{v_{i1}, e_{i1}, v_{i2}, e_{i2}, \cdots, v_{i(t-1)}, e_{i(t-1)}, v_{it}\}$$

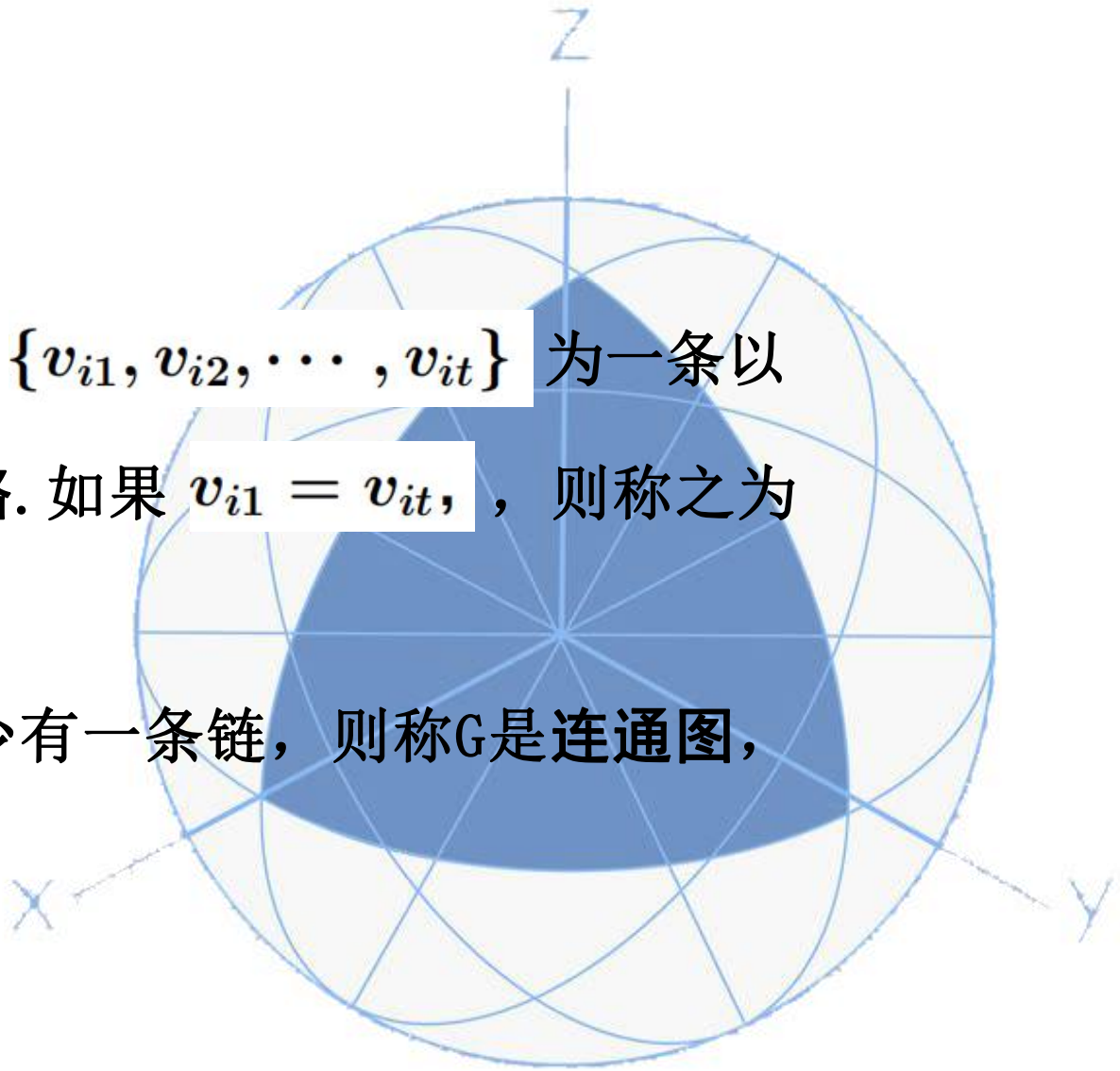
为连接 v_{i1}和v_{it} 的一条**链**，简记为 $\{v_{i1}, v_{i2}, \cdots, v_{it}\}$. 其中

$e_{ik} = (v_{ik}, v_{i(k+1)}), \quad k = 1, 2, \cdots, t-1.$ 两个端点重合的链，称为**圈**.



(4)在有向图D=(V, A)中，称链 $\{v_{i1}, v_{i2}, \cdots, v_{it}\}$ 为一条以 v_{i1} 为始点，通向终点 v_{it} 的**路**. 如果 $v_{i1} = v_{it}$，，则称之为**回路**.

(5)如果图G中任何两顶点间至少有一条链，则称G是**连通图**，否则为不连通图.

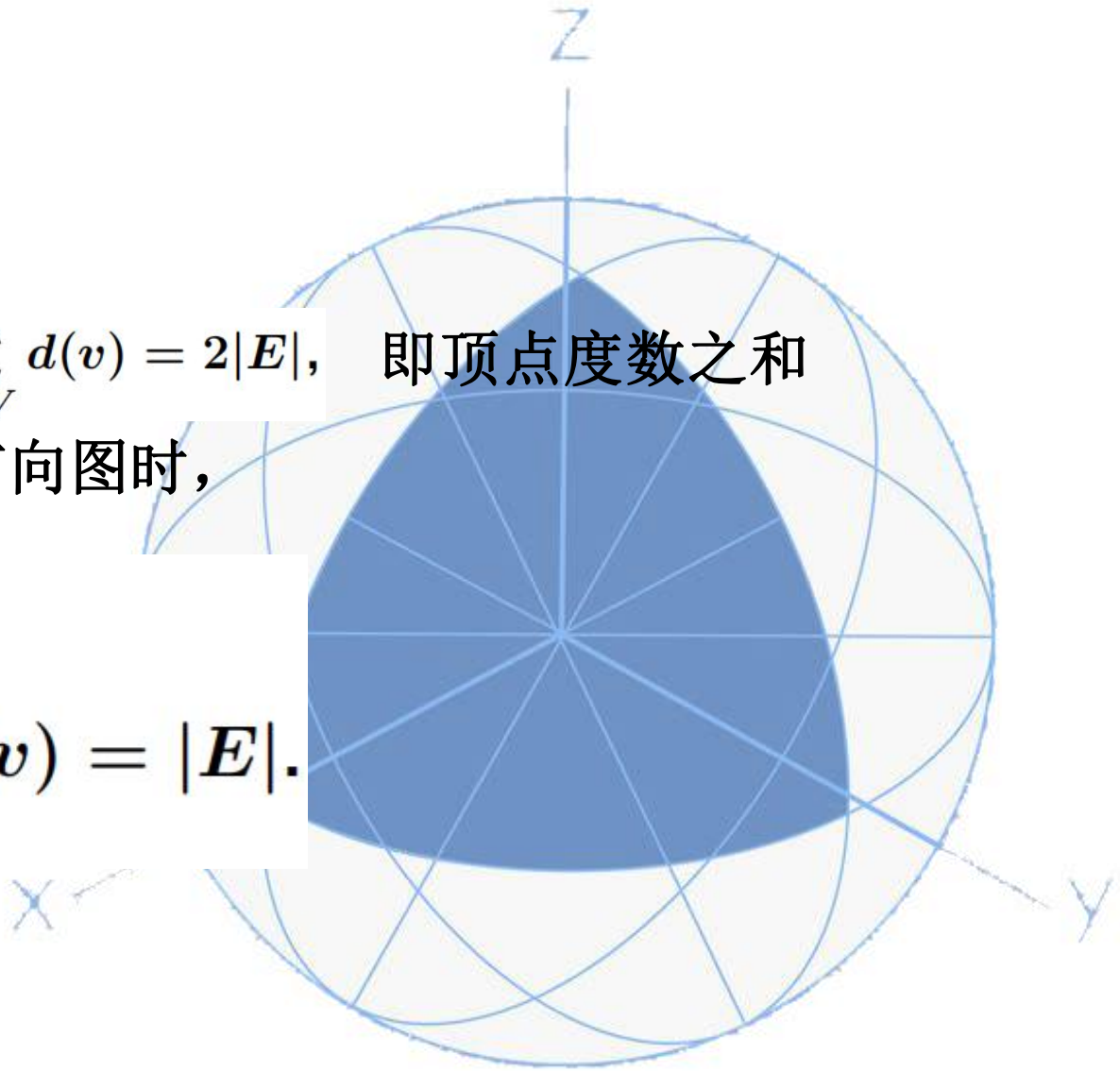


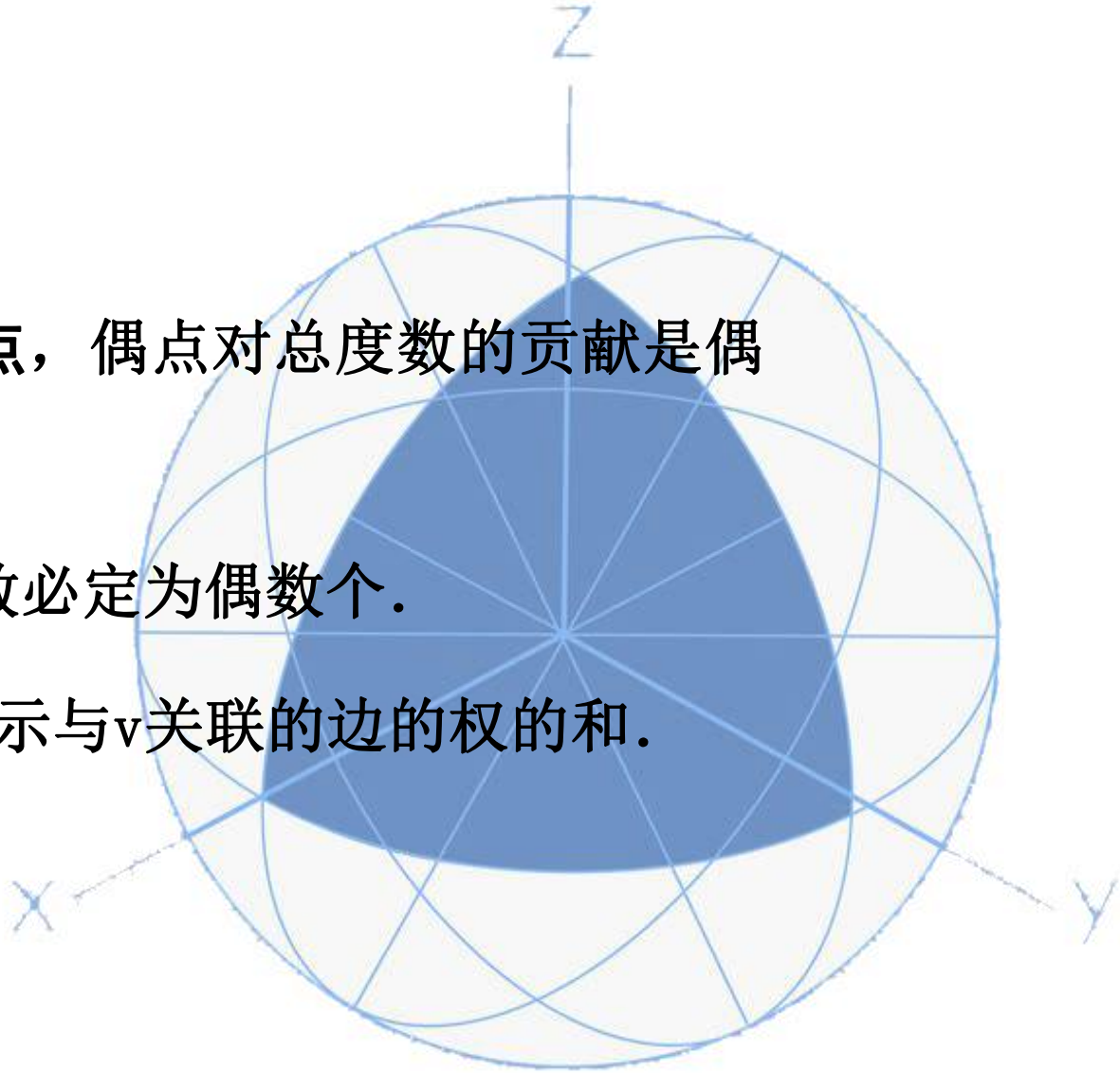
二、顶点的度数

我们先考虑无向图．设G是个无向图，$v \in V(G)$，v在图G中的**度数** $d_G(v)$(简记为d(v))定义为G中与v关联的边的条数(其中，环计算两次)．若G是有向图，在G 中以顶点v为尾的弧的数目称为出度，记作 $d^+ = d_G^+(v)$,以顶点v 为头的弧的数目称为入度，记作 $d^- = d_G^-(v)$. v的度 $d(v) = d_G(v)$ 定义为它的出度和入度之和，记为 $d(v) = d^+(v) + d^-(v)$.

定理12.1 设G=(V, E)，则 $\sum_{v\in V} d(v) = 2|E|$， 即顶点度数之和为边数的 2 倍. 进一步，当G 是有向图时，

$$\sum_{v\in V} d^{+}(v) = \sum_{v\in V} d^{-}(v) = |E|.$$

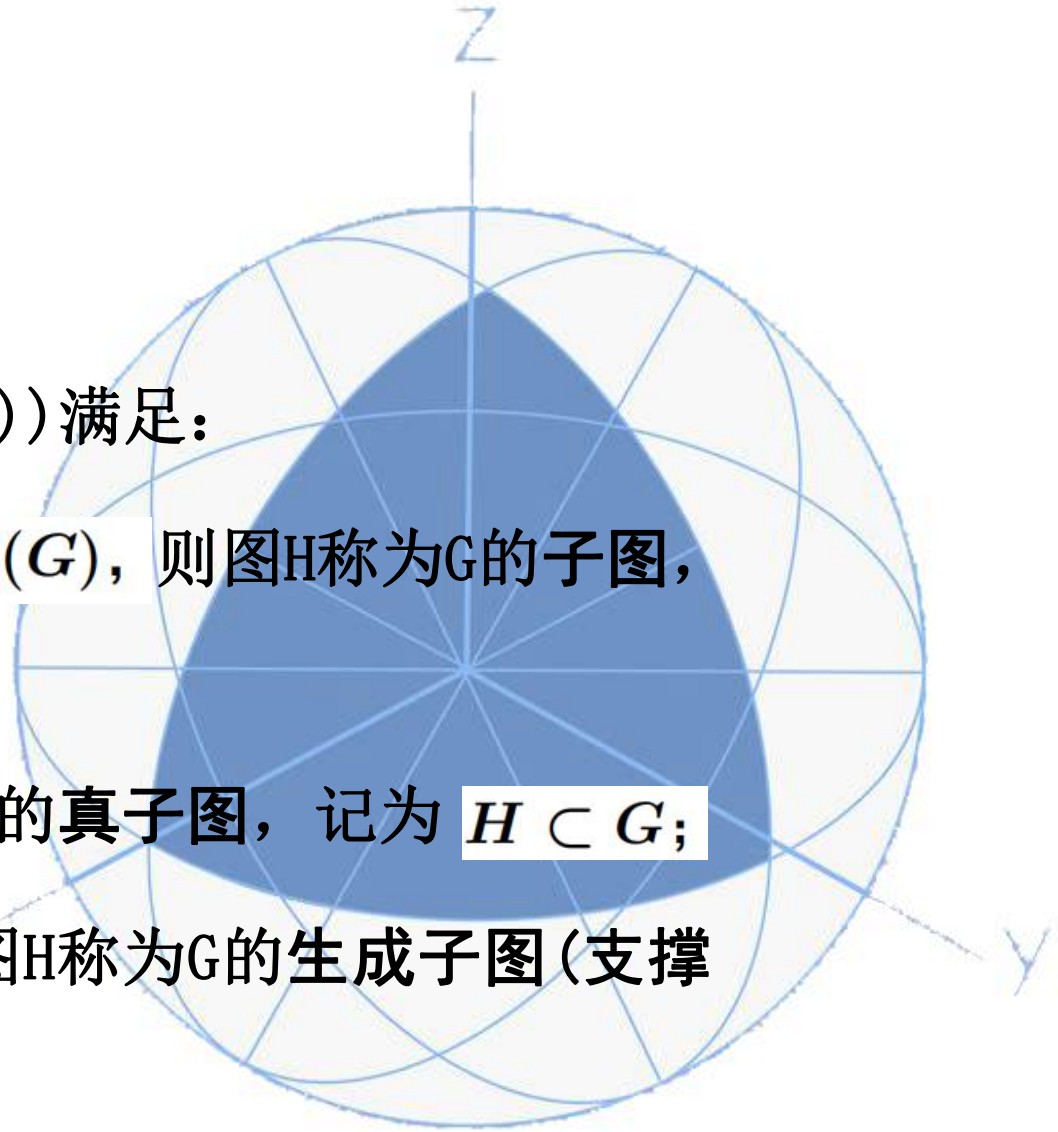




由于图的顶点可划分为**偶点**和**奇点**，偶点对总度数的贡献是偶数，因此有：

推论12.1 任意图中，奇点的个数必定为偶数个.

当G是赋权图时，$d(v) = d_G(v)$ 表示与v关联的边的权的和.



三、子图

图H=(V(H), E(H))和G=(V(G), E(G))满足：

(1)若 $V(H) \subseteq V(G)$ 且 $E(H) \subseteq E(G)$，则图H称为G的**子图**，记为 $H \subseteq G$.

(2)若 $H \subseteq G$ 且 $H \neq G$，则图H称为G的**真子图**，记为 $H \subset G$；

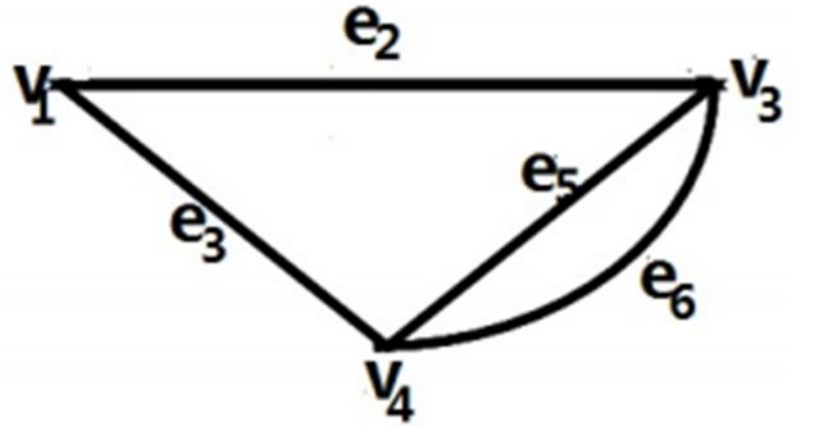
(3)若 $H \subseteq G$ 且 $V(H) = V(G)$，则图H称为G的**生成子图(支撑子图)**.

如右图， $H=(V',E')$

其中 $V'=\{v_1,v_3,v_4\}$,

$E'=\{e_2,e_3,e_5,e_6\}$

这就是例题12.3图G的一个子图。



四、树与最小生成树

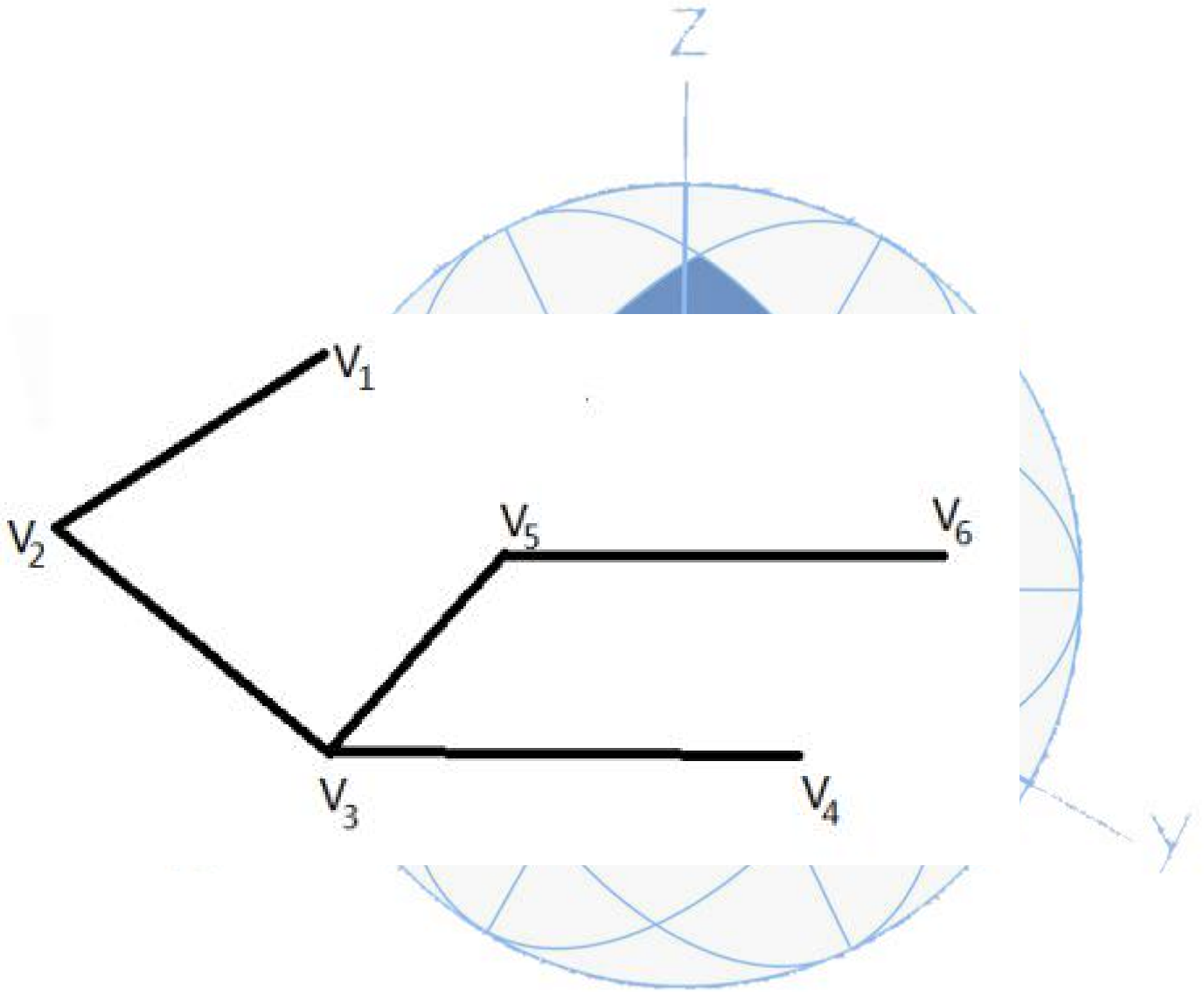
如果G=(V,E)是

一个无圈的连通图，

则称G为**树**，

记为T=(V,E).

如右图所示.



树的简单性质：

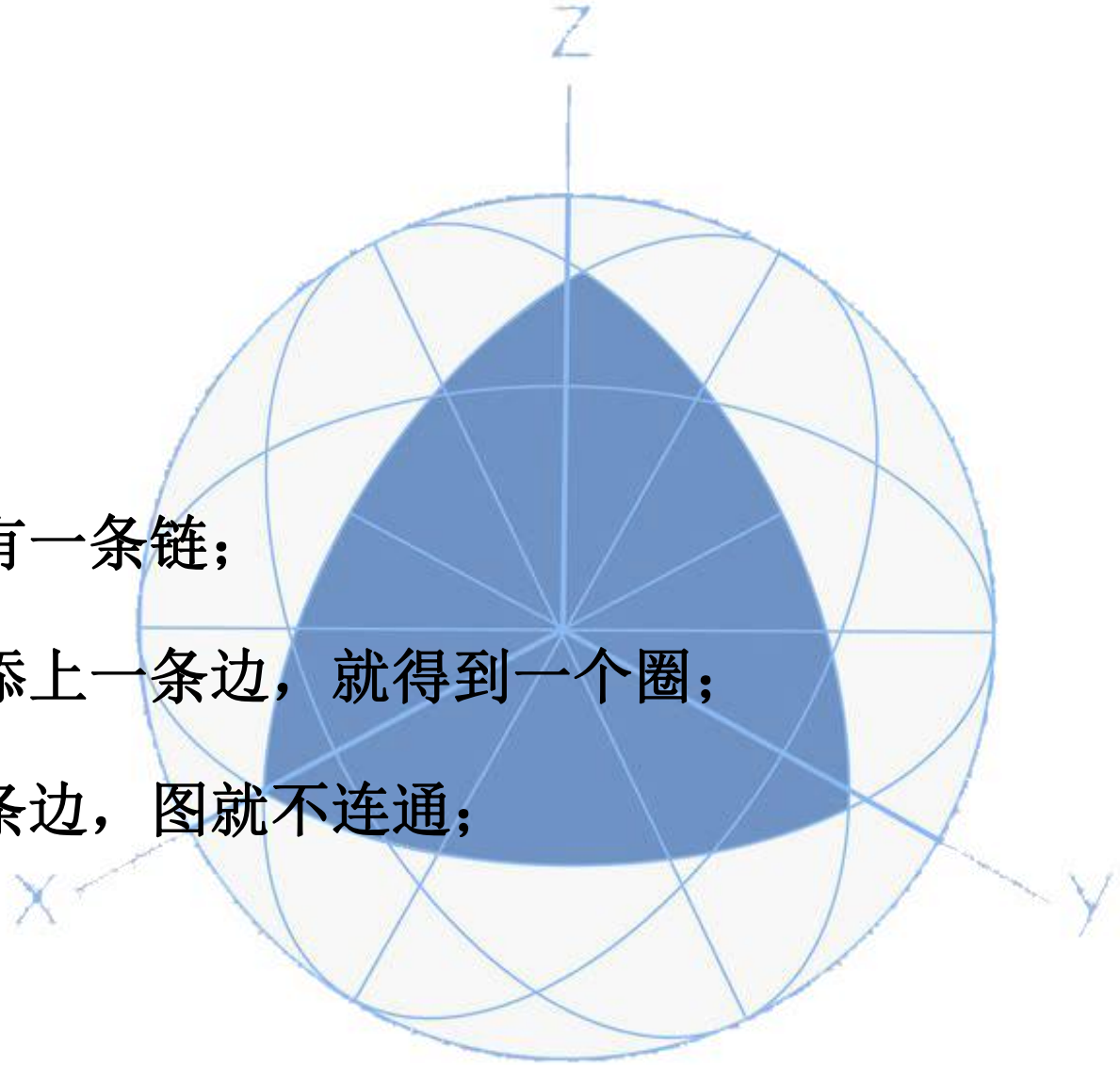
(1)树必连通且无圈；

(2)树中任意两顶点间必有且仅有一条链；

(3)在树的两个不相邻的顶点间添上一条边，就得到一个圈；

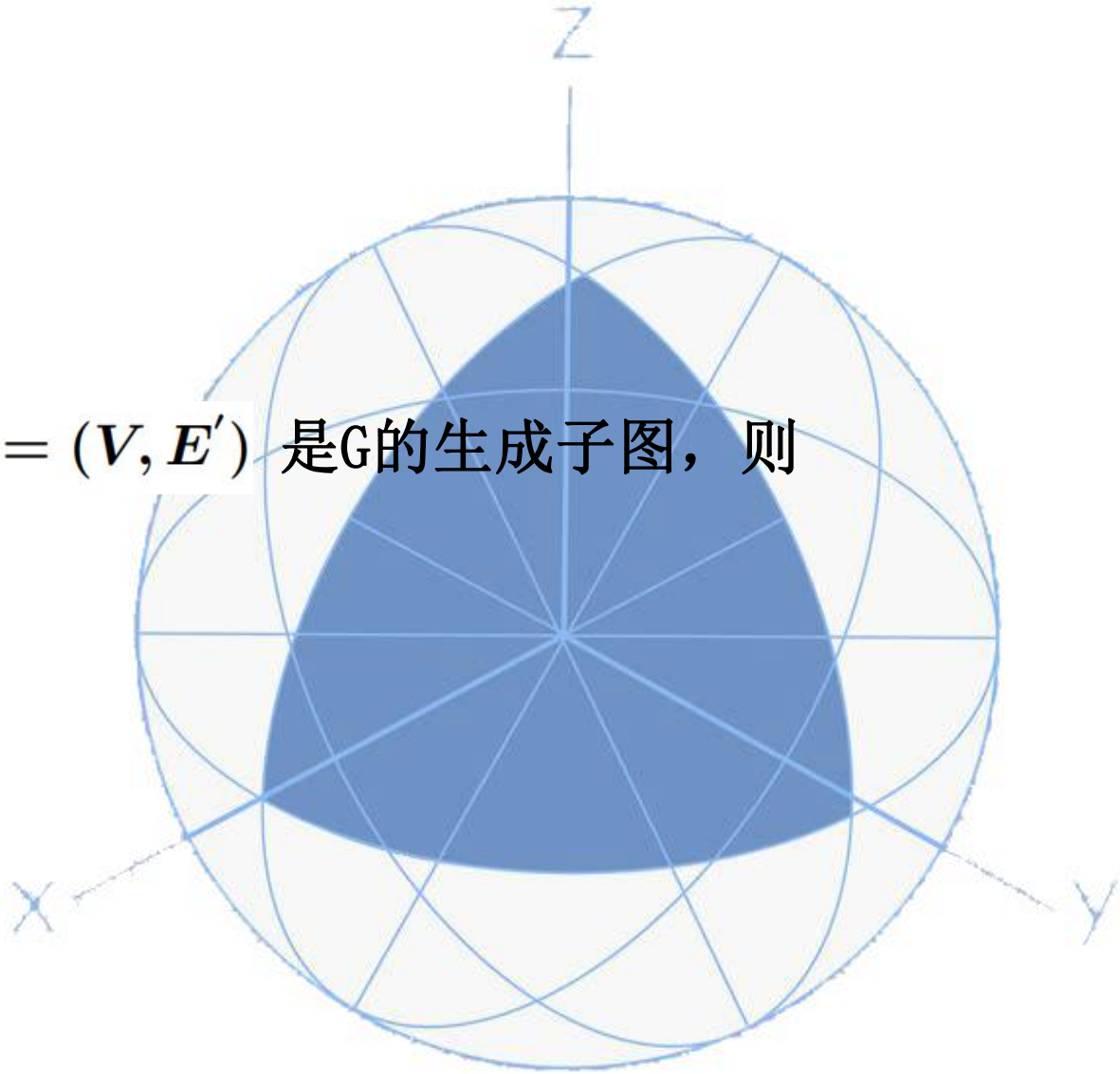
(4)树连通，在树中去掉任何一条边，图就不连通；

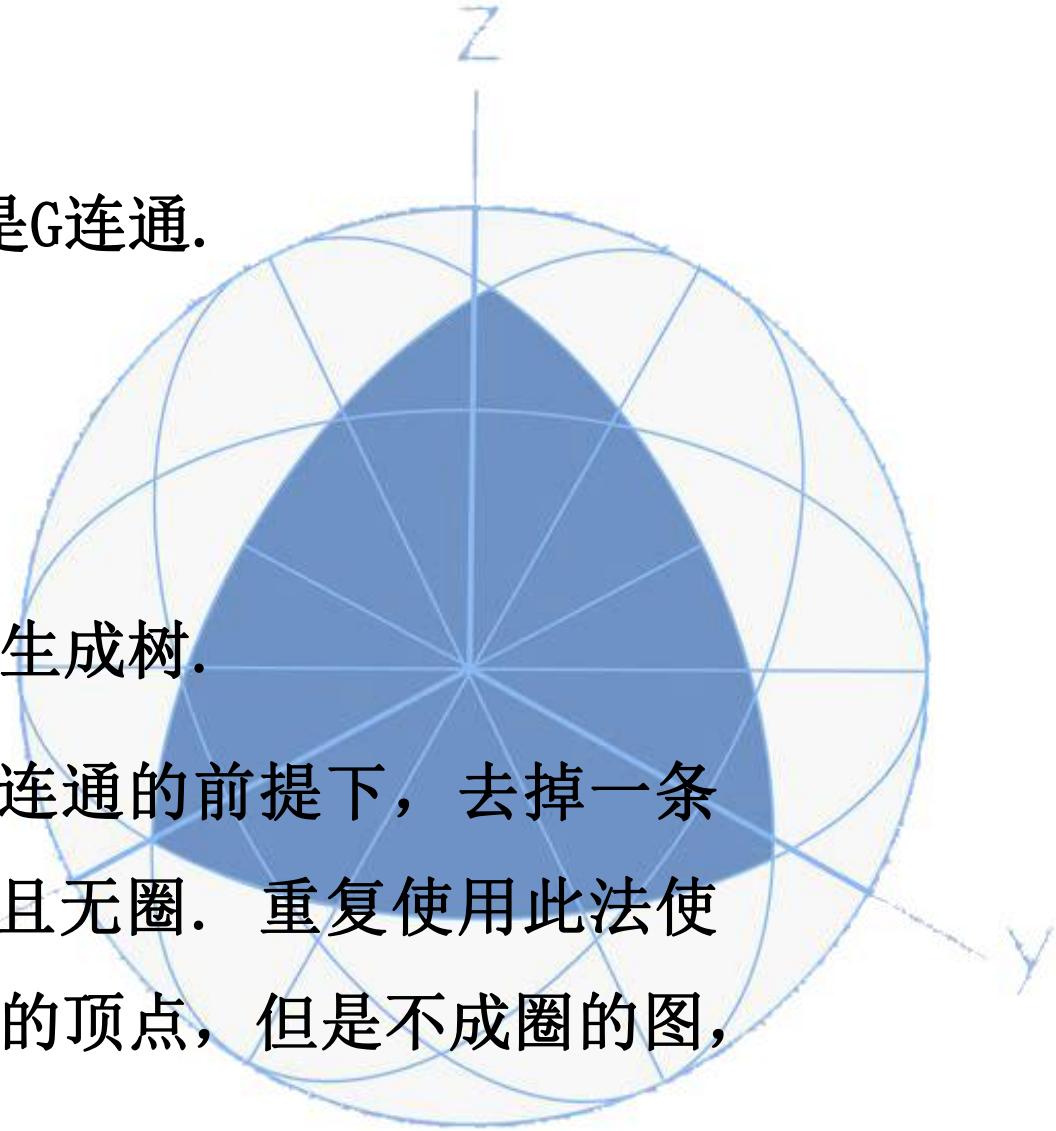
(5)含有n个顶点的树有n-1条边.



设G=(V, E)为连通图，若树 $T=(V,E')$ 是G的生成子图，则称T是G的生成树.

什么样的图才有生成树？





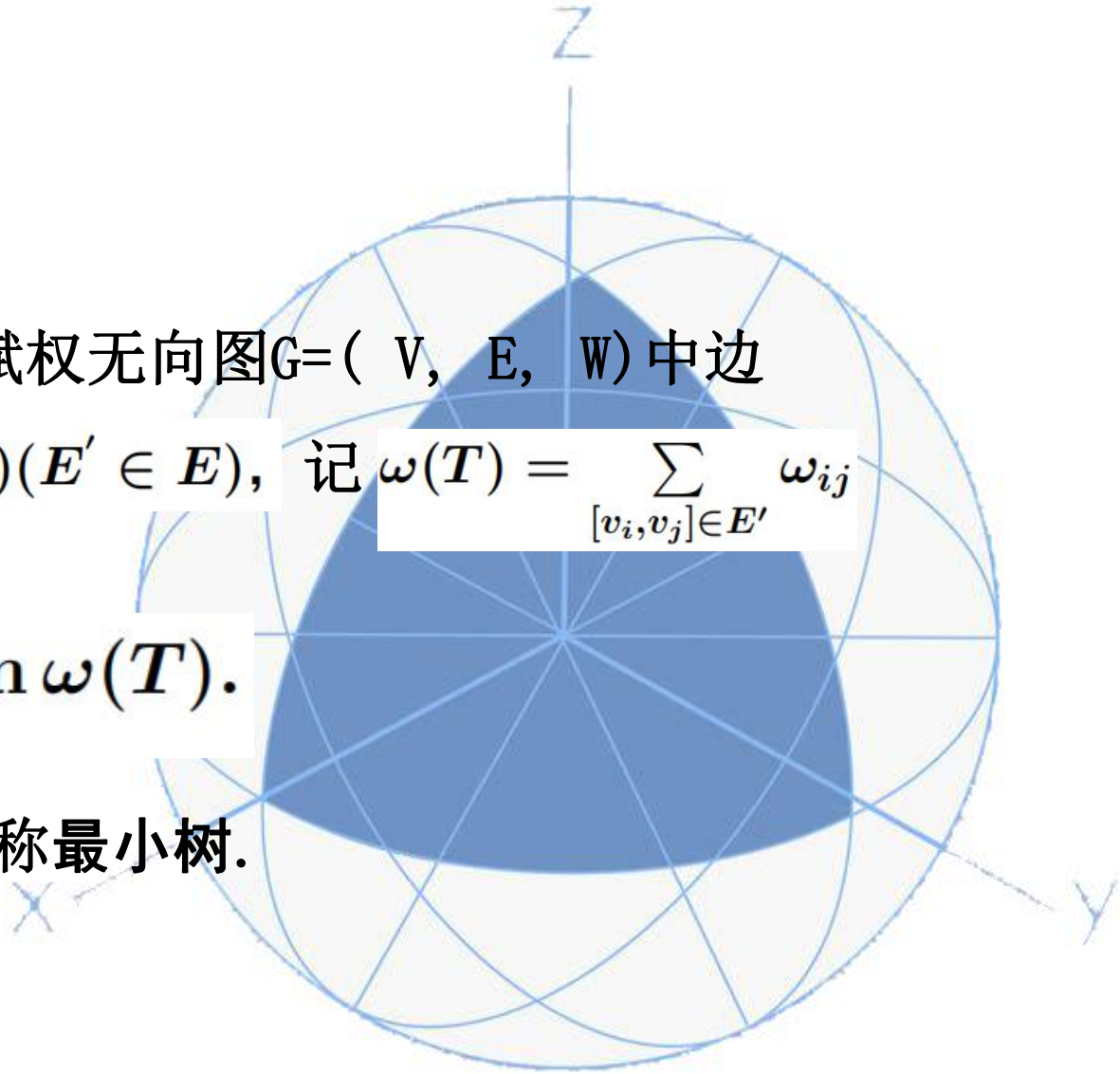
定理12.2 图G有生成树的充要条件是G连通.

证 必要性由定义直接可得.

充分性：设G是连通图.

(1) 若G不含圈，则G就是自身的一棵生成树.

(2) 若G含圈，任取一个圈，在保证连通的前提下，去掉一条或几条边，要包含这个圈上的所有点，且无圈. 重复使用此法使每个圈都受到破坏，最后保留下G中所有的顶点，但是不成圈的图，这就是G的生成树.



最小生成树问题：对一个连通的赋权无向图G=(V, E, W)中边权 $\omega_{ij} \geq 0$,设G 的生成树 $T=(V,E')(E' \in E)$，记 $\omega(T)=\sum_{[v_i,v_j]\in E'} \omega_{ij}$ 为树T的权满足：

$$W(T^*)=\min_{T} \omega(T).$$

的生成树 T^* 称为**最小生成树**，简称**最小树**.

对给定的连通图G用两种方法可以得到最小生成树.

(1) 用（丢边）破圈法求最小树的具体步骤是：

第一步：先任取一个圈，从圈中去掉一条权最大的边，若在同一个圈中有几条都是权最大边，则任选其中一条边去掉.

第二步：在余下的子图中，重复上述步骤，直至没有圈为止.

总之，将图G的边按边权从大到小依次排列后，依次按排定顺序去掉一条能构成圈的边，直至无圈为止.

(2)（取边）避圈法

将图G的边按边权从小到大依次排列后，依次按排定顺序考虑各边是否应该加上，原则为加上此边后不能构成圈.

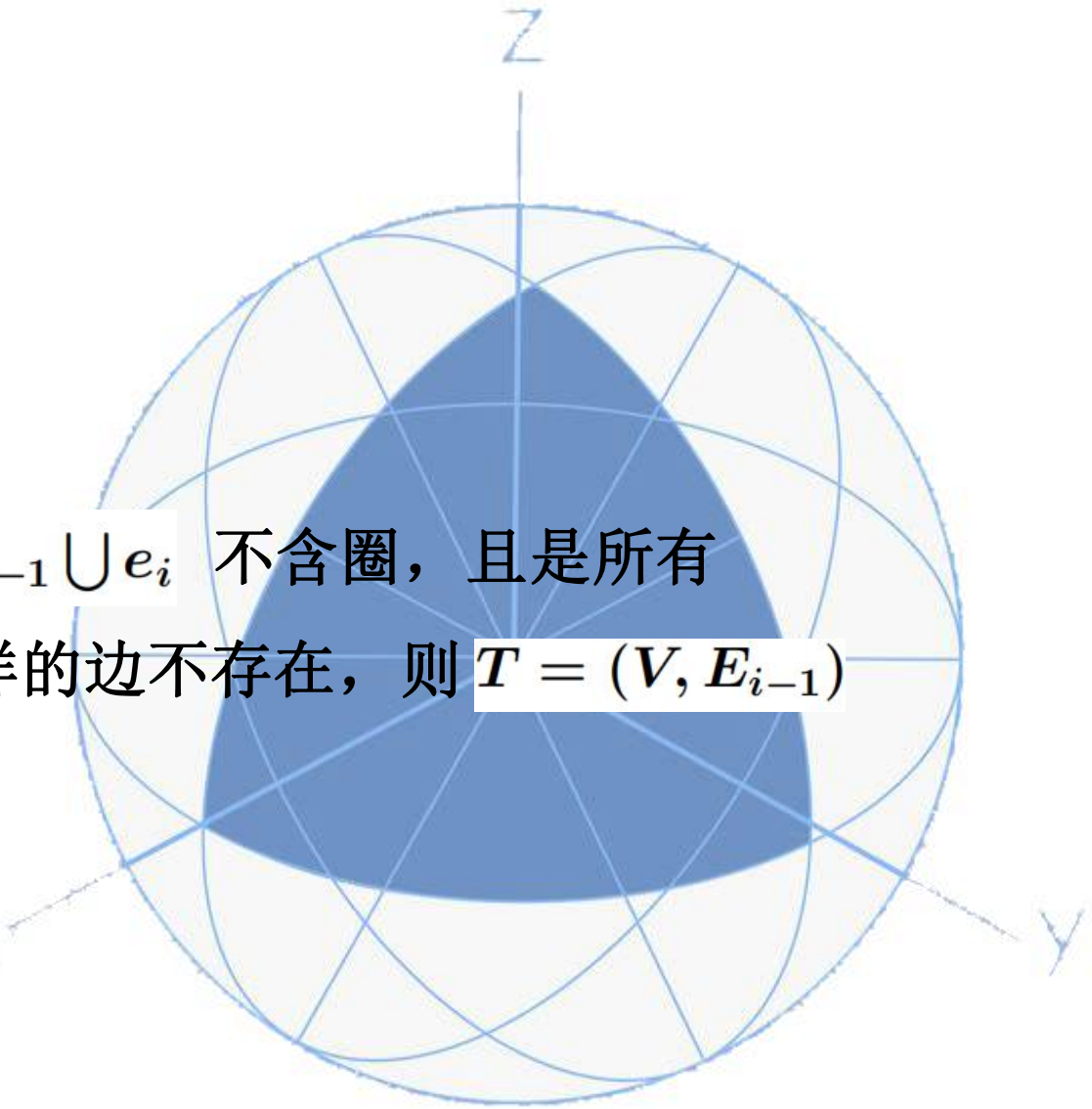
Kruskal算法的思想是：开始选一条最小权的边，在以后的每步中，总从未被选取的边中选一条权最小的边，并使之与以选取的边不构成圈.如果同时有几条都是最小权的边，则可从中任选一条.

具体步骤是：

第一步：令 $i=1, E_0=\emptyset$;

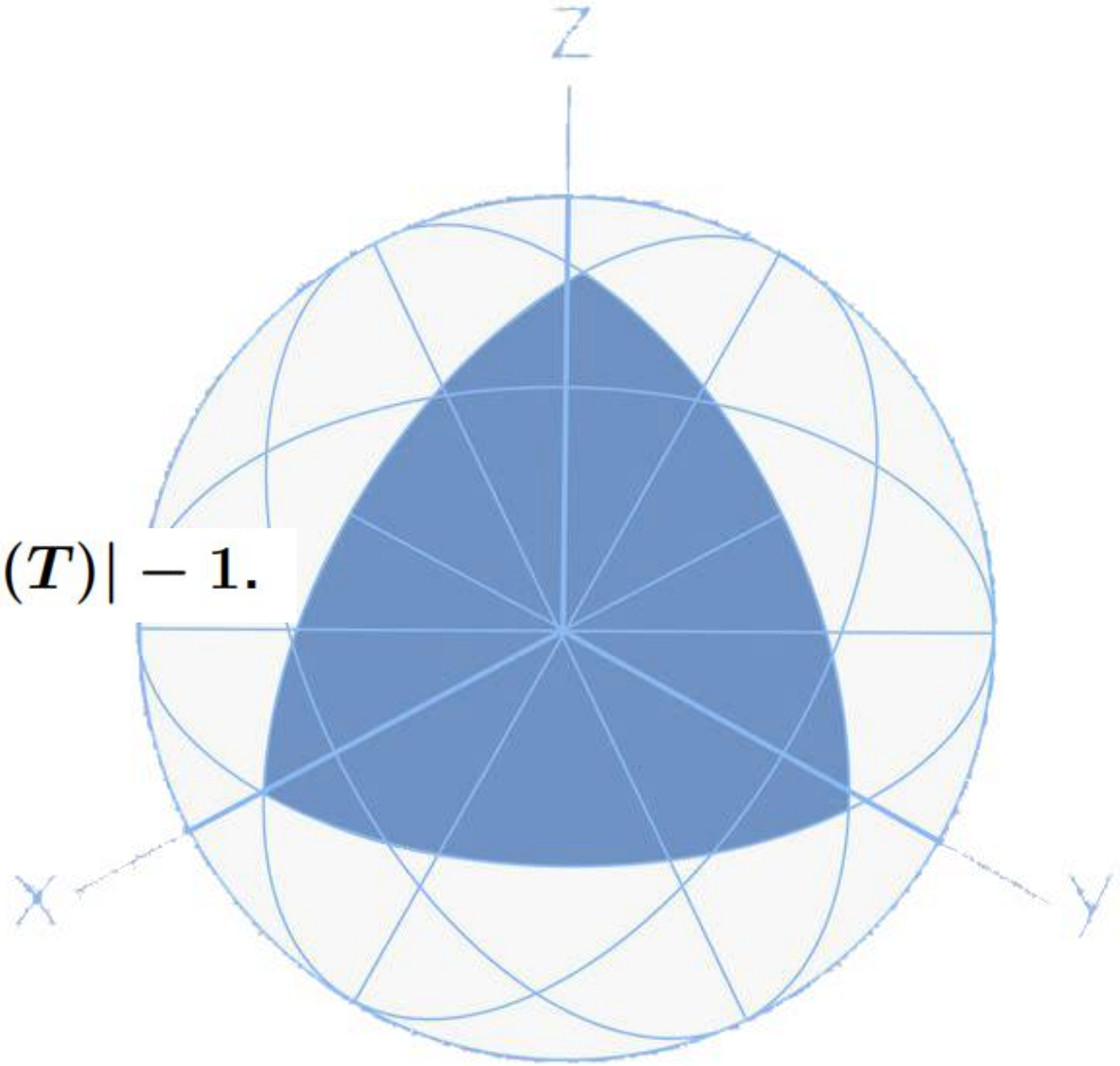
第二步：选边 $e_i \in E/E_{i-1}$，使 $E_{i-1} \bigcup e_i$ 不含圈，且是所有未被选取的边中权最小的一条. 如果这样的边不存在，则 $T=(V, E_{i-1})$ 就是最小树.

第三步：置 $i=i+1$， 转入第二步.



定理12.3： 设T是树，则

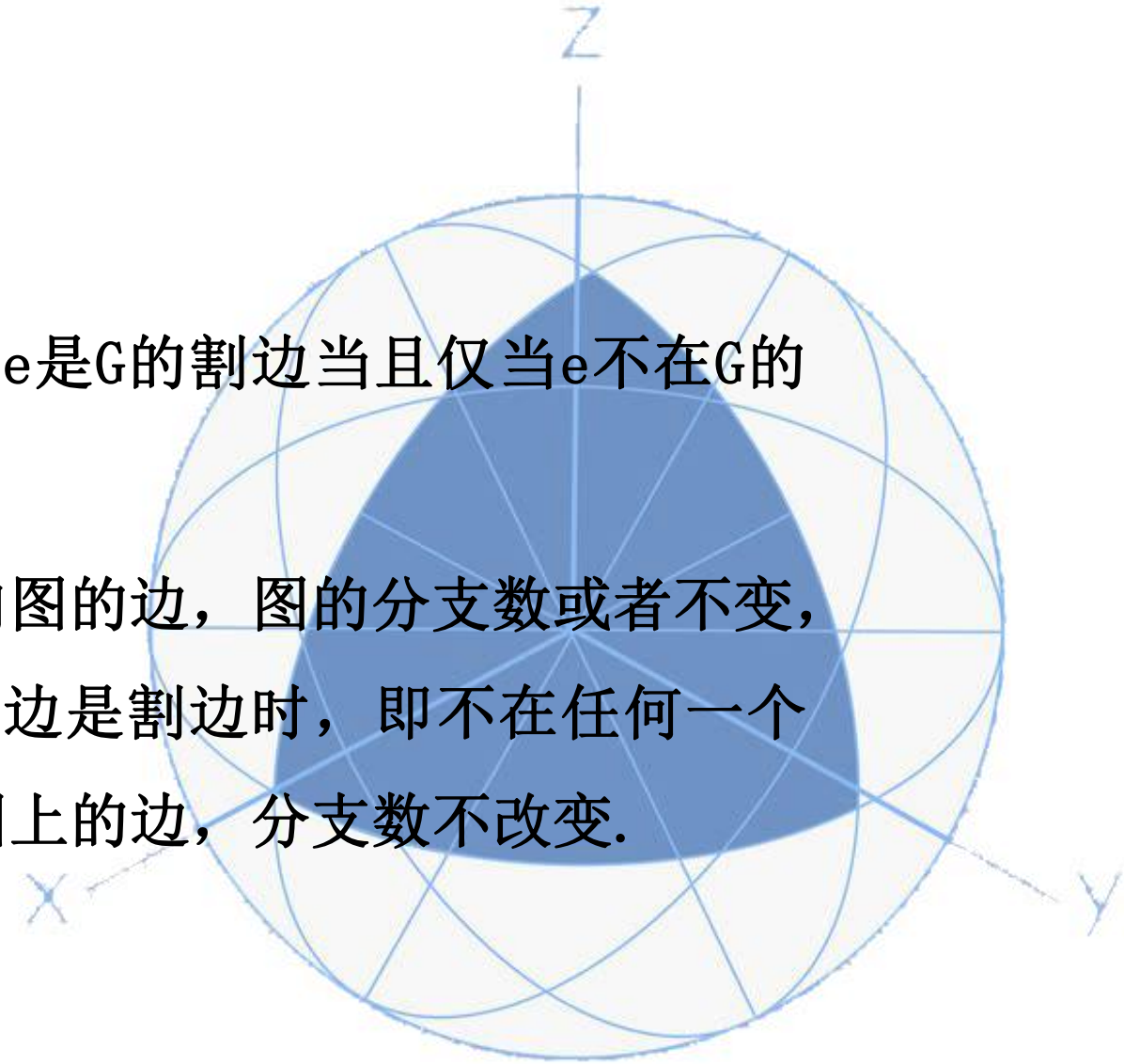
$$|E(T)| = |V(T)| - 1.$$



五、图的点割集、边割集与割边

一连通图中两顶点 x，y的距离记为 $d(x,y)=d_G(x,y)$，定义为x和y之间长度最小的路(最短路)的长度.

设G是一连通图，V' 表示G的顶点子集，若 $G-V'$ 为不连通的，则G的顶点子集 V' 称为一个**点割集**. 一个**k-点割集**是一个具有k个顶点的点割集. 类似地，也有**边割集**的概念. 特殊地，若边割集E'是单点集，即 $E'=\{e\}$，则称e为**割边或桥**.



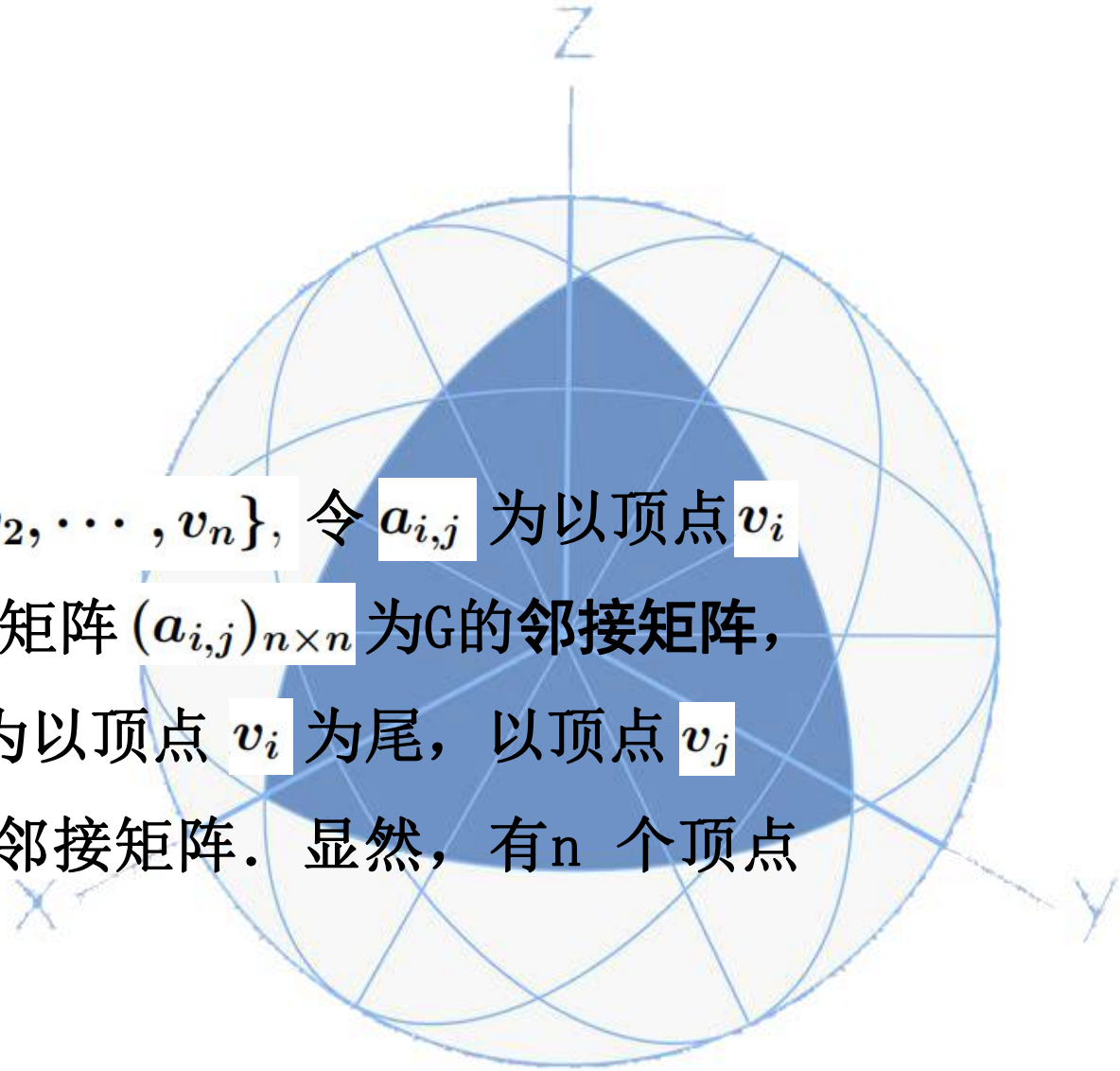
定理12.4 设G是一无向图. 则e是G的割边当且仅当e不在G的任何圈中.

从定理不难看出：去掉一个无向图的边，图的分支数或者不变，或者仅增加1. 并且当且仅当去掉的边是割边时，即不在任何一个圈上的边，分支数增加1，而去掉圈上的边，分支数不改变.

六、图的存储结构表示

1. 邻接矩阵表示法

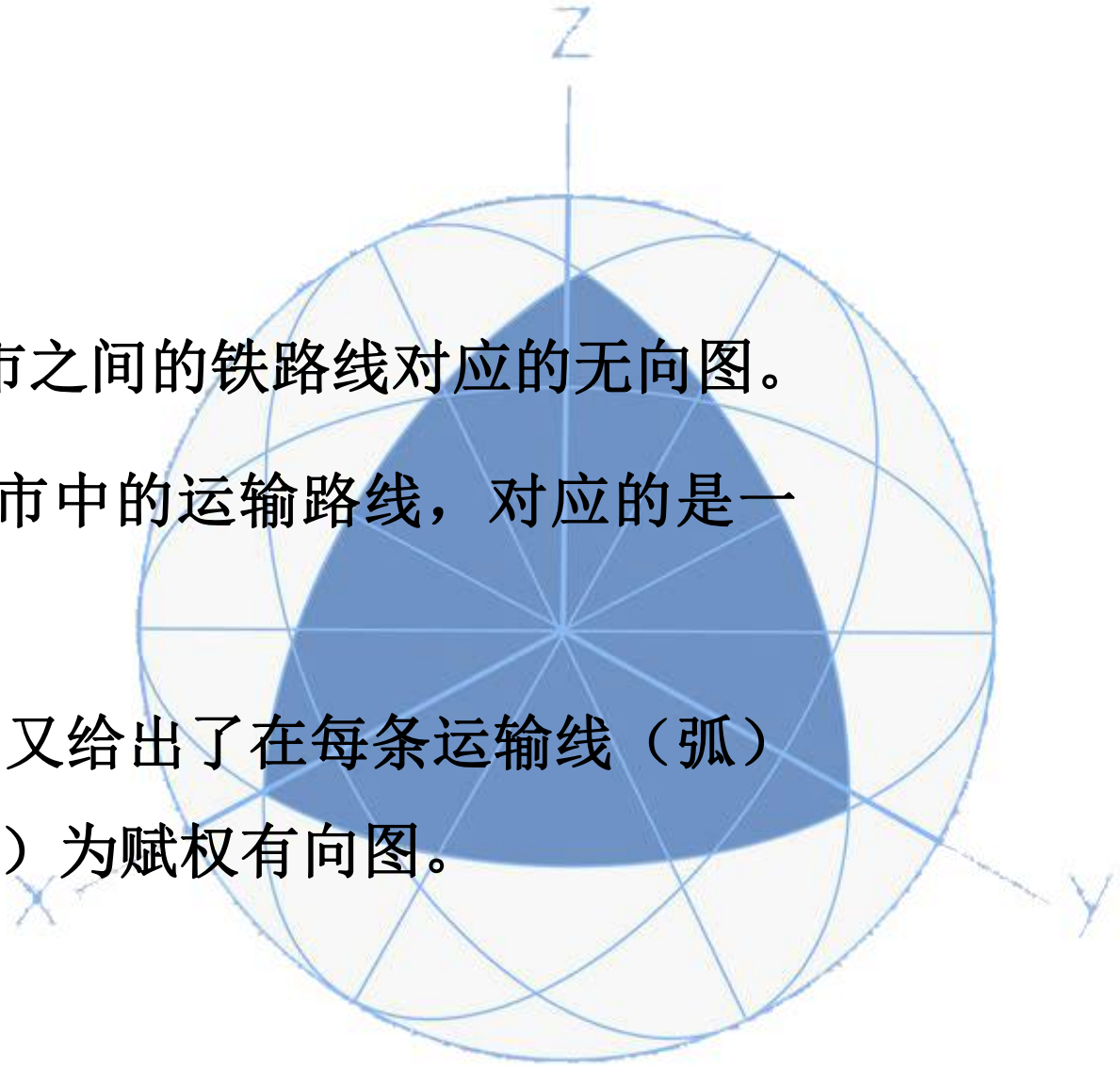
设无向图G=(V, E)，$V=\{v_1, v_2, \cdots, v_n\}$，令 $a_{i,j}$ 为以顶点 v_i 与顶点 v_j 为端点的边的条数，则称矩阵 $(a_{i,j})_{n\times n}$ 为G的**邻接矩阵**，记作A(G). 当G是有向图时，令 $a_{i,j}$ 为以顶点 v_i 为尾，以顶点 v_j 为头的边的条数，则得到有向图的邻接矩阵. 显然，有n 个顶点的图的邻接矩阵为n阶方阵.



如果顶点i到j有一条弧，则邻接矩阵中对应的元素(i, j) 的值为1；否则为0。无向图的邻接矩阵是对称矩阵，有向图的邻接矩阵则不一定对称。

对于赋权图的邻接矩阵表示，一条弧所对应的元素不再是1，而是相应的权值。对应权值代表的意义不同，通常(i, i)处的值为0；没有弧对应的元素(i, j)处的值为 ∞。

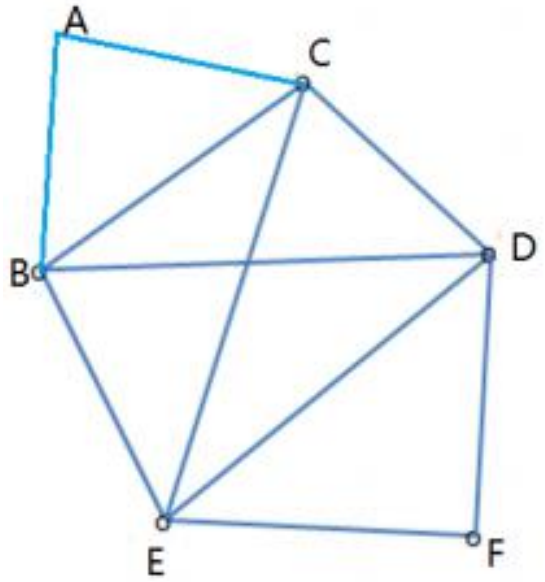
邻接矩阵表示的是图的顶点与顶点之间的关系。



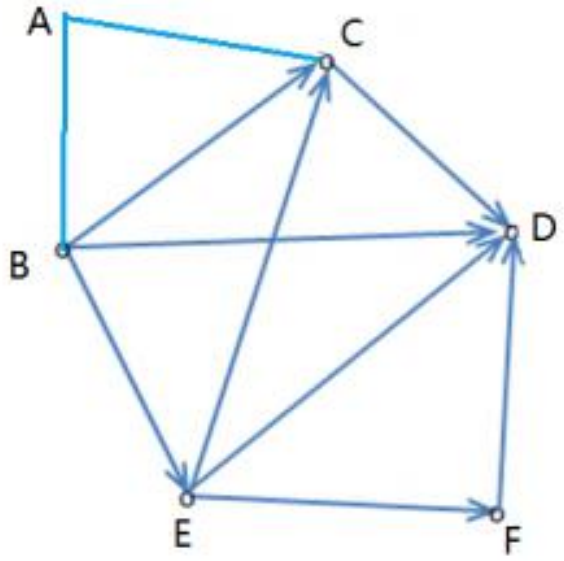
例12.4　图（1）为连通六个城市之间的铁路线对应的无向图。

图（2）为某种物资在这六个城市中的运输路线，对应的是一个有向图。

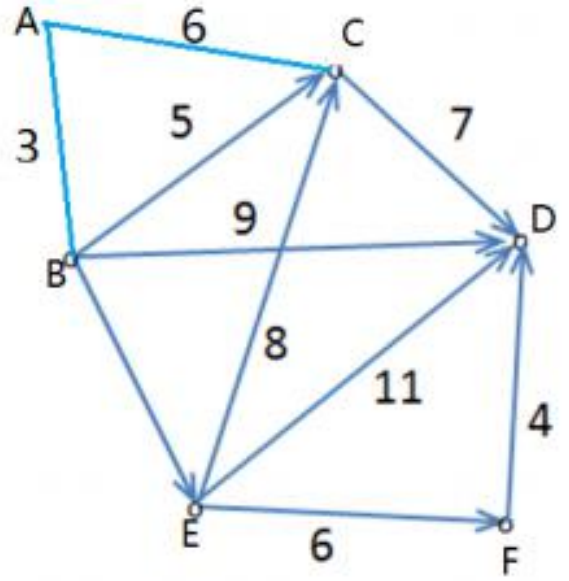
图（3）在（2）的运输路线中，又给出了在每条运输线（弧）上所花费的费用，称为弧权，图（3）为赋权有向图。



(1)



(2)



(3)

上图（1）、（2）、（3）分别为无向图，有向图，赋权有向图，它们的存储可用下列邻接矩阵分别表示为：

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 3 & 6 & \infty & \infty & \infty \\ \infty & 0 & 5 & 9 & 10 & \infty \\ \infty & \infty & 0 & 7 & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & 8 & 11 & 0 & 6 \\ \infty & \infty & \infty & 4 & \infty & 0 \end{bmatrix}$$

2. 关联矩阵表示法

设无向图G=(V, E)，$V=\{v_1,v_2,\cdots,v_n\}$，$E=\{e_1,e_2,\cdots,e_m\}$，令 $m_{i,j}$ 为顶点 v_i 与边 e_j 的关联次数，有n 个顶点，m 条弧（边）的图的关联矩阵为n × m 阶矩阵。

在关联矩阵中，每行对应于图的一个顶点，每列对应于图的一条弧（边）。如果一个顶点是一条弧的起点，则关联矩阵中对应的元素为1；如果一个顶点是一条弧的终点，则关联矩阵中对应的元素为-1(无向图的边则两端点均为1)；如果一个顶点与一条弧不关联，则关联矩阵中对应的元素为0。

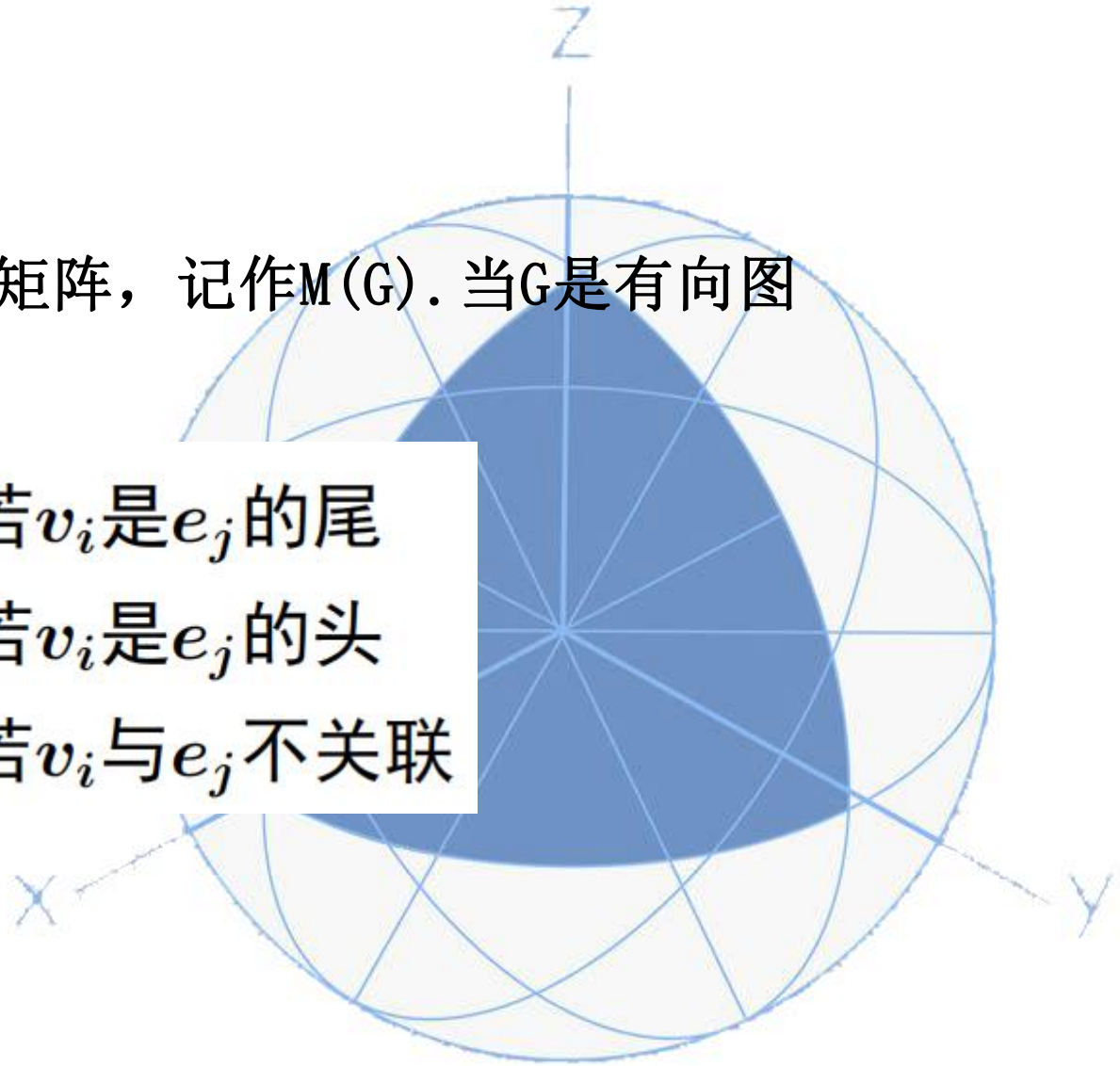
如果图中的弧赋有权，可以把关联矩阵增加一行，把每一条弧所对应的权存储在增加的行中. 即：

$$m_{i,j}=\begin{cases}1, & \text{若}v_i\text{与}e_j\text{关联且}e_j\text{是连杆}\\ 2, & \text{若}v_i\text{与}e_j\text{关联且}e_j\text{是环}\\ 0, & \text{若}v_i\text{与}e_j\text{不关联}\end{cases}$$

则称矩阵 $(m_{i,j})_{n\times m}$ 为G的关联矩阵，记作M(G).当G是有向图时，令

$$m_{i,j}=\begin{cases}1, & \text{若}v_i\text{是}e_j\text{的尾}\\ -1, & \text{若}v_i\text{是}e_j\text{的头}\\ 0, & \text{若}v_i\text{与}e_j\text{不关联}\end{cases}$$

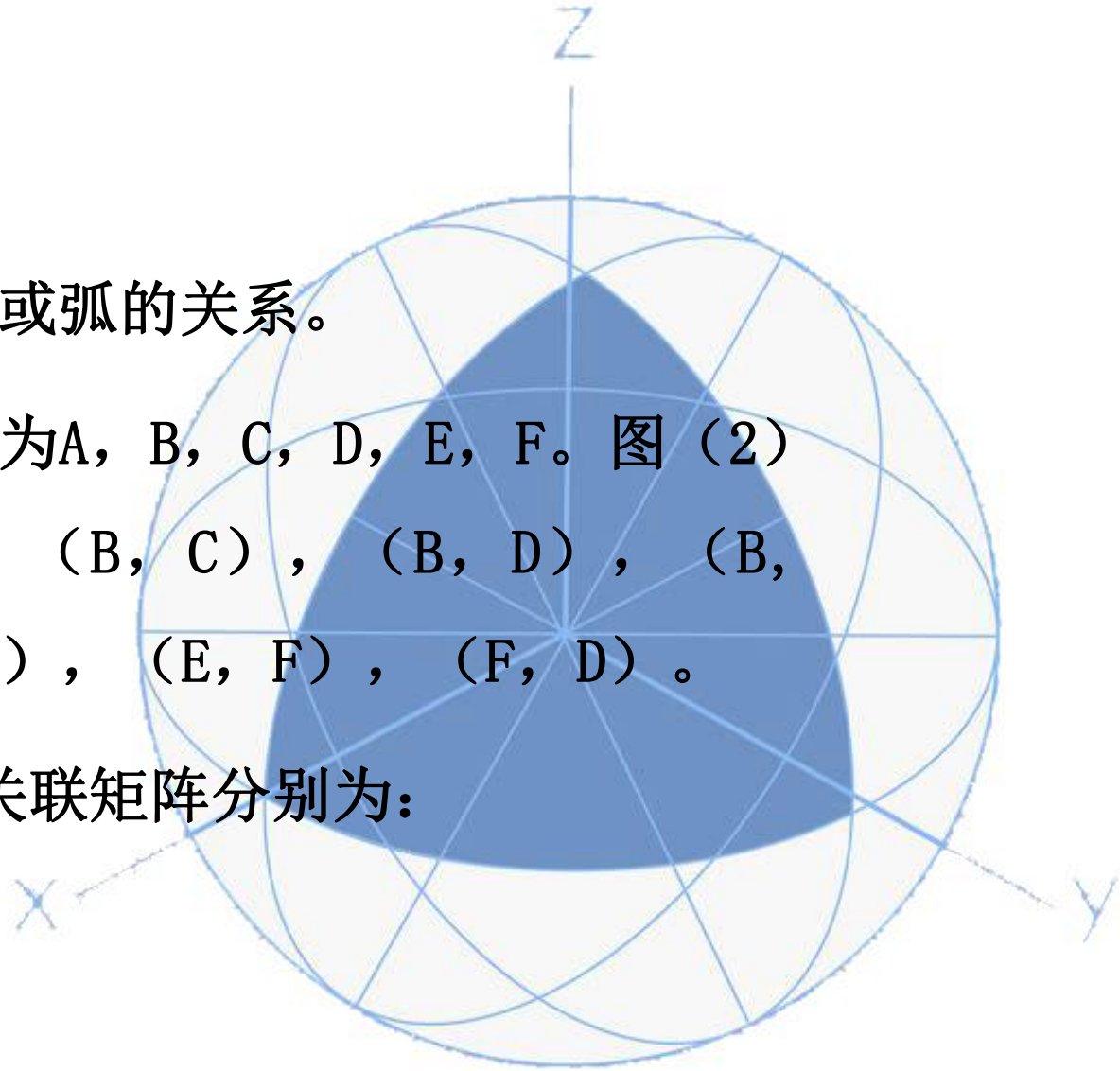
则得到有向图的关联矩阵.



例12.5:例12.3中图G的关联矩阵和邻接矩阵分别为：

$$M(G)=\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \qquad A(G)=\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 2 \\ 1 & 0 & 2 & 0 \end{bmatrix}$$

关联矩阵表示的是图的顶点与边或弧的关系。



关联矩阵表示的是图的顶点与边或弧的关系。

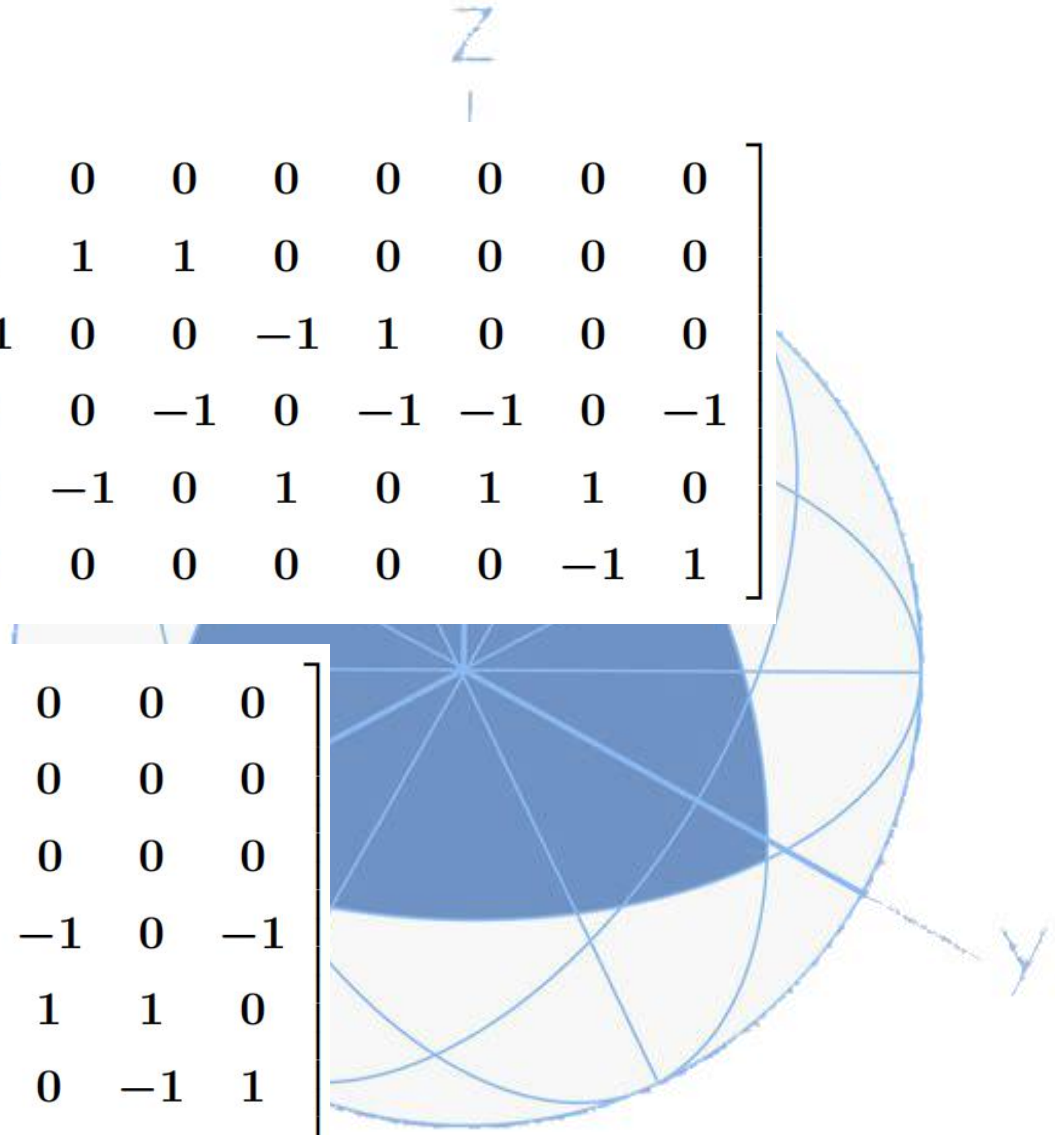
记例12.4中图（1）中的节点1-6为A，B，C，D，E，F。图（2）中弧1-10为（A，B），（A，C），（B，C），（B，D），（B，E），（C，D），（C，E），（E，D），（E，F），（F，D）。

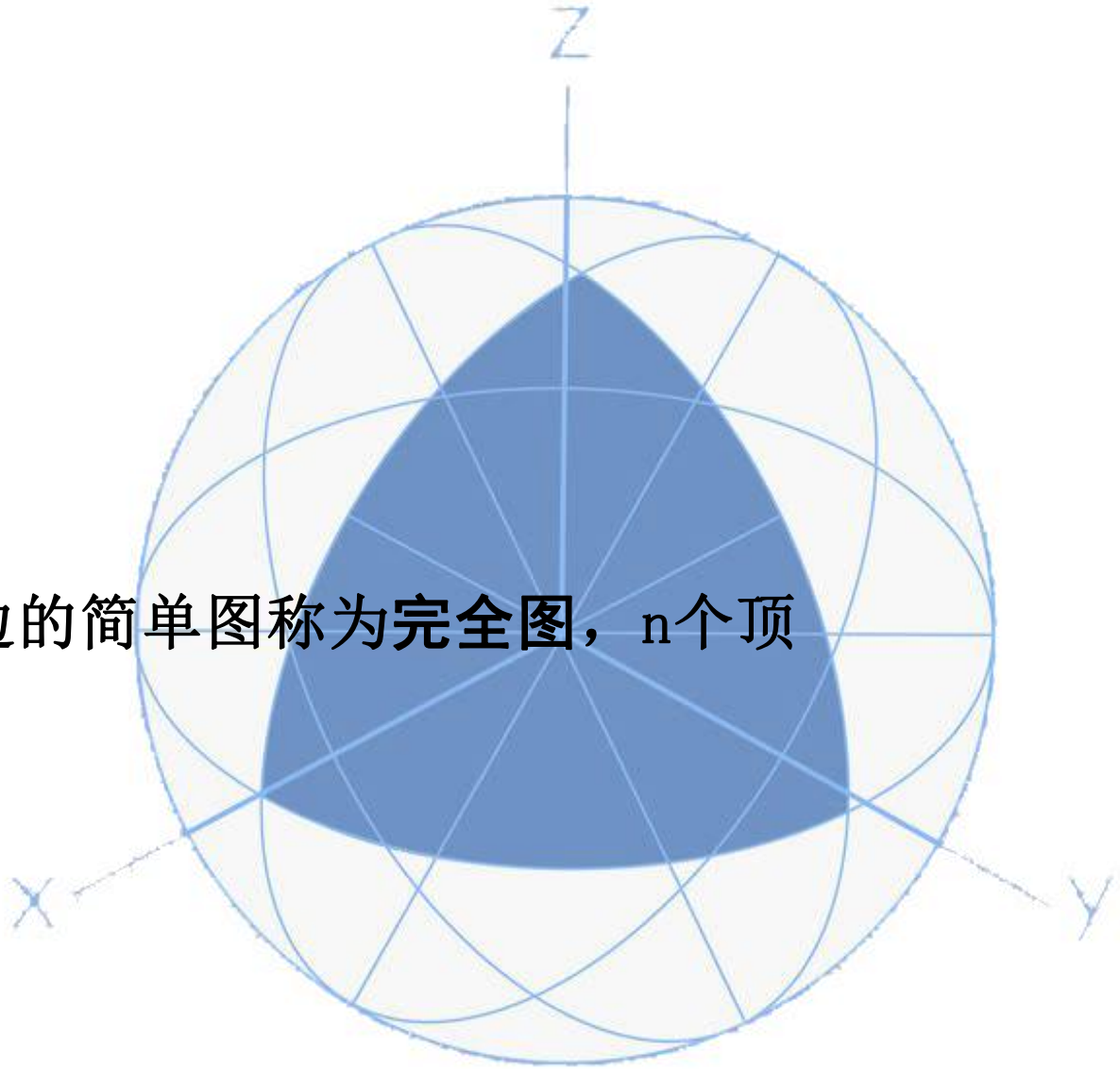
图12.3（1）、（2）、（3）的关联矩阵分别为：

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 3 & 6 & 5 & 10 & 9 & 8 & 7 & 11 & 6 & 4 \end{bmatrix}$$

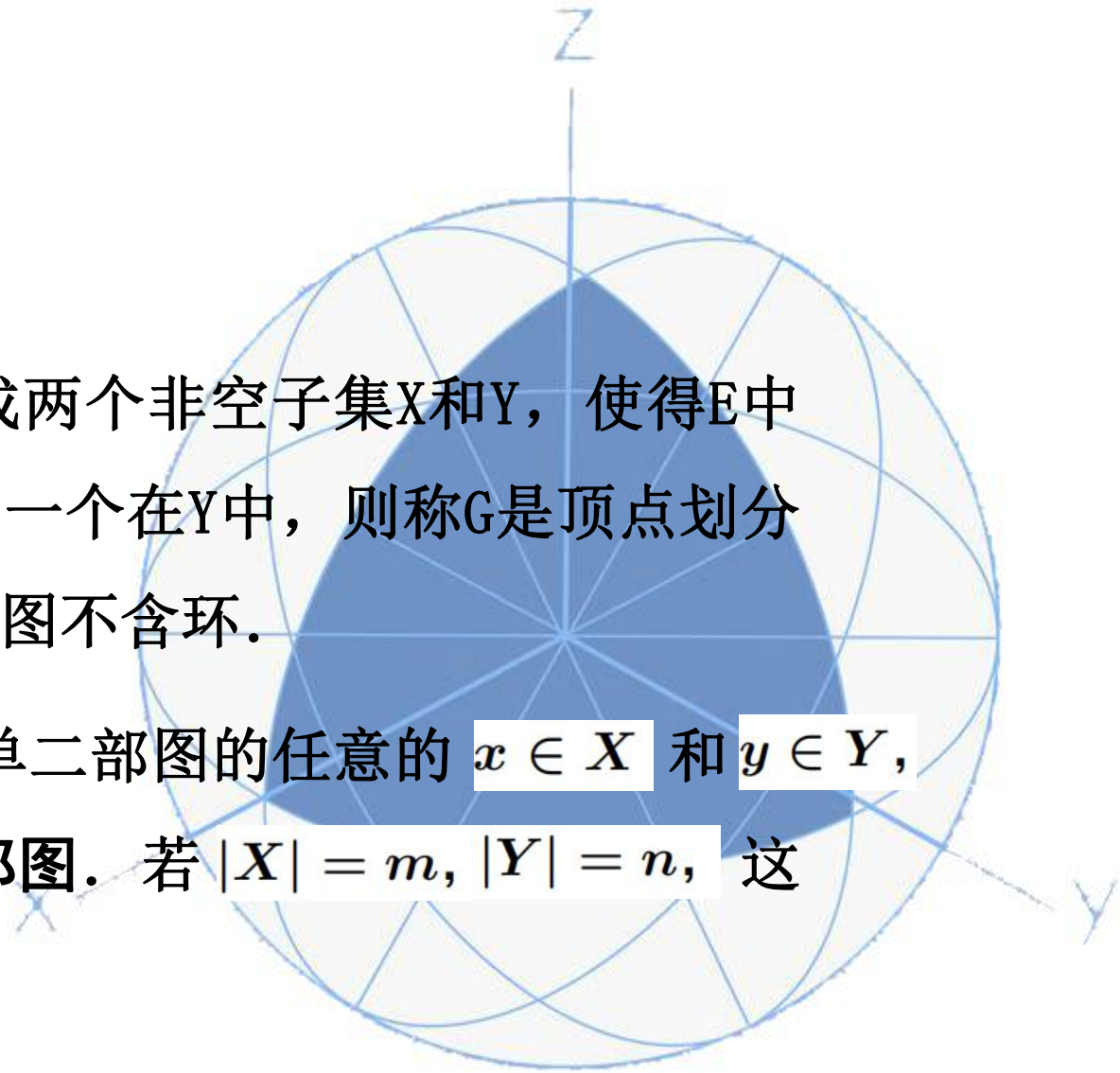




七、一些特殊的图

(1) 完全图

每对不同的顶点之间都有一条边的简单图称为**完全图**，n个顶点的完全图记为 K_n



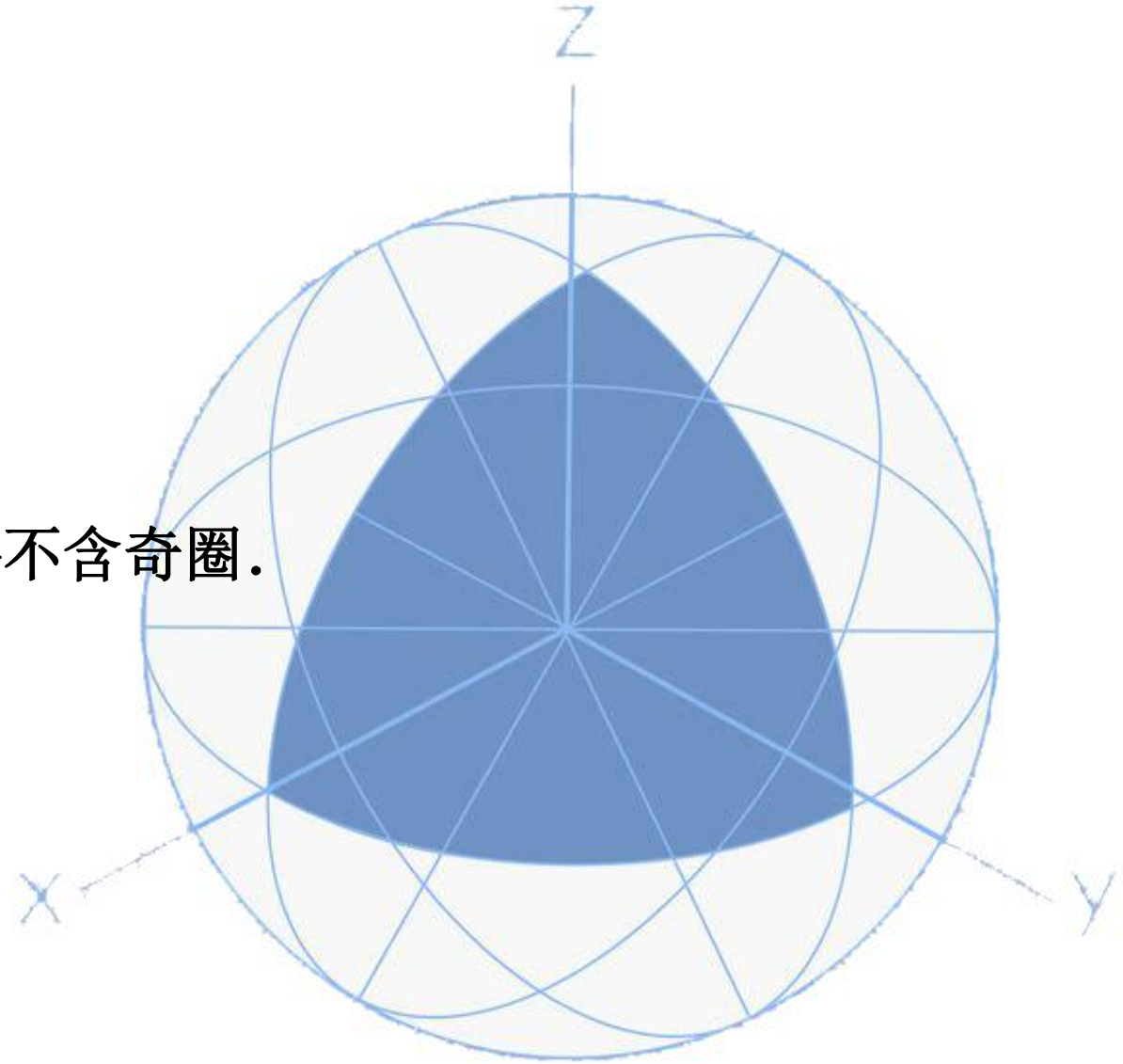
(2) 二部图

设无向图G=(V,E)，若V可划分成两个非空子集X和Y，使得E中任意一边的两个端点都一个在X中，一个在Y中，则称G是顶点划分为(X,Y)的一个**二部图**. 显然，二部图不含环.

若一个顶点划分为(X, Y)的简单二部图的任意的 $x \in X$ 和 $y \in Y$, 都有边(x, y)，则称之为**完全二部图**. 若 $|X| = m, |Y| = n,$ 这样的完全二部图记为 $K_{m,n}$.

定理12.5

一个无向图G是二部图当且仅当G不含奇圈.

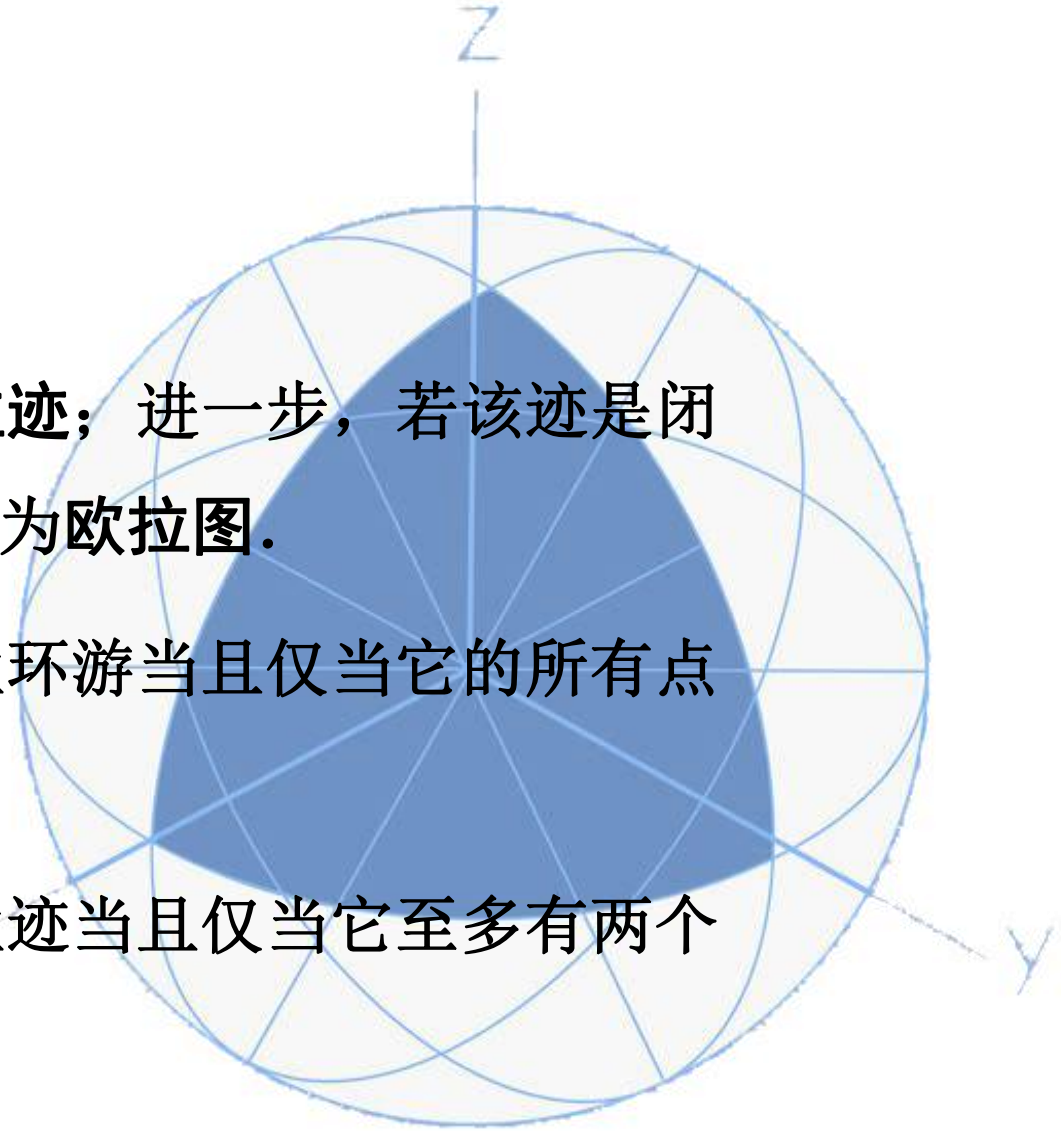


八、欧拉环游

包含连通图G的所有边的迹称G的**欧拉迹**；进一步，若该迹是闭迹，则称为**欧拉环游**.有欧拉环游的图称为**欧拉图**.

定理12.6　一个非平凡连通图有欧拉环游当且仅当它的所有点都是偶点.

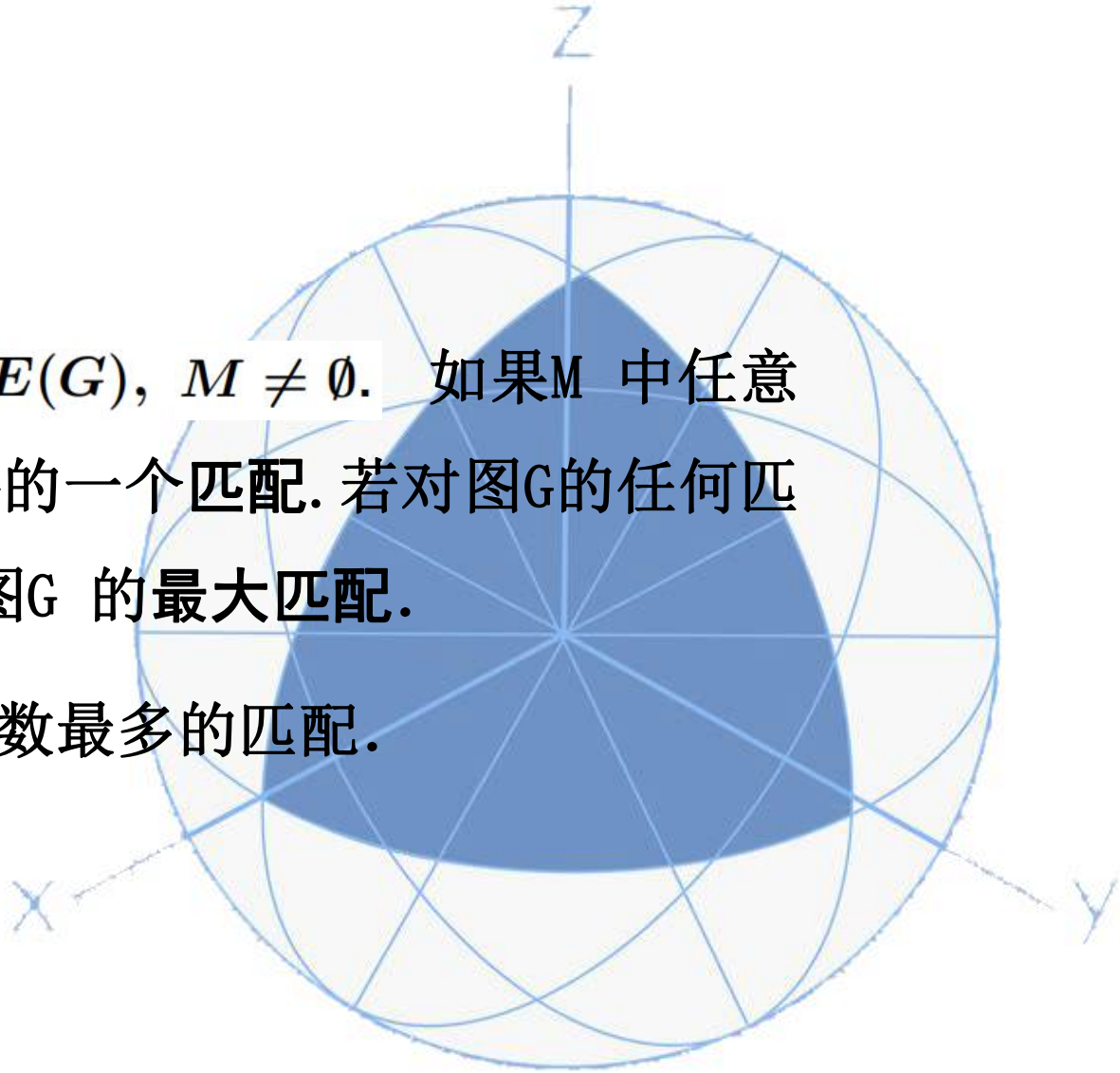
推论12.2　一个非平凡连通图有欧拉迹当且仅当它至多有两个奇点.



九、匹配

定义12.1：设G是无环图，$M \subseteq E(G)$，$M \neq \emptyset$. 如果M 中任意两条边在G中均不相邻，则称M是图G的一个**匹配**. 若对图G的任何匹配M'，均有 $|M'| \leq |M|$，则称M为图G 的**最大匹配**.

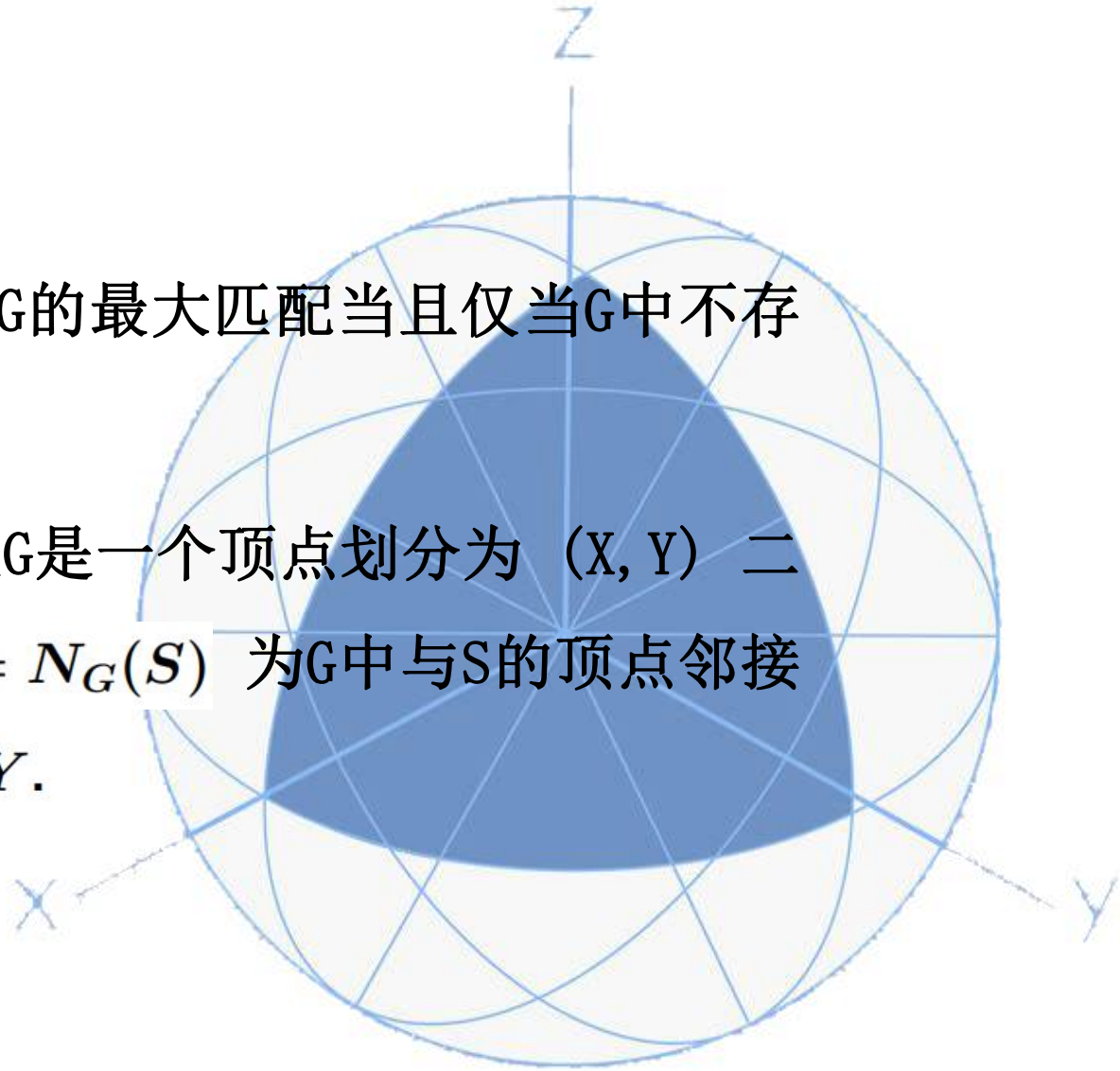
对非赋权图，最大匹配也就是边数最多的匹配.



定义12.2:设M是图G的匹配，$v \in V(G)$. 若v与M中某边(也只能是某一条边)关联，则称v为**M-饱和点**，否则称为**M-非饱和点**.若图G的顶点都是M-饱和点，则称M为G的**完美匹配**.

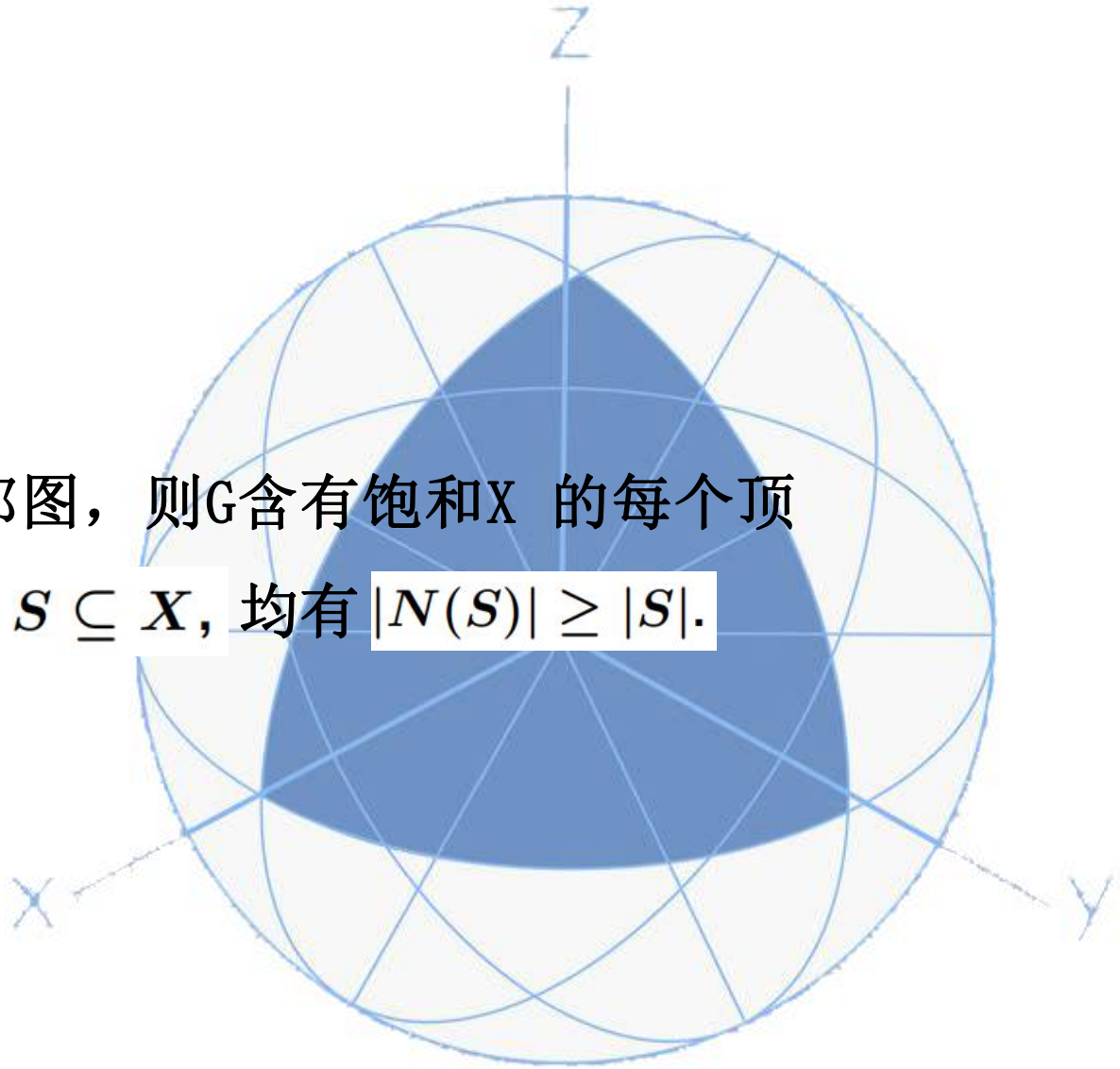
由定义可知，完美匹配一定是最大匹配，反之则未必.

定义12.3　设M是图G的匹配，P是G的一条路．若在路P上，M的边和E(G)-M的边交替出现，则称P是G的一条**M-交错路**．若M-交错路P的两个端点为M-非饱和点，则称P为**M-增广路**.



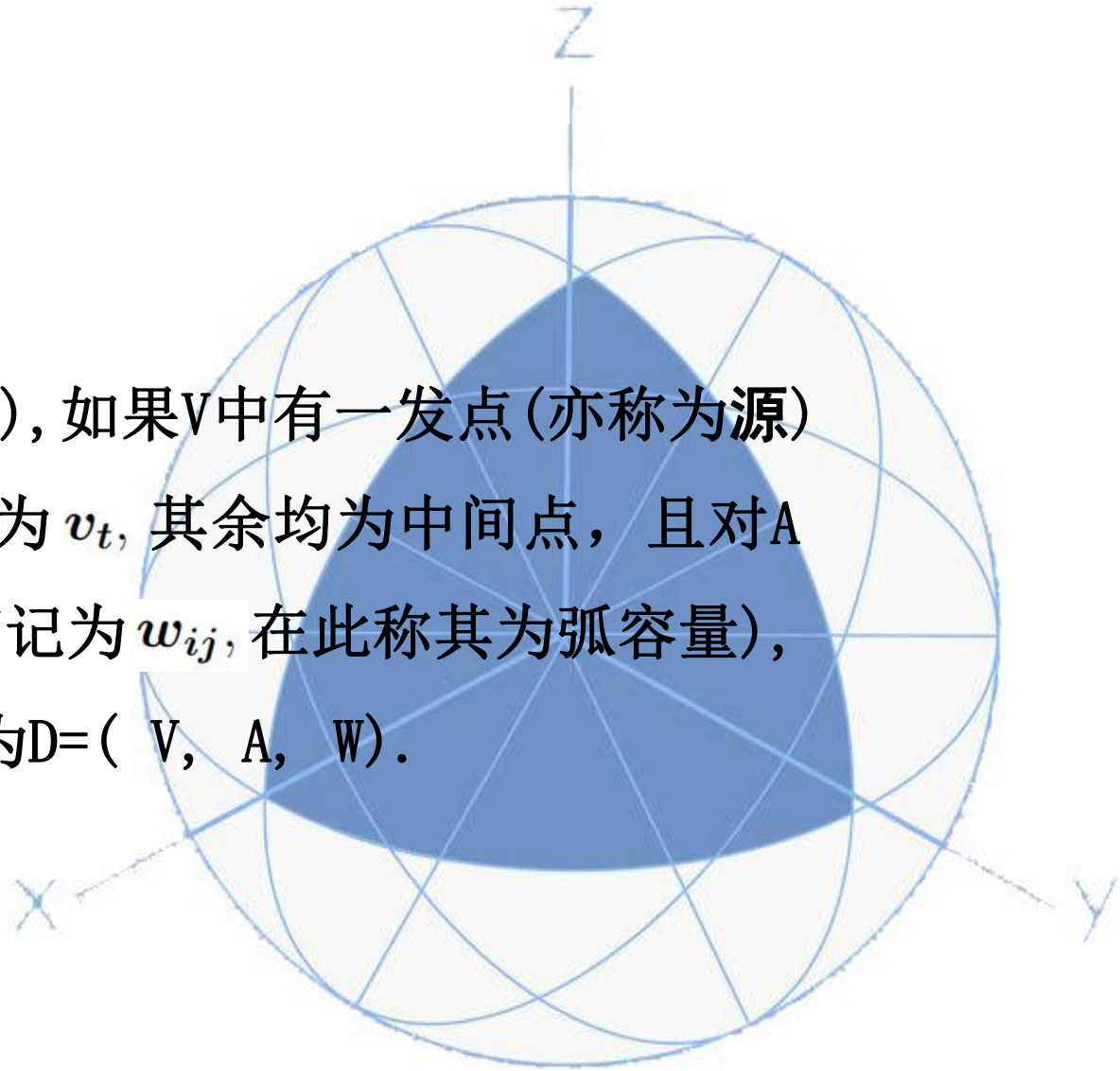
定理12.7　(Berge,1957)M是图G的最大匹配当且仅当G中不存在M-增广路.

现在我们回到二部图的情形. 设G是一个顶点划分为 (X,Y) 二部图，对任意 $S \subseteq X$，记 $N(S) = N_G(S)$ 为G中与S的顶点邻接的所有顶点的集合.显然，$N(S) \subseteq Y$.



定理12.8　(Hall,1935年)

设G是顶点划分为(X, Y)的二部图，则G含有饱和X 的每个顶点的匹配 M 的充要条件是，对任意 $S \subseteq X$，均有 $|N(S)| \geq |S|$.



十、网络与流

定义12.4：对于有向图D=(V, A),如果V中有一发点(亦称为**源**)记为 v_s，还有一收点(亦称为**汇**)记为 v_t，其余均为中间点，且对A的每条弧均赋权 $W(v_i, v_j) \geq 0$ (简记为 w_{ij}，在此称其为弧容量)，则称这样的赋权有向图D为**网络**,记为D=(V, A, W).

定义12.5：在D中，通过弧 (v_i, v_j) 的物运量 f_{ij} 称为通过弧 (v_i, v_j) 的**流量**，所有弧上流量的集合 $f = \{f_{ij}\}$ 称为该网络D的一个**流**.

在运输网络的实际问题中，我们可以看出，对于流有两个明显的要求：

一是每个弧上的流量不能超过该弧的最大通过能力（即弧的容量）；

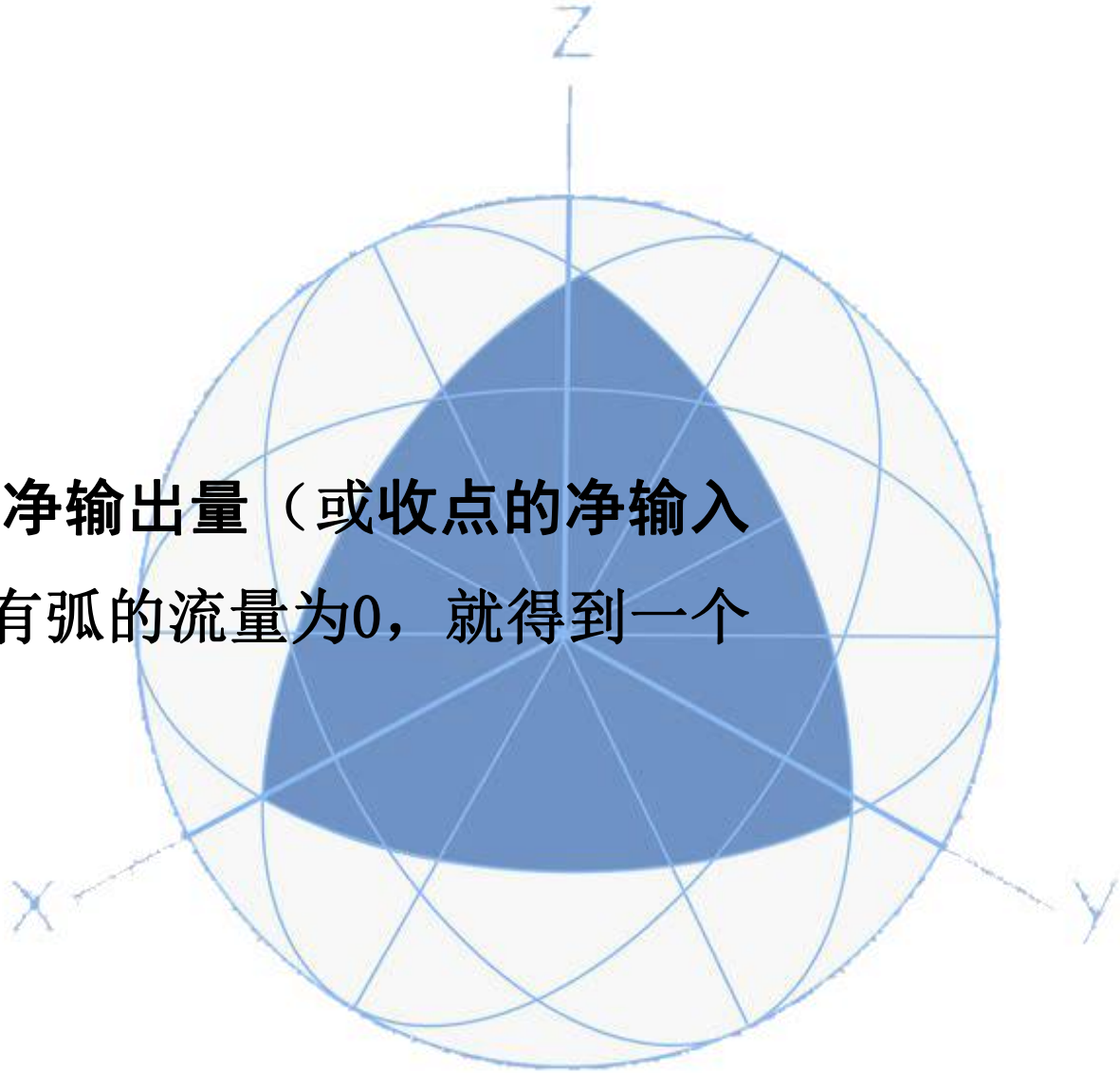
二是对每个中间点，运出这点的产品总量与运进这点的产品总量相同，这是由于中间点只起转运站的作用. 因此有：

定义12.6：满足下述两个条件的流 f 称为**可行流**：

(1)弧流量限制条件：$0 \leq f_{ij} \leq w_{ij}, (v_i, v_j) \in A.$

(2)平衡条件：

$$\sum f_{ij} - \sum f_{ji} = \begin{cases} v(f) & i = s \\ 0 & i \neq s, t \\ -v(f) & i = t \end{cases}$$



可行流f的流量v(f)定义为**发点的净输出量**（或**收点的净输入量**）. 可行流总是存在的，比如令所有弧的流量为0，就得到一个可行流（称为零流），其流量为0.

定义12.7：设f是一个可行流，μ 是从v_s到 v_t 的一条路，若 μ 满足下列条件：

（i）在弧 $(v_i, v_j) \in \mu^+$ (称为正向弧)上，$0 \leq f_{ij} < c_{ij}$, 即正向弧的集合 μ^+ 中每一弧是f-非饱和弧；

（ii）在弧 $(v_i, v_j) \in \mu^-$ (反向弧)上，$0 < f_{ij} \leq c_{ij}$, 即反向弧的集合 μ^- 中每一弧是f-非零流弧，那么称之为**f-增广路.**

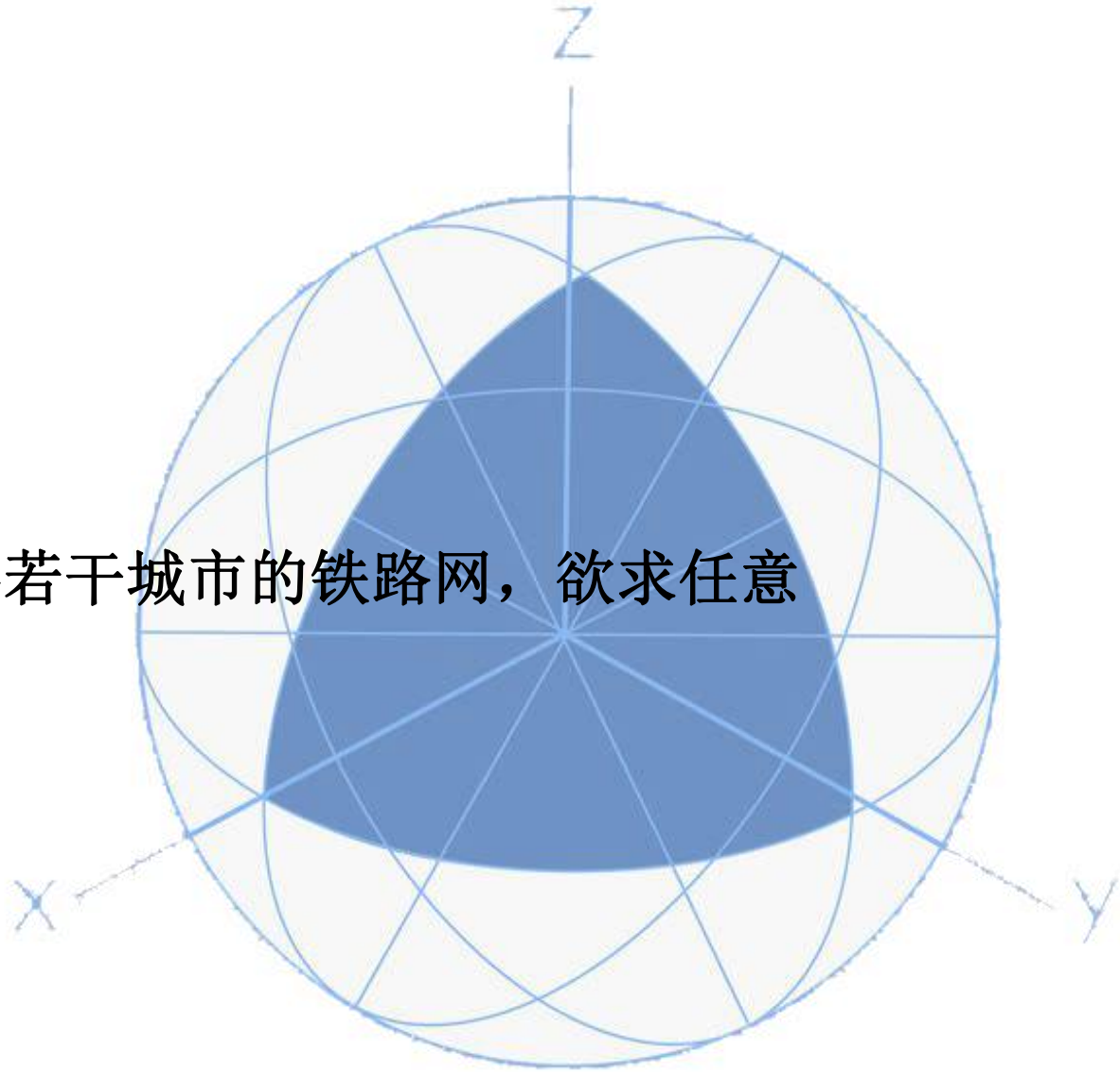
引理12.1：可行流f^* 是网络D的最大流当且仅当D中不存在f^*-增广路.

定义12.8：如果将网络D=(V, A, W) 的V剖分为两部分 V_s 和$\overline{V_s}$，使$v_s \in V_s, v_t \in \overline{V_s}$ 且 $V_s \cap \overline{V_s} = \emptyset$, $V_s \cup \overline{V_s} = V$ 则把从 V_s 指向 $\overline{V_s}$ 的弧的全体称为分离 V_s 和 $\overline{V_s}$ 的一个**截集**，记为 $(V_s, \overline{V_s})$，即 $(V_s, \overline{V_s}) = \{v_i, v_j \mid v_i \in V_s, v_j \in \overline{V_s}, (v_i, v_j) \in A\}$，截集 $(V_s, \overline{V_s})$ 中所有弧的容量之和称为该截集的截量，记为 $W(V_s, \overline{V_s})$. 在D的所有截集中，称截量最小的为最小截集. 直观地说，所有弧是 V_s 到 V_t 的必经之路. 如果从D中丢去这一截集，则发点就不能通往收点.

Part 3

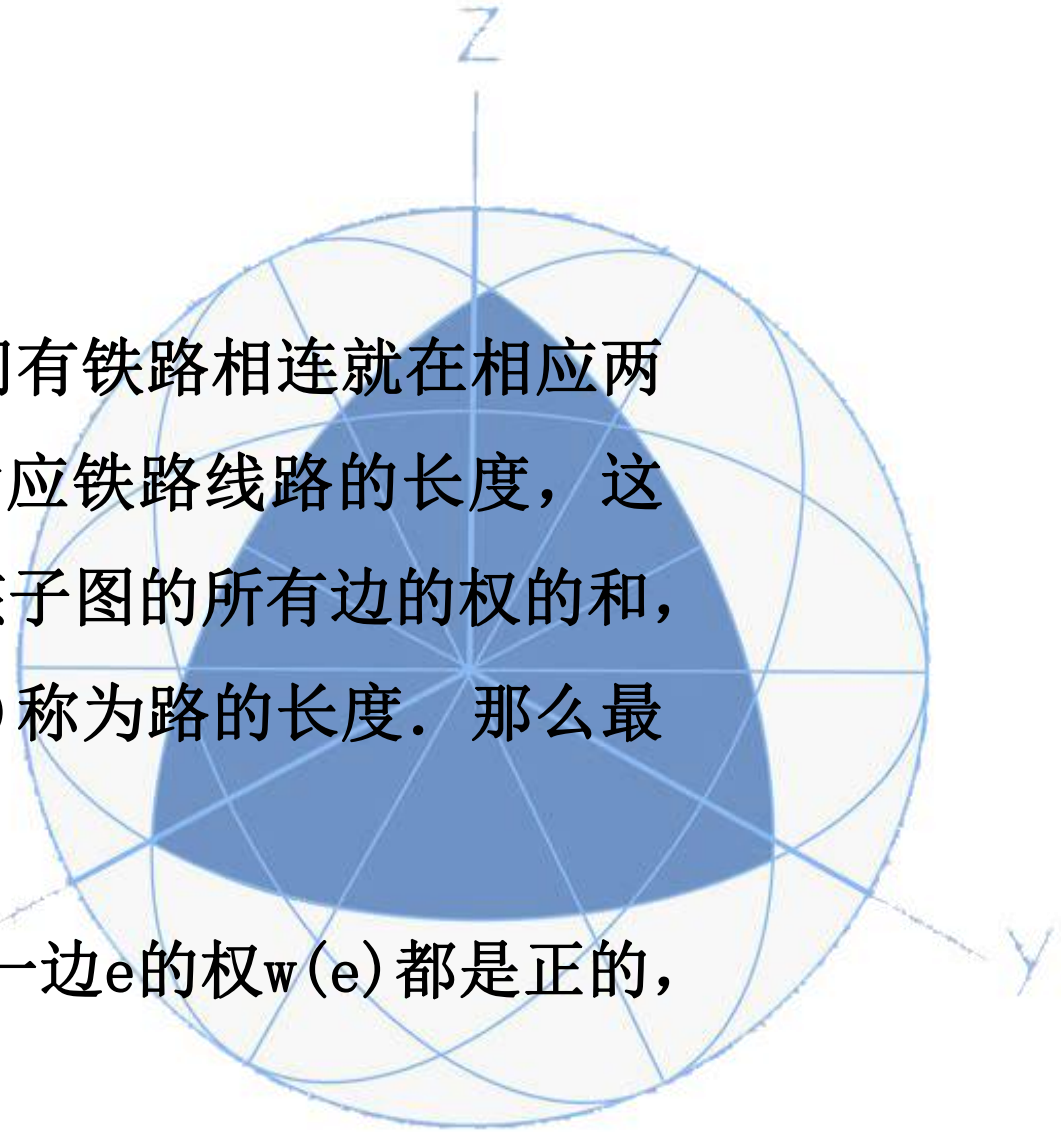
案例分析

12.3 案例分析



案例一、 最短路问题

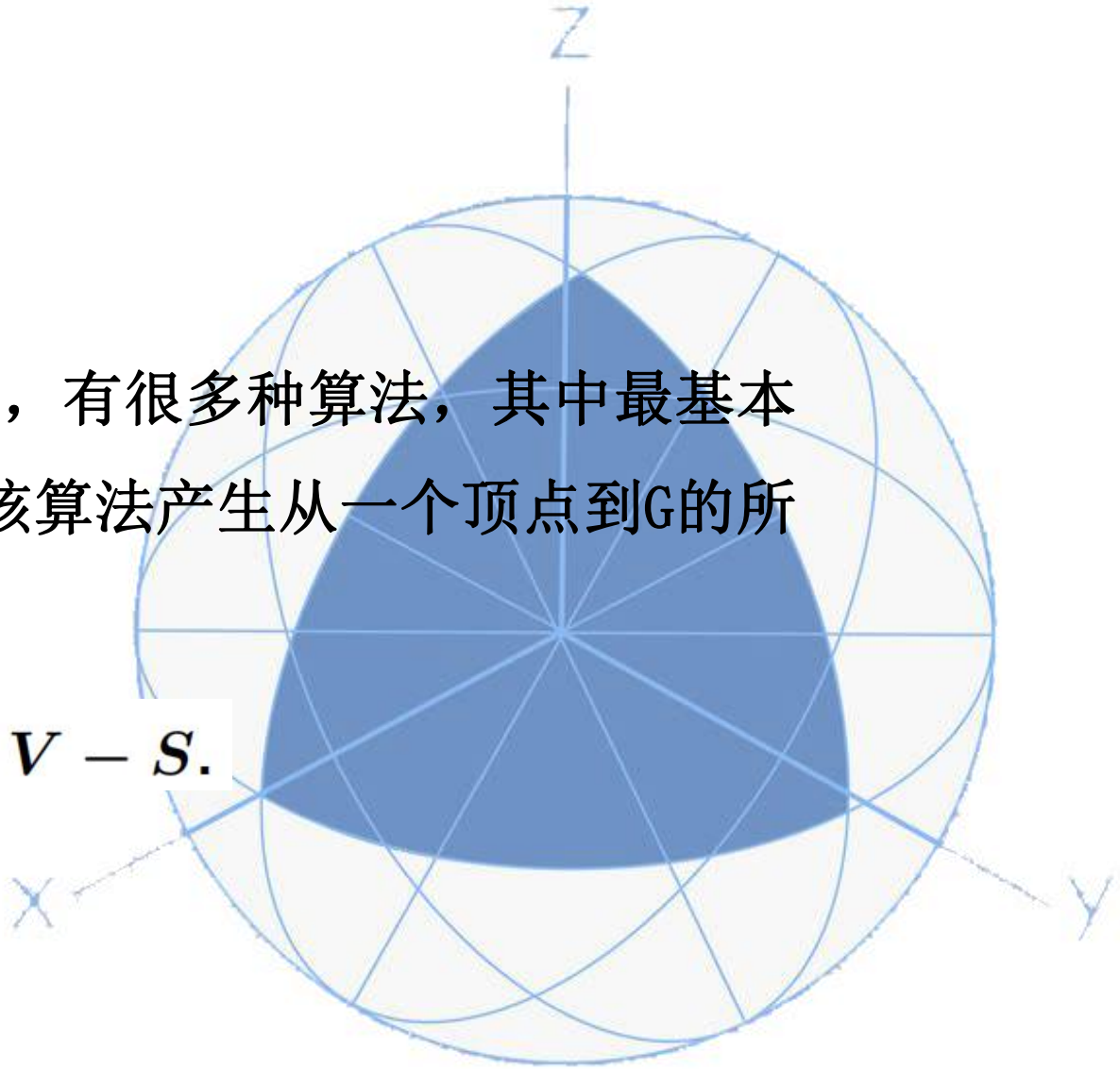
问题背景：在某个区域，有连接若干城市的铁路网，欲求任意两个城市之间的最短的铁路线路？



【问题分析】

将该区域的城市作为顶点，若两城市间有铁路相连就在相应两顶点之间连一条边，在每条边上标注上对应铁路线路的长度，这样就得到一赋权图. 其子图的权定义为该子图的所有边的权的和，特殊地，路的权（即路上所有边的权的和)称为路的长度．那么最短路问题即转化为：

给定一简单连通赋权图G，且它的任意一边e的权w(e)都是正的，求G中任意两点间最短路及其长度.



【模型构建】

最短路问题一般采取最短路算法，有很多种算法，其中最基本的一种是1959年Dijkstra提出的．该算法产生从一个顶点到G的所有其它顶点的最短路.

设 $S \subset V$ 且 $u_0 \in S$. 令 $\bar{S} = V - S$.

若 $P = u_0 \cdots \bar{u}\bar{v}$ 是 u_0到$\bar{S}$ 的最短路，即 u_0与$\bar{S}$ 的各顶点的最短路中最短的一条，则必有：

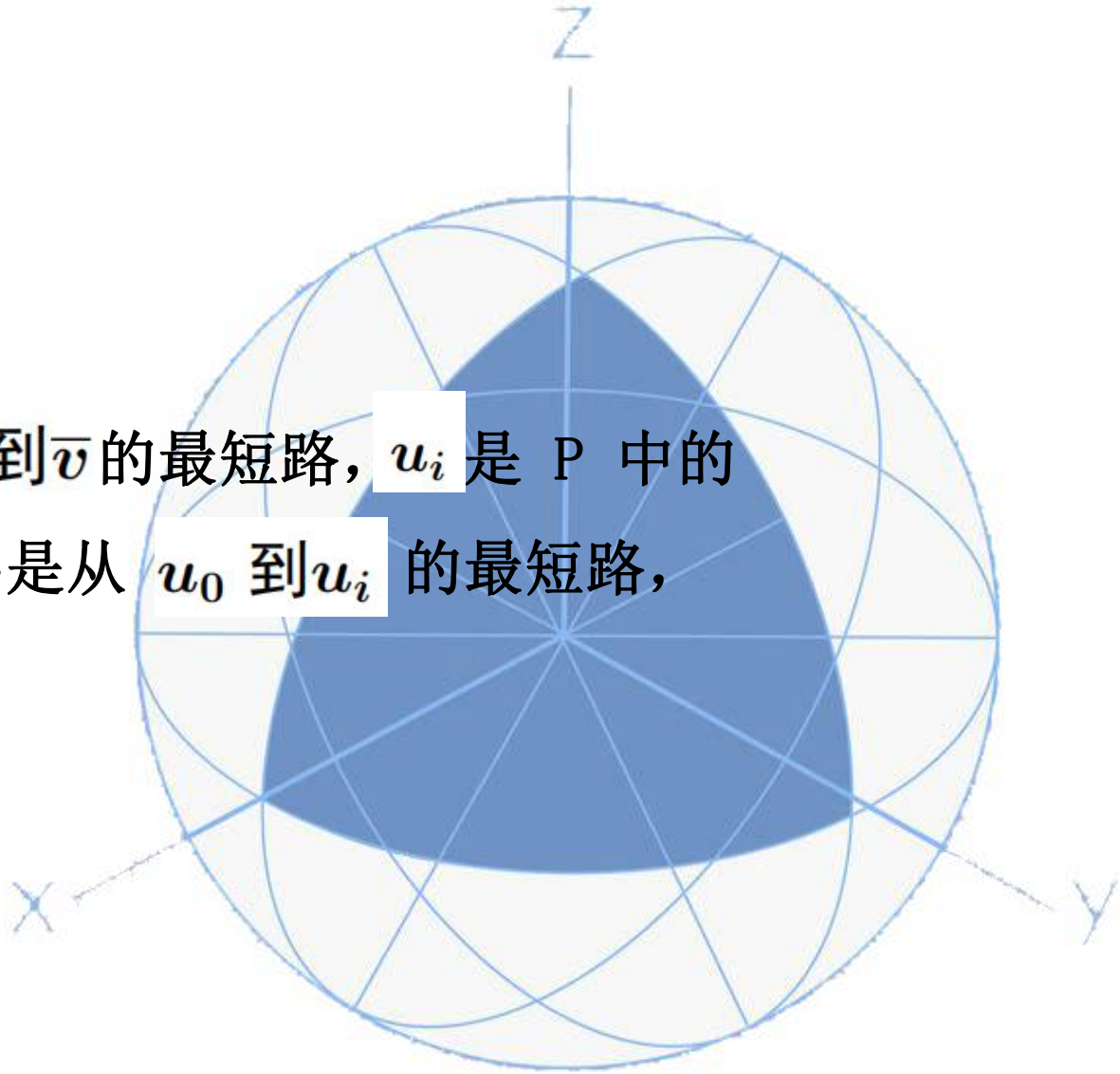
(a) P必定是 u_0到$\bar{v}$ 的最短路；

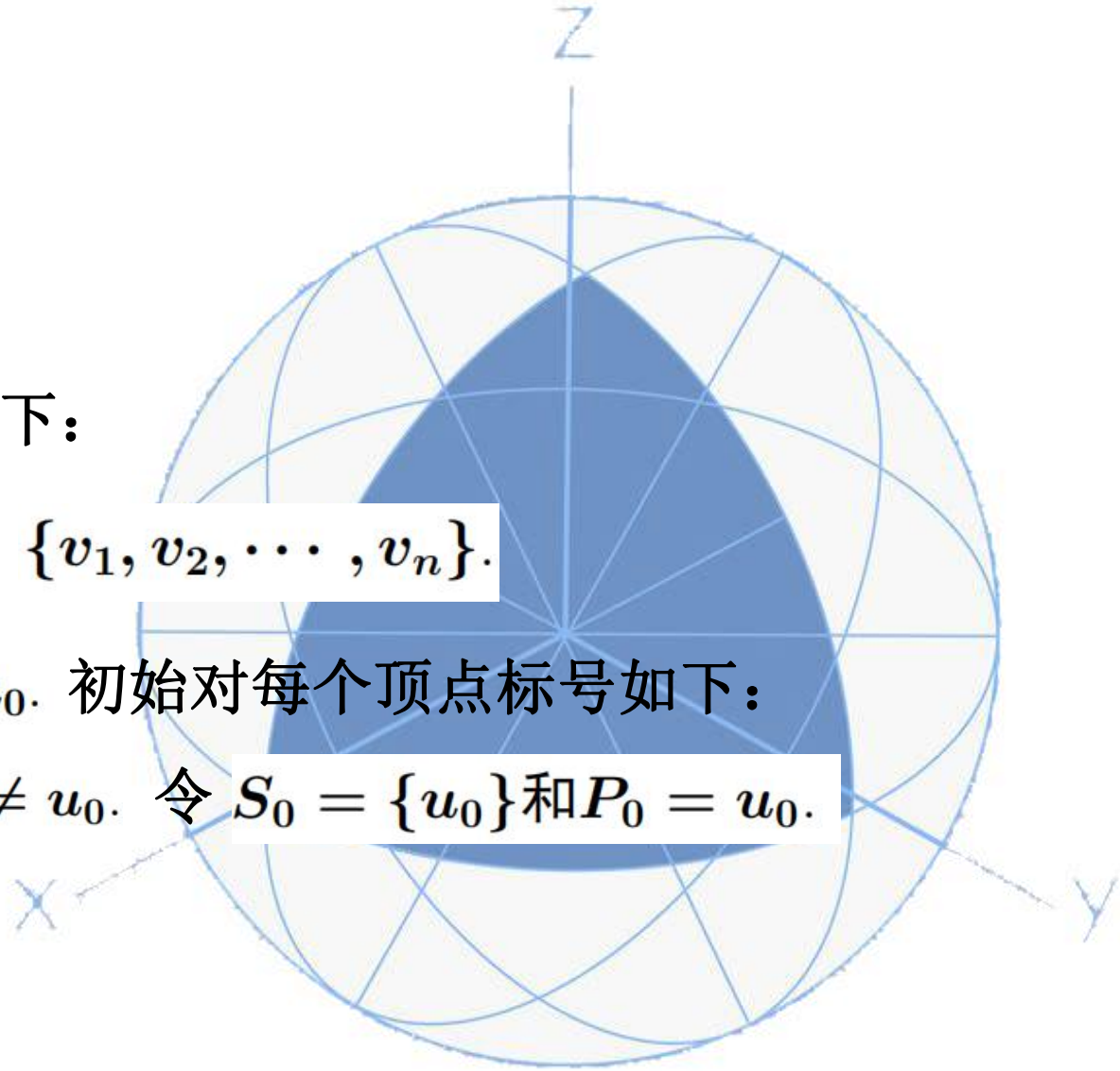
(b) $\bar{u} \in S$ 且P上的 $(u_0, \bar{u})$ -路是 u_0和$\bar{u}$ 间的最短路.

由(a)和(b)得，$d(u_0, \bar{v}) = d(u_0, \bar{u}) + w(\bar{u}, \bar{v})$. 因此有

$$d(u_0, \bar{S}) = \min_{\substack{u \in S \\ v \in \bar{S}}} \{d(u_0, u) + w(u, v)\}.$$

容易理解，如果 P 是 D 中从u_0到$\overline{v}$的最短路，u_i是 P 中的一个中间点，则u_0沿 P 到u_i的路必是从u_0到u_i的最短路，这就是求最短路方法的理论依据.





【模型求解】

据此，我们设计Dijkstra算法如下：

输入：连通赋权图G，其中V(G)= $\{v_1, v_2, \cdots, v_n\}$.

Step 1 任取G的某个顶点作为 u_0. 初始对每个顶点标号如下：

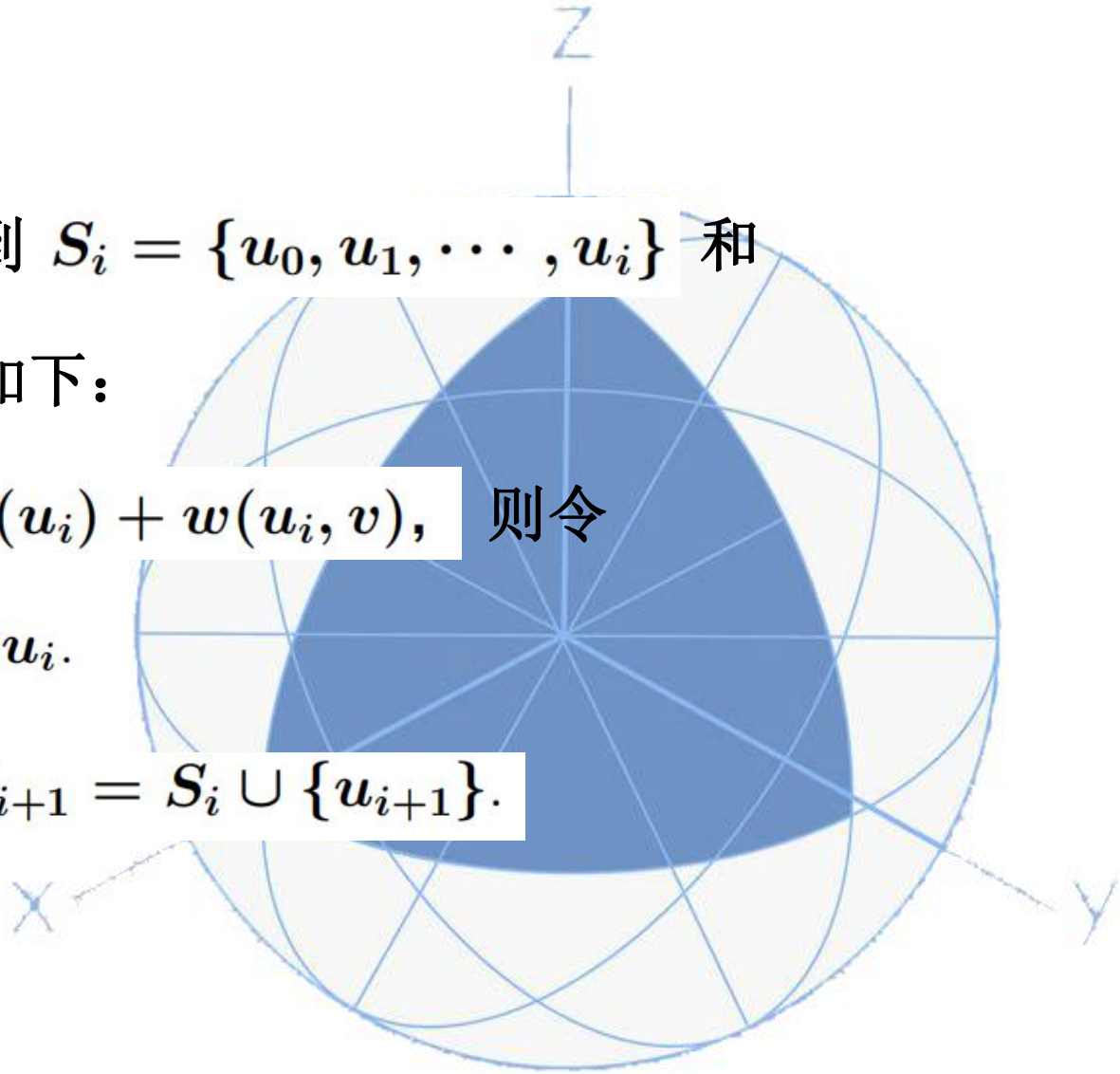
令 $l(u_0) = 0$, $l(v) = \infty$ 对所有 $v \neq u_0$. 令 $S_0 = \{u_0\}$和$P_0 = u_0$.

Step 2　一般地，假设已经得到 $S_i = \{u_0, u_1, \cdots, u_i\}$ 和 $P_0, P_1, \cdots, P_i$，现求 S_{i+1}和P_{i+1} 如下：

(a) 对每个 $v \in \bar{S}_i$，若 $l(v) > l(u_i) + w(u_i, v)$，则令 $l(v) = l(u_i) + w(u_i, v)$ 和 $b(v) = u_i$.

(b) 取 $u_{i+1} = \min_{v \in \bar{S}_i}\{l(v)\}$，令 $S_{i+1} = S_i \cup \{u_{i+1}\}$.

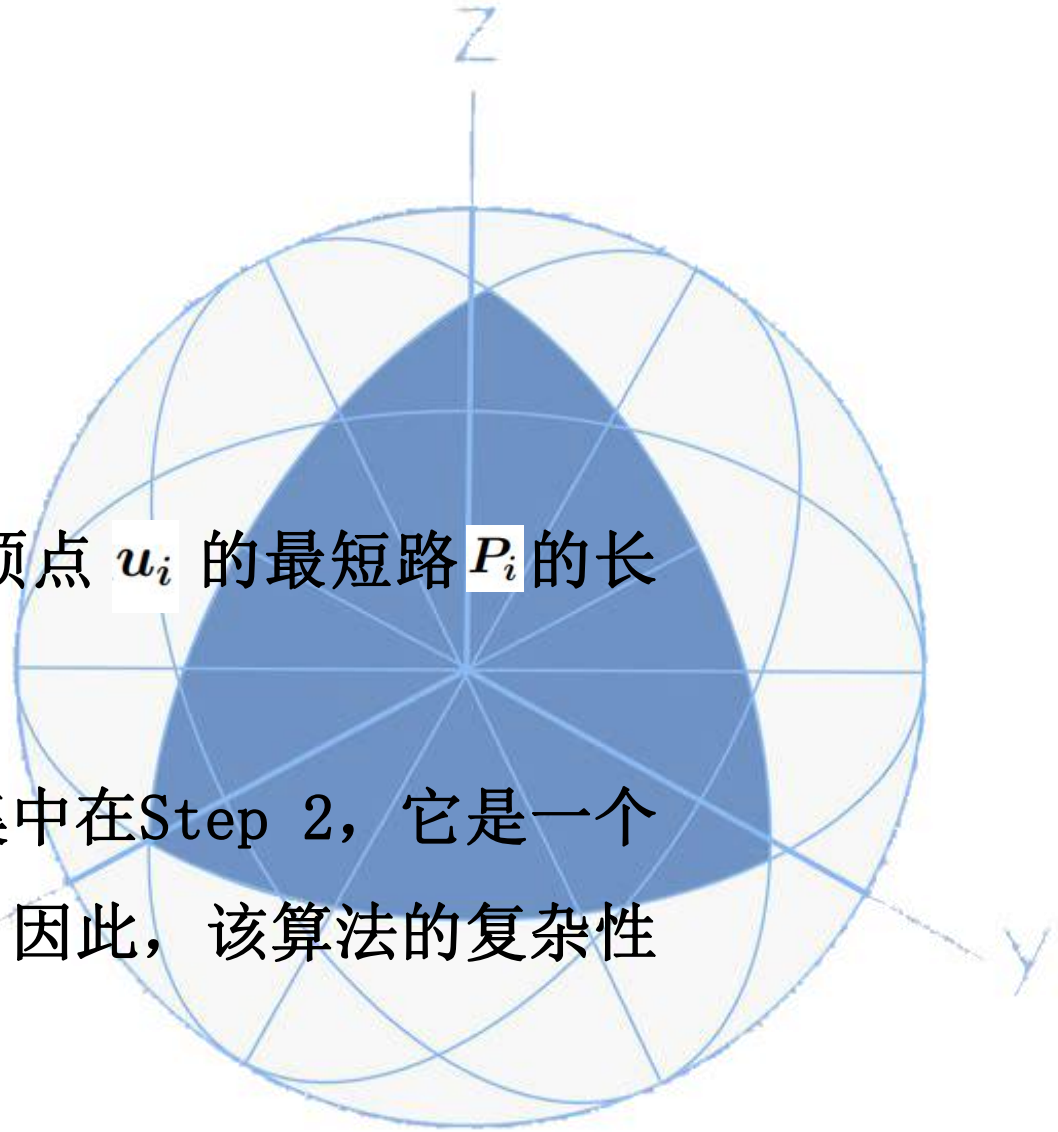
(c) 令 $P_{i+1} = P_{b(u_{i+1})}u_{i+1}$.



Step 3　当Step 2不能执行时停止.

输出：$P_1, P_2, \cdots, P_{n-1}$.

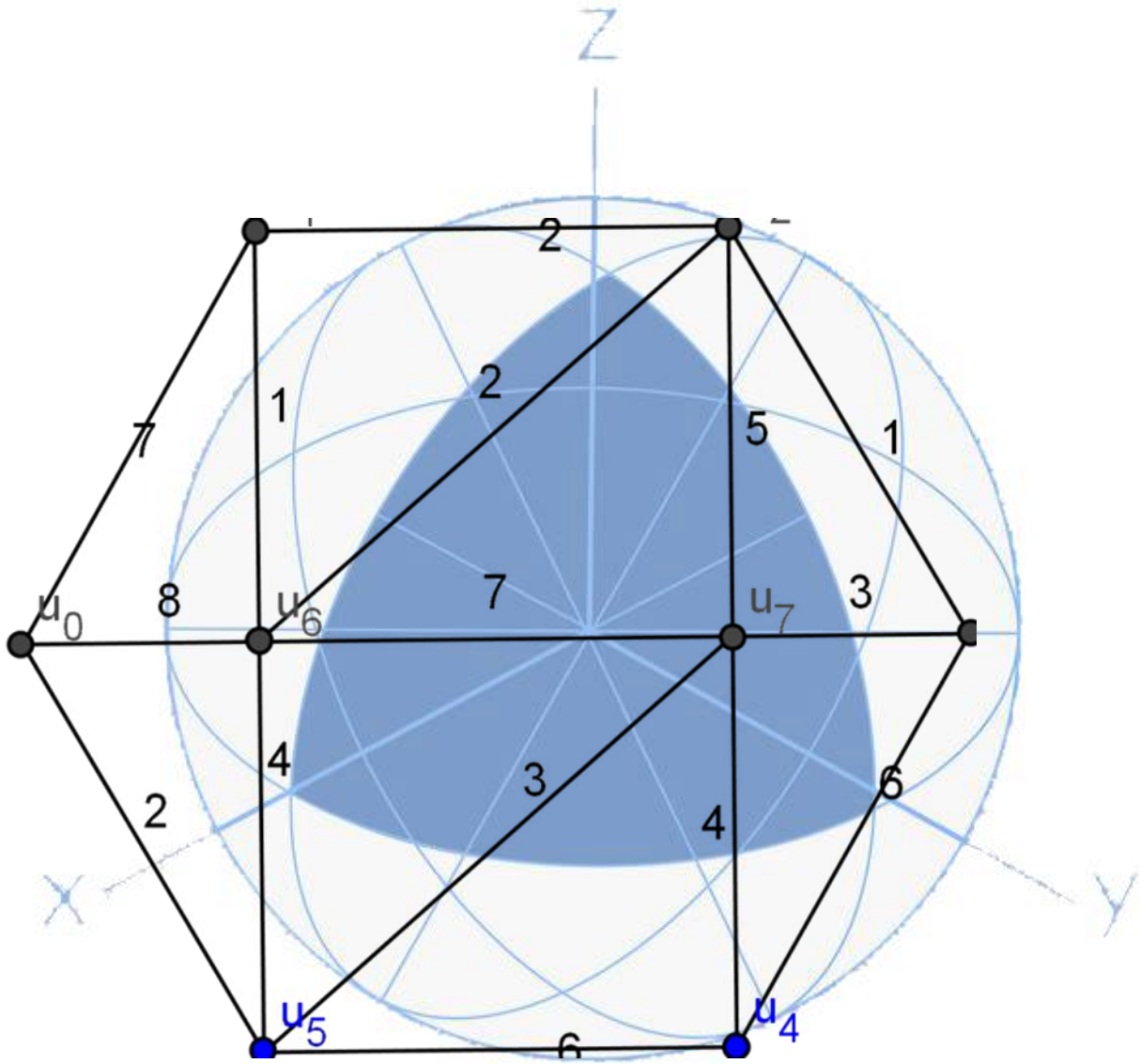
注意算法结束后 $l(u_i)$ 的值即为 u_0 到顶点 u_i 的最短路 P_i 的长度，$i = 1, 2, \cdots, n-1$.



最短路Dijkstra算法的计算步骤主要集中在Step 2，它是一个循环过程.该过程里面的(a)也是一个循环.因此，该算法的复杂性为 $O(n^2)$,是个多项式算法.

例12.5　求右边赋权图中

从 u_0 到其它各点的最短路.



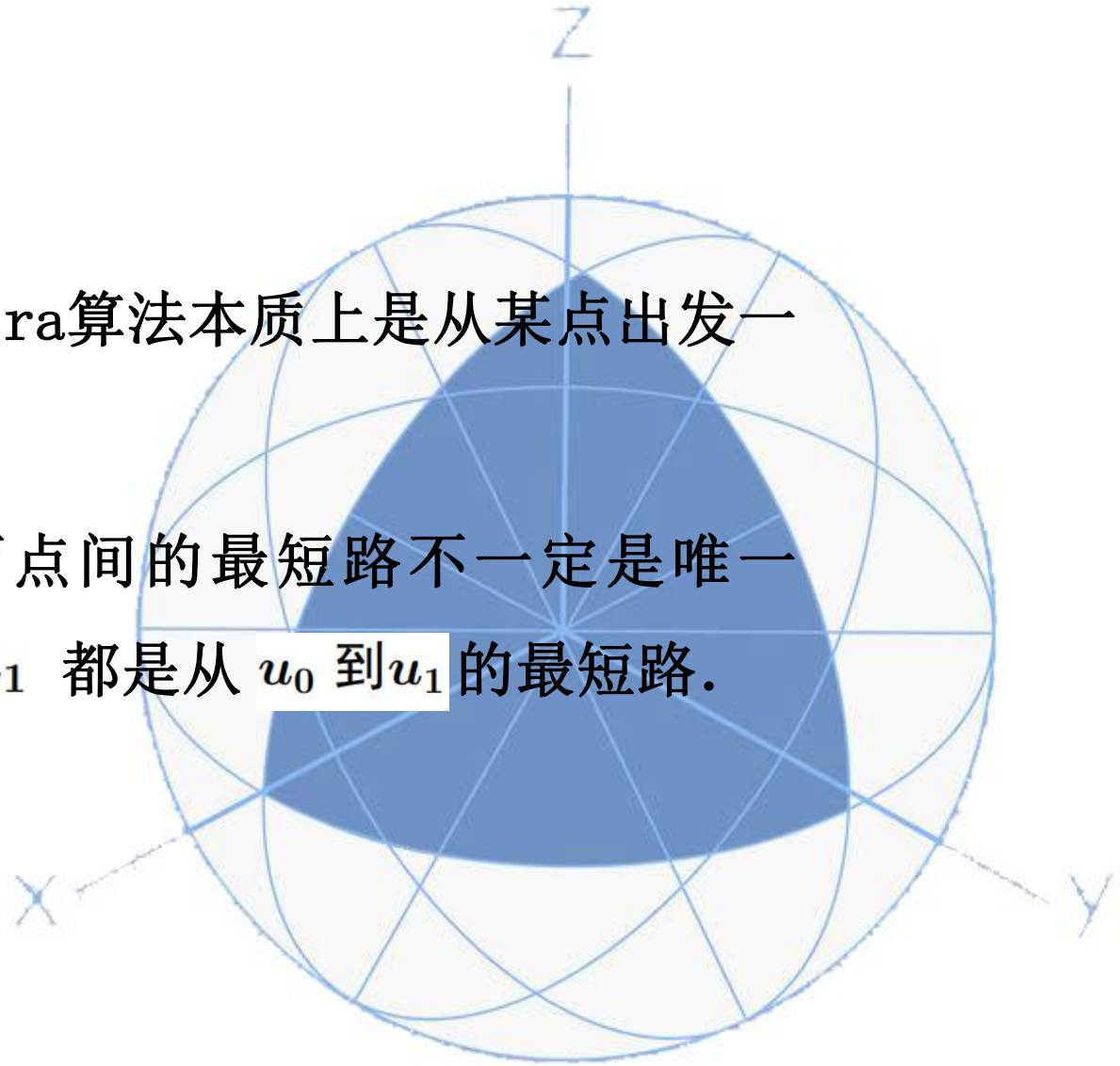
解：用表格形式表示Dijkstra算法的求解过程如下：

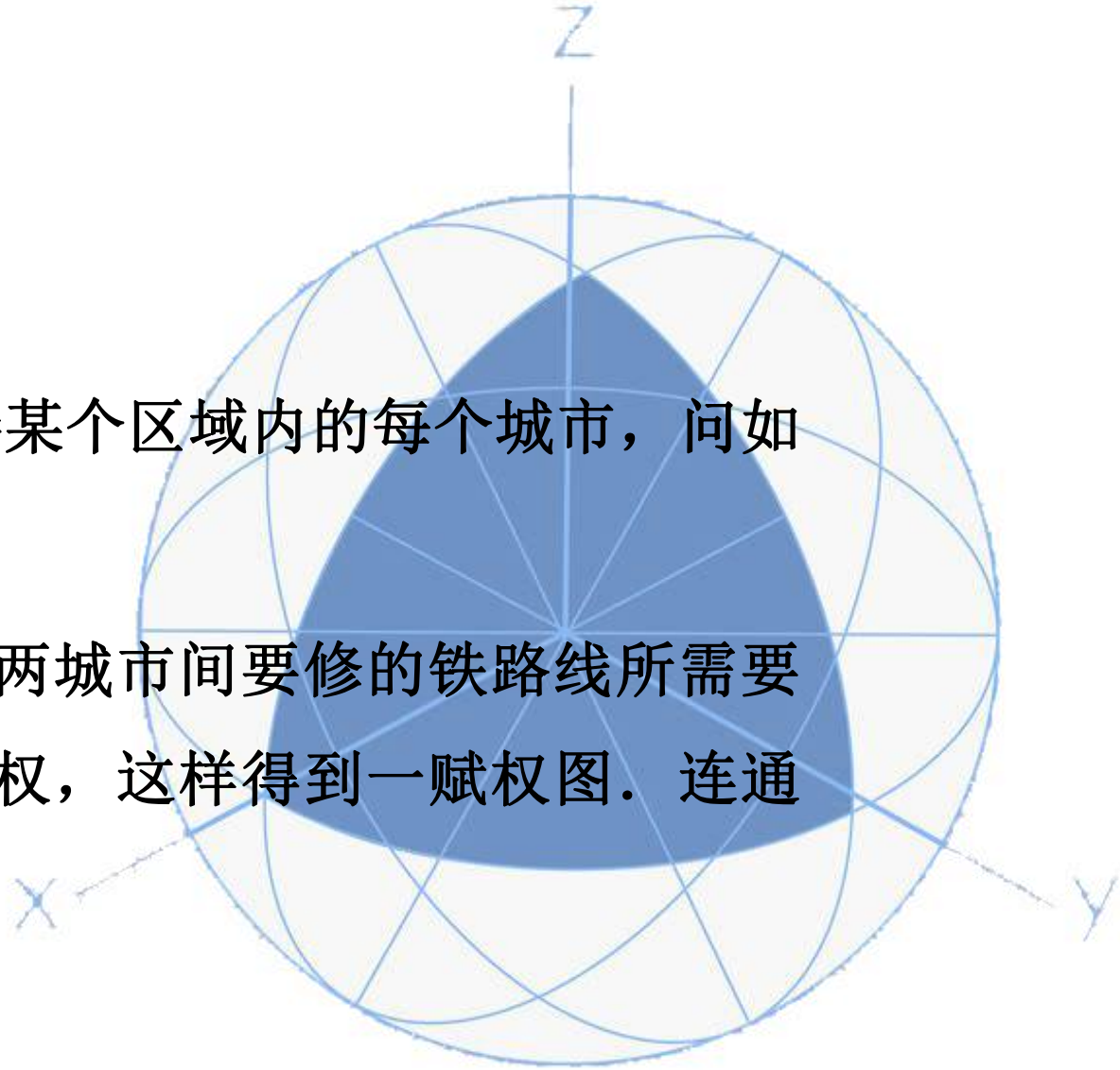
m	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7
0	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	0	$(0,1)=7$	$+\infty$	$+\infty$	$+\infty$	$(0,5)=2$	$(0,6)=8$	$+\infty$
2	0	$(6,1)=7$	$(1,2)=9$	$+\infty$	$(5,4)=8$	$(0,5)=2$	$(5,6)=6$	$(5,7)=5$
3	0	$(6,1)=7$	$(6,2)=8$	$(7,3)=8$	$(5,4)=8$	$(0,5)=2$	$(5,6)=6$	$(5,7)=5$
4	0	$(6,1)=7$	$(6,2)=8$	$(7,3)=8$	$(5,4)=8$	$(0,5)=2$	$(5,6)=6$	$(5,7)=5$

至m=4，已求出u0到其他点的最短路.

从该例的过程可以看出，Dijkstra算法本质上是从某点出发一步步向外长成一棵生成树的过程.

需要注意的是，一般来说，两点间的最短路不一定是唯一的. 例如上例中 u_0u_1 和 $u_0u_5u_6u_1$ 都是从 u_0 到 u_1 的最短路.

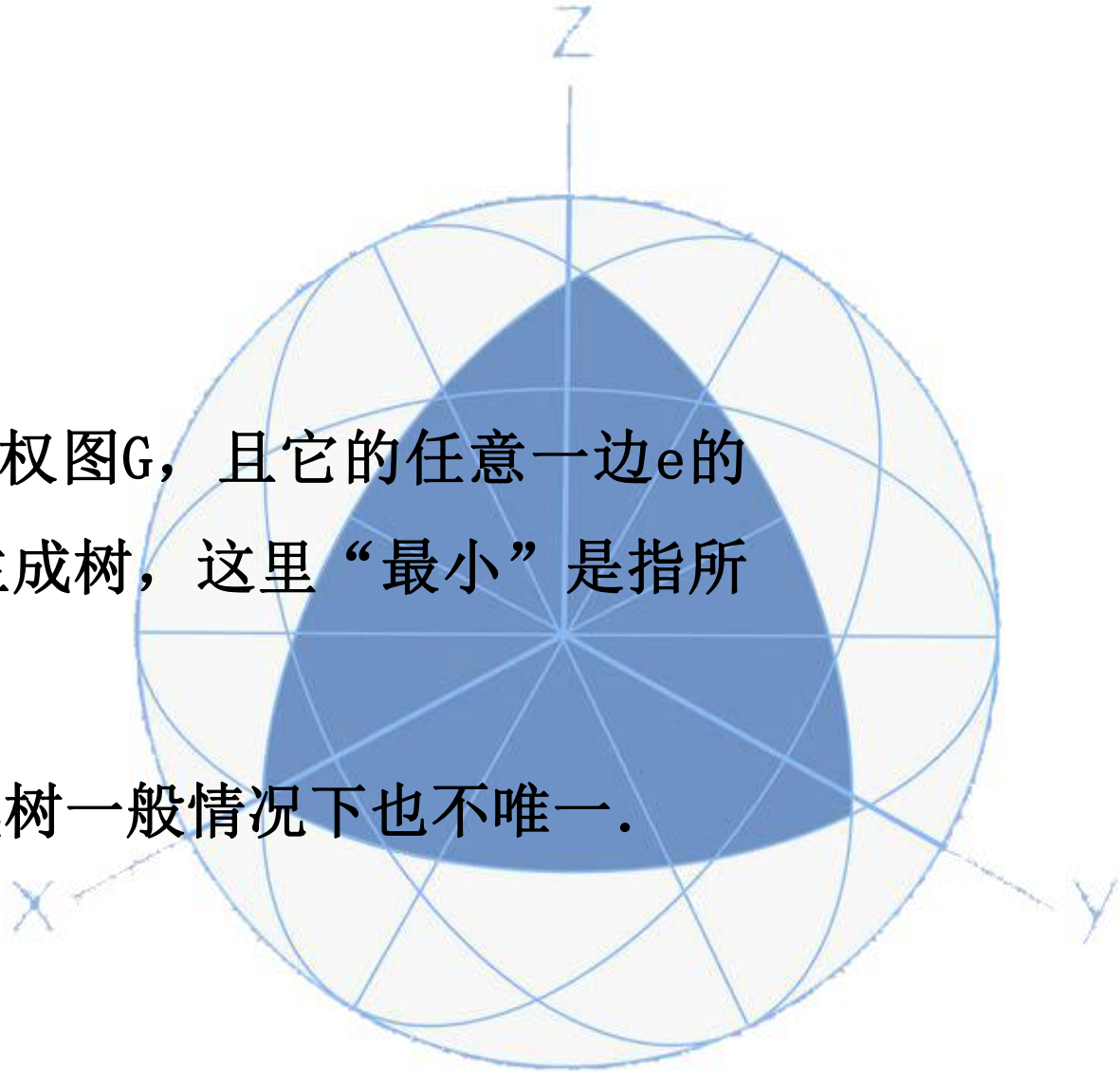




案例二、 铁路网问题

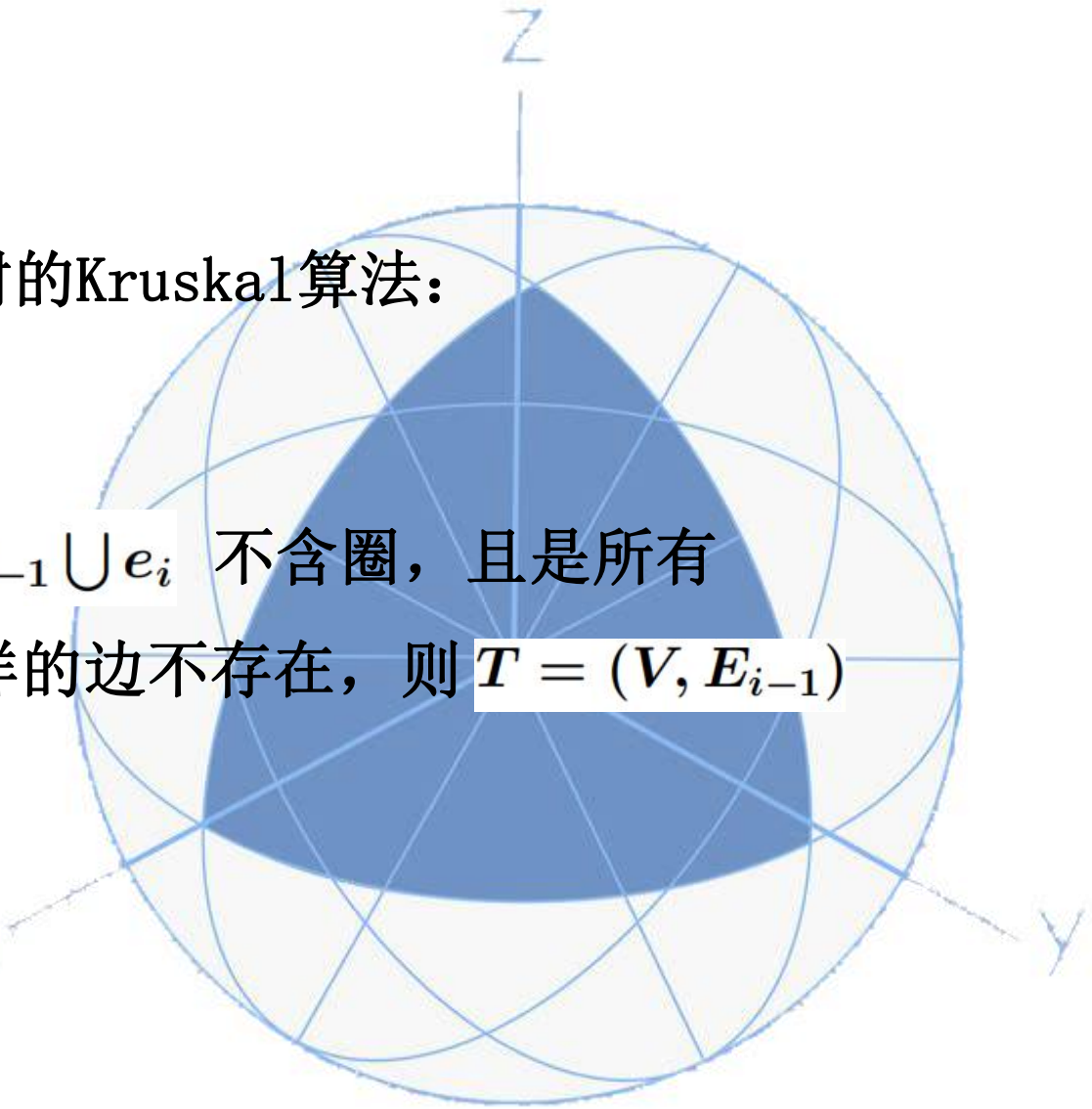
问题背景：要建造一铁路网连接某个区域内的每个城市，问如何设计铁路网使得修建费用最低？

问题分析： 以城市作为顶点，两城市间要修的铁路线所需要的费用作为连接这两个城市的边的权，这样得到一赋权图．连通问题即转化为下面的图论问题.



模型构建： 给定一个连通简单赋权图G，且它的任意一边e的权w(e) 都是正的，求它的一棵最小生成树，这里“最小”是指所有生成树中权最小的一棵.

当然和最短路问题一样，最小生成树一般情况下也不唯一.



【模型求解】求赋权图的最小生成树的Kruskal算法：

第一步：令 $i=1, E_0=\emptyset$;

第二步：选边 $e_i \in E/E_{i-1}$，使 $E_{i-1} \bigcup e_i$ 不含圈，且是所有未被选取的边中权最小的一条. 如果这样的边不存在，则 $T=(V, E_{i-1})$ 就是最小树.

第三步：置 $i=i+1$， 转入第二步.

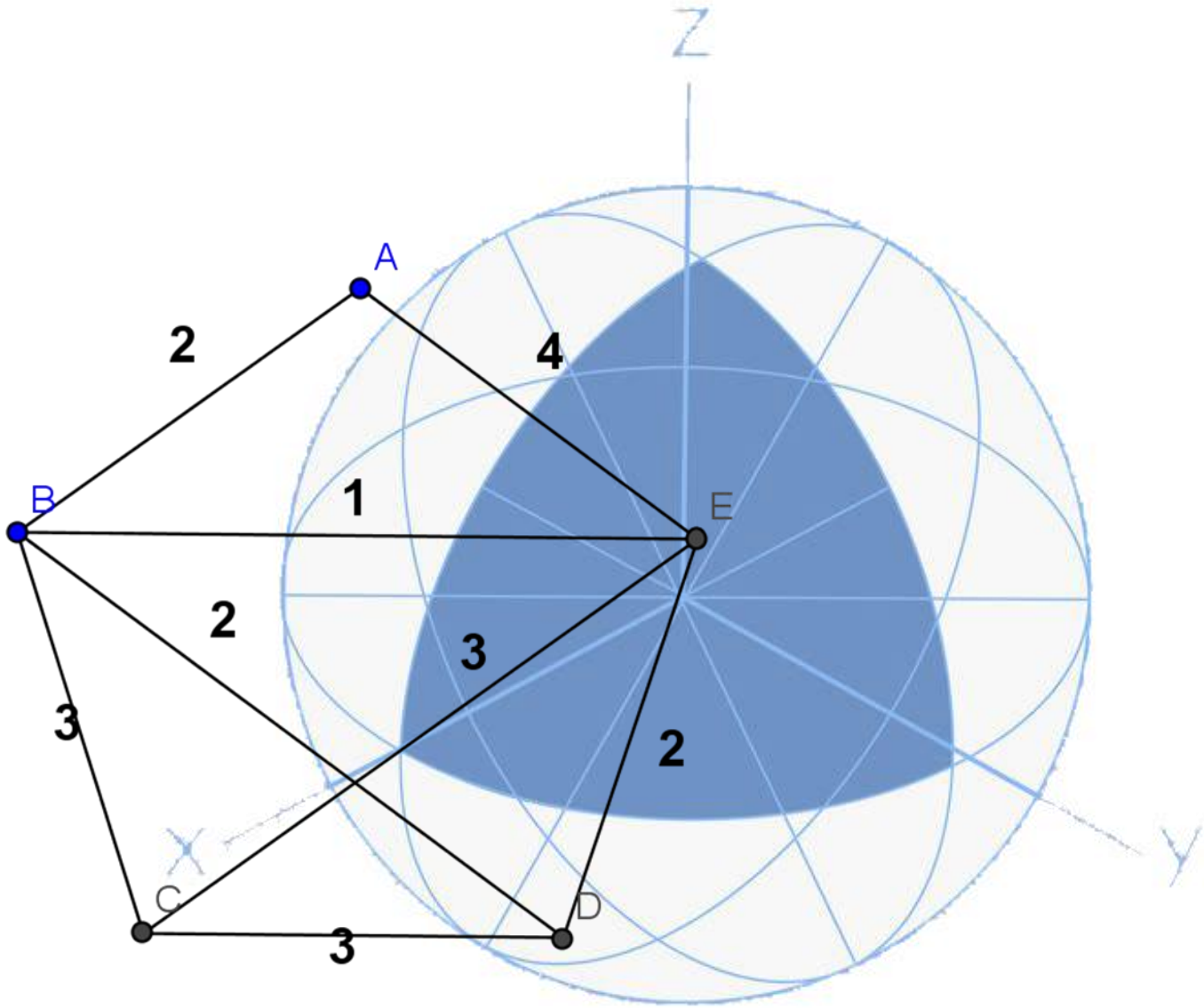
【模型应用】

例12.7:

求右图所示的

赋权图的

最小生成树.

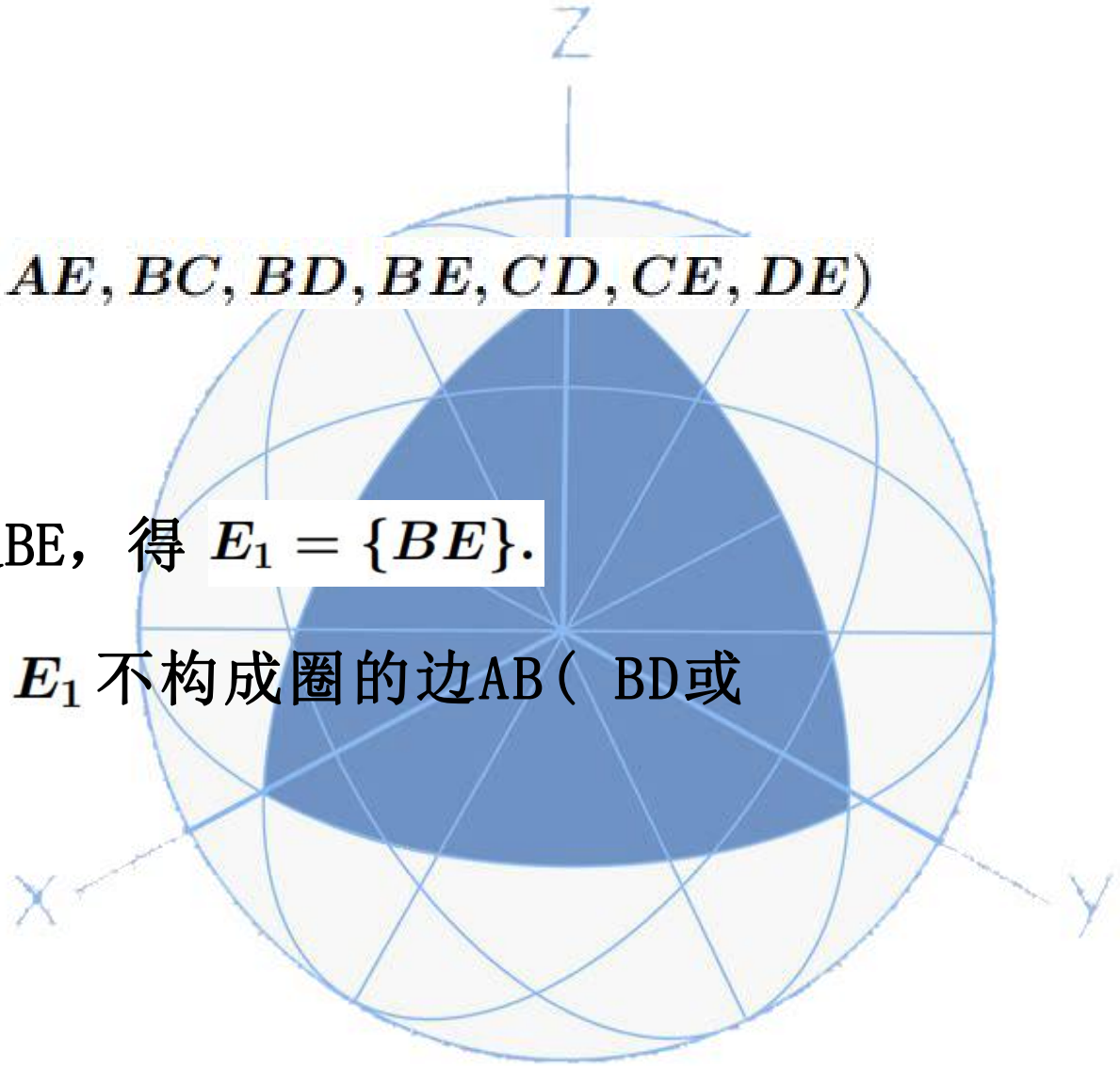


解：根据图像可知，$E(G)=(AB,AE,BC,BD,BE,CD,CE,DE)$

利用Kruskal算法求解过程如下：

$i=1,E_0=\emptyset$，从E中选权最小的边BE，得 $E_1=\{BE\}$.

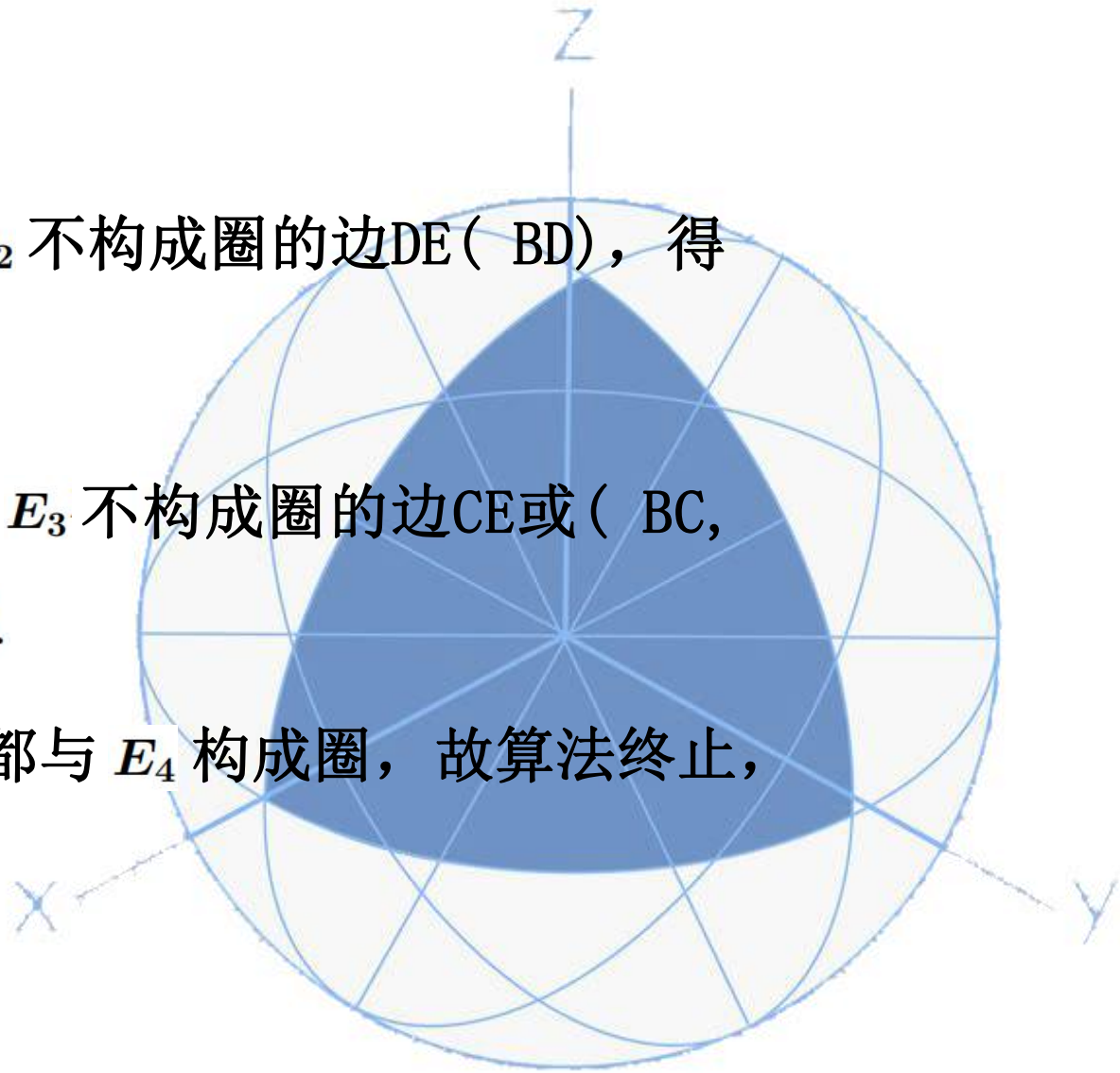
$i=2$，从 E/E_1 中选最小权且与 E_1 不构成圈的边AB（ BD或DE），得 $E_2=\{BE,AB\}$.



$i=3$，从 E/E_2 中选最小权且与 E_2 不构成圈的边DE(BD)，得 $E_3=\{BE,AB,DE\}$.

$i=4$，从 E/E_3 中选最小权且与 E_3 不构成圈的边CE或(BC, CD)，得 $E_4=\{BE,AB,DE,CE\}$.

$i=5$，由于在 E/E_4 中的任一边都与 E_4 构成圈，故算法终止，

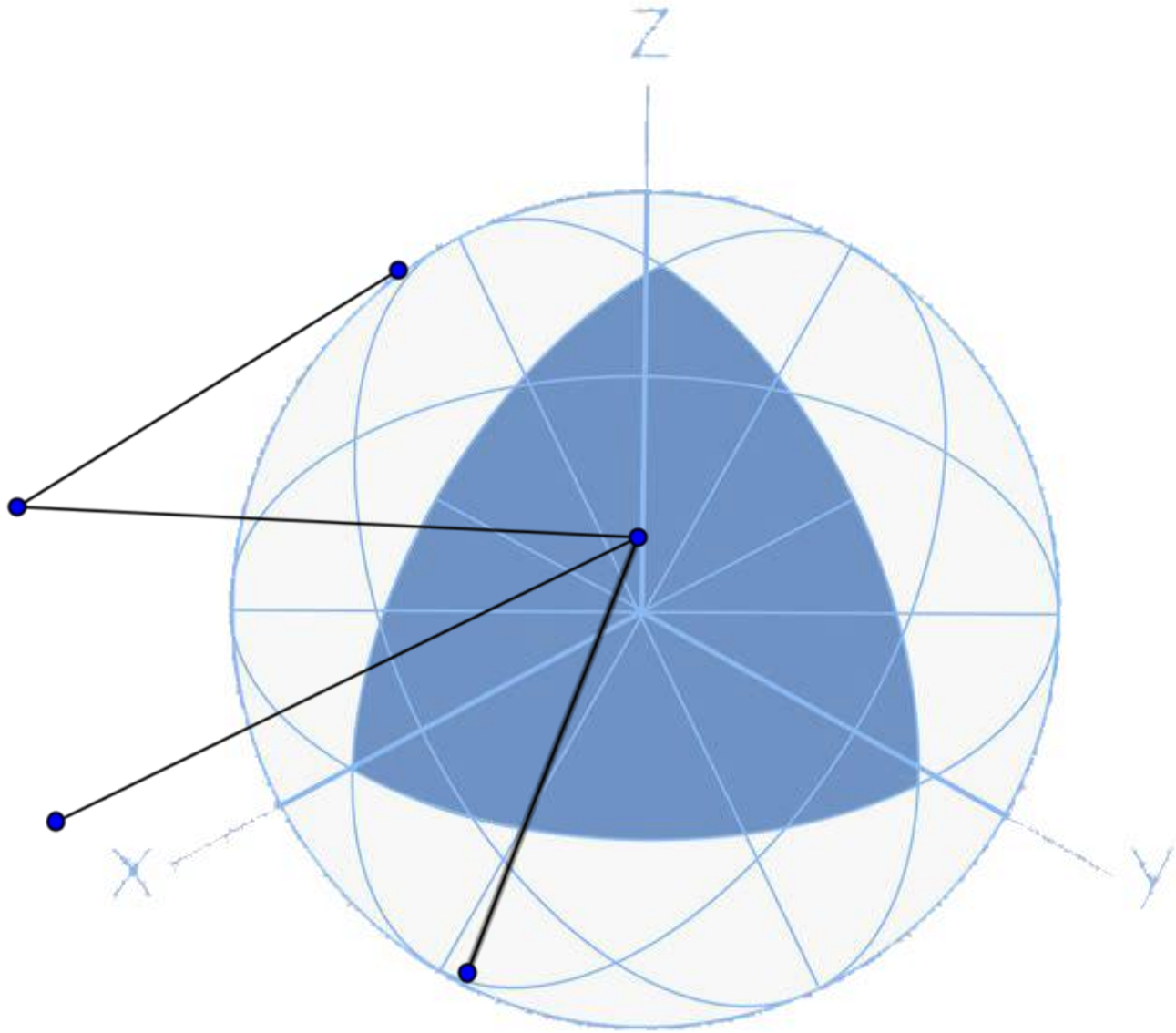


$T = (V, E_4)$ 就是

要求的最小支撑树，

右图给出了

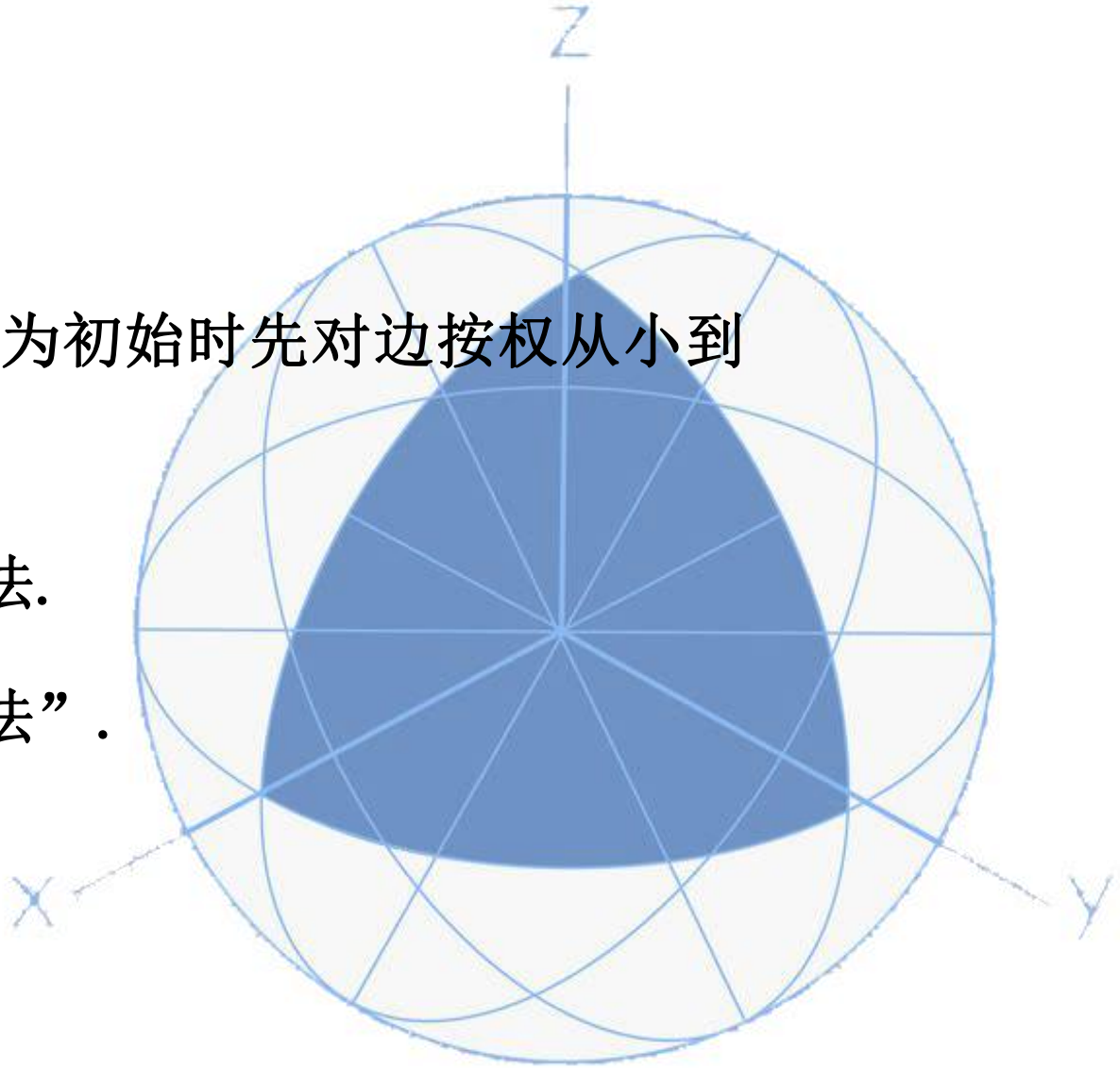
一棵最小支撑树.



上述的Kruskal算法可以稍加改进为初始时先对边按权从小到大排序，从而简化Step 2(ii).

因此Kruskal算法也是个多项式算法.

上述的Kruskal算法也称为“避圈法”.



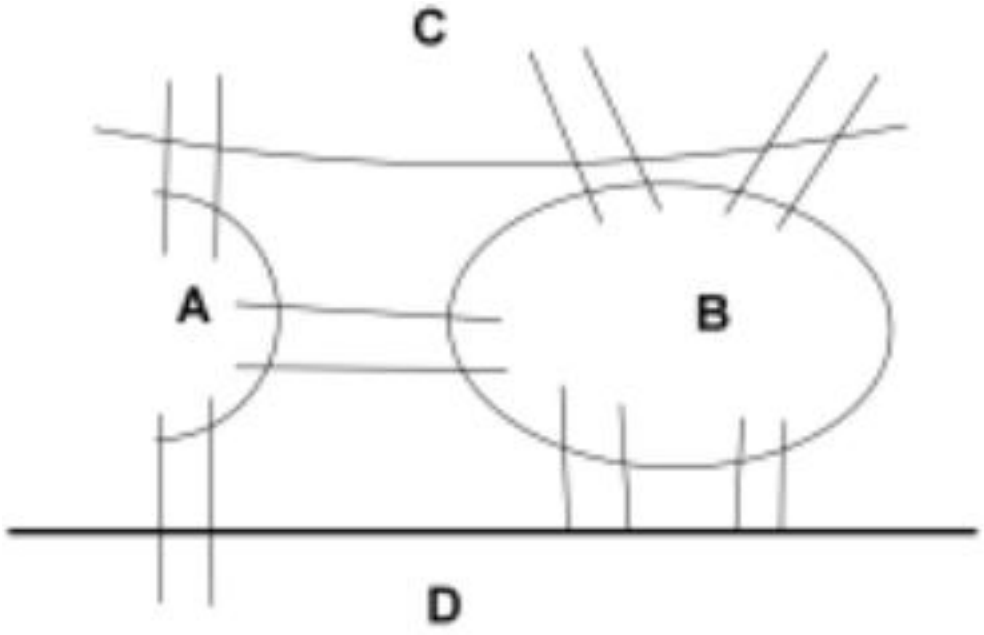
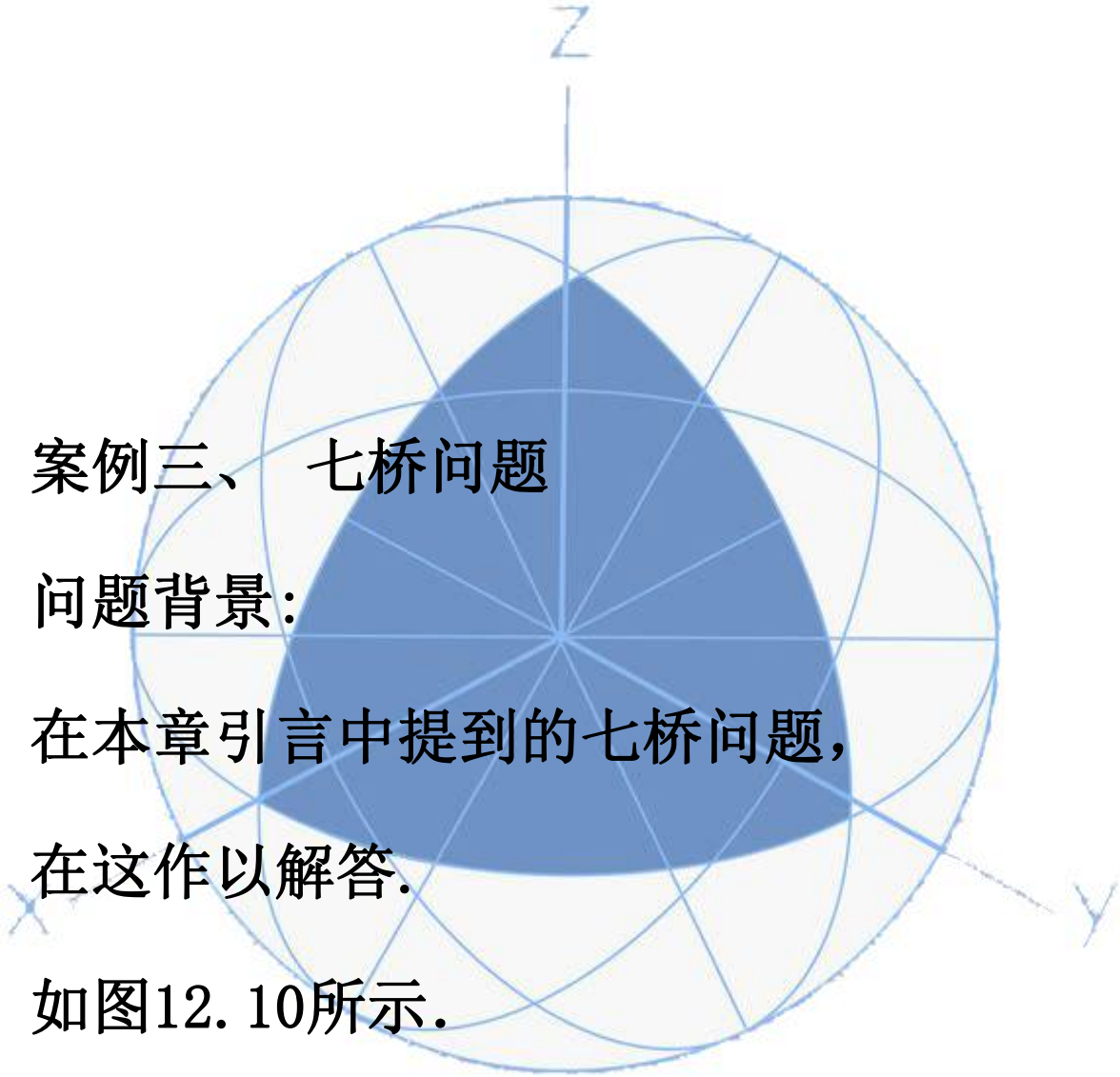


图 12.10　Königsberg 七桥图示



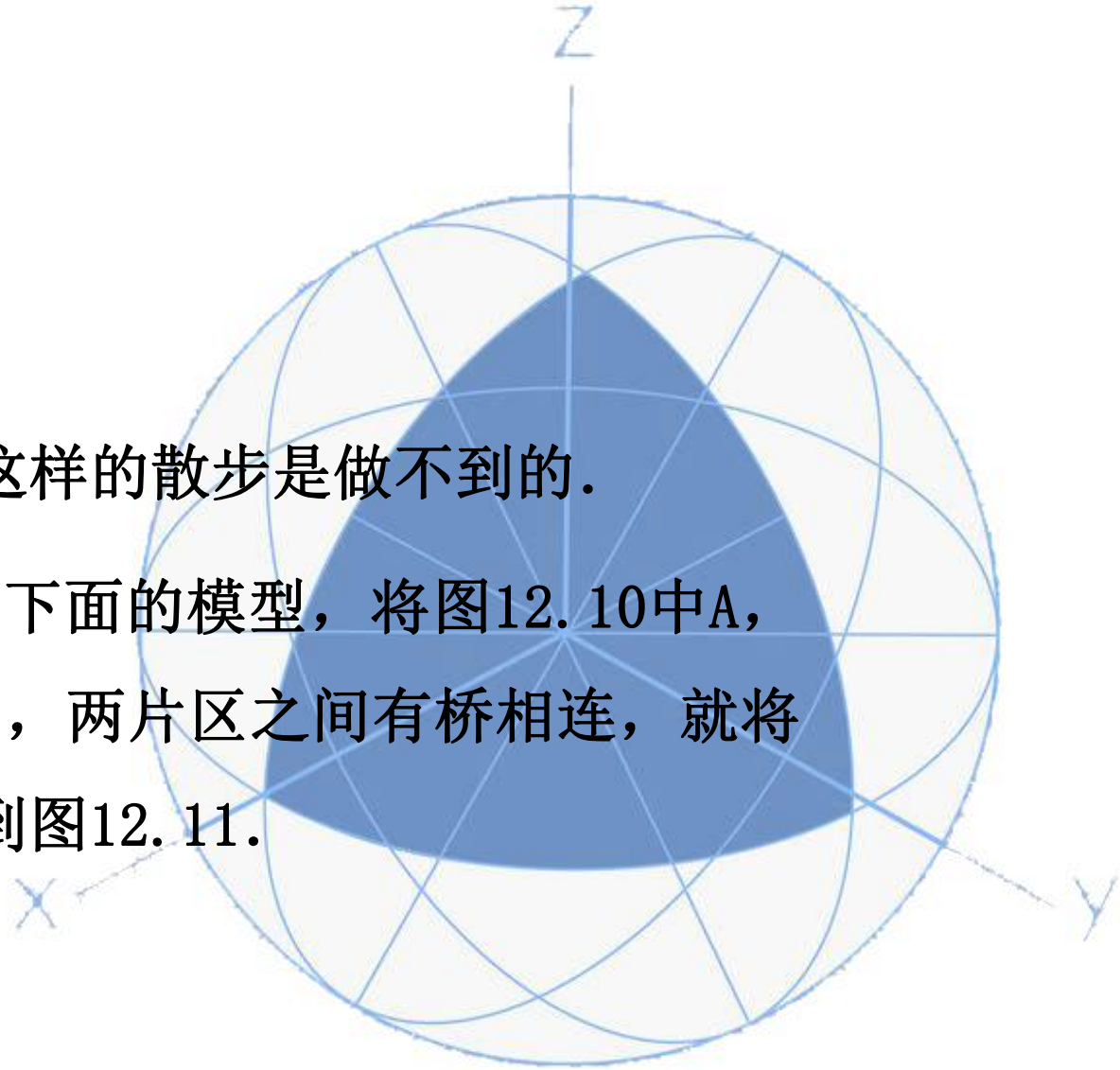
案例三、 七桥问题

问题背景:

在本章引言中提到的七桥问题，

在这作以解答.

如图12.10所示.



【问题分析】

大数学家Euler在1736年证明了这样的散步是做不到的.

他的思路是将这个问题抽象成为下面的模型，将图12.10中A，B，C和D这四个片区各看成一个顶点，两片区之间有桥相连，就将相应的两个顶点连一条边，这样得到图12.11.

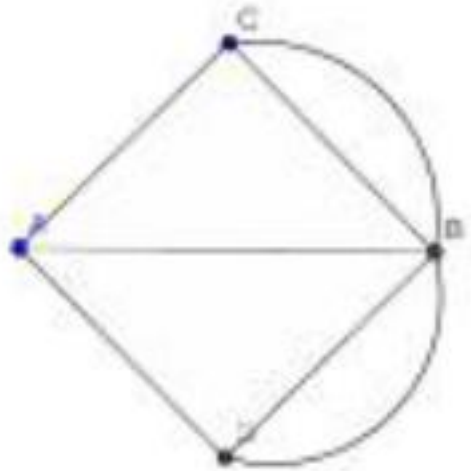
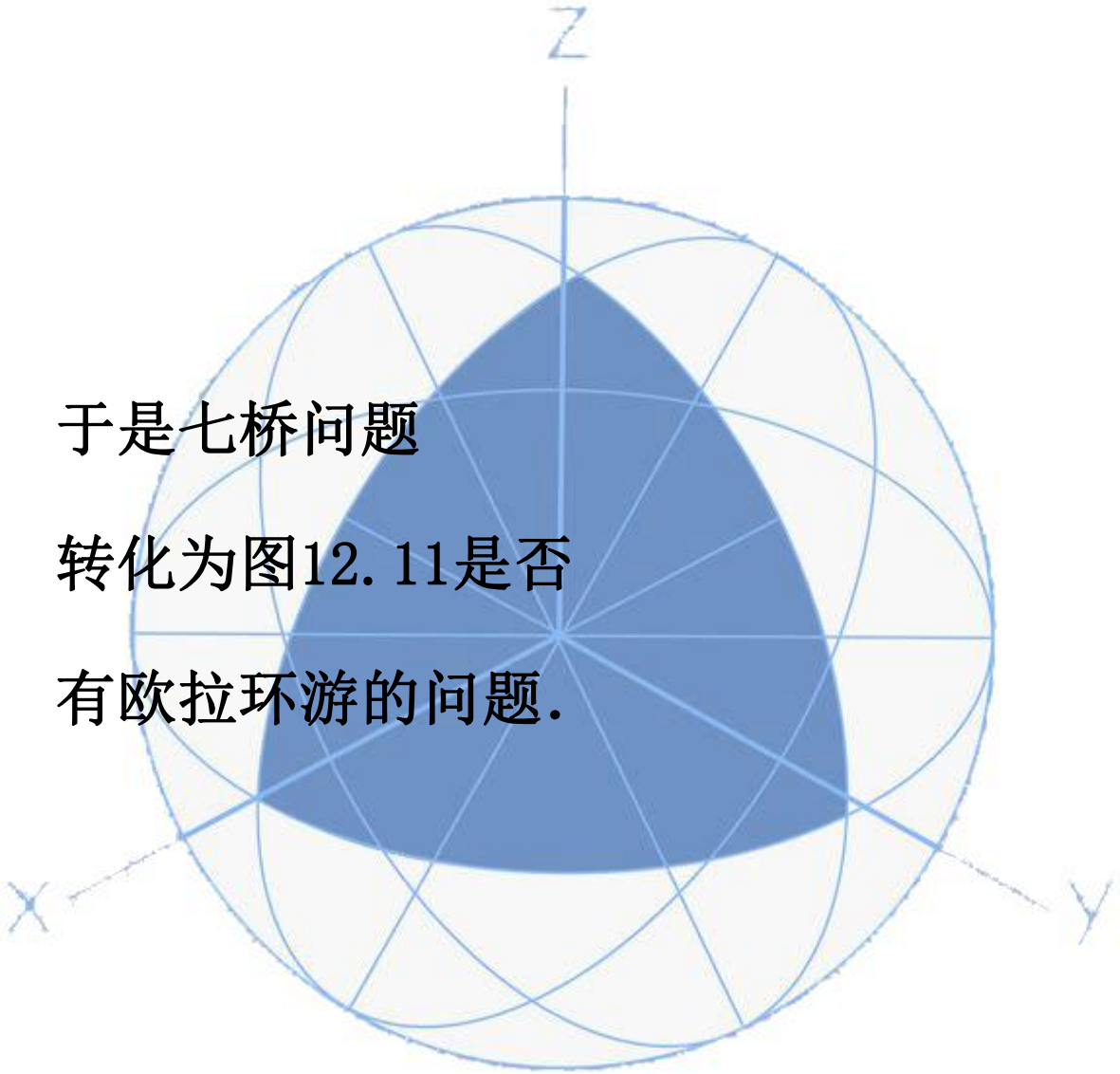


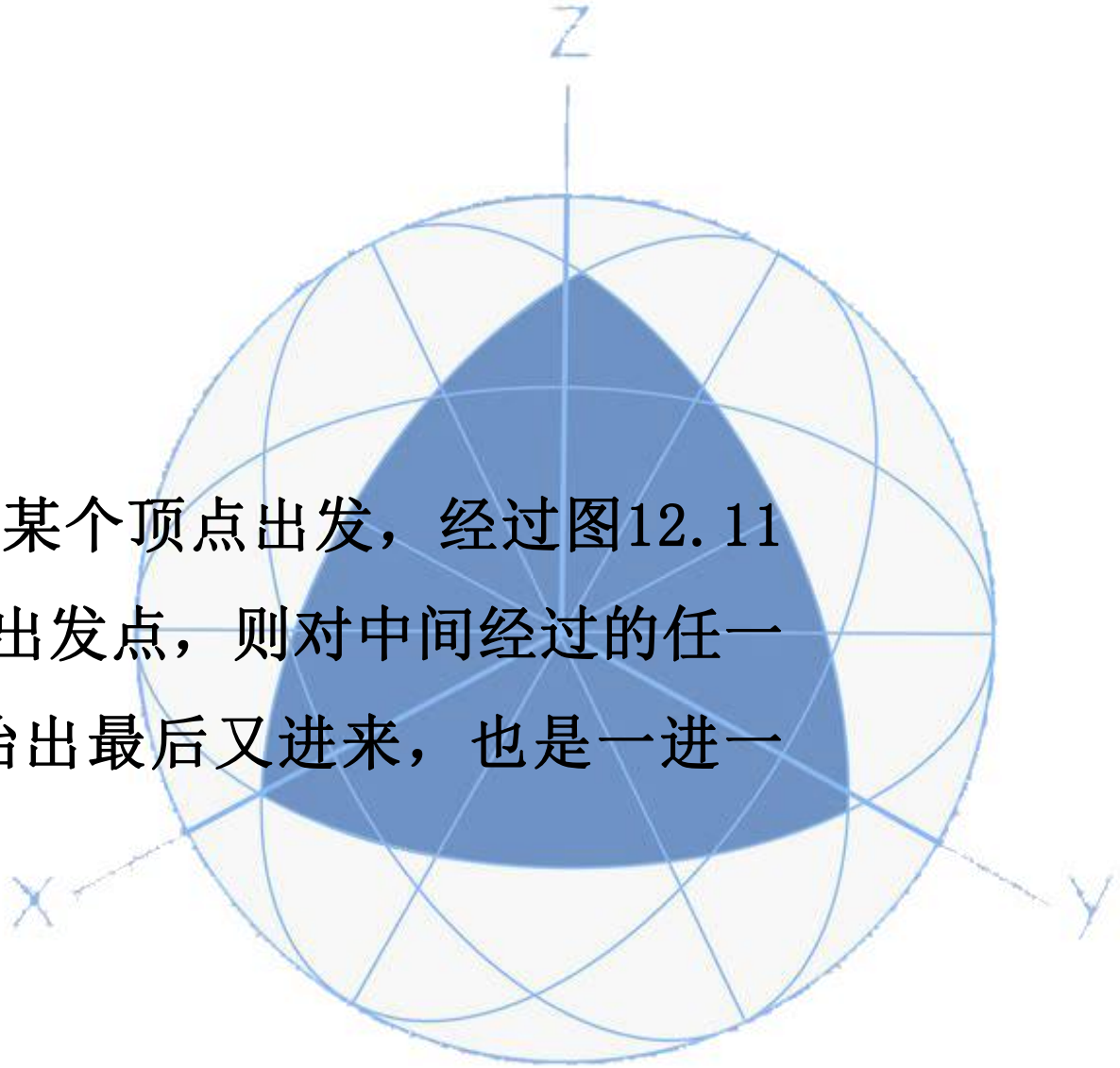
图 12.11　Königsberg 七桥的图模型

于是七桥问题

转化为图12.11是否

有欧拉环游的问题.





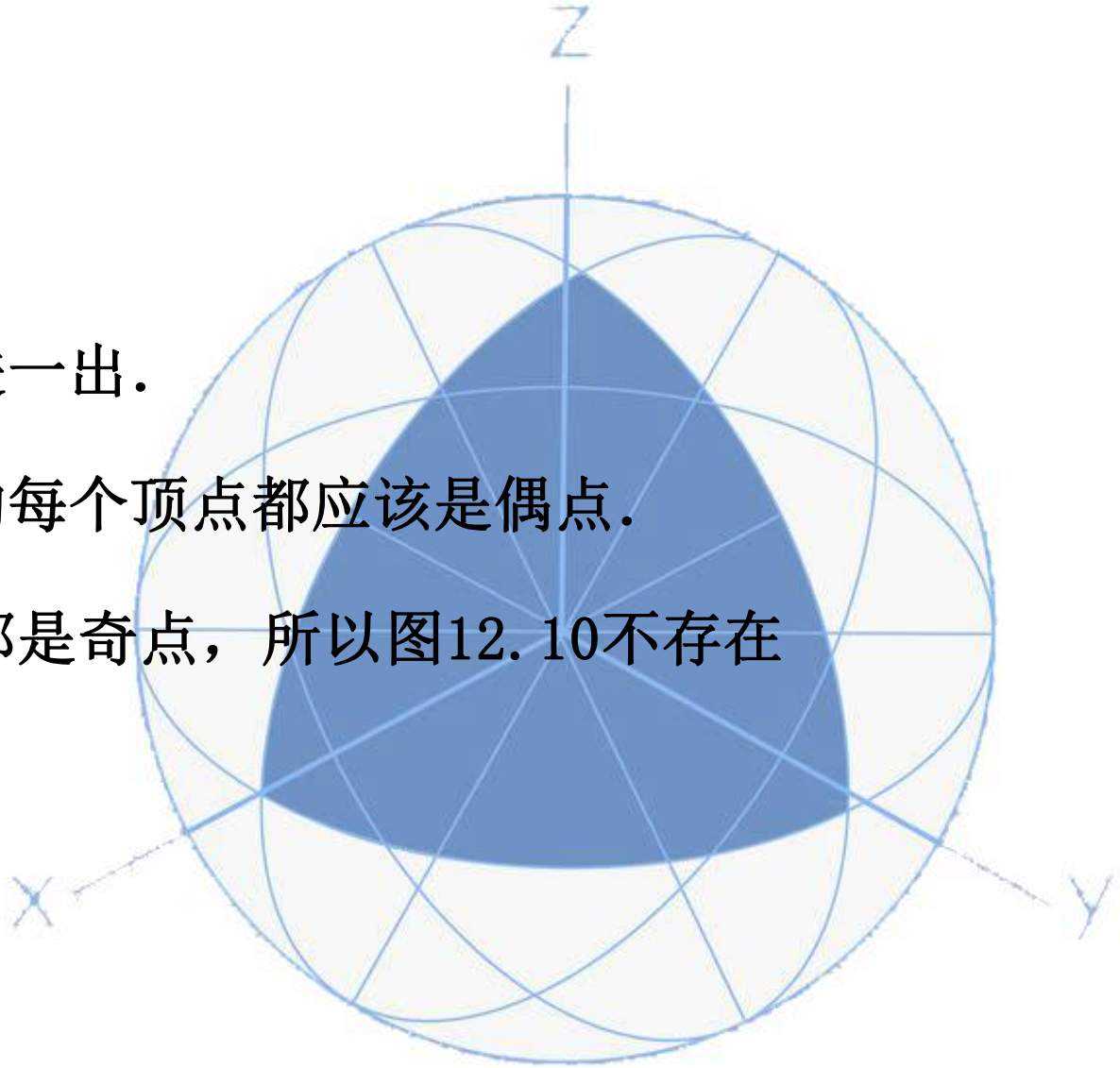
【模型构建】

如果图12.11有欧拉环游，即从某个顶点出发，经过图12.11所有边一次，且不重复，最后回到出发点，则对中间经过的任一顶点都是一进一出，而出发点开始出最后又进来，也是一进一出.

注意有的顶点可能有若干次一进一出.

不论如何，都意味着图12.11 的每个顶点都应该是偶点.

而实际上图12.11 的每个顶点都是奇点，所以图12.10不存在欧拉环游. 这就解决了七桥问题.



厦门大学
XIAMEN UNIVERSITY

THANK YOU