第十六章 DNA 数术

第一节 DNA 数术的动机

为了方便中医与西医的结合, 辩证将 宏观医学与微观医学进行紧密的结合起来, 我一直在研究一种迅捷的有价值的元基观测角度, 于是动机明显. 开始实践. 罗盘最大的作用是趋吉避凶，因果关联， 我之后的研发数据都会优化合并归纳在数据预测API[5]最新版本中

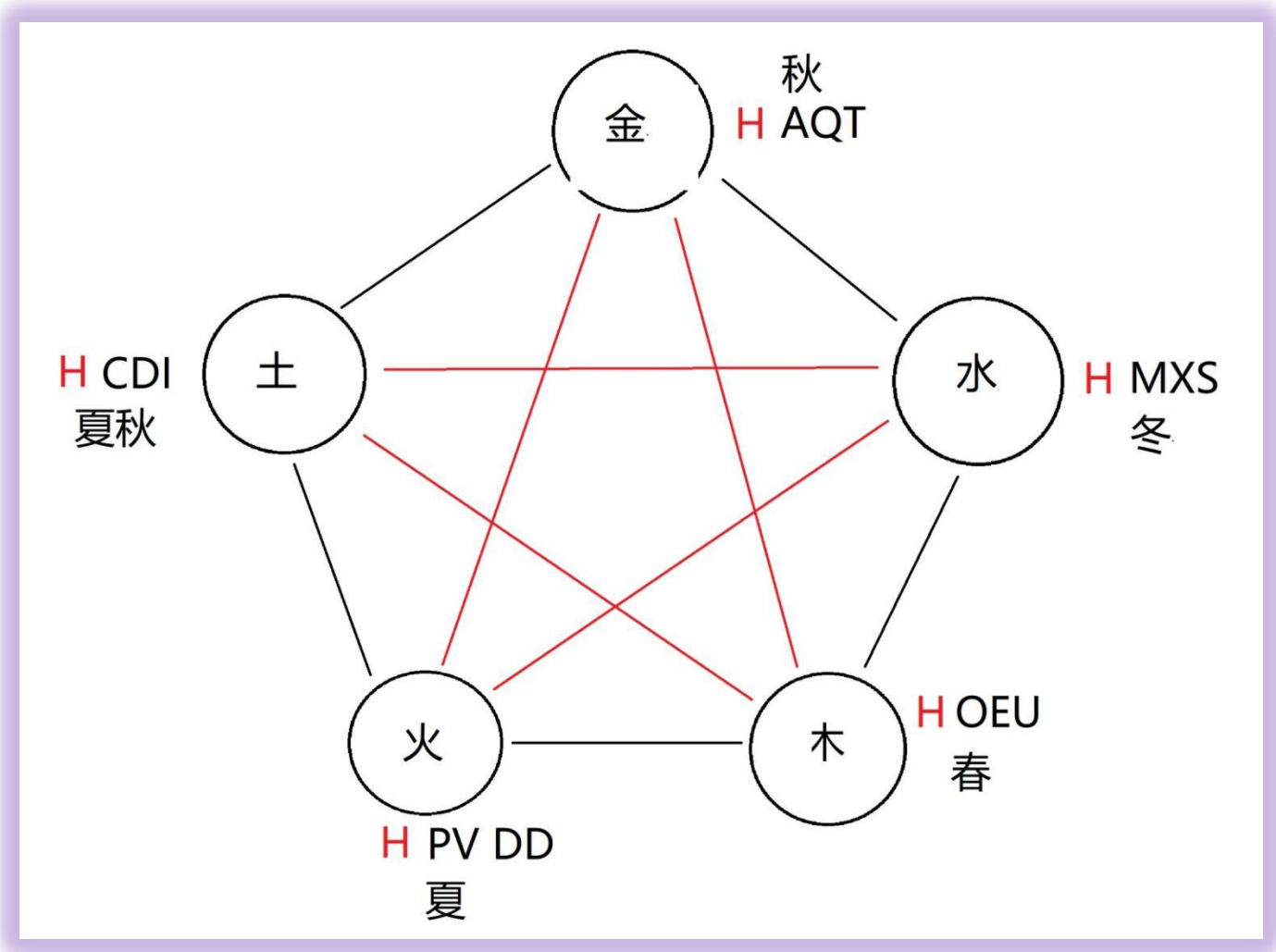
第二节 DNA 数术的应用需求

元基数术, 活性, 腐蚀性排序表

A	O	P	M	V	E	C	S	I	D	U	Q	T	X	H
0	1	2	1	0	1	1	0	2	1	1	2	1	1	2
5	3	3	3	3	1	2	2	1	2	3	1	4	5	1
-5	-2	-1	-2	-3	0	-1	-2	1	-1	-2	1	-3	-4	1
I	**Q**	**H**	E	**P**	**C**	**D**	O	M	S	U	**V**	**T**	X	**A**
D	**A**	**Q**	I	**P**	**T**	S	O	U	V	M	**X**	**C**	E	**H**
O	**U**	**V**	**M**	**P**	T	S	**X**	**C**	I	**E**	D	A	Q	**H**

今天来仔细描述这一节. 看过我 CSDN, 知乎等媒体的相信一定有了解了. 我设计的模式很简单, 仅仅把元基解码的数据中的酮基与甲基, 氨基 进行了计数, 形成了第一和第二排, 然后进行差值计算, 然后进行排序, 然后将元基图谱进行了离散的观测排列到八卦罗盘形式进行了小简单变换然后连线排序了下活性, 于是生成了下面的各种基础元基罗盘图.

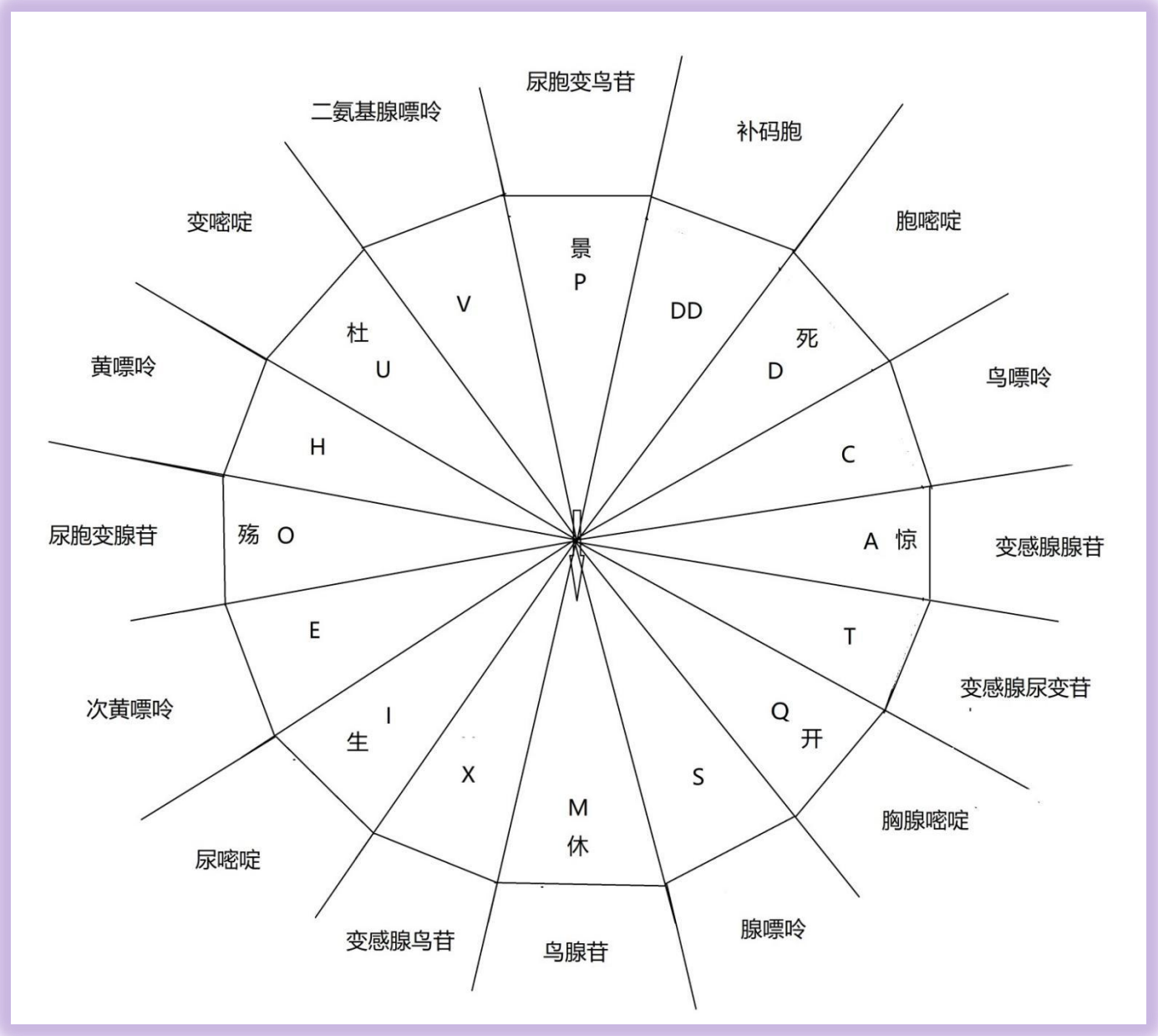
第三节 DNA 数术的具体描述



元基语义五行排序图

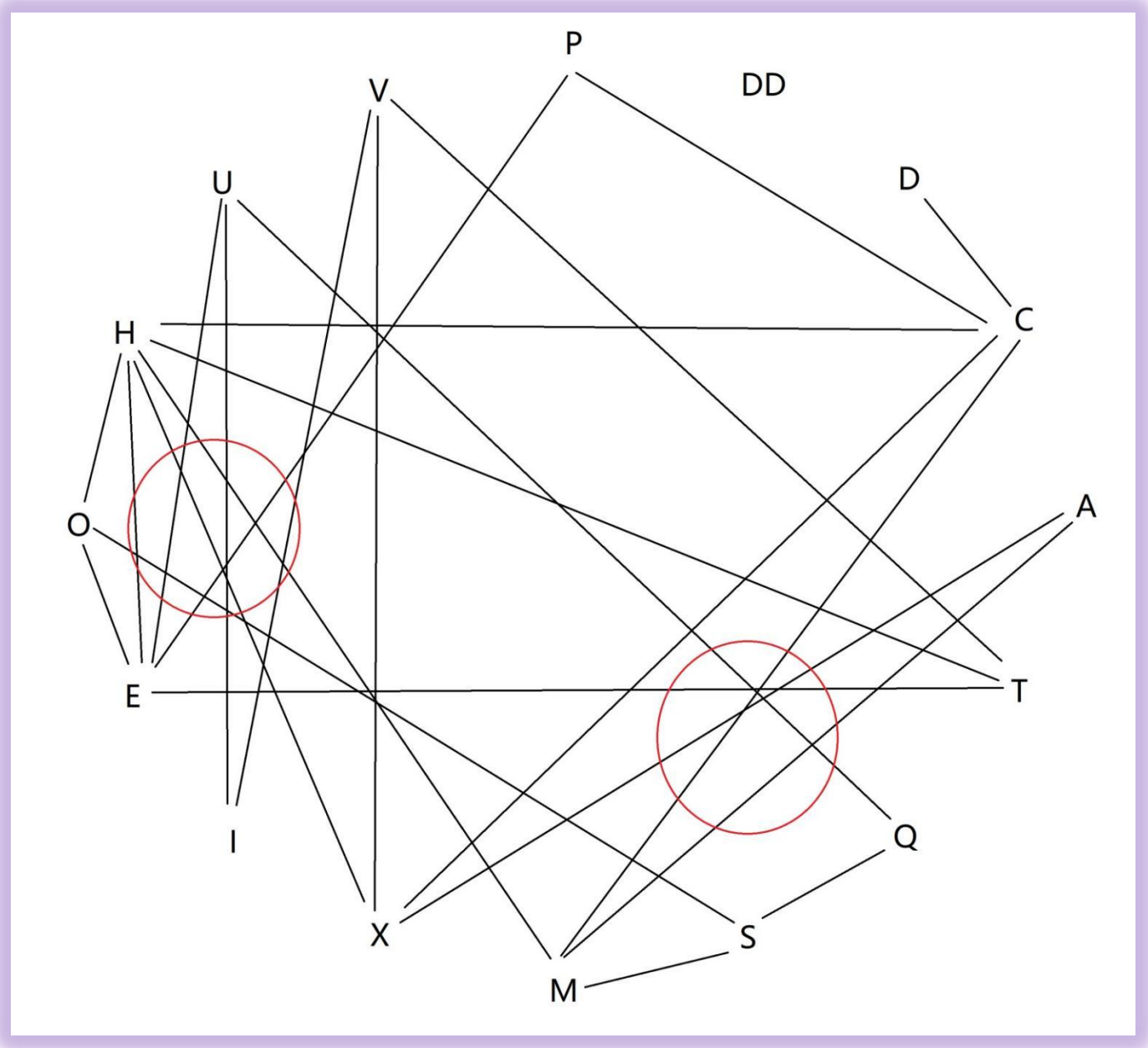
这里为什么我为什么会把 H 元基五行满排, 进行红色标注, 依据是它的活性. 与罗盘 7 个方位都有简洁变换逻辑. 其它元基布局就好解释了, 按语义编码推导出来的.

第四节 DNA 数术的应用实现



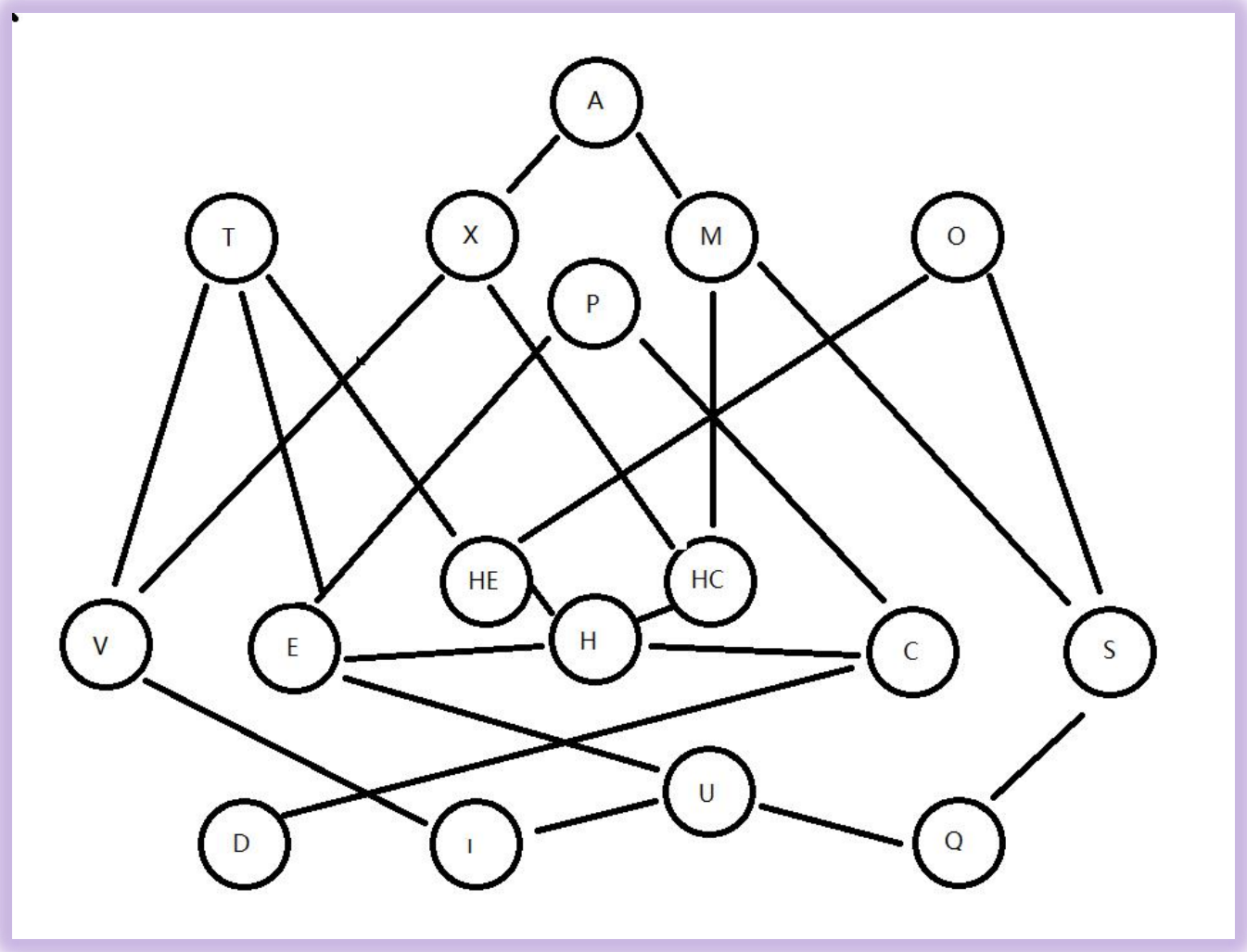
元基语义排序罗盘

上面这个罗盘, 我是根据 DNA 解码一著进行了语义的罗盘排序, 没有变动, 一开始少了一个 DD, 后来我不妨大胆了一点, 把元基补码的 DD 按比值排序加进去, 计算发现也刚好在 PD 之间. 于是就不改了, 作为第一代元基罗盘放在第二卷里.



元基语义肽展　活性排序罗盘

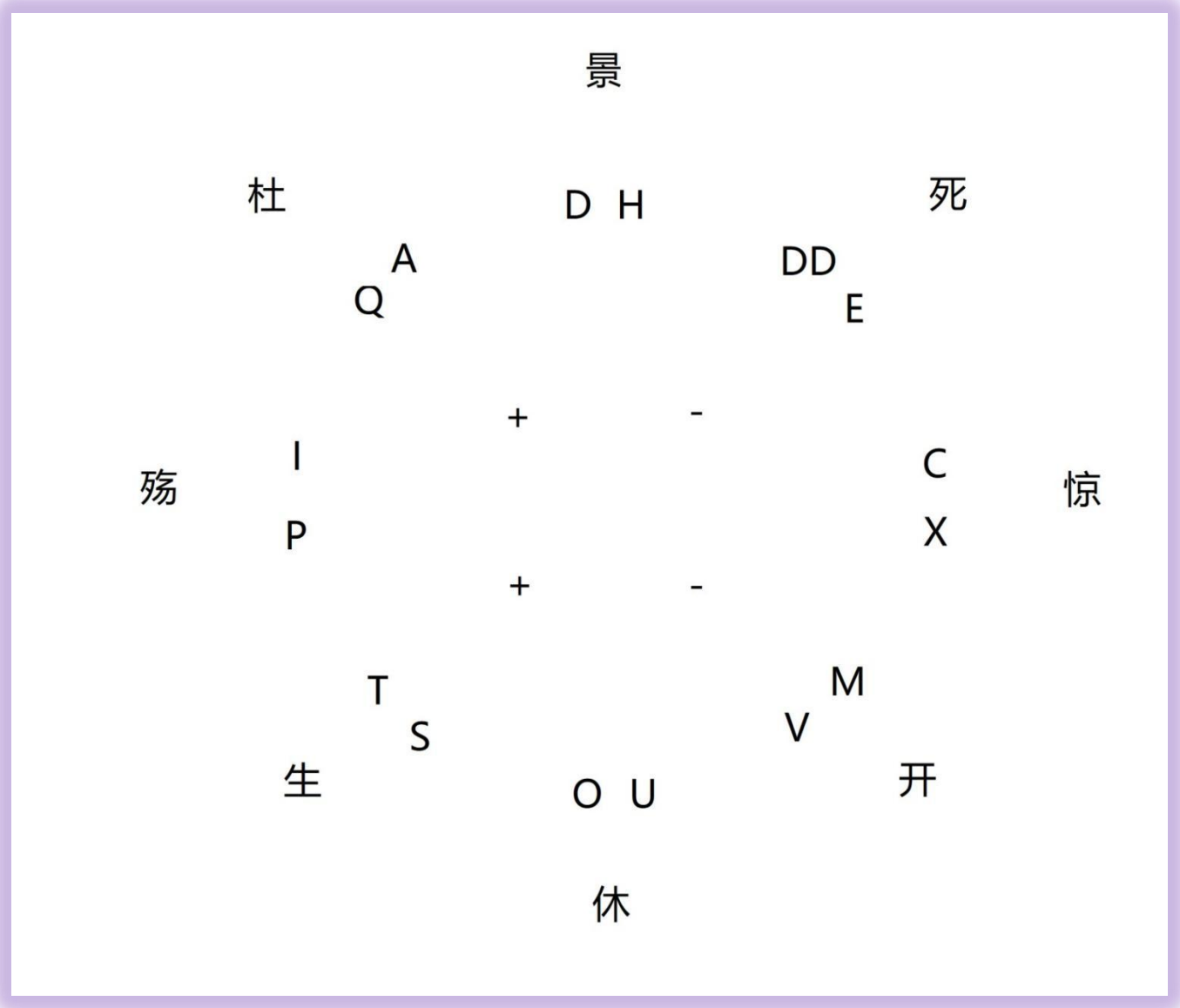
元基语义肽展 活性排序罗盘 是我将元基图谱进行了离散的观测排列到八卦罗盘形式进行了小简单变换然后连线排序了下活性, 于是生成了下面的各种基础元基罗盘图, 我在这里举个例子怕大家不理解, 如下:



元基肽展公式关系图

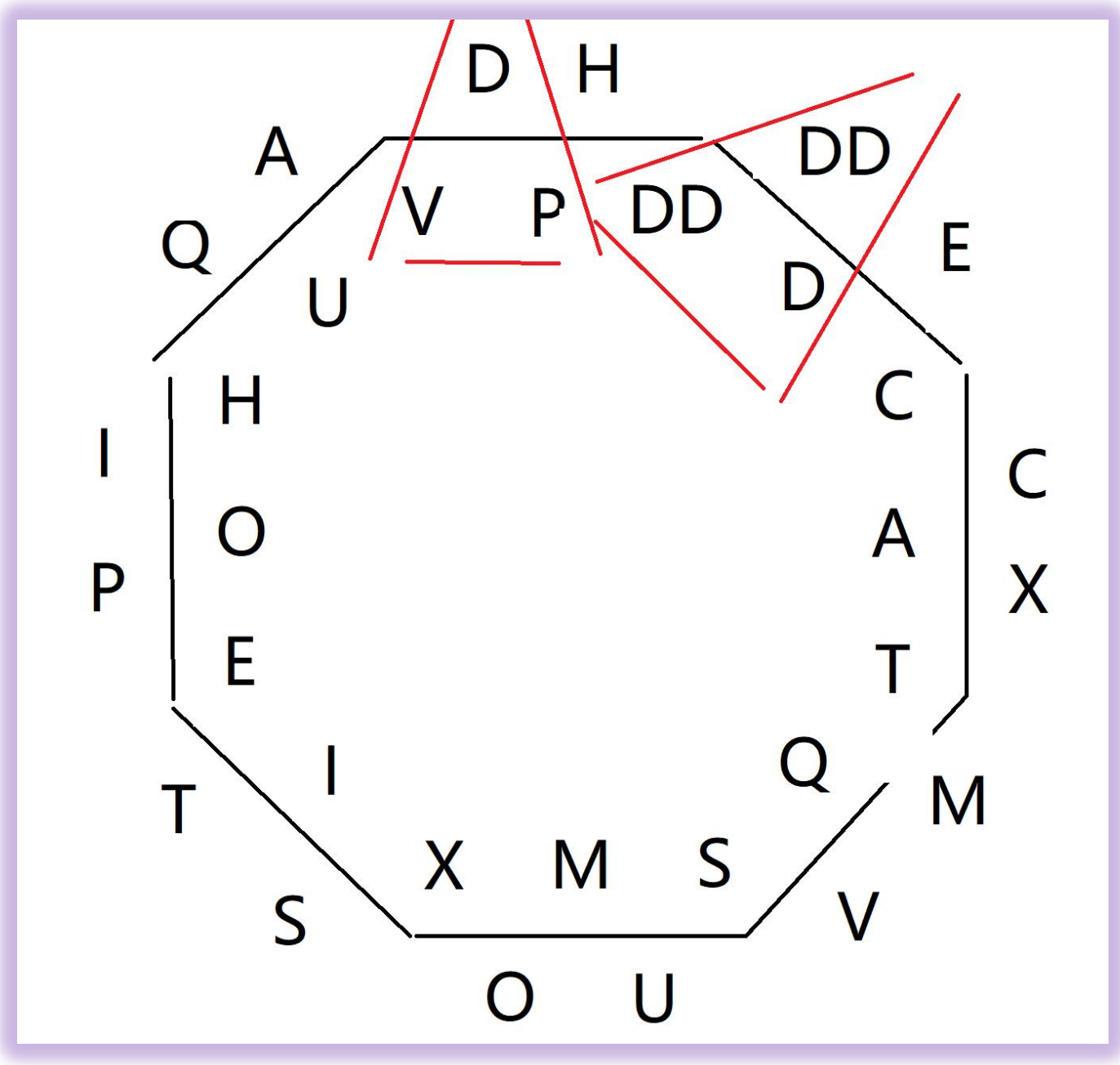
上面这个图不是第二卷产物, 是第一卷已有的 DNA 解码文件. 根据肽展公式的生化计算解码推导出来的. 第一卷中我已经描述的很详细了, 这里就停止介绍. 这里我要感谢 班加罗尔大学基督学院的维嘉斯神父, 08 年教我离散数学, 很认真, 谢谢他当年传授我这个离散观测变换的思维.

元基语义肽展 活性排序罗盘是无向的, 而元基肽展公式关系图是有向的, 因为应用场景不同, 这里观测方式不同. 于是我准备结合肽展公式关系与酸碱分类进行生化模式排序如下:



元基腐蚀性排序罗盘

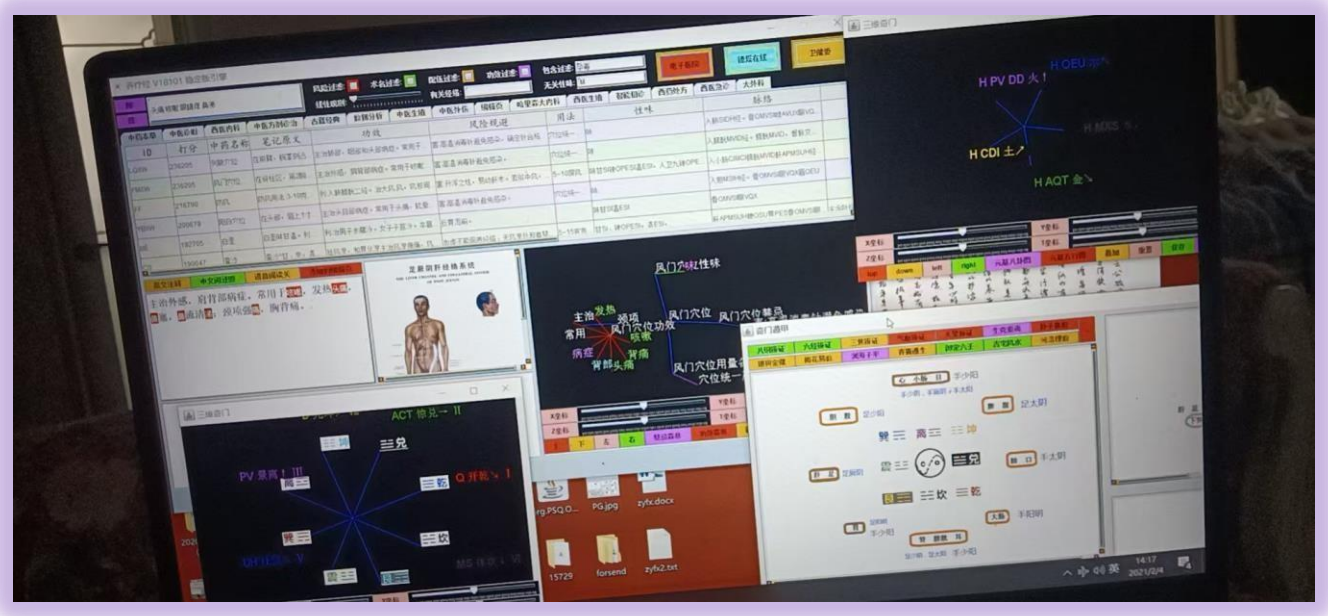
上图元基是我计算了元基数术, 活性, 腐蚀性排序表 后将第六行进行腐蚀中性重新定义然后进行强度排序的. 效果不错, 录在第二卷里先, 稍后整稿.



语义生化双元基叠加罗盘

这个图很好解释，我将语义元基与腐蚀性排序元基 进行了无理级嵌套，于是得到了这个无理级语义生化双元基叠加罗盘催化映射. 功能强大. 稍后描述. 这里右上角的 DD D DD 的双元催化双角, 胞嘧啶大堆, 这里是一个巨大的生化医学突破口, 第三卷 第四卷我会好好研究下. 先别想那么远，我先把基础做好, 医学家有幸看到我的文章, 蠢蠢欲动，让医学家去实验，我按我的思路来，求严谨，不求跳跃. 另外，语义肽展公式和生化肽展公式 开始有区别了，我准备定个计划，将语义肽展公式统一

用生化肽展公式, 虽然会少几个, 但效率增加是必然.



元基罗盘目前已经应用在德塔养疗经[17]医学大数据软件中, 用于中药数据的八纲辨证观测. 这只是一种简单应用, 我设计了更多的实例. 我把这个图加入了养疗经[17]的奇门遁甲页面, 生成三维立体图, 用于环境, 中药. 为中医观测提供多种观测和数据统计手段.

无机罗盘术数

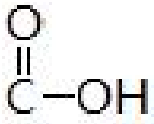
上面有了元基语义罗盘和元基生化罗盘后, DNA 催化的表达开始双元耦合, 于是第一能想到的是元基术数的 DNA 催化必定有钥匙来开启, 同时生化计算中催化钥匙同样可以用罗盘来表达, 于是我将下面八种非 DNA 单肽也通过酸碱活泼属性进行了方位归纳如下作为第一代钥匙参照物. 这酸醚醇酶 酚酯酰酮，八种肽键和其化学结构表达式来自人卫九[14] 等 生物化学教材书籍, 不是罗瑶光先生发明的, 卷头在前言中已经声明. .

单肽	语义	成份	酸	碱	DNA	RNA	钥匙
酰	浸润	COR	T		T	T	核酸
酸	腐蚀	CHO	T		T	T	消化
酮	调节	COR	T			T	激素
酯	聚能	CO	T			T	体能
酚	免疫	CHOR		T		T	免疫
醚	溶入	ROR		T		T	受体
醇	扩散	COH		T		T	循环
酶	诱导	CNOH		T	T	T	介质

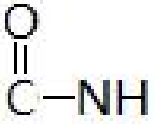
于是生成的语义钥匙和生化钥匙如下:

生化钥匙罗盘

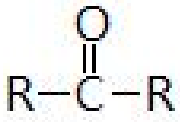
酰



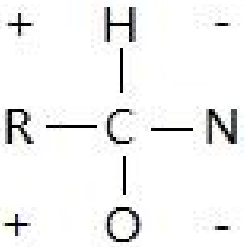
酸



酶

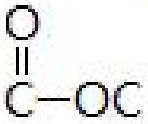


酮



醇

酯

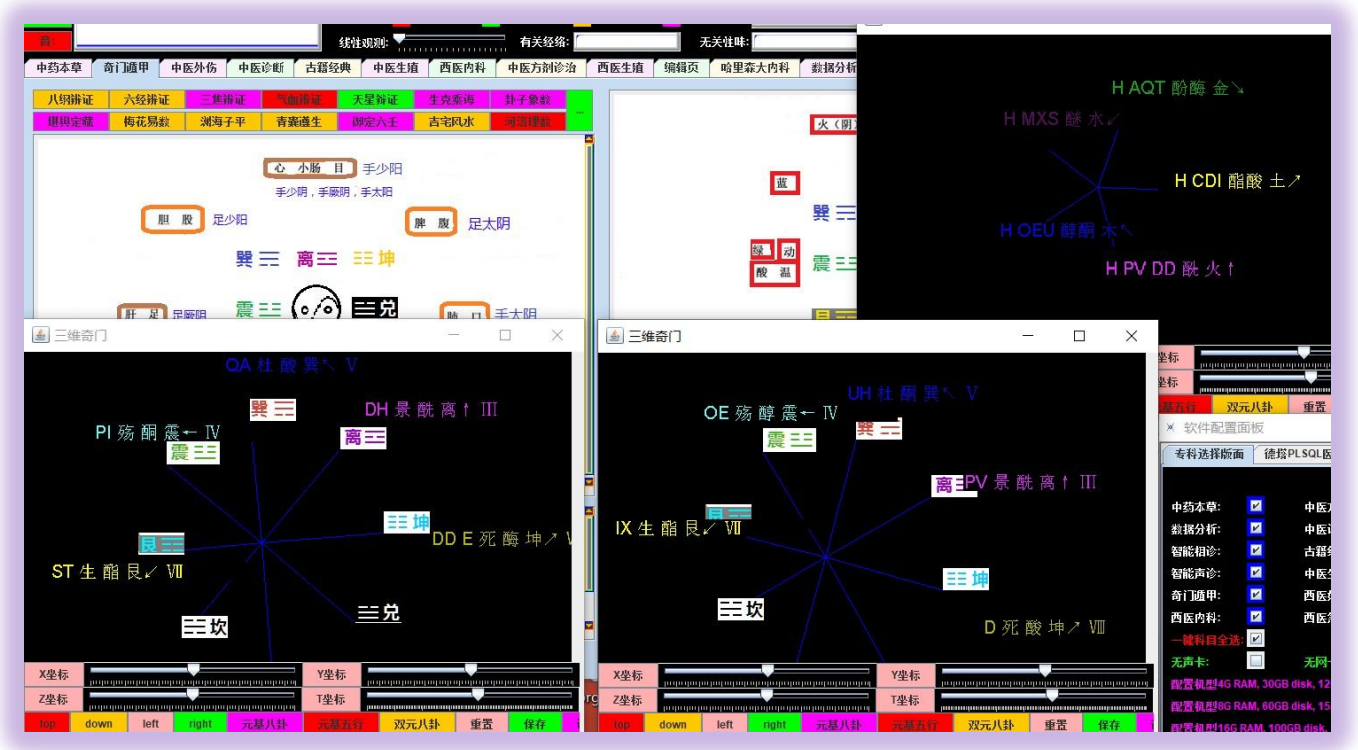


酚

醚

RC—OH


语义钥匙罗盘
浸润
火
调节
酰
腐蚀
木
土
酮
酸
扩散
木
醇
酚
金
免疫
酯
酶
土
金
聚能
诱导
醚
受体
水



催化钥匙应用于 罗盘中观测展示, 我在这里描述下, 为什么我把笔记中的胺替换为醚, 因为胺的活性剧烈, 属于毒品原型, 在元基催化过程中已经类似于破坏的性质. 而醚类激素已经具备了其活性属性, 就去掉了.

```
package OSI.SSI.ASU.OSU.PSU.MSU.pde;
import java.util.HashMap;
import java.util.Map;
import OSI.SSI.ASU.OSU.PSU.MSU.pds.PDE_PDS_DL;

//这个函数集用于将常数变换成十七进制元基数字,
//这个函数集用于将十七进制元基数字进行元基变换
//这个函数集用于将元基变换进行肽展概率丝化展开
//这个函数用于将肽展丝化的肽增十七进制进行元基变换
//这个函数用于将肽展丝化的肽增十七进制变换成元基数字
//这个函数用于十七进制元基数字进行十进制还原。
public class DecadeToPDS{
	//思想:肽展公式 1.2.2, 元基数字逻辑; 十七进制元基组合数学;概率论
	//算法:进制计算, 肽展计算
	//程序员: 罗瑶光,
	public Map<String, String> initonsMap= new HashMap<>();
	public Map<String, String> initonsCode= new HashMap<>();
	public Map<String, String> initonsSet= new HashMap<>();
	public Map<String, Integer> numberSet= new HashMap<>();
	public static void main(String[] Args) {
		DecadeToPDS decadeToPDS= new DecadeToPDS();
		decadeToPDS.IV_(decadeToPDS);
		int decade= (int)(Math.random()*1000 % 256);//随便写一个数
		double pDE_KEY_rate= 0.25;//随便模拟一个0-1之间的概率钥匙, 假设 0~0.5为酸, 0.5~1 为碱;

		decadeToPDS.doPDS(decadeToPDS, decade, pDE_KEY_rate);
	}
//	//元基符号变元基数字
//	//System.out.println("输入十进制数:"+ decade);
//	String seventeen= decadeToPDS.decadeToSeventeen(decade, decadeToPDS);
//	//System.out.println("元基进制数为:"+ seventeen);
//	String initons= decadeToPDS.seventeenToIntons(seventeen, decadeToPDS);
//	//System.out.println("变换为元基:"+initons);
//	//initons= "AOPMVE";
//	//System.out.println("输入元基:"+ initons);
//	//System.out.println("输入概率:"+ pDE_KEY_rate);
```

```
//		String pDS= decadeToPDS.initonsToPDS(initons, pDE_KEY_rate, decadeToPDS);
//		//System.out.println("输出肽丝:"+ pDS);
//		pDS= pDS.replace(".", "");
//		String pDSInitons= decadeToPDS.PDSToInitons(pDS, pDE_KEY_rate, decadeToPDS);
//		//System.out.println("肽丝增元:"+ pDSInitons);
//		//第二卷的肽展公式 可以用到了
//		//String pDEInitons= decadeToPDS.PDSToPDE(pDSInitons, pDE_KEY_rate, decadeToPDS);
//		//System.out.println("肽展增元:"+ pDEInitons);
//		String pDSSeventeen= decadeToPDS.initonsToSeventeen(pDSInitons, decadeToPDS);
//		//System.out.println("元基数字:"+ pDSSeventeen);
//		String pDSDecade= decadeToPDS.seventeenToDecade(pDSSeventeen, decadeToPDS);
//		//System.out.println("输出十进制数:"+ pDSDecade);
//		return Integer.valueOf(pDSDecade).intValue();
		private int doPDS(DecadeToPDS decadeToPDS, int decade, double pDE_KEY_rate) {
			String seventeen= decadeToPDS.decadeToSeventeen(decade, decadeToPDS);
			String initons= decadeToPDS.seventeenToIntons(seventeen, decadeToPDS);
			String pDS= decadeToPDS.initonsToPDS(initons, pDE_KEY_rate, decadeToPDS);
			pDS= pDS.replace(".", "");
			String pDSInitons= decadeToPDS.PDSToInitons(pDS, pDE_KEY_rate, decadeToPDS);
			String pDSSeventeen= decadeToPDS.initonsToSeventeen(pDSInitons, decadeToPDS);
			int pDSDecade= decadeToPDS.seventeenToDecade(pDSSeventeen, decadeToPDS);
			return pDSDecade;
		}
		public void IV_(DecadeToPDS decadeToPDS) {
			decadeToPDS.initonsMap.put("A", "7");
			decadeToPDS.initonsMap.put("O", "A");
			decadeToPDS.initonsMap.put("P", "2");
			decadeToPDS.initonsMap.put("M", "8");
			decadeToPDS.initonsMap.put("V", "D");
			decadeToPDS.initonsMap.put("E", "3");
			decadeToPDS.initonsMap.put("C", "1");
			decadeToPDS.initonsMap.put("S", "9");
			decadeToPDS.initonsMap.put("I", "E");
			decadeToPDS.initonsMap.put("D", "0");
			decadeToPDS.initonsMap.put("U", "F");
			decadeToPDS.initonsMap.put("Q", "G");
			decadeToPDS.initonsMap.put("T", "C");
			decadeToPDS.initonsMap.put("X", "6");
			decadeToPDS.initonsMap.put("+", "B");
```

```
decadeToPDS.initonsMap.put("-", "5");
decadeToPDS.initonsMap.put("H", "4");
//元基数字变元基符号
decadeToPDS.initonsCode.put("0", "D");
decadeToPDS.initonsCode.put("1", "C");
decadeToPDS.initonsCode.put("2", "P");
decadeToPDS.initonsCode.put("3", "E");
decadeToPDS.initonsCode.put("4", "H");
decadeToPDS.initonsCode.put("5", "-");
decadeToPDS.initonsCode.put("6", "X");
decadeToPDS.initonsCode.put("7", "A");
decadeToPDS.initonsCode.put("8", "M");
decadeToPDS.initonsCode.put("9", "S");
decadeToPDS.initonsCode.put("A", "O");
decadeToPDS.initonsCode.put("B", "+");
decadeToPDS.initonsCode.put("C", "T");
decadeToPDS.initonsCode.put("D", "V");
decadeToPDS.initonsCode.put("E", "I");
decadeToPDS.initonsCode.put("F", "U");
decadeToPDS.initonsCode.put("G", "Q");
//阿拉伯数字变元基数字
decadeToPDS.initonsSet.put("0", "0");
decadeToPDS.initonsSet.put("1", "1");
decadeToPDS.initonsSet.put("2", "2");
decadeToPDS.initonsSet.put("3", "3");
decadeToPDS.initonsSet.put("4", "4");
decadeToPDS.initonsSet.put("5", "5");
decadeToPDS.initonsSet.put("6", "6");
decadeToPDS.initonsSet.put("7", "7");
decadeToPDS.initonsSet.put("8", "8");
decadeToPDS.initonsSet.put("9", "9");
decadeToPDS.initonsSet.put("10", "A");
decadeToPDS.initonsSet.put("11", "B");
decadeToPDS.initonsSet.put("12", "C");
decadeToPDS.initonsSet.put("13", "D");
decadeToPDS.initonsSet.put("14", "E");
decadeToPDS.initonsSet.put("15", "F");
decadeToPDS.initonsSet.put("16", "G");
```

```
        //元基数字变阿拉伯数字
        decadeToPDS.numberSet.put("0", 0);
        decadeToPDS.numberSet.put("1", 1);
        decadeToPDS.numberSet.put("2", 2);
        decadeToPDS.numberSet.put("3", 3);
        decadeToPDS.numberSet.put("4", 4);
        decadeToPDS.numberSet.put("5", 5);
        decadeToPDS.numberSet.put("6", 6);
        decadeToPDS.numberSet.put("7", 7);
        decadeToPDS.numberSet.put("8", 8);
        decadeToPDS.numberSet.put("9", 9);
        decadeToPDS.numberSet.put("A", 10);
        decadeToPDS.numberSet.put("B", 11);
        decadeToPDS.numberSet.put("C", 12);
        decadeToPDS.numberSet.put("D", 13);
        decadeToPDS.numberSet.put("E", 14);
        decadeToPDS.numberSet.put("F", 15);
        decadeToPDS.numberSet.put("G", 16);
    }
    //      //准备集成第二卷的AOPM 级别 肽展公式，已经并入PDSToInitons 函数中
    //      private String PDSToPDE(String pds, double pDE_KEY_rate, DecadeToPDS decadeToPDS) {
    //
    ////pds= pds.replace("UQ", "V");
    ////pds= pds.replace("DI", "C");
    ////pds= pds.replace("IQ", "S");
    ////pds= pds.replace("VS", "A");
    ////pds= pds.replace("ES", "O");
    ////pds= pds.replace("EC", "P");
    ////pds= pds.replace("CS", "M");
    ////pds= pds.replace("VE", "T");
    ////pds= pds.replace("VC", "X");
    ////
    //                  return pds;
    //      }
    //这个函数集用于将常数变换成十七进制元基数字,
    public String decadeToSeventeen(int decade, DecadeToPDS decadeToPDS) {
        String seventeen= "";
        int decad= decade;
        while(0< decad/ 17) {
```

```
                int seventeenth= decad% 17;
                seventeen= decadeToPDS.initonsSet.get(""+ seventeenth)+ seventeen;
                decad= decad/ 17;
            }
            seventeen= decadeToPDS.initonsSet.get(""+ decad)+ seventeen;
            //
            return seventeen;
        }
        //这个函数集用于将十七进制元基数字进行元基变换
        public String seventeenToIntons(String seventeen, DecadeToPDS decadeToPDS) {
            String initons= "";
            for(int i= 0; i< seventeen.length(); i++) {
                initons+= decadeToPDS.initonsCode.get(""+ seventeen.charAt(i));
            }
            //
            return initons;
        }
        //这个函数集用于将元基变换进行肽展概率丝化展开
        public String initonsToPDS(String initons, double pDE_KEY_rate, DecadeToPDS decadeToPDS) {
            String PDS= "";
            StringBuilder PDEKey= new StringBuilder("");
            for(int i= 0; i< initons.length(); i++) {
                PDS+= new PDE_PDS_DL().initonPDSwithBYS(""+ initons.charAt(i), pDE_KEY_rate, PDEKey,
true)+ ".";
            }
            //System.out.println("生成钥匙:"+ PDEKey);
            //
            return PDS;
        }
        //这个函数用于将肽展丝化的肽增十七进制进行元基变换
        public String PDSToInitons(String pDS, double pDE_KEY_rate, DecadeToPDS decadeToPDS) {
            String initons= "";
            //initons= new PDE_PDS_DL().initonPDIwithBYS(pDS, 0, new StringBuilder(), false);
            //initons= new PDE_PDS_DL().initonPDEwithBYS(pDS, pDE_KEY_rate, new StringBuilder(), true);
            initons= new PDE_PDS_DL().initonPDE_DCDLwithBYS(pDS, pDE_KEY_rate, new StringBuilder(),
true);
            return initons;
        }
        //这个函数用于将肽展丝化的肽增十七进制变换成元基数字
```

```
	public String initonsToSeventeen(String initons, DecadeToPDS decadeToPDS) {
		String seventeen= "";
		//
		for(int i= 0; i< initons.length(); i++) {
			seventeen+= decadeToPDS.initonsMap.get(""+ initons.charAt(i));
		}
		return seventeen;
	}
	//这个函数用于十七进制元基数字进行十进制还原。
	public int seventeenToDecade(String seventeen, DecadeToPDS decadeToPDS) {
		int decade= 0;
		//A11   10*17*17 + 1*17 + 1
		for(int i= 0; i< seventeen.length(); i++) {
			int value= decadeToPDS.numberSet.get(""+ seventeen.charAt(i)).intValue();
			decade+= value* Math.pow(17,  seventeen.length()- 1- i);
		}
		return decade;
	}
	//这个函数用于十七进制元基数字进行十进制矩阵变换。
	public int[][] doPDSMatrix(DecadeToPDS decadeToPDS, int[][] rp, double facx) {
		for(int i= 0; i< rp.length; i++) {
			for(int j= 0; j< rp[0].length; j++) {
				rp[i][j]= decadeToPDS.doPDS(decadeToPDS, rp[i][j], facx);
			}
		}
		return rp;
	}
}
```

```
package OSI.SSI.ASU.OSU.PSU.MSU.pds;
//这个函数用于元基进行数字逻辑丝化变换
//思想:肽展公式，十七进制元基数字，元基数字逻辑
//作者:罗瑶光
//算法参考如下(肽展公式在离散数学中根据贝叶斯进行数字逻辑变换)
//#### 元基数字 =  元基符号=  肽展公式元基数字变换 = (肽概率展开数字逻辑集合)
//#### 0        =          D =    DD =(D, DD)
//#### E        =          I =    I =(I)
//#### F        =          U =    I++ OR Q--  =(I, Q)
//#### G        =          Q =   Q  =(Q)
//
//#### 1        =          C =    DI  =(DI)
//#### 3        =          E =    IU, DU  =(IU, DU)
//#### 4        =          H =    (IU, DU) OR DI  =(IU, DU, DI) OR (IUDI, DUDI)
//#### D        =          V =    UQ  =(UQ)
//#### 9        =          S =    QI  =(QI)
//
//
//#### 2        =          P =    (IU, DU) + DI  =(IUDI, DUDI)
//#### 5        =          HC =   ((IU, DU) OR DI) + DI  =(IUDI, DUDI, DIDI) OR (IUDIDI, DUDIDI)
//#### B        =          HE =    ((IU, DU) OR DI) + (IU, DU)  =(IUIU, IUDU, DUIU, DUDU, DIIU, DIDU) OR (IUDIIU, IUDIDU, DUDIIU, DUDIDU)
//#### A        =          O =    (IU, DU) + QI  =(IUQI, DUQI)
//#### 7        =          A =    UQQI  =(UQQI)
//
//
//#### 8        =          M =    ((IU, DU) OR DI) + DI + QI  =(IUDIQI, DUDIQI, DIDIQI) OR (IUDIDIQI, DUDIDIQI)
//#### 6        =          X =    UQ + ((IU, DU) OR DI) + DI  =(UQIUDI, UQDUDI, UQDIDI) OR (UQIUDIDI, UQDUDIDI)
//#### C        =          T =    UQ + ((IU, DU) OR DI) + (IU, DU)  =(UQIUIU, UQIUDU, UQDUIU, UQDUDU, UQDIIU, UQDIDU) OR (UQIUDIIU, UQIUDIDU, UQDUDIIU, UQDUDIDU)

public class PDE_PDS_DL {
	public String initonPDSwithBYS(String initon, double bys, StringBuilder pDEKey, boolean isBys) {
		if(initon.equalsIgnoreCase("D")) {
			return "D";
		}
```

```
if(initon.equalsIgnoreCase("I")) {
	return "I";
}
if(initon.equalsIgnoreCase("U")) {
	if(!isBys) {
		if(Math.random()< 0.5) {
			pDEKey.append("0");
			return "I";
		}else {
			pDEKey.append("1");
			return "Q";
		}
	}else {
		if(Math.random()< bys) {
			pDEKey.append("0");
			return "I";
		}else {
			pDEKey.append("1");
			return "Q";
		}
	}
}
if(initon.equalsIgnoreCase("Q")) {
	return "Q";
}
if(initon.equalsIgnoreCase("C")) {
	return "DI";
}
if(initon.equalsIgnoreCase("E")) {
	if(!isBys) {
		if(Math.random()< 0.5) {
			pDEKey.append("0");
			return "IU";
		}else {
			pDEKey.append("1");
			return "DU";
		}
	}else {
		if(Math.random()< bys) {
			pDEKey.append("0");
```

```
                        return "IU";
                }else {
                        pDEKey.append("1");
                        return "DU";
                }
        }
}
//#### 4 =        H  =     (IU, DU) OR DI  =(IU, DU, DI) OR (IUDI, DUDI)
if(initon.equalsIgnoreCase("H")) {
        if(!isBys) {
                if(Math.random()< 0.5) {
                        pDEKey.append("0");
                        return "IUDI";
                }else {
                        pDEKey.append("1");
                        return "DUDI";
                }
        }else {
                if(Math.random()< bys) {
                        pDEKey.append("0");
                        return "IUDI";
                }else {
                        pDEKey.append("1");
                        return "DUDI";
                }
        }
}
//+- 符号见 FindOulerRing 函数 的 332行。
//#### 5          =                   HC =    ((IU, DU) OR DI) + DI
//=(IUDI, DUDI, DIDI) OR (IUDIDI, DUDIDI)
if(initon.equalsIgnoreCase("-")) {
        if(!isBys) {
                if(Math.random()< 0.5) {
                        pDEKey.append("0");
                        return "IUDIDI";
                }else {
                        pDEKey.append("1");
                        return "DUDIDI";
                }
```

```
		}else {
			if(Math.random()< bys) {
				pDEKey.append("0");
				return "IUDIDI";
			}else {
				pDEKey.append("1");
				return "DUDIDI";
			}
		}
	}
	//+- 符号见 FindOulerRing 函数 的 332行。
	//#### B =	HE =	((IU, DU) OR DI) + (IU, DU)
	//=(IUIU, IUDU, DUIU, DUDU, DIIU, DIDU) OR (IUDIIU, IUDIDU, DUDIIU, DUDIDU)
	if(initon.equalsIgnoreCase("+")) {
		if(!isBys) {
			if(Math.random()< 0.5) {
				pDEKey.append("0");
				if(Math.random()< 0.5) {
					pDEKey.append("0");
					return "IUDIIU";
				}else {
					pDEKey.append("1");
					return "IUDIDU";
				}
			}else {
				pDEKey.append("1");
				if(Math.random()< 0.5) {
					pDEKey.append("0");
					return "DUDIIU";
				}else {
					pDEKey.append("1");
					return "DUDIDU";
				}
			}
		}else {
			if(Math.random()< bys) {
				pDEKey.append("0");
				if(Math.random()< bys) {
					pDEKey.append("0");
```

```
                                return "IUDIIU";
                        }else {
                                pDEKey.append("1");
                                return "IUDIDU";
                        }
                }else {
                        pDEKey.append("1");
                        if(Math.random()< bys) {
                                pDEKey.append("0");
                                return "DUDIIU";
                        }else {
                                pDEKey.append("1");
                                return "DUDIDU";
                        }
                }
        }
}
if(initon.equalsIgnoreCase("V")) {
        return "UQ";
}
if(initon.equalsIgnoreCase("S")) {
        return "QI";
}
//#### 2 =        P  =      (IU, DU) + DI  =(IUDI, DUDI)
if(initon.equalsIgnoreCase("P")) {
        if(!isBys) {
                if(Math.random()< 0.5) {
                        pDEKey.append("0");
                        return "IUDI";
                }else {
                        pDEKey.append("1");
                        return "DUDI";
                }
        }else {
                if(Math.random()< bys) {
                        pDEKey.append("0");
                        return "IUDI";
                }else {
                        pDEKey.append("1");
                        return "DUDI";
```

```
                }
        }
}
//#### A =        O  =     (IU, DU) + QI  =(IUQI, DUQI)
if(initon.equalsIgnoreCase("O")) {
        if(!isBys) {
                if(Math.random()< 0.5) {
                        pDEKey.append("0");
                        return "IUQI";
                }else {
                        pDEKey.append("1");
                        return "DUQI";
                }
        }else {
                if(Math.random()< bys) {
                        pDEKey.append("0");
                        return "IUQI";
                }else {
                        pDEKey.append("1");
                        return "DUQI";
                }
        }
}
if(initon.equalsIgnoreCase("A")) {
        return "UQQI";
}
//#### 8 =        M  =     ((IU, DU) OR DI) + DI + QI
//=(IUDIQI, DUDIQI, DIDIQI) OR (IUDIDIQI, DUDIDIQI)
if(initon.equalsIgnoreCase("M")) {
        if(!isBys) {
                if(Math.random()< 0.5) {
                        pDEKey.append("0");
                        return "IUDIDIQI";
                }else {
                        pDEKey.append("1");
                        return "DUDIDIQI";
                }
        }else {
                if(Math.random()< bys) {
                        pDEKey.append("0");
```

```
                        return "IUDIDIQI";
                    }else {
                        pDEKey.append("1");
                        return "DUDIDIQI";
                    }
                }
            }
            //#### 6        =           X =    UQ + ((IU, DU) OR DI) + DI
            // =(UQIUDI, UQDUDI, UQDIDI) OR (UQIUDIDI, UQDUDIDI)
            if(initon.equalsIgnoreCase("X")) {
                if(!isBys) {
                    if(Math.random()< 0.5) {
                        pDEKey.append("0");
                        return "UQIUDIDI";
                    }else {
                        pDEKey.append("1");
                        return "UQDUDIDI";
                    }
                }else {
                    if(Math.random()< bys) {
                        pDEKey.append("0");
                        return "UQIUDIDI";
                    }else {
                        pDEKey.append("1");
                        return "UQDUDIDI";
                    }
                }
            }
            //#### C =      T =    UQ + ((IU, DU) OR DI) + (IU, DU)
            //=(UQIUIU, UQIUDU, UQDUIU, UQDUDU, UQDIIU, UQDIDU) OR (UQIUDIIU, UQIUDIDU,
UQDUDIIU, UQDUDIDU)
            if(initon.equalsIgnoreCase("T")) {
                if(!isBys) {
                    if(Math.random()< 0.5) {
                        pDEKey.append("0");
                        if(Math.random()< 0.5) {
                            pDEKey.append("0");
                            return "UQIUDIIU";
                        }else {
                            pDEKey.append("1");
```

```
                                return "UQIUDIDU";
                        }
                }else {
                        pDEKey.append("1");
                        if(Math.random()< 0.5) {
                                pDEKey.append("0");
                                return "UQDUDIIU";
                        }else {
                                pDEKey.append("1");
                                return "UQDUDIDU";
                        }
                }
        }else {
                if(Math.random()< bys) {
                        pDEKey.append("0");
                        if(Math.random()< bys) {
                                pDEKey.append("0");
                                return "UQIUDIIU";
                        }else {
                                pDEKey.append("1");
                                return "UQIUDIDU";
                        }
                }else {
                        pDEKey.append("1");
                        if(Math.random()< bys) {
                                pDEKey.append("0");
                                return "UQDUDIIU";
                        }else {
                                pDEKey.append("1");
                                return "UQDUDIDU";
                        }
                }
        }
        return null;
}
//IUQ D CEVS H POA -+ MXT
//反向排列后如下
//TXM +- AOP H SVEC D QUI
```

```
public String initonPDIwithBYS(String pds, double bys, StringBuilder pDEKey, boolean isBys) {
		pds= pds.replace("UQIUDIIU", "T");
		pds= pds.replace("UQIUDIDU", "T");
		pds= pds.replace("UQDUDIIU", "T");
		pds= pds.replace("UQDUDIDU", "T");
		pds= pds.replace("UQIUDIDI", "X");
		pds= pds.replace("UQDUDIDI", "X");
		pds= pds.replace("IUDIDIQI", "M");
		pds= pds.replace("DUDIDIQI", "M");
		pds= pds.replace("IUDIIU", "+");
		pds= pds.replace("IUDIDU", "+");
		pds= pds.replace("DUDIIU", "+");
		pds= pds.replace("DUDIDU", "+");
		pds= pds.replace("IUDIDI", "-");
		pds= pds.replace("DUDIDI", "-");
		pds= pds.replace("UQQI", "A");
		pds= pds.replace("IUQI", "O");
		pds= pds.replace("DUQI", "O");
		pds= pds.replace("IUDI", "P");
		pds= pds.replace("DUDI", "P");
		pds= pds.replace("IUDI", "H");
		pds= pds.replace("DUDI", "H");
		pds= pds.replace("QI", "S");
		pds= pds.replace("UQ", "V");
		pds= pds.replace("IU", "E");
		pds= pds.replace("DU", "E");
		pds= pds.replace("DI", "C");
		pds= pds.replace("D", "D");
		pds= pds.replace("Q", "Q");
		pds= pds.replace("U", "U");
		pds= pds.replace("I", "I");
		return pds;
}
//用于肽展公式逐级变换
//QUI  D  SVEC  H AOP +-  TXM
public String initonPDEwithBYS(String pds, double pDE_KEY_rate, StringBuilder pDEKey, boolean isBys) {
		if(!isBys) {
			pds= pds.replace("Q", "Q");
			pds= pds.replace("U", "D");
```

```
            pds= pds.replace("I", "I");
            pds= pds.replace("D", "D");
            pds= pds.replace("QI", "S");
            pds= pds.replace("UQ", "V");
            pds= pds.replace("IU", "E");
            pds= pds.replace("DU", "E");
            pds= pds.replace("DI", "C");
            pds= pds.replace("IUDI", "H");
            pds= pds.replace("DUDI", "H");
            pds= pds.replace("UQQI", "A");
            pds= pds.replace("IUQI", "O");
            pds= pds.replace("DUQI", "O");
            pds= pds.replace("IUDI", "P");
            pds= pds.replace("DUDI", "P");
            pds= pds.replace("IUDIIU", "+");
            pds= pds.replace("IUDIDU", "+");
            pds= pds.replace("DUDIIU", "+");
            pds= pds.replace("DUDIDU", "+");
            pds= pds.replace("IUDIDI", "-");
            pds= pds.replace("DUDIDI", "-");
            pds= pds.replace("UQIUDIIU", "T");
            pds= pds.replace("UQIUDIDU", "T");
            pds= pds.replace("UQDUDIIU", "T");
            pds= pds.replace("UQDUDIDU", "T");
            pds= pds.replace("UQIUDIDI", "X");
            pds= pds.replace("UQDUDIDI", "X");
            pds= pds.replace("IUDIDIQI", "M");
            pds= pds.replace("DUDIDIQI", "M");
            return pds;
    }
    pds= pds.replace("Q", "Q");
    pds= pds.replace("U", "U");
    pds= pds.replace("I", "I");
    pds= pds.replace("D", "D");
    pds= pds.replace("QI", "S");
    pds= pds.replace("UQ", "V");
    if(Math.random()<pDE_KEY_rate) {
            pds= pds.replace("IU", "E");
    }else {
            pds= pds.replace("DU", "E");
```

```
}
pds= pds.replace("DI", "C");
if(Math.random()< pDE_KEY_rate) {
	pds= pds.replace("IUDI", "H");
}else {
	pds= pds.replace("DUDI", "H");
}
pds= pds.replace("UQQI", "A");

if(Math.random()<pDE_KEY_rate) {
	pds= pds.replace("IUQI", "O");
}else {
	pds= pds.replace("DUQI", "O");
}
if(Math.random()< pDE_KEY_rate) {
	pds= pds.replace("IUDI", "P");
}else {
	pds= pds.replace("DUDI", "P");
}
if(Math.random()< pDE_KEY_rate) {
	if(Math.random()<pDE_KEY_rate) {
		pds= pds.replace("IUDIIU", "+");
	}else {
		pds= pds.replace("IUDIDU", "+");
	}
}else {
	if(Math.random()< pDE_KEY_rate) {
		pds= pds.replace("DUDIIU", "+");
	}else {
		pds= pds.replace("DUDIDU", "+");
	}
}
if(Math.random()< pDE_KEY_rate) {
	pds= pds.replace("IUDIDI", "-");
}else {
	pds= pds.replace("DUDIDI", "-");
}
if(Math.random()< pDE_KEY_rate) {
	if(Math.random()< pDE_KEY_rate) {
		pds= pds.replace("UQIUDIIU", "T");
```

```
                }else {
                        pds= pds.replace("UQIUDIDU", "T");
                }
        }else {
                if(Math.random()< pDE_KEY_rate) {
                        pds= pds.replace("UQDUDIIU", "T");
                }else {
                        pds= pds.replace("UQDUDIDU", "T");
                }
        }
        if(Math.random()< pDE_KEY_rate) {
                pds= pds.replace("UQIUDIDI", "X");
        }else {
                pds= pds.replace("UQDUDIDI", "X");
        }

        if(Math.random()<pDE_KEY_rate) {
                pds= pds.replace("IUDIDIQI", "M");
        }else {
                pds= pds.replace("DUDIDIQI", "M");
        }
        return pds;
    }
    //融合肽展公式，离散数学和数字逻辑 的元基变换
    //      pds= pds.replace("UQ", "V");
    //      pds= pds.replace("DI", "C");
    //      pds= pds.replace("IQ", "S");
    //      pds= pds.replace("VS", "A");
    //      pds= pds.replace("ES", "O");
    //      pds= pds.replace("EC", "P");
    //      pds= pds.replace("CS", "M");
    //      pds= pds.replace("VE", "T");
    //      pds= pds.replace("VC", "X");
    // 我的思维逻辑是先将PDS的数字逻辑和离散数学归纳识别，然后走肽展识别，最大缩短元基长度
    public String initonPDE_DCDLwithBYS(String pds, double pDE_KEY_rate, StringBuilder pDEKey, boolean isBys) {
        if(!isBys) {
                pds= pds.replace("UQIUDIIU", "T");
                pds= pds.replace("UQIUDIDU", "T");
```

```
            pds= pds.replace("UQDUDIIU", "T");
            pds= pds.replace("UQDUDIDU", "T");
            pds= pds.replace("UQIUDIDI", "X");
            pds= pds.replace("UQDUDIDI", "X");
            pds= pds.replace("IUDIDIQI", "M");
            pds= pds.replace("DUDIDIQI", "M");
            pds= pds.replace("IUDIIU", "+");
            pds= pds.replace("IUDIDU", "+");
            pds= pds.replace("DUDIIU", "+");
            pds= pds.replace("DUDIDU", "+");
            pds= pds.replace("IUDIDI", "-");
            pds= pds.replace("DUDIDI", "-");
            pds= pds.replace("IUDI", "H");
            pds= pds.replace("DUDI", "H");
            pds= pds.replace("UQQI", "A");
            pds= pds.replace("IUQI", "O");
            pds= pds.replace("DUQI", "O");
            pds= pds.replace("IUDI", "P");
            pds= pds.replace("DUDI", "P");
            pds= pds.replace("QI", "S");
            pds= pds.replace("UQ", "V");
            pds= pds.replace("IU", "E");
            pds= pds.replace("DU", "E");
            pds= pds.replace("DI", "C");
            pds= pds.replace("Q", "Q");
            pds= pds.replace("U", "D");
            pds= pds.replace("I", "I");
            pds= pds.replace("D", "D");
            //PDE
            pds= pds.replace("VS", "A");
            pds= pds.replace("ES", "O");
            pds= pds.replace("EC", "P");
            pds= pds.replace("HE", "+");
            pds= pds.replace("HC", "-");
            pds= pds.replace("VE", "T");
            pds= pds.replace("VC", "X");
            pds= pds.replace("CS", "M");
            return pds;
    }
    if(Math.random()< pDE_KEY_rate) {
```

```
		if(Math.random()< pDE_KEY_rate) {
			pds= pds.replace("UQIUDIIU", "T");
		}else {
			pds= pds.replace("UQIUDIDU", "T");
		}
	}else {
		if(Math.random()< pDE_KEY_rate) {
			pds= pds.replace("UQDUDIIU", "T");
		}else {
			pds= pds.replace("UQDUDIDU", "T");
		}
	}
	if(Math.random()< pDE_KEY_rate) {
		pds= pds.replace("UQIUDIDI", "X");
	}else {
		pds= pds.replace("UQDUDIDI", "X");
	}
	if(Math.random()<pDE_KEY_rate) {
		pds= pds.replace("IUDIDIQI", "M");
	}else {
		pds= pds.replace("DUDIDIQI", "M");
	}
	if(Math.random()< pDE_KEY_rate) {
		if(Math.random()<pDE_KEY_rate) {
			pds= pds.replace("IUDIIU", "+");
		}else {
			pds= pds.replace("IUDIDU", "+");
		}
	}else {
		if(Math.random()< pDE_KEY_rate) {
			pds= pds.replace("DUDIIU", "+");
		}else {
			pds= pds.replace("DUDIDU", "+");
		}
	}
	if(Math.random()< pDE_KEY_rate) {
		pds= pds.replace("IUDIDI", "-");
	}else {
		pds= pds.replace("DUDIDI", "-");
	}
```

```
        if(Math.random()< pDE_KEY_rate) {
            pds= pds.replace("IUDI", "H");
        }else {
            pds= pds.replace("DUDI", "H");
        }
        pds= pds.replace("UQQI", "A");
        if(Math.random()<pDE_KEY_rate) {
            pds= pds.replace("IUQI", "O");
        }else {
            pds= pds.replace("DUQI", "O");
        }
        if(Math.random()< pDE_KEY_rate) {
            pds= pds.replace("IUDI", "P");
        }else {
            pds= pds.replace("DUDI", "P");
        }
        pds= pds.replace("QI", "S");
        pds= pds.replace("UQ", "V");
        if(Math.random()<pDE_KEY_rate) {
            pds= pds.replace("IU", "E");
        }else {
            pds= pds.replace("DU", "E");
        }
        pds= pds.replace("DI", "C");
        pds= pds.replace("D", "D");
        pds= pds.replace("Q", "Q");
        pds= pds.replace("U", "U");
        pds= pds.replace("I", "I");
        //PDE
        pds= pds.replace("VS", "A");
        pds= pds.replace("ES", "O");
        pds= pds.replace("EC", "P");
        pds= pds.replace("HE", "+");
        pds= pds.replace("HC", "-");
        pds= pds.replace("VE", "T");
        pds= pds.replace("VC", "X");
        pds= pds.replace("CS", "M");
        return pds;
    }
}
```