算　法　图　论

云南大学数学系

李建平

2016年9月

第二章　树与路

2.1 树的概念与性质

定义2.1 不包含圈的图称为**无圈图**(acyclic graph);连通的无圈图称为**树**(tree),常用字母T表示.

一个无圈图也称为**森林**(forest)，树也是森林。

在一棵树中，度数为1的顶点称为**树叶**(leaf)，度数大于1的顶点称为**分支点**.

例如 图2.1画给出了一些 6 个顶点的树，它们的任一种组合均为森林.

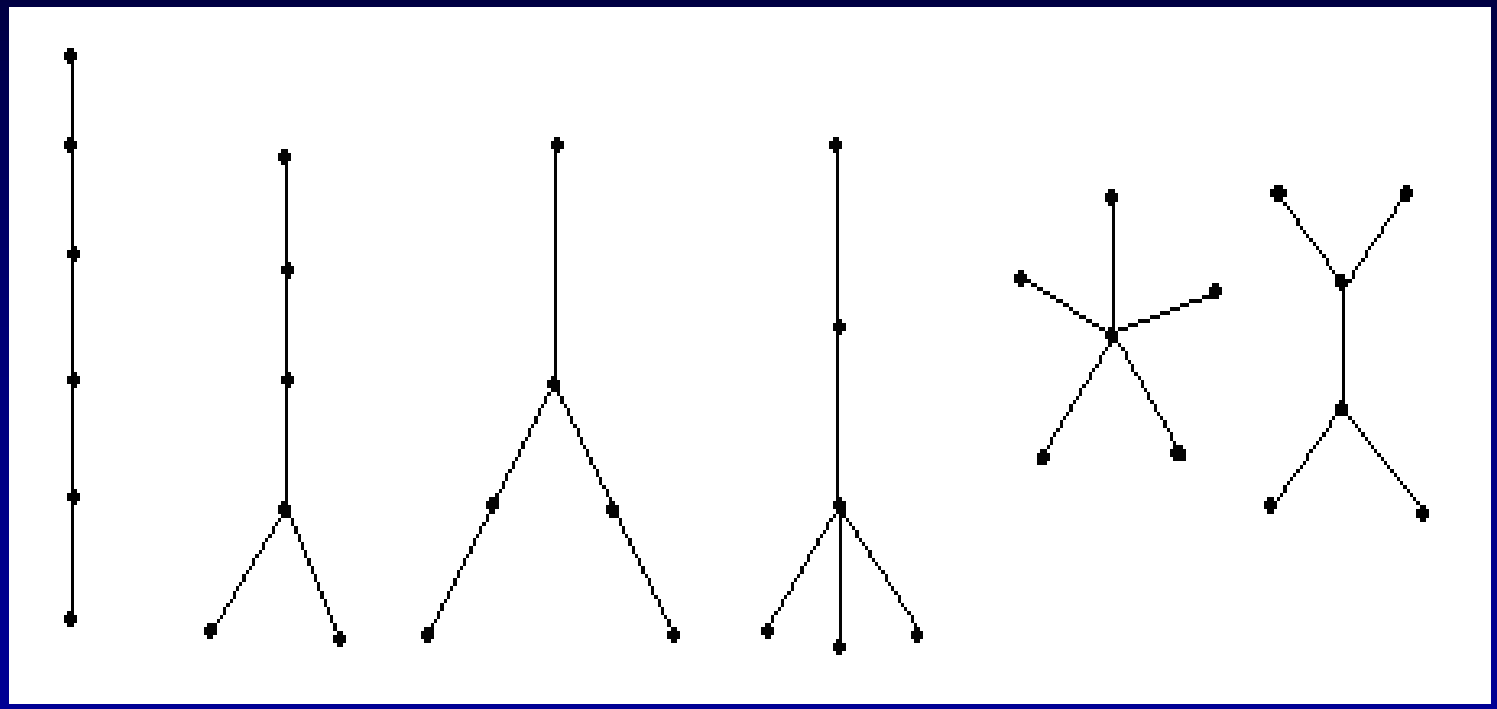


图2.1

树有许多等价的定义，在下面定理2.1中列出了 5 个等价命题。

定理2.1 设无向图 $G=(V, E)$ 的顶点数为 n，边数为 m，则下列命题等价：

(1) G是树；

(2) G中任意两顶点间有且仅有一条路相连；

(3) G是连通的，且 $m=n-1$；

(4) G中无圈，且 $m=n-1$；

(5) G中无圈，但在 G中任意不相邻的两顶点间增加一条边，就得到唯一一个圈。

证明：(1)→(2)设图G是树，由于连通性，任意两顶点u, v之间必有路相连。设$P_1=uu_1u_2\cdots u_sv$和$P_2=uv_1v_2\cdots v_kv$是图G中的两条$u-v$路，且$i+1$是使$u_{i+1}\neq v_{i+1}$的最小整数，$j(j>i)$是使v_{j+1}在P_1上的最小整数，即设$v_{j+1}=u_h$，那么，$u_iu_{i+1}\cdots u_hv_jv_{j-1}\cdots v_{i+1}v_i$是$G$中的一个圈，这与$G$是树矛盾。

(2) →(3)由于图G中任意两点间有一条路相连，所以G是连通的。

我们对图G的顶点个数作数学归纳，证明$m=n-1$.

当$n=1$时，$m=0$，结论显然成立。

设命题对$n-1$个顶点的图成立。现设G中有n个顶点、m条边的图。令$e=uv$是图G中任意

一条边，由（2），在G中去掉该边后，G成为两个连通分支G_1与G_2. 设G_1、G_2的顶点数分别为n_1、n_2，边数分别为m_1、m_2，由归纳假设：

$$m_1=n_1-1,\ m_2=n_2-1,$$

注意到$m=m_1+m_2+1$，从而

$$m=m_1+m_2+1=n_1-1+n_2-1+1=n-1。$$

（3）→（4）只须证明图G中无圈。若不然，对G中每一个圈，去掉其中的一条边，则该圈被去掉，但G中的顶点数与连通性不变，如此作下去，最后得到一棵树。这说明：$m>n-1$，得出与(3)矛盾的结论。所以，图G中不能有圈。

（4）→（5）先证明G是连通的。设G有k个分支，由（4），G中无圈，故每个分支是一棵树。设这k棵树分别是$G_1, G_2, \cdots G_k$，且G_i有n_i个顶点，m_i条边，$1 \leqslant i \leqslant k$，从而$m_i=n_i-1$，于是，

$$m=n_1+n_2+\cdots n_k-k=n-k。$$

与（4）比较得$k=1$。因此G是连通的，于是G是连通而无圈的图，即（1）成立，从而（2）成立。

设u和v是G中不邻接的两个顶点，由（2），在G中有一条连接u与v的路P，路P和边uv作成图$G+uv$的一个圈。因为G中无圈，所以$G+uv$的任何一个圈含有边uv。于是G中若有两个不同的圈，则由此可以得到连接u, v的两条不同的路，这与（2）是矛盾的。

（5）→（1）若G不连通，则存在属于G的两个不同分支的点u与v，于是$G+uv$中不含圈，这与（5）矛盾。

推论 2.1：由k棵树组成的森林满足：$m=n-k$, 其中n为G的顶点数，m为G的边数。

证明：设森林中每棵树的顶点数和边数分别是n_i和m_i(i=1, 2, …, k). 则由定理2.1,

$$m_i=n_i-1,\ i=1, 2, \cdots, k.$$

因此

$$\sum_{i=1}^{k} m_i = \sum_{i=1}^{k} (n_i - 1) = n - k$$

即

$$m=n-k.$$

推论2.2 一棵阶大于或等于2的树至少有两片树叶.

证明： 设$d_1 \leqslant d_2 \leqslant \cdots \leqslant d_n$是树$T$的度序列，因为$T$是连通的，所以树$T$的最小度$\delta(T)=d_1 \geqslant 1$。如果在树T中至多有1片树叶，则与

$$2n-2=2m(T)=\sum_{i=1}^{n} d_i \geq 1+2(n-1)$$

矛盾。

也可以利用最长路（或者极长路）的思想来证明。

问题：

1　给定一个图G, 如何判定G是一棵树?

2　如何判定一个图是否是连通图?

3　如何找到图G中的一个圈?

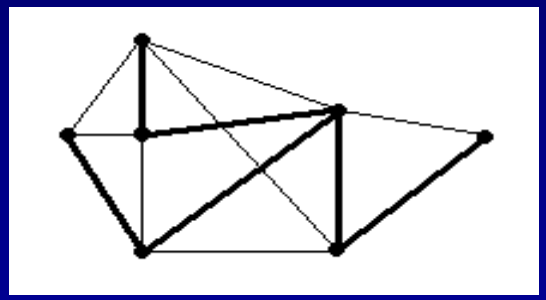
4　如何判定图G是森林?

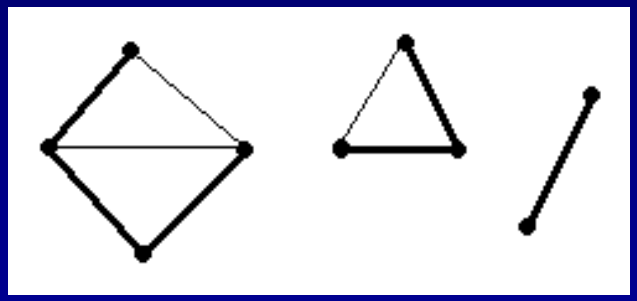
利用搜索算法**Searching**

2.2 支撑树(spanning tree)

定义2.2 设T是图G的一个支撑子图，如果T是一棵树，则称它为图G的一棵**支撑树**；若T为森林，则称它为G的支撑森林(spanning forest)。

下图分别描绘了一个连通图和一个分离图的支撑树和支撑森林。





(a)连通图和它的一棵支撑树　(b)图和它的支撑森林

图2.2

定理2.2每个连通图都至少包含一棵支撑树。

证明：设G是任意一个连通图，若在G中没有圈，则G本身是一棵树，其支撑树是G本身。若G中有圈C，去掉C中任意一条边后，不影响G的连通性，若G中还有圈，再去掉该圈的一条边，如此下去，直到G中无圈为止。最后得到图G的一个不含圈的连通支撑子图，该子图即为G的一棵支撑树。

推论2.3若$G=(V,E)$是连通图，则$|E|\geq|V|-1$.

证明：设G是连通图，由上述定理，G包含一棵支撑树T，所以

$$|E(G)|\geq|E(T)|=|V(T)|-1=|V(G)|-1.$$

问题： 设$G=(V, E)$为一个连通图, 如何求G的一棵支撑树?

求连通图支撑树的方法很多，下面介绍三种常用的方法。

算法1（反圈法）

①设$G=(V, E)$为一个图, 取$X^0=\{u_1\}$, u_1为V中的任意一点, $E^0=\phi$.

②若已知有X^k及$E^k(k\geqslant 0)$， $\Phi(X^k)$为X^k的反圈集合, 在$\Phi(X^k)$中选取一条或多条边, 使得所选边在$V-X^k$中具有不同的端点, 记所选边的集合为F^k, 所选顶点的集合为V^k, 取$X^{k+1}=X^k\cup V^k$, $E^{k+1}=E^k\cup F^k$.

③当$X^k=V$时, 停止, 有支撑树(V, E^k)；在某步，当$X^{(k)}\neq V$，但 $\phi(X^{(k)})=\varnothing$，表明G是不连通的，没有支撑树。

利用反圈法来求一些优化问题的基本步骤：

取 $X^{(0)}=\{v_1,v_2,\cdots,v_r\}\subset V, E^{(0)}=\varnothing,$（下面已存在 $X^{(k)},E^{(k)}$），按下面方式构造 $X^{(k+1)},E^{(k+1)}$ 。在 $\phi(X^{(k)})$ 中，按所需的条件选边，使得所选边在 $V-X^{(k)}$ 中具有不同端点，记这些边组成集合 $F^{(k)}$ ，这些被选端点为 $Y^{(k)}$ 。取 $X^{(k+1)}\Leftarrow X^{(k)}\cup Y^{(k)}$，$E^{(k+1)}\Leftarrow E^{(k)}\cup F^{(k)}$ 重复上述过程或停止。

该算法有三个因素要确定：（1）初值；（2）选边条件；（3）算法停止条件。

方法2（避圈法）：设G=（V，E）是给定的连通图，$G^k=(V，E^k)$是G的无圈支撑子图（开始k=0时，令$E^0=\varnothing$）。若G^k是连通的，则G^k就是G的支撑树；若G^k是不连通的，任选E中的一条边e_k，使e_k的两个端点分别在G^k的两个不同分图之中。令$G^{k+1}=（V，E^{k+1}）$，这里$E^{k+1}=E^k \cup \{e_k\}$，则G^{k+1}仍是G的无圈支撑子图。对G^{k+1}重复上述过程，则G^{n-1}必是G的一棵支撑树。

在构造过程中，方法2总保持支撑子图的无圈性，而逐步向连通性过渡。当G=（V，E）不是连通图时，只能得到支撑森林。

算法3（破圈法）

设 $G=(V,E)$为一连通图,在图G中任取一个圈,去掉该圈上的一条边,图G仍为连通图,这样进行下去,最终可得G的一个无圈连通支撑子图,即得到G的一棵支撑树.

当G=（V，E）是连通图时，在构造过程中，方法3总保持支撑子图的连通性，而逐步向无圈性过渡。当G=（V，E）不是连通图时，只能得到支撑森林。

思考：对于不连通的图 G=（V，E），如何求出它的支撑森林？

我们容易发现一个连通图的支撑树一般不唯一，只有当G本身是树时，其支撑树才唯一。

研究一个连通图的支撑树的数目是有一定意义的。只是要注意不同支撑树的定义。

问题：给定图G=(V,E),如何求G的所有不同的支撑树数目？

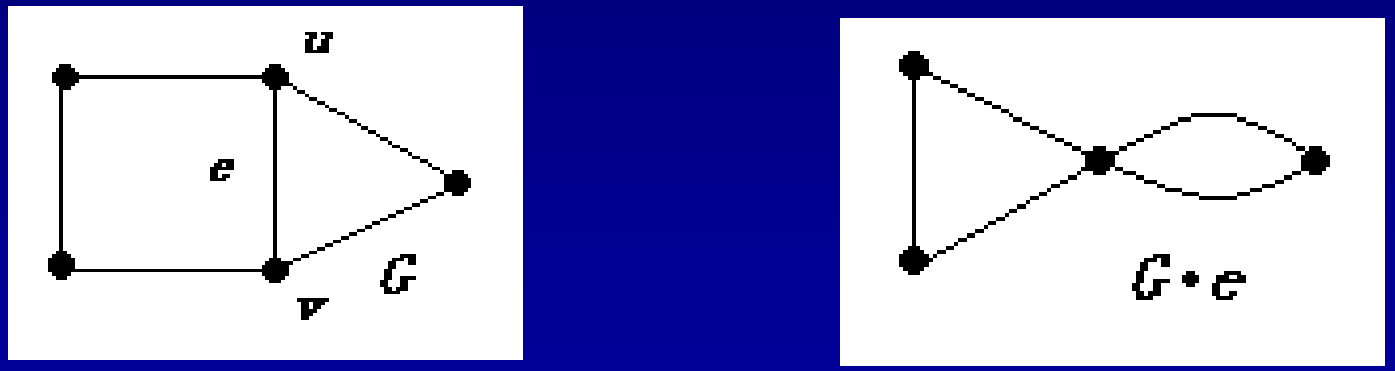
注：1. 设 $T_1=(V,E_1)$和$T_2=(V,E_2)$ 为支撑树，称 T_1,T_2 是不同的，如果 $E_1\neq E_2$ 。

2. 设$T_1=(V,E_1)$和$T_2=(V,E_2)$ 为支撑树，称 T_1,T_2 是相同的（或同构的），如果存在双射 $\varphi:T_1\to T_2$ ，当 $e=uv\in E_1$时，则有 $\varphi(u)\varphi(v)\in E_2$ ，i.e.，$T_1\cong T_2$

对于第一种形式的支撑树的数目，可以利用递推算法来求得，对于第二种形式的支撑树的数目，目前还没有好的办法求解，也没有证明是NP-难的。

下面介绍求连通图的支撑树的数目的递推算法（按第一种形式的支撑树的数目）。

图$G=(V, E)$的边e称为**被收缩**，是指删去边e，并使它的两个端点重合，如此得到的图记为$G \cdot e$. 下图(b)表示(a)收缩边e后得到的图。



(a)　　图2.3　　(b)

可知，若e是G的一条边，则有

$|V(G\cdot e)|=|V(G)|-1$,

$|E(G\cdot e)|=|E(G)|-1$和$\omega(G\cdot e)=\omega(G)$.

所以，若T是树，则$T\cdot e$也是树.

定理2.3 若e是图G的边，则$\tau(G)=\tau(G-e)+\tau(G\cdot e)$，这里用$\tau(G)$表示$G$的支撑树的数目.

证明 由于G的每一棵不包含e的支撑树也是$G-e$的支撑树，所以，$\tau(G-e)$就是G的不包含e的支撑树的棵数。

对于G的每棵包含e的支撑树T，相应的有一棵$G\cdot e$的支撑树$T\cdot e$. 这个对应显然是一一对应。所以$\tau(G\cdot e)$恰好是G的包含e的支撑树的棵树。由此推知：$\tau(G)=\tau(G-e)+\tau(G\cdot e)$。

当图G是完全图时，我们有Cayley 公式

定理2.4（Cayley 公式） $\tau(K_n)=n^{n-2}$.

（此定理在此不做证明.）

注： n^{n-2}不是K_n的非同构支撑树的数目，它是K_n的不同支撑树的数目（按第一种形式的支撑树的数目）；K_6有6棵非同构的支撑树，而它的不同支撑树有6^4棵.

2.3 最小支撑树(minimum spanning tree)

定义2.3 设G=（V,E;w）是一个连通赋权图，w：E→R^+，T是G的一棵支撑树，T的每条边所赋权值之和称为T的**权重**，记为w(T)。G中具有最小权的支撑树称为G的最小支撑树。

问题：在连通赋权图G=（V,E;w）中寻找一个支撑树T，使得w(T)达到最小，称该问题为最小支撑树问题（MST）。

一个赋权图的最小支撑树不一定是唯一的。求赋权图的最小支撑树的方法很多，以下介绍三种常用算法。

算法1（反圈法）

① 设$G=(V, E; w)$为一个赋权图, 取$X^0=\{u_1\}$, u_1为V中的任意一点, $E^0=\phi$.

② 若已知有X^k及$E^k(k\geqslant 0)$， $\Phi(X^k)$为X^k的反圈集合, 在$\Phi(X^k)$中选取一条边, 使得所选边的权尽可能小, 记所选边为e, 所选顶点为v, 取$X^{k+1}=X^k\cup\{v\}$， $E^{k+1}=E^k\cup\{e\}$.

③当$X^k=V$时, 停止, 这时有最小支撑树(V, E^k). 当 $X^{(k)}\neq V$ 时，但 $\phi(X^{(k)})=\varnothing$，没有支撑树，当然无最小支撑树。

方法2（**Kruskal**，避圈法）

设$G=（V，E）$是给定的图，$G^k=（V，E^k）$是G的无圈支撑子图（开始k=0时，令$E^0=\varnothing$）。若G^k是连通的，则G^k就是G的支撑树；若G^k是不连通的，任选E中的一条权重最小的边e_k，使e_k的两个端点分别在G^k的两个不同分图之中。令$G^{k+1}=（V，E^{k+1}）$，这里$E^{k+1}= E^k \cup \{e_k\}$，则$G^{k+1}$仍是G的无圈支撑子图。对$G^{k+1}$重复上述过程，则$G^{n-1}$必是G的一棵支撑树。

算法3（Rosenstiehl，破圈法）

设$G=(V,E;w)$是赋权图，在保证图G连通的情况下，（寻找适当的圈C）逐次删掉图G中权值最大的边，直到获得图G的一棵支撑树为止，该支撑树是图G的最小支撑树。

值得一提的是，我国学者管梅谷在1975年给出了求赋权图最小支撑树的一个简单方法，该方法也称为**破圈法**，即在赋权图中任取一个回路（即圈），删去该回路中的一条权值最大的边，如此进行下去，直到图中没有回路为止，最后得到的赋权图为原图的一棵最小支撑树。

2.4 最短路（shortest path）

定义2.4 设$G=(V, E; w)$为赋权图，$w: E\to R^+$，s、t为G中固定的两个顶点，设P是G中从s到t的一条路，如果$w(P)=\min\{w(Q) \mid Q$为G中从s到t的路，$w(Q)=\sum_{e\in E(Q)} w(e)\}$，称P为G中从$s$到$t$的最短路，$w(P)$称为$G$中从$s$到$t$的距离，记为$d(s, t)$（若$G$中从$s$到$t$无路，取$d(s, t)=+\infty$）.

问题：设$G=(V, E, w; s, t)$为赋权图，如何计算$d(s, t)$？

利用反圈法来求s到t的最短路：

（1）初值：$X^{(0)}=\{s\}$，给s一个标号 $\lambda(s)=0$

（$\lambda(v)$:从s到v的距离，从s到v的最短长度）

（2）选边条件：在 $\phi(X^{(k)})$ 中选边e=uv，

满足：$\lambda(u)+w(u,v)$ 达到最小， i. e.

$$\lambda(u)+w(u,v)=\min\left\{\lambda(u')+w(u',v')\middle|u'\in X^{(k)},v'\notin X^{(k)},e'=u'v'\in\phi(X^{(k)})\right\}$$

取$\lambda(v)=\lambda(u)+\omega(u,v)$

（3）停止条件：当 $t \in X^{(k)}$，则存在从s到t的最短路，其长度为 $\lambda(t)$

当 $t \notin X^{(k)}$，但 $\phi(X^{(k)}) = \varnothing$，说明从s到t没有路，当然亦无最短路。

（当 $X^{(k)} = V$ 时，说明存在s到所有点的最短路;当 $X^{(k)} \neq V$，但 $\phi(X^{(k)}) = \varnothing$，说明图G是不连通的。）

我们称一个算法A是好的算法（或有效算法），如果在任何一个m条边的n阶图G上实施此算法的计算步数都可由n和m的一多项式为其上界.

定理2.5 利用反圈法能求得从s到t的一条最短路，其复杂性为O(nm)。

下述算法是Dantig发现的,称为**标号法**.这个算法不仅找到了最短的(s, t)路,而且给出了s到所有其他顶点的最短路.算法中$w(e)$表示e的权,$p(a_k)$是赋权图G中s到a_k的最短路长度,其基本步骤如下:

①记$s=a_1$, $p(a_1)=0$, $A_1=\{a_1\}$, $T_1=\varphi$;

②若已得到$A_i=\{a_1,a_2,\ldots,a_i\}$, 且对$1\leqslant n\leqslant i$, 已知$p(a_n)$,对每一个$a_n\in A_i$, 求一点 $b_n^{(i)}\in N(a_n)-A_i=B_n^{(i)}$ 使 $w(a_n,b_n^{(i)})=\min\limits_{v\in B_n^{(i)}} w(a_n,v)$,

其中$N(a_n)$是与a_n邻接的所有的点的集合, 又称为点a_n的**邻域**或**邻集**;

③设有m_i, $1\leqslant m_i\leqslant i$, 而 $b_{m_i}^{(i)}$使$\mathrm{p}(a_{m_i})+w(a_{m_i},b_{m_i}^{(i)})$取最小值, 令

$$b_{m_i}^{(i)}=\mathrm{a}_{\mathrm{i}+1},\ \mathrm{p}(a_{i+1})=\mathrm{p}(a_{m_i})+w(a_{m_i},\mathrm{a}_{\mathrm{i}+1}),T_{i+1}=T_i\cup\{a_{m_i}\mathrm{a}_{\mathrm{i}+1}\}$$

④若$a_{i+1}=t$, 停止. 否则, 记$A_{i+1}=A_i\cup\{a_{i+1}\}$回到②.

定理2.6 算法中的函数$p(a_i)$给出了s与a_i的距离.

证明：对i用数学归纳法. $i=1$时显然成立. 若对所有j, $1\leqslant j<i$, $p(a_j)=d(s,a_j)$(表示s到a_j的距离). 令$P=v_0v_1\cdots v_d$, $v_0=s$, $v_d=a_i$是连接s与a_i的一条最短路, 从而$d=d(s,a_i)$, 令v_n是P中第一个不在A_{i-1}中的点. 由于a_i不属于A_{i-1}, 故这样的点v_n存在. 又因$v_0\in A_{i-1}$知$n\geqslant 1$, 设$v_{n-1}=a_k$, 则有$k\leqslant i-1$. 记P中由s到v_n的一段长度为l, s到v_{n-1}的一段长度为l_1. 由归纳法假设, 有$l_1\geqslant p(a_k)$, 且

$$d(s,a_i)=d\geq l=l_1+w(v_{n-i},v_n)\geq p(a_k)+w(a_k,b_k^{(i-1)})$$

$$\geq p(a_{m_{i-1}})+w(a_{m_{i-1}},a_i)=p(a_i)$$

其中$a_{m_{i-1}}$由上面算法的第三步得到，$1 \leqslant m_{i-1} \leqslant i-1$. 又由于存在一条长度为$p(a_i)$连结的$s$与$a_i$的路，可知

$p(a_i) \geq d(s,a_i)$

联合此两个不等式，即得

$$p(a_i)=d(s,a_i)$$

按归纳法原理，知定理成立.

因为Dantig算法总共需要作$n(n-1)/2$次加法和$n(n-1)$次比较，运算数量级为$0(n^2)$，故Dantig算法是一个好算法.

2.5 直径与半径(diameter and radius)

定义2.5 设$G=(V,E)$为一个连通图，可定义$d(s,t)$，这里$s,t\in V$，称$\max\{d(s,t)\mid s,t\in V\}$为G的**直径**，记为$d(G)$.

对给定$v\in V$，令$e(v)=\max\{d(u,v)\mid u\in V\}$，则称$e(v)$为顶点$v$的偏心率；

令 $\rho(G)=\min\{e(v)\mid v\in V\}$，称$\rho(G)$为图$G=(V,E)$的半径.

比较图G的直径与偏心率的定义可知，图G的直径是G的最大偏心率.

问题： 如何求连通图G的直径与半径?

2.6 支撑树的插点问题

定义2.6（支撑树的插点问题）给定一个连通赋权图G=(V,E;w,c)及两个正整数d和B，这里w,c：$E \to R^+$, 在图G中找一个支撑树T，使树T的权重w(T)不超过常数B，并向树T的一些特殊边上插入一些顶点，不妨设得到的树为T′（称T′为T的加细树）,使得树T′中任意边的权重都不超过常数d，目标是在满足上述条件情况下，求出插入顶点后所产生的费用达到最小。这里用insert(e)表示插入到边e中新点的个数，于是插点费用为$\sum_{e \in E(T)} \text{insert}(e)c(e)$.

定义2.7 （支撑树的插点数目问题）给定一个连通赋权图G=(V,E;w)及两个正整数d和B，这里w：$E \to R^+$, 在图G中找一个支撑树T，使树T的权重w(T)不超过常数B，并向树T的一些特殊边上加入一些顶点，不妨设得到的树为T′（称T′为T的加细树）,使得树T′中任意边的权重都不超过常数d，目标是在满足上述条件情况下，求出插入顶点数目达到最小。这里用insert(e)表示插入到边e中新点的个数，于是目标函数为 $\sum_{e \in E(T)} \text{insert}(e)$.

支撑树的插点问题是NP-完备的，而支撑树的插点数目问题是多项式可以解决的。具体算法的策略是：利用求最小支撑树的算法求得G的最小支撑树T，然后在T上加入相应的顶点即可，即对于边e，当w(e)>d时，加入insert(e) =[w(e)/d]-1个顶点，而当w(e) ≦d时，加入的顶点数为0，也即insert(e) =0。

具体算法如下：

算法：支撑树的插点数

Begin

Step 1. 由Prim算法或Kruskal算法求出G的最小支撑树T

Step 2. 如果w(T)>B，则停止，无可行解。

Step 3. 对于边e∈E(T)，它的权重为w(e)，在边e中插入insert(e)个顶点，其中insert(e)=[w(e)/d]-1。

Step 4. 输出支撑树T及相应的加细树T′。

End

定理2.7（1）支撑树的插点问题是NP-完备的；

（2）算法“支撑树的插点数”能够解决支撑树的插点数目问题，其算法的复杂性为求最小支撑树问题的复杂性，即$O(n^2)$。

（证明省略）

定义2.8（支撑树插点的总费用问题）给定一个连通赋权图G=(V,E;w,c)及两个正整数d和B，这里$w,c: E\rightarrow R^{+}$, 在图G中找一个支撑树T，并向树T的一些特殊边上插入一些顶点，不妨设得到的树为T′（称T′为T的加细树）,使得树T′中任意边的权重都不超过常数d，目标是在满足上述条件情况下，求出插入顶点后所产生的总费用达到最小。这里用insert(e)表示插入到边e中新点的个数，于是插点后的总费用为$w(T)+\sum_{e\in E(T)}insert(e)c(e)$.

算法：支撑树插点的总费用

Begin

Step 1.由图G构造一个新图$G^*=(V,E;w^*)$，顶点集和边集合与图$G=(V,E;w)$的相同,新图的边赋权函数为$w^*:E\rightarrow R^+$，这里$w^*(e)=w(e)+insert(e)c(e)$,对于任意$e\in E(G^*)$;

Step 2. 由Prim算法或Kruskal算法求出G^*关于权重函数w^*的最小支撑树T。

Step 3. 对于边$e\in E(T)$，它的权重为$w(e)$，在边e中插入$insert(e)$个顶点，其中 $insert(e)=[w(e)/d]-1$。

Step 4. 输出支撑树T及相应的加细树T′。

End

定理2.8 算法“支撑树插点的总费用”能够

解决支撑树插点的总费用问题，其算法的复杂性为求最小支撑树问题的复杂性，即$O(n^2)$。

（证明省略）

2.7 路的插点问题

定义2.9（路的插点问题）给定一个连通赋权图G=(V,E;w,c;s,t)及两个正整数d和B，这里w,c：$E \to R^+$, s和t是两个固定的顶点，在图G中寻找一条从s到t的路$P_{s,t}$，使得路$P_{s,t}$的权重$w(P_{s,t})$不超过常数B，并向路$P_{s,t}$的一些特殊边上插入一些顶点，不妨设得到的新路为$P^*_{s,t}$（称$P^*_{s,t}$为P的加细路）,使得路$P^*_{s,t}$中任意边的权重都不超过常数d，目标是在满足上述条件情况下，求出插入顶点后所产生的费用达到最小。这里用insert(e)表示加入到边e中新点的个数，于是插点费用为$\sum_{e \in E(Ps,t)} \text{insert}(e)c(e)$.

定义2.10（最短路的插点问题）给定一个连通赋权图G=(V,E;w,c;s,t)及正整数d，这里$w,c: E\rightarrow R^{+}$, s和t是两个固定的顶点，在图G中寻找一条从s到t的最短路$P_{s,t}$，并向最短路$P_{s,t}$的一些特殊边上插入一些顶点，不妨设得到的新路为$P^{*}_{s,t}$（称$P^{*}_{s,t}$为P的加细路），使得最短$P^{*}_{s,t}$中任意边的权重都不超过常数d，目标是在满足上述条件情况下，求出插入顶点后所产生的费用达到最小。这里用insert(e)表示插入到边e中新点的个数，于是插点费用为$\sum_{e\in E(Ps,t)}$insert(e)c(e).

定义2.11（路的插点数目问题）给定一个连通赋权图G=(V,E;w;s,t)及两个正整数d和B，这里w：$E \to R^+$, s和t是两个固定的顶点，在图G中寻找一条从s到t的路$P_{s,t}$，使得路$P_{s,t}$的权重$w(P_{s,t})$不超过常数B，并向路$P_{s,t}$的一些特殊边上插入一些顶点，不妨设得到的新路为$P^*_{s,t}$（称$P^*_{s,t}$为P的加细路）,使得路$P^*_{s,t}$中任意边的权重都不超过常数d，目标是在满足上述条件情况下，求出插入顶点数目达到最小。这里用insert(e)表示插入到边e中新点的个数，于是加点数目为$\sum_{e \in E(Ps,t)} \text{insert}(e)$.

路的插点问题是NP-完备的，而最短路的插点问题和路的插点数目问题是多项式可以解决的。

具体算法的策略是: (1)利用求最短路的算法求得G中从s到t的所有最短路构成的辅助有向图D，然后修正D上的权重，再在新权重下求从s到t最短路即可（解决最短路的插点问题）；(2) 利用(1)中的算法，再利用最短路的性质，就可设计算法来解决路的插点数目问题。

算法： 最短路的插点（HSCSP）

Begin

Step 1. 由反圈算法或Dijkstra算法求出 G中从s到t的最短路构成的子图，并构造辅助有向图D。

Step 2. 对于D中的每条弧e，构造新的权重w*(e)= insert(e)c(e)，其中 insert(e)=[w(e)/d]-1。

Step 3. 在D中，再利用反圈算法或Dijkstra算法求出从s到t的最短路$P_{s,t}$。

Step 4. 输出的最短路$P_{s,t}$及相应的加细最短路$P'_{s,t}$。

End

引理2.9 对于辅助有向图D=(V,A;w,c;s)中的固定顶点s和每个顶点u ∈V, 令$P_{s,u}$是一条从s到u路，并满足$w(P_{s,u})\leq B$, $Q_{s,u}$是一条从s到u最优路，也满足$w(Q_{s,u})\leq B$，其中B是常数, 则在最优路$Q_{s,u}$上的插点数目至少是$P_{s,u}$上的插点数目减去 (n-1), 即

$$insert(Q_{s,u})\geq insert(P_{s,u}) - (n-1).$$

（证明省略）

对于顶点u ∈V(D)及两个整数k≥0和i, 定义$d_k(u,i)$表示从s到u至多经过k条弧，恰好含有i个插入的新顶点，并具有最短路的长度, 如果没有这样的路存在，取$d_0(u,c)=+\infty$。

显然，对于辅助有向图D=(V,A;w,c;s)中的每个顶点u, 如果没有从s可以到达的负费用有向路, 从s到u最短路长度恰好就是 $d_n(u,i)$，这里$insert(P_{s,u})-(n-1)) \le I \le insert(P_{s,u})$, n是辅助有向图D的顶点数。

Algorithm SCSP

Step 1 For each u and s in D, the algorithm HSCSP computes the minimum number insert($P_{s,u}$), where the minimum is taken over all shortest paths from s to u.

Step 2 Denote $d_0(s,c)=0$ for each $c\geq 0$ and $d_0(u,c)=+\infty$ if $u\neq s$.

Step 3 k=1,2,…, n-1, do

For each vertex u in D, do

For (i=insert($P_{s,u}$)-(n-1)) to insert($P_{s,u}$), do

$d_k(u,i)=\min\{d_{k-1}(u,i),$

$\min\{d_{k-1}(v,i\text{-insert}(v,u)) + w(v,u)\}\};$

Step 4 Output the number $i^*=\min\{i \mid d_{n-1}(t,i) \leq B$ and the related path

定理2.10（1）路的插点问题是NP-完备的；

（2）算法“最短路的插点（HSCSP）”能够解决最短路的插点问题，其算法的复杂性为求最短路问题的复杂性，即$O(n^2)$

（3）算法SCSP能够解决路的插点数目问题，其算法的复杂性为求最短路问题的复杂性，即$O(n^2)$。

（证明省略）

2.8 增广问题(Augmentation Problem)

问题1 给定一个图$G=(V,E;w)$，如何增加一些边（包括权重），使得增加边后的图具有某种性质P，并要求所增加边的权重之和尽可能小。

问题2 给定图$G=(V,E;w)$，要在G中找权重最小的k-连通支撑子图。当$k=1$时，这是最小支撑树问题（MST）；当$k\geq 2$时，这是NP-完备问题

问题3 给定图$G=(V,E)$，要在G中找一个k-连通支撑子图，使其所含边数尽可能少。当$k=1$时这是支撑树问题；当$k\geq 2$时，这是NP完备问题

问题4 给定完全图 K_n，要在 K_n 中找一个 k-连通支撑子图，使该子图所含边数尽可能少。

记 $f(k,n)$ 表示完全图中所得到的 k-连通支撑子图的边数，则 $f(k,n)\geq kn/2$。

上面的问题都是在无向图中，在有向图中也有类似问题。

问题5 给定有向图 $D=(V,E;\omega)$，要在 D 中找权重最小的 k-强连通支撑子图。当 $k=1$ 时，这是NP-完备的问题。

问题6 给定完全有向图$D_n=(V,A)$，要在D_n中找一个k-强连通支撑子图，使该子图所含边数尽可能少。此时$f(k,n)\geq kn$。

2.9 2013年大学生数学建模B题

碎纸片的拼接复原

问题1. 对于给定的来自同一页印刷文字文件的碎纸机破碎纸片（仅纵切），建立碎纸片拼接复原模型和算法，并针对附件1、附件2给出的中、英文各一页文件的碎片数据进行拼接复原。如果复原过程需要人工干预，请写出干预方式及干预的时间节点。复原结果以图片形式及表格形式表达。

问题2. 对于碎纸机既纵切又横切的情形，请设计碎纸片拼接复原模型和算法，并针对附件3、附件4给出的中、英文各一页文件的碎片数据进行拼接复原。如果复原过程需要人工干预，请写出干预方式及干预的时间节点。复原结果表达要求同上。

【数据文件说明】

1. 每一附件为同一页纸的碎片数据；

2. 附件1、附件2为纵切碎片数据，每页纸被切为19条碎片；

3. 附件3、附件4为纵横切碎片数据，每页纸被切为11×19个碎片。

【结果表达格式说明】

复原图片表格表达格式如下：

1. 附件1、附件2的结果：将碎片序号按复原后顺序填入1×19的表格；

2.附件3、附件4的结果：将碎片序号按复原后顺序填入11×19的表格。

附件1

城上层楼叠巘。城下清淮古汴。举手揖吴云，人与暮天俱远。魂断。魂断。后夜松江月满。点点衣巾莎枣花。村里村北响绿车。牛衣古柳卖黄瓜。海棠珠缀一重重。清晓近帘栊。胭脂谁与匀淡，偏向脸边浓。小郑非常强记，二南依旧能诗。更有鲈鱼堪切脍，儿辈莫教知　自古相从休务日，何妨低唱浅吟。天垂云玉作春阴。坐中人半醉，帘外雪将深。双鬟绿坠。娇眼横波眉黛翠。妙舞翔蹁跹。掌上身轻意态妍　碧霄轻笼两凤，彩翅淡拂双鸦。为谁流睇不归家。错认门前过马。

我劝髯张归去好，从来自己忘情。尘心消尽道心平。江南与塞北，何处不堪行。闲离阻。谁念莱损斓王，何曾梦云雨。旧恨前欢，心事两无据。要知欲见无由，痴心犹自，倩人道、一声传语。风卷珠帘白上钩。萧萧乱叶报新秋。独携纤手上高楼。临水纵横回晚辔。归来转觉情怀动。梅笛烟中闻几弄。秋阴重。西山雪淡云凝冻。凭高眺远，见长空万里，云无留迹。珪璋飞来光射处，冷浸一天秋碧。玉宇琼楼，乘鸾来去，人在清凉国。江山如画，望中烟树历历。省可清言挥玉尘，真须保啬全真。风流何似道家纯。不应同蜀客，惟爱卓文君。自惜风流云雨散。关山有限情无限。待君重见寻芳伴。为说相思，目断画楼燕。莫恨黄花未吐。且教红粉相扶。酒阑不必看茱萸。俯仰人间今古。玉骨那愁瘴雾，冰姿自有仙风。海仙时遣探芳丛　倒挂绿毛么凤。

姐豆唐桑真过矣，凭君说与南荣。愿闻吴越报丰登。君王如有问，结袜顾王生。师唱谁家曲，宗风嗣阿谁。借君拍板与门槌。我也逢场作戏、莫相疑。攀腮嫌枕印。印枕嫌腮晕。闲照晚妆残。残妆晚照闲。可恨相逢能几日，不知重会是何年。茱萸仔细更重看。午夜风翻幔，三更月到床。簟纹如水玉肌凉。何物与侬归去、有残妆。金炉犹暖麝煤残。惜香更把宝钗翻。重闻处，余熏在，这一番、气味胜从前。菊暗荷枯一夜霜。新苞绿叶照林光。竹篱茅舍出青黄。霜降水痕收。浅碧鳞鳞露远洲。酒力渐消风力软，飒飒。破暖多情却恋头。烛影摇风，一枕伤春绪。归不去。凤楼何处。芳草迷归路。汤发云腴酽白，盏浮花乳轻圆。人间谁敢更争奴。斗取红窗粉面，炙手无人傍屋头。萧萧晚雨脱梧楸。谁怜季子敝貂裘。

附件2

fair of face.

The customer is always right. East, west, home's best. Life's not all beer and skittles. The devil looks after his own. Manners maketh man. Many a mickle makes a muckle. A man who is his own lawyer has a fool for his client.

You can't make a silk purse from a sow's ear. As thick as thieves. Clothes make the man. All that glisters is not gold. The pen is mightier than sword. Is fair and wise and good and gay. Make love not war. Devil take the hindmost. The female of the species is more deadly than the male. A place for everything and everything in its place. Hell hath no fury like a woman scorned. When in Rome, do as the Romans do. To err is human; to forgive divine. Enough is as good as a feast. People who live in glass houses shouldn't throw stones. Nature abhors a vacuum. Moderation in all things.

Everything comes to him who waits. Tomorrow is another day. Better to light a candle than to curse the darkness.

Two is company, but three's a crowd. It's the squeaky wheel that gets the grease. Please enjoy the pain which is unable to avoid. Don't teach your Grandma to suck eggs. He who lives by the sword shall die by the sword. Don't meet troubles half-way. Oil and water don't mix. All work and no play makes Jack a dull boy.

The best things in life are free. Finders keepers, losers weepers. There's no place like home. Speak softly and carry a big stick. Music has charms to soothe the savage breast. Ne'er cast a clout till May be out. There's no such thing as a free lunch. Nothing venture, nothing gain. He who can does, he who cannot, teaches. A stitch in time saves nine. The child is the father of the man. And a child that's born on the Sab-

附件3

便邮。温香熟美。醉惬云鬟垂两耳。多谢春工。不是花红是玉红。一颗樱桃荣寿口。不要黄金，只要人长久。学画鸦儿犹未就。眉尖已作伤春皱。清泪斑斑，挥断柔肠寸。顺人间。昔灯偷温拭尽残妆粉　春事阑珊芳草歇。客里风光，又过清明节。小院黄昏人忆别。落红处处闻啼鴂。岁云暮，须早计，要褐裘。故乡归去千里，佳处辄迟留　我醉歌时君和，醉倒须君扶我，惟酒可忘忧。一任刘玄德，相对卧高楼　记取西湖西畔，正暮山好处，空翠烟霏。算诗人相得，如我与君稀。约他年、东还海道，愿谢公、雅志莫相违。西州路，不应回首，为我沾衣。料峭春风吹酒醒。微冷。山头斜照却相迎。回首向来萧瑟处。归去。也无风雨也无晴。寻陌寻春去，红尘拂面来。无人不道看花回。惟见石榴新蕊、一枝开。

九十日春都过了，贪忙何处追游。三分春色一分愁。雨打梨花深闭门，风转柳花球。白雪清词出坐间。爱君才器两俱全。异乡风景却依然。团扇只堪题往事，新丝那解系行人。酒阑滋味似残春。

缺月向人舒窈窕，三星当户照绸缪。香生浅黛见纤柔。搔首踟躇归欤。自觉功名懒更疏。若问使君才与术，何如。占得人间一味愚。海东头，山尽处。自古空槎来去。槎有信，赴秋期。使君行不归。别酒劝君君一醉。清润潘郎，又是何郎婿。记取锹头新利市。莫将分付东邻子。西塞山边白鹭飞。散花洲外片帆微。桃花流水鳜鱼肥　主人顺小　欲向东风先醉倒。已属君家　且更从容等待他。愿我已无当世望，似君须向古人求　岁寒松柏肯惊秋

水涵空，山照市。西汉二疏乡里　新白发，旧黄金。故人恩义深。谁道东阳都瘦损，凝然点漆精神。瑶林终自隔风尘。试看披鹤氅，仍是谪仙人。三过平山堂下，半生弹指声中。十年不见老仙翁。壁上龙蛇飞动。暖风不解留花住，片片吹人无数。楼上望春归去。芳草迷归路。犀钱玉果。利市平分沾四坐。多谢无功。此事如何到得侬。元宵似是欢游好。何况公庭民讼少。万家游赏上春台，十里神仙迷海岛。

虽抱文章，开口谁亲。且闲闲、乐尽天真。几时归去，作个闲人。对一张琴，一壶酒，一溪云。相如未老。梁苑犹能陪俊少。莫惹闲愁。且折

附件4

bath day. No news is good news.

Procrastination is the thief of time. Genius is an infinite capacity for taking pains. Nothing succeeds like success. If you can't beat em, join em. After a storm comes a calm. A good beginning makes a good ending.

One hand washes the other. Talk of the Devil, and he is bound to appear. Tuesday's child is full of grace. You can't judge a book by its cover. Now drips the saliva, will become tomorrow the tear. All that glitters is not gold. Discretion is the better part of valour. Little things please little minds. Time flies. Practice what you preach. Cheats never prosper.

The early bird catches the worm. It's the early bird that catches the worm. Don't count your chickens before they are hatched. One swallow does not make a summer. Every picture tells a story. Softly, softly, catchee monkey. Thought is already is late, exactly is the earliest time. Less is more.

A picture paints a thousand words. There's a time and a place for everything. History repeats itself. The more the merrier. Fair exchange is no robbery. A woman's work is never done. Time is money.

Nobody can casually succeed, it comes from the thorough self-control and the will. Not matter of the today will drag tomorrow. They that sow the wind, shall reap the whirlwind. Rob Peter to pay Paul. Every little helps. In for a penny, in for a pound. Never put off until tomorrow what you can do today. There's many a slip twixt cup and lip. The law is an ass. If you can't stand the heat get out of the kitchen. The boy is father to the man. A nod's as good as a wink to a blind horse. Practice makes perfect. Hard work never did anyone any harm. Only has compared to the others early, diligently

问题1. 仅有纵切文本的复原问题

1.由于“仅有纵切”，碎纸片较大，所以信息特征较明显。一种直观的建模方法是：按照某种特征定义两条碎片间的（非对称）距离，建立赋权有向图，采用固定两个顶点的最短有向Hamilton路思想建立优化模型，其中固定两个顶点就是纸张的两个边界。关于最短有向Hamilton路的求解方法有很多，可以求得顶点数目为19（即弧数为18）的权重最小的有向Hamilton路。该实例可以求固定两个顶点之间的最短路.

2.可以利用向量排序方法来进行碎纸片拼接复原。

具体解法：

构造赋权有向图D=(V,A;w)如下：

用v_i表示纵切文本的第i文件， i=1,2,…,19

用弧(v_i,v_j)表示第i文件的右边能够与第j文件的左边相接，定义弧(v_i,v_j)的权重为文本v_i与文本v_j的距离，其中可以确定最左边与最右边的文本文件，不妨设为v_1和v_{19}。

于是，我们能够得到如下命题。

命题1：附件1能够正确地复原成为一个文本当且仅当D=(V,A;w)中存在一条从v_1到v_{19}的一条Hamilton路，i.e.，一条从v_1到v_{19}的一条路，并且包含19个顶点。

命题2：D=(V,A;w)中存在一条从v_1到v_{19}的一条Hamilton路，当且仅当D=(V,A;w)中存在一条从v_1到v_{19}的一条路，其权重达到最小，并且包含19个顶点。

算法 1：纵切碎片复原算法

Input：附件1（附件2）中的所有碎片文件

Output：拼接方案

Begin

Step 1：利用Matlab中的读写命令，把碎片文件转变为对应矩阵；

Step 2：通过人工干预或者Matlab编程，在附件1（附件2）的所有碎片文件中，找出位于原始完好文件最左边的碎片和最右边的碎片；

Step 3：根据附件1（附件2），按照上述方式构造赋权有向图；

Step 4：找出有向图中从顶点v_1到顶点v_{19}的总权重最小的Hamilton有向路；

Step 5：根据Step 4中的所找到的Hamilton有向路，按照有向路经过每个顶点的次序，把相应的碎片依次拼接起来，便得到正确的方案。

End

利用算法1，得到附件1中文碎片文件的拼接顺序如下：

008	014	012	015	003	010	002	016	001	004	005	009	013	018	011	007	017	000	006

利用算法1，得到附件2英文碎片文件的拼接顺序如下：

006	003	002	007	015	018	011	000	005	001	009	013	010	008	012	014	017	016	004

问题2. 有横、纵切文本的复原问题

一种建模方法是：首先利用文本文件的行信息特征，建立同一行碎片的聚类模型，这里可以修正问题1中的求解方法，共11次地利用问题1中的算法，每次寻找从一个固定顶点到其它11个固定顶点所含弧数恰好等于18（=19-1）最短路径，以达到聚类效果。在得到行聚类结果后，再利用类似于问题1中的方法完成每行碎片的排序工作，同时注意行间距离（尤其是有空白行）。最后对排序后的行，再作纵向排序。

先找到第一列和最后一列的文件对应的顶点，还可以构造辅助网络，利用最小费用流的算法求得流量为9的最小费用整数流，得到9个行的聚类，再利用类似于问题1中的方法完成每行碎片的排序工作，同时注意行间距离（尤其是有空白行）。

最后对排序后的行，再作纵向排序。

利用算法，求得附件3中所有中文碎片文件的拼接复原图顺序如下：

049	054	065	143	186	002	057	192	178	118	190	095	011	022	129	028	091	188	141
061	019	078	067	069	099	162	096	131	079	063	116	163	072	006	177	020	052	036
168	100	076	062	142	030	041	023	147	191	050	179	120	086	195	026	001	087	018
038	148	046	161	024	035	081	189	122	103	130	193	088	167	025	008	009	105	074
071	156	083	132	200	017	080	033	202	198	015	133	170	205	085	152	165	027	060
014	128	003	159	082	199	135	012	013	160	203	169	134	039	031	051	107	115	176
094	034	084	183	090	047	121	042	124	144	077	112	149	097	136	164	127	058	043
125	013	182	109	197	016	184	110	187	066	106	150	021	173	157	181	204	139	145
029	064	111	201	005	092	180	048	037	075	055	044	206	010	104	098	172	171	059
007	208	138	158	126	068	175	045	174	000	137	053	056	093	153	070	166	032	196
089	146	102	154	114	040	151	207	155	140	185	108	117	004	101	113	194	119	123

利用算法，求得附件4中所有英文碎片文件的拼接复原图顺序如下

191	075	011	154	190	184	002	104	180	064	106	004	149	032	204	065	039	067	147
201	148	170	196	198	094	113	164	078	103	091	080	101	026	100	006	017	028	146
086	051	107	029	040	158	186	098	024	117	150	005	059	058	092	030	037	046	127
019	194	093	141	088	121	126	105	155	114	176	182	151	022	057	202	071	165	082
159	139	001	129	063	138	153	053	038	123	120	175	085	050	160	187	097	203	031
020	041	108	116	136	073	036	207	135	015	076	043	199	045	173	079	161	179	143
208	021	007	049	061	119	033	142	168	062	169	054	192	133	118	189	162	197	112
070	084	060	014	068	174	137	195	008	047	172	156	096	023	099	122	090	185	109
132	181	095	069	167	163	166	188	111	144	206	003	130	034	013	110	025	027	178
171	042	066	205	010	157	074	145	083	134	055	018	056	035	016	009	183	152	044
081	077	128	200	131	052	125	140	193	087	089	048	072	012	177	124	000	102	115