**第一章\_德塔自然语言图灵系统**

**测试速度：**单机联想Y7000笔记本win10 实测峰值每秒 中文分词1630~1650万+中文字， 词库65000+，函数准确率100%，缺失语法函数 0.3%-， 算法准确率 99.7%+， 100%完整开放源码，在api与书籍中。

**测试效果：输入：**如果从容易开始于是从容不迫天下等于是非常识时务必为俊杰沿海南方向逃跑他说的确实在理结婚的和尚未结婚的提高产品质量中外科学名著内科学是临床医学的基础内科学作为临床医学的基础学科重点论述人体各个系统各种疾病的病因发病机制临床表现诊断治疗与预防

**输出结果：**如果+从+容易+开始+于是+从容不迫+天下+等于+是非+常识+时务+必+为+俊杰+沿海+南+方向+逃跑+他+说+的+确实+在理+结婚+的+和+尚未+结婚+的+提高+产品质量+中外+科学+名著+内科学+是+临床+医学+的+基础+内科学+作为+临床+医学+的+基础+学科+重点+论述+人体+各个+系+统+各种+疾病+的+病因+发病+机制+临床+表现+诊断+治疗+与+预防+++++



定义：德塔分词是一种-- 基于神经网络索引字典切割-- 进行前序遍历词性组合匹配-- 按文学语法定义搭配 的切词引擎。

**德塔分词的催化切词优化方式主要包含：**

1 索引字典进行细化拆分加速。

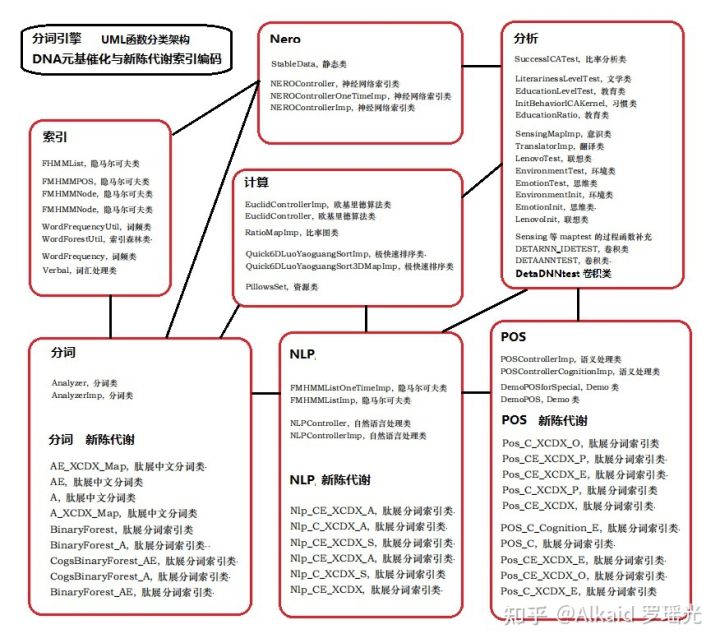
2 函数进行使用频率统计排列加速优化。

3 动态类卷积遍历内核的关键字优化。

4 函数文件和 函数文件名 进行新陈代谢，二次新陈代谢优化索引编码加速。

5 文学切词语法函数的细化优化加速。

定义者 罗瑶光



**分词，**

1 德塔的分词是一种前序《排队论》逐字遍历文字索引，通过索引中的词汇匹配 按长度进行提取，然后将提取的词汇串 进行词性切分的过程。refer page 12 ~

2 德塔的分词文字索引采用关联分类生成小文件map集（词性map，词长map，词类map）， 进行整体加速，作为一个催化细化过程。refer page 44，54, 92，

3 德塔的词汇匹配目前有多个国家语言字符集，可统一，可拆分，目前最大划分处理长度为4，划分切词采用动态 类似CNN 卷积（遍历pos函数语句的内核计算，非卷积的积分叠加计算） StringBuilder核做POS识别。refer page 45，119,120，

4 德塔的词性切分按照4字词 3字词 2字词 单字 进行逐级按词汇的 POS搭配语法模式进行归纳，按文本的POS出现频率进行流水阀门方式优化。refer page 97,116，



（德塔分词逻辑， 已经纠正红色字 ‘卷积’改为‘内核’，因为第四修订版本已经在申请中，ppt所有书中的原图纠正内容统一更新在第5版，罗瑶光）

**排序，**

1 德塔分词排序思想原型采用 Sir Charles Antony Richard Hoare 的 快速排序思想。

refer page 版权原因无文字收录 已经refer [快速排序算法\_百度百科](https://link.zhihu.com/?target=https://baike.baidu.com/item/%E5%BF%AB%E9%80%9F%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95/369842?fr=aladdin" \t "https://zhuanlan.zhihu.com/p/_blank)

2 德塔分词排序源码原型采用 Introduction to Algorithms 的 快速排序4代源码。

refer page 版权原因无源码收录 已经refer [https://github.com/yaoguangluo/Data\_Processor/blob/master/DP/sortProcessor/Quick\_4D\_Sort.java](https://link.zhihu.com/?target=https://github.com/yaoguangluo/Data_Processor/blob/master/DP/sortProcessor/Quick_4D_Sort.java" \t "https://zhuanlan.zhihu.com/p/_blank)

3 基于1 和 2原型，德塔分词排序 采用 Theory on YAOGUANG's Array Split Peak Defect 的微分催化算子优化思想 2013年开始优化。refer page 247,248,250,529,620，

4 优化过程为 小高峰左右比对法， 波动算子过滤思想，离散条件归纳微分思想(如狄摩根计算，流水阀门计算等)，目前为TopSort5D。refer page 658，下册134

5 德塔分词的函数优化方式和算法优化方式，包括分词引擎，读心术，NLP分析等核心组件均采用 微分催化系统。 refer page 661，

**神经网络索引，**

1 德塔分词的词汇字典用map进行索引，因为jdk8+的map对象的key支持2分搜索，搜索速度到了峰值。refer page,129,131

2 德塔分词的索引不断的将大map进行细化分类，如词长map，词类map，词性map，让搜索再次加速。refer page 55，

3 德塔分词的索引map支持 2次组合计算，支持分布式服务器进行索引cache。关于2次组合计算作者不建议单机使用。refer page 92，

4 德塔分词map的key用string的 char对应ASCII int进行标识来执行find key，方便二分搜索存储和 StringBuilder高速计算，实现底层核统一。refer page 92

**分词在线性文本搜索中应用，**

1 德塔分词的搜索建立在map类的权重计算方法上，不同的权重叠加产生的打分进行排序输出。refer page 下册64

2 权重的计算方法按词性的主谓宾如代 名动形 ，和 POS如 动名形谓介分类。refer page 下册66

3 权重与词长，词频进行耦合bit叠加计算(bit位计算比乘法要快一个数量级)，生成最终输出结果。 refer page 下册68

4 权重与词长的 比值可以精度调节，确定搜索的精确性和记录个人搜索偏好。refer page 下册68

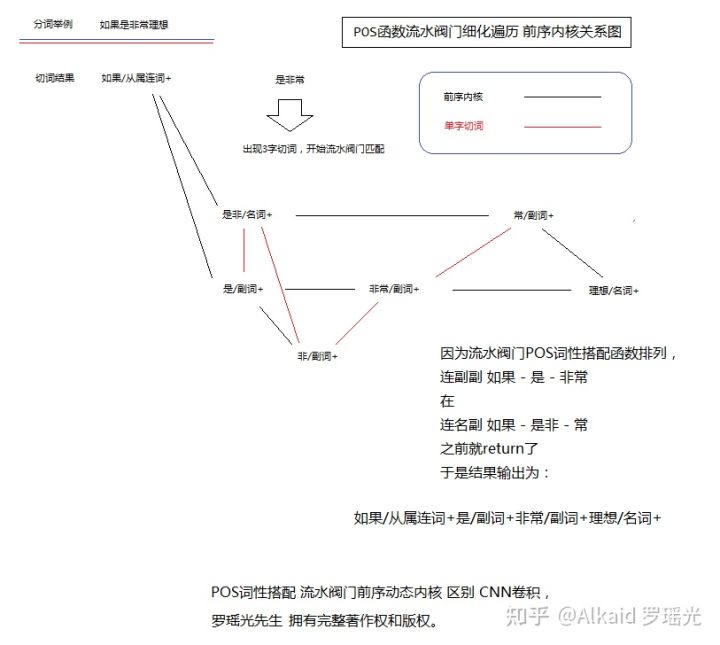
**动态 POS函数流水阀门细化遍历 内核匹配，**

1 动态的核分为前序核和后序核两种。根据词汇分析的位置进行实时变动更新。refer page 97

2 前序核主要缓存存储词汇的位置和词性，用于POS词性搭配的 POS函数流水阀门细化遍历 计算。refer page 97

3 后序核主要缓存词汇的切词链 后面准备 跟进的词语。用于POS语法的修正计算，如连词匹配。refer page 97

4 内核采用StringBuilder做核载体进行计算加速。refer page 97

2019年3月18日之前作者Github的 该算法函数编码框架已经出现

[https://github.com/yaoguangluo/Deta\_Parser/commit/25b90c9847d15df85c5c991448f2c271e0ad8106](https://link.zhihu.com/?target=https://github.com/yaoguangluo/Deta_Parser/commit/25b90c9847d15df85c5c991448f2c271e0ad8106" \t "https://zhuanlan.zhihu.com/p/_blank)

注意：链接的**CNN 关键词**的 历史记录 属于作者用词错误，作者当年基础学术累积不够，关于卷积的知识仅仅学了**计算机视觉**的理论课，以为带内核计算的都叫CNN卷积，

另外作者发现自己还有一个错误， 就是以为序列链表方式计算就叫隐马科夫链计算。所以**CNN+隐马可夫**这两个技术词汇，伴随作者10年之久。今天进行ppt严谨定义，翻阅大量定义文献资料，才发现这些错误。予以纠正。作者的ANN和RNN 出现的文本分析内核计算才是真正的CNN卷积计算。

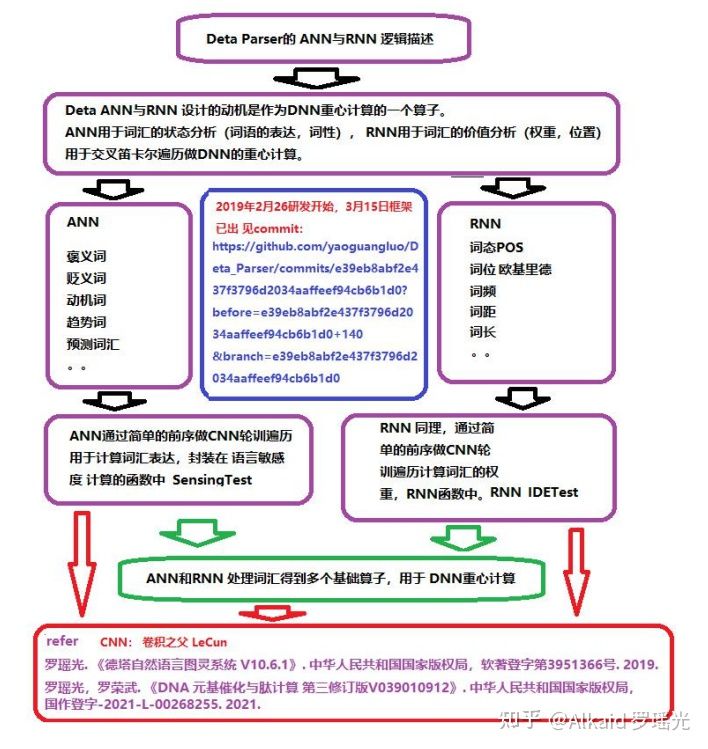
**POS,**

1 德塔分词的核心类，包含了词性的搭配切分所有函数。refer page 97,116



**NLP,**

1 德塔分词的核心类，包含了词性的词长切分所有函数。refer page 119，120



**ANN,**

**德塔词性的卷积计算ANN，主要包含意识比率算子,环境比率算子,动机比率算子,情绪比率算子。这个四个算子 的组合计算产生了一些高级决策,如 情感比重,动机比重,词权比重,持续度,趋势比重,预测比重,猜想比重,意识综合。这些决策在文本分析的领域可以拥有实际评估和决策的价值。同时意识综合 summing 也是德塔DNN计算的一个输入参数组件，用于文本中心思想词汇标识计算。**

1词性卷积计算refer page 182

2用于确定文本的中心

2.1 算子组成

2.1.1 S SENSING 意识比率

2.1.2 E ENVIRONMENT 环境比率

2.1.3 M MOTIVATION 动机比率

2.1.4 E EMOTION 情绪比率

refer page 18

**RNN,**

**德塔的词位卷积计算RNN, 主要包含词性比率, 词距比率算子和欧基里德熵算子。这三个算子主要用于求解 POS距离, COVEX距离, EUCLID距离.这些权距 在一篇文章中能够很清楚的计算每一个词汇的使用度，出现的价值，和应用频率以及分布规律。用于文本的主要描述语句的重心所在位置计算。**

1词位卷积计算refer page 178

2用于确定文本的重心

2.1 算子组成

2.1.1 P POS 词性比率

2.1.2 C CORRELATION 词距比率

2.1.3 E E-DISTANCE 欧基里德熵

refer page 18

**DNN,**

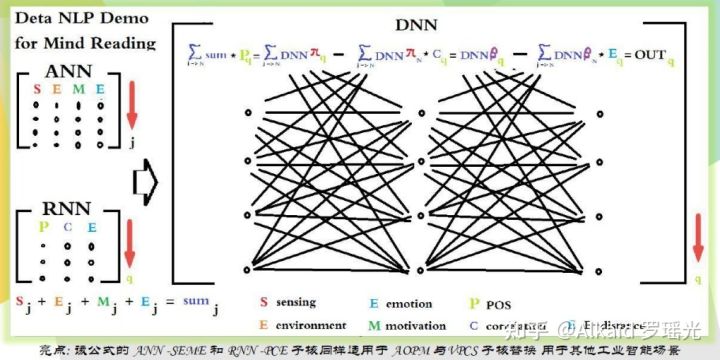
**德塔的词汇深度计算 可以理解为 德塔词性的卷积计算ANN 与 德塔的词位卷积计算RNN 的前序笛卡尔卷积计算。因为参数 由 文章中心思想 和 文章的重心词位 两类组成，因此适用于分析和计算 文章的 核心思想词汇的价值**

1词汇深度计算refer page 183

2用于确定文本的核心

2.1 深度计算 (ANN sum核 -> RNN PCE)

refer page 18



**图灵机，**

1 文学分析refer page 168



德塔文学分析主要用于文章的思想分析和挖掘，如确定多语意识的场景，当时的环境，动机，意识形态倾向和决策思维表达等。（**多语意识 ：通过人物的对话方式，语言特征，模式场景等因素 来 分析当时的人文情感，大众思想，从而了解所处时代的民族风情，社会建筑，时代背景。 教授人：作者导师白育芳，2007年，总参解放军炮兵学院南京分院。**）

**2 作品评估refer page 167**

德塔作品评估 可理解为教育程度评估，如语法，词汇的词性统计，专业词汇的统计，成语，三字词的词长词汇的统计，等等。如一个句子中含有的高级词汇的比率，4字名词的比率，形容词的比率。（作者最早意识出现在**2009年** 在上海章鑫杰那 处理**法国ESIEE亚眠大学**的法语邮件项目， **Pascal教授**曾传授作者关于**FLECH**法语**元音**比重**单词**分析的表述。设计这个项目，进行了灵感发散。**德塔图灵分词全文没有任何单词分析和 非中文的语言分析，不涉及flech任何思想和逻辑，因此一直没有refer**。 作者拥有完整著作权和版权）

**3 动机分析refer page 169**

德塔动机分析 基于动机词典的map key匹配 进行决策表达。比较简单。因为词典定义 带有作者个人主观思维特征。所以没有太多描述。

适用3，4，5

**4 情感分析refer page 159**

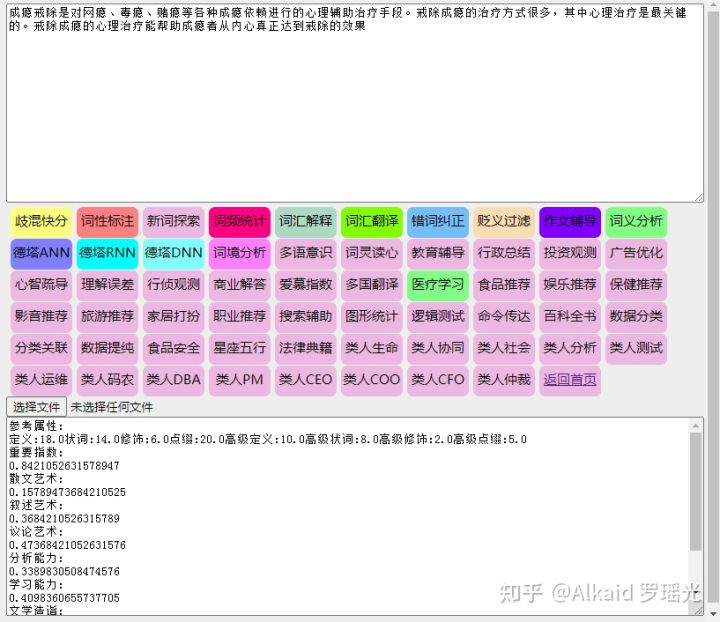
**德塔情感分析 基于 褒义词 贬义词 和中性词 的 map key匹配 进行决策表达。比较简单。因为词典定义 带有作者个人主观思维特征。所以没有太多描述。**

**5 习惯分析refer page 169**

**德塔习惯分析 基于 褒义词 贬义词 和中性词，动机词， 文学分析数据，作品评估比率，教育程度等数据 的全文比重，来确定一个人写作特征，和写作习惯。写作风格。因为词典定义 带有作者个人主观思维特征。所以没有太多描述。**

**6 教育程度评估refer page 168**

德塔教育程度评估体现在文章中的（**有效词汇**如词长超过2位）的 （**有价值词汇**如名动形谓状）的全文，全句，其它POS词性的比率来确定文章的句法特征。举个简单的例子，一个句子中有效有价值的形容词比重大的文章通常代表作者的分析表达和散文修饰能力比较强势。，思维来自作者初中语文学习。



应用

第二章 Java 数据分析算法引擎系统

**定义： 微分催化排序 一般指 将传统的排序在数据排列计算过程中 进行 内存峰值波动平均，计算逻辑减少，计算算子减少，计算条件减少，计算的频率减少，计算关系减少的催化过程。**

价值是方便函数 元基索引 和 新陈代谢，二次新陈代谢。（见之后的 象契排序算法的 新陈代谢催化优化实例）

内存峰值波动平均，(见小高峰过滤左右比对算法)

计算逻辑减少，(见比较函数的 缩进优化)

计算算子减少，(见增序，与减序替换)

计算条件减少，(见狄摩根离散条件or变换)

计算的频率减少，(见选择排序的小于deep的堆栈检测替换，和阀门逻辑序列频率统计 代码排列优化)

计算关系减少 (见算子减少和条件减少的相互关系优化)

的催化过程。

定义人 罗瑶光

函数集合

1 德塔的数据分析包，最早是作者在大学的处理 Rohini教授的 C语言数据结构《Data Structure》 和 Renhart教授 计算机视觉卷积的《Computer Vision》课后作业。

《Data Structure》refer page 226,230,235,238,253,作者没有把当年的计算器四则运算器和rotation tree等作业算法归纳在该作品中。

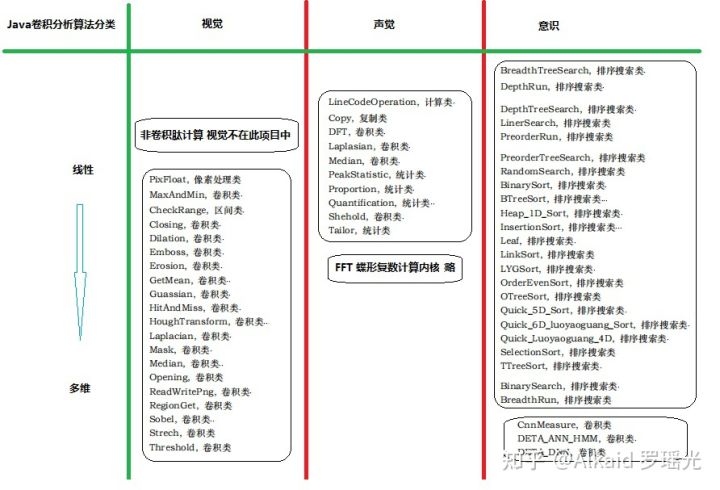
《Computer Vision》refer page 202,204,205,206,209,211,212,213,214,217,218,220,221，259,260，

2 德塔的卷积在2013年后不断的完善，发现其在仿生听觉和视觉计算中都能进行系统的应用，于是开始优化。refer page 191

作者一开始设计卷积是路德大学图片上的应用，2013年，当ETL设计成了节点处理图片像素后，作者开始设计声音 java sound API的处理， 2014年。这个引擎逐渐在计算机仿生系统中进行集成应用。论证了其在具体应用工程中的实践价值。作者当时设计了主要用来测测作者自己的心跳。

3 优化方式为将计算函数进行插件接口模式封装成jar，方便上层调用。refer page 190

4 封装的过程中，不断的进行细化优化，衍生出多个辅助计算函数集，如催化排序，仿生滤波。 refer page 247,655

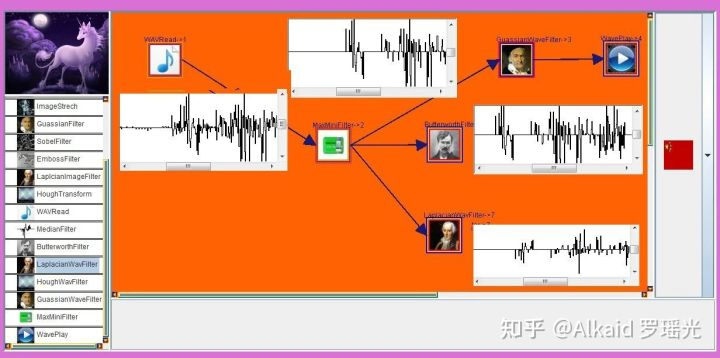
UML

**线性，**

1 德塔的数据分析包 包含array的线性排序处理 refer page 见排序

2 德塔的数据分析包 包含array的线性卷积处理 refer page 见卷积

3 ANN RNN DNN 线性深度卷积计算处理 refer page 222,223,223,



**非线性，**

1 德塔的数据分析包 包含图论的非线性广度建模 refer page 226,230

2 德塔的数据分析包 包含图论的非线性深度建模 refer page 230,232

3 德塔的数据分析包 包含图论的非线性树建模 refer page 236,243,253

**维度，**

1 德塔的数据分析包 包含1维 语音数组计算实例 refer page 见智能声诊

2 德塔的数据分析包 包含2维 图片卷积计算实例 refer page 见智能相诊

3 德塔的数据分析包 包含3维 数据循环阶计算实例 refer page 见噪音识别，三阶傅里叶应用，animation等



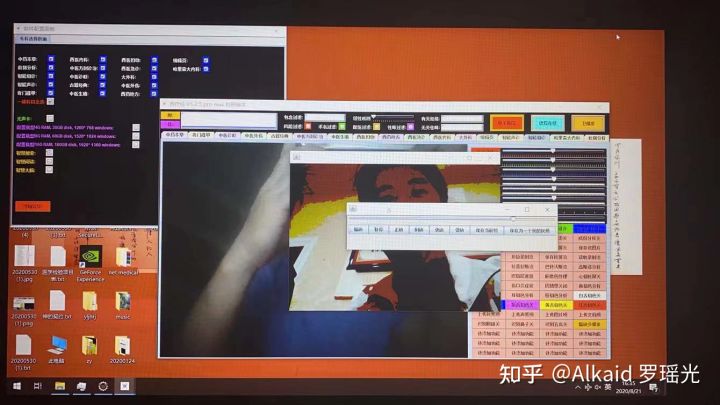
**德塔三阶傅里叶计算定义：一般指将线性的时序语音波进行傅里叶变换，此时的波为 频率域波，通过简单的噪声频率过滤后，让后再进行第二次傅里叶变换。于是输出的时序波结果会非常的均匀和格式化，产生优美的平滑间隔峰区间，于是将此时序波第三次傅里叶变换，再次得到的频率波输出具有明确的间隔峰区间生物特征标记。用于德塔语音识别。**

**定义人 罗瑶光**

**场景，**

1 图片的操作。refer page 214

2 像素的操作。refer page 见视觉



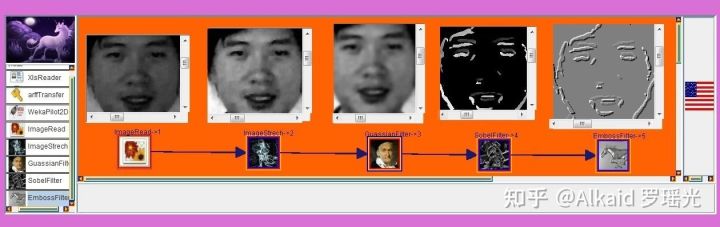
3 文件的存储。refer page 214

4 语音的处理。refer page 见听觉

**仿生听觉，**

1 滤噪计算 高斯1D，median refer page 206, 213, 260

2 频率变换 傅里叶， 快速傅里叶 refer page 258



**视觉，**

1 德塔的视觉主要包含常见2维卷积滤波函数。refer page

2 边缘计算 索贝尔凸蚀，索贝尔梯度，索贝尔向量，拉普拉斯refer page 218, 212

3 凹度计算 emboss浮雕，索贝尔mask，refer page 204,

4 频率计算 傅里叶时序域，傅里叶频率域，哈尔计算，refer page 258, 211,

5 腐蚀计算 膨胀计算，侵蚀计算，均值计算，高斯计算1D一字， 高斯2D十字。refer page 202, 204, 205, 206



**排序，**

1 德塔的排序作者早期2009年设计《算法导论》 黑皮书 ，北邮出版社有其 数据结构 影印教材 的 快速排序4代，进行了10年优化，refer page [https://github.com/yaoguangluo/Data\_Processor/blob/master/DP/sortProcessor/Quick\_4D\_Sort.java](https://link.zhihu.com/?target=https://github.com/yaoguangluo/Data_Processor/blob/master/DP/sortProcessor/Quick_4D_Sort.java" \t "https://zhuanlan.zhihu.com/p/_blank)

2 优化过程归纳，逐渐的形成了一个微分催化排序体系。refer page 247,248,250, 658，下册134，

3 左右比对算法优化，小高峰过滤优化，缺陷峰归纳，催化算子优化，离散逻辑优化。refer page 658，下册134，

**左右比对算法优化，**一般指在不对称的数列中，为了寻找对称性观测面，作者设计了一种比较简单的方法，如将数列逐渐拆分， 取出拆分后的小数列的初值和尾值进行比较，作为一个参照点，用于躲避计算高峰。测试发现具有强大的实用性。

**小高峰过滤优化，**一般指为了躲避内存计算高峰而导致的延迟，卡顿，死锁，堆栈溢出等问题 而设计的一类高效率算法集合。

**缺陷峰归纳，**一般指计算数列在不断的拆分中的中值基偶问题导致了变量，算子，函数的使用频率不对称而出现的一系列蝴蝶效应问题集的归纳。

**催化算子优化，**一般指 计算的中间过程中 因 变量，算子，函数的使用频率 不对称，不稳定导致的各种问题 ，为了解决这类问题而 进行的不断 的对 变量，算子，函数优化与校正过程。

**离散逻辑优化，**一般指 对 变量，算子，函数优化与校正过程中 通过离散数学， 迪摩根定律，等客观存在的逻辑定律进行 不断优化与校正过程。

**定义人 罗瑶光**

4 目前代表作为TopSort5D 极速催化排序。refer page 下册134

**搜索，**

1 德塔的搜索计算主要做一个编码参照，没有工程用途。refer page 226

2 编码参照有利于作者在设计图论计算和非线性搜索时候发散思维用途。refer page

3 编码开始于作者2009年 完成 Rohini教授布置的作业。refer page 我 qq 313699483 有完整作业备份日记。

4 对作者研究Hash空间 有发散思维的用途，如作者数据预测包设计的辅助。refer page

**应用**

1 TopSort5D 包含深度算子，包含广度算子，包含滤波算子。refer page 下册134

TopSort5D 版权源码

本人调通的算法导论的quicksort4D算法链接如下，可直接区别，再次Refer 快速排序之父 霍尔先生：

[https://github.com/yaoguangluo/Data\_Processor/blob/master/DP/sortProcessor/Quick\_4D\_Sort.java](https://link.zhihu.com/?target=https://github.com/yaoguangluo/Data_Processor/blob/master/DP/sortProcessor/Quick_4D_Sort.java" \t "https://zhuanlan.zhihu.com/p/_blank)

2 索贝尔 dir 向量差 区别三维的立体面特征趋势。refer page 219

3 噪音识别。refer page 720

4 小波分离。refer page 不在此章 涉及鸡尾酒调度，被略去先

5 极速商旅TSP。refer page 538,541,547

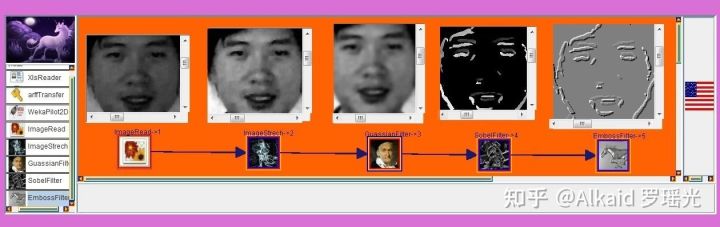
6 股票数据抓取 refer page 不在此章， 261, 263, 264, 266可以处理 股票数据线波。

**涉及著作权文件：**

第三章 德塔 ETL 人工智能可视化数据流分析引擎系统.



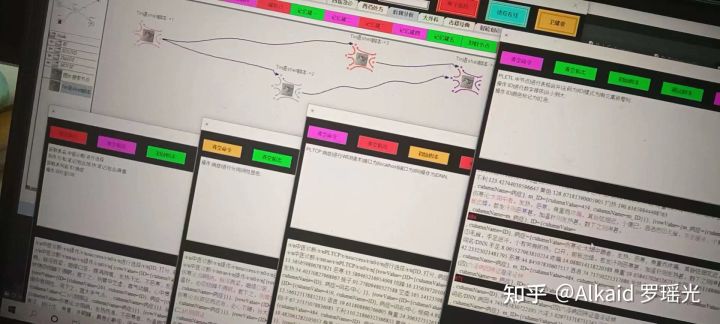
1 德塔ETL 又叫 ETL UNICRON, 是一个数据节点流计算的可视化操作工具.refer page 267



2 最早由作者在路德大学设计java卷积视觉包，为了方便 像素矩阵流的流水观测。refer page 186

3 德塔ETL采用APPLET,可以嵌入在网页上作为 rich web架构，与flash应用相似。refer page 287

4 德塔ETL已成为当前的PLETL,和元基花模拟神经元计算的基础组件。refer page 774



**界面，**

1 德塔ETL 的界面采用Jsplitpane分区 主要包含节点显示树区，节点画布操作区，计算状态反馈区 和系统配置区。refer page 286~

2 节点的显示区 采用Jtree进行鼠标操作，左键选择，右键弹框。refer page 286~

3 画布操作区 的 节点采用画线描点实现，操作为左键拖拽，右键连线和弹框。refer page 311~

早期的节点处理界面弹框 设计成inner弹框模式作者发现关闭按钮被屏蔽了，于是就改成frame组件跳出canvas画布来显示节点处理界面。

4 状态反馈与系统配置区用于实时了解ETL的运行状态。refer page 见jtextPanel

**最新 养疗经 版本，作者将状态数据的jtextpanel 写在了元基枝全局变量进行集成，如果要单独将etl做插件使用，要区别这个 函数，可以改写下。**

**皮肤，**

1 德塔ETL 皮肤采用bitmap实现，可以自由替换。refer page 333,334

（**之后我会进行代码的优化优化， 将 bitmap的表达形式 array[][] swap成 array[]+array[]，这样50\*50的界面 含有200个点， 2500次遍历 就变成了200次， 界面刷新时间缩减到原来的十分之一甚至更短。罗瑶光**）

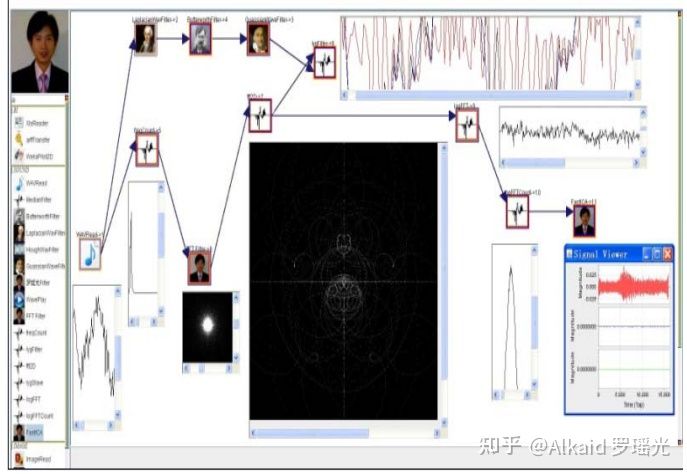
**写清楚点， array[10行][10位]= 1 or true 如果坐标矩阵第10行第10位 是像素显示标记1， 那么可以swap成 array[0]=10行， + array[0]=10位， 这样极大的减少计算array的内存buffer占用。再举例 array[13行][9位]= 1 or true 如果坐标矩阵第13行第9位 是像素显示标记1， 那么可以接下来swap成 array[1]=13行， + array[1]=9位，方便理解。**

2 德塔ETL 皮肤包含指标，箭头，连线，节点外形，控间外形设计。refer page 332,319~

3 德塔ETL 的控件采用jdk的组件 componient进行2次开发。refer page 334~

4 德塔ETL 的引擎界面的描点画线基于AWT Swing的canvas 画布系统实现。refer page 311~

另外德塔早期曾用界面 LYG-AI 如下：



**流存储，**

1 德塔ETL的流存储是一个object类，可以包含多种状态。refer page 329~332

2 德塔ETL的流采用单例的this.clone,确定了中间态存储模式。refer page 见 节点 clone()

3 德塔ETL的流可保存，可观测，但不可逆。refer page 329,330

**节点，**

1 德塔ETL的节点作为一个计算单元，模拟神经元作为最小单位计算。refer page 327~

2 德塔ETL的节点存在UI多样化，可界面设计如orange， knime，weka那样。refer page 335

3 德塔ETL的节点也可以语言化，如PLETL的语句IO模式研发设计。refer page 774~,790

4 德塔ETL的节点大体为从左到右的IO模式，节点流计算目前可循环但不可逆。refer page 329~332

**[德塔可视化ETL操作例子](https://www.zhihu.com/zvideo/1483183087273684992" \t "https://zhuanlan.zhihu.com/p/_blank)**

[](https://www.zhihu.com/zvideo/1483183087273684992" \t "https://zhuanlan.zhihu.com/p/_blank)

[Alkaid 罗瑶光的视频](https://www.zhihu.com/zvideo/1483183087273684992" \t "https://zhuanlan.zhihu.com/p/_blank)

[· 11 播放](https://www.zhihu.com/zvideo/1483183087273684992" \t "https://zhuanlan.zhihu.com/p/_blank)

**插件，**

1 德塔ETL的插件类似OSGI的jar开发模式。严谨的说只是继承的classloader模式。refer page 286,290,777

2 德塔ETL的插件可以加元基标识认证组件，避免错误插件扩展。refer page 290,777,779

3 德塔ETL目前支持插件进行平台配置，页面扩展和节点扩展。refer page 286,290

4 德塔ETL目前的3元基文件名索引肽化支持插件的分类管理和加密标识。refer page 781

**档案，**

1 德塔ETL的档案包含节点流信息和节点配置信息。refer page 279,282

2 德塔ETL的存储采用节点的画布状态单例信息存储方式.refer page,279,,282

3 单例信息包含画布中节点的坐标，名称，ID，连线，配置信息等实体信息。refer page 282

4 德塔ETL的流存储用文件读写形式.etl后缀存储.refer page 282

5 德塔ETL的存储可支持加密和batch模式运行。refer page 养疗经应用略

**拓扑，**

1 德塔ETL的拓扑体现在节点的神经元模拟计算观测。refer page 273

2 节点的神经元模拟拓扑体现在从左到右的从高到低拓扑模式。refer page 273

3 第2点或许是个拓扑缺陷，但是却因此又确定了固有 的向量方位。refer page 273

**神经网络，**

1 德塔ETL的神经网络计算在流数据计算中有实际价值。refer page 274

2 德塔ETL神经网络在PLETL和tinshell中逐渐体现其更多学术价值。refer page 783

3 德塔ETL的向量拓扑模式确定了神经网络的加权方式。refer page 274

4 德塔ETL的神经网络可循环不可逆。refer page 274

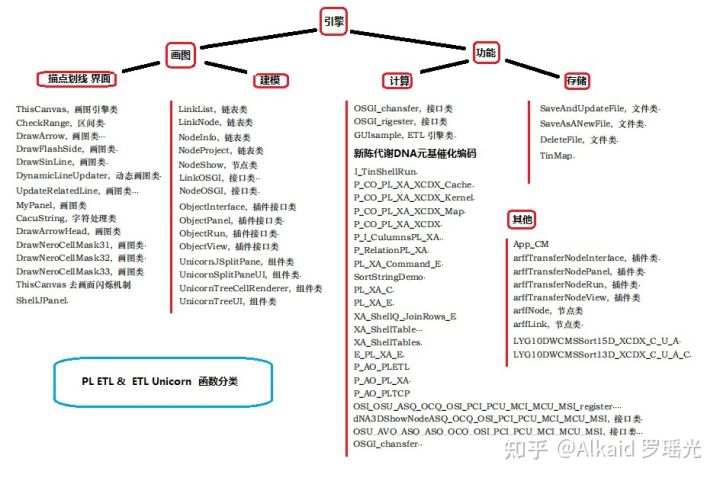
**一键执行，**

1 德塔ETL支持 一键保存 。refer page 277, 282

2 德塔ETL支持 一键读取。refer page 302

3 德塔ETL支持 一键执行。refer page 308

德塔的一键执行最早作者想设计成Knime的那种batch的批处理脚本节点流那样，随着作者TinShell出来后，作者觉得批处理跨语言，即繁琐又没效率。于是想做成JAVA脚本语言，伴随着个思路，目前作者设计了TinShell语言和元基花语言。



第四章 德塔 Socket 流可编程数据库语言引擎系统

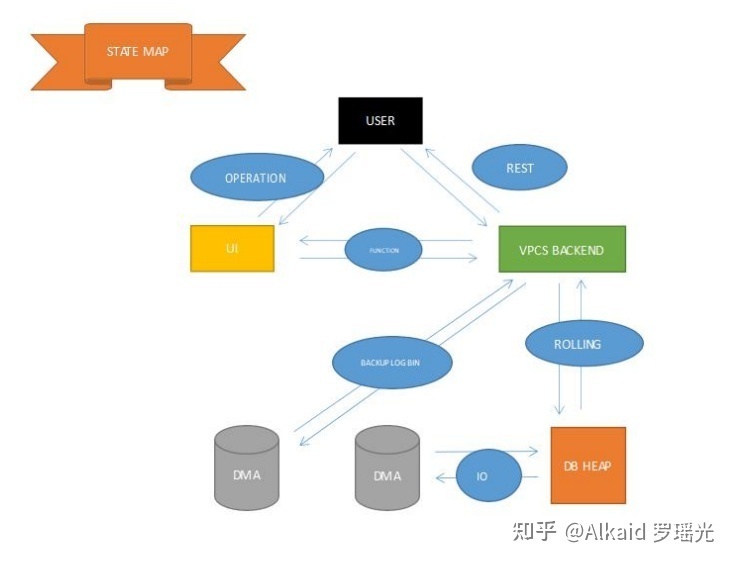
**Socket rest TCP握手协议，**

1 德塔数据库的 admin界面采用 web页进行配置操作。refer page 376

2 web页配置操作采用TCP握手访问模式，基于socket的http请求握手。refer page 464~

3 德塔数据库将socket握手进行线程封装，然后多线程组织页面。refer page 392，

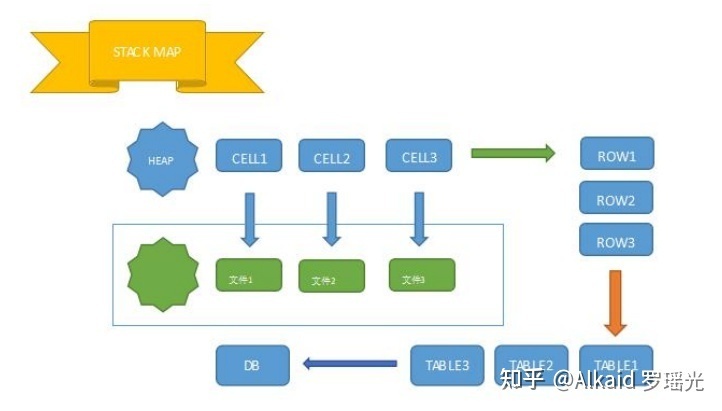
4 封装和组织页面设计过程逐步进行优化形成VPCS后端管理体系。refer page 383,476



**文件数据库，**

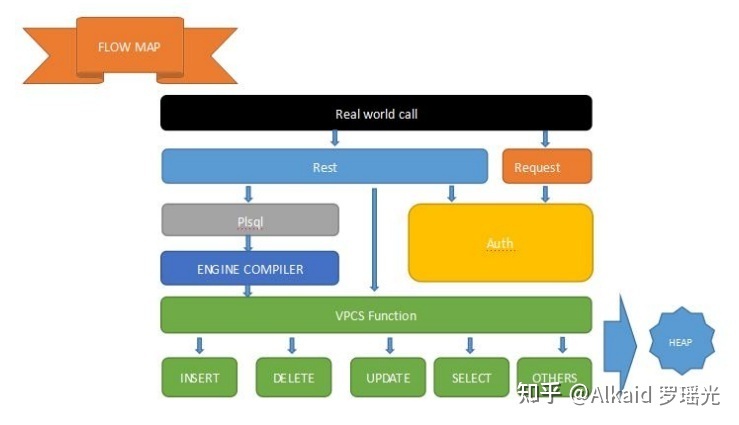
1 德塔数据库的数据存储是一种文件存储模式。refer page 408,409,469,473

2 文件的读写进行子集，行，表，映射，表头，按1范式分类。refer page 375,434,



3 数据库的数据读写支持加密。refer page 见元基加密

4 每一个文件不但有物理空间，还有相应的内存空间。refer page 375



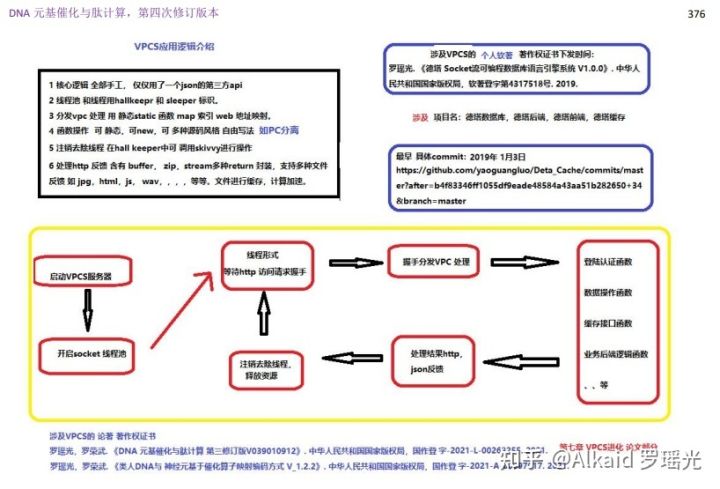
**VPCS服务器，**

1 VPCS服务器支持每秒400万QPS的web请求。refer page 389,

2 VPCS服务器采用TCP rest request模式，标准化http response。refer page 388,395

3 VPCS服务器可自由设计前端和后端集成。refer page 见德塔官网 和 养料经admin 两个实例

4 VPCS服务器完全支持post 个 get 2种请求模式，可扩展。refer page 481,488



**VPCS调度架构，**

1 VPCS服务器包含 视觉模块，处理模块，控制模块，资源模块。refer page 396,394,392,383

2 每一种模块有各自的名称标识 和 内存标识，方便精确查找。refer page 492,493,

3 VPCS服务器包含执行者-生产者-造梦者-sleeper，管理者-分配者-登记者-HallKeeper，运维者-服务员-清洁员-skivvy 3个模式。refer page 392,394,

4 支持控制与执行分离，线程与资源分离。refer page 385~389,486,490,492

**作者最早设计 VPCS 服务器的动机，是为了弥补VPC的计算过程观测困难的问题。因为作者设计的VPC是采用springboot + mybatis的结构，底层全是是开源插件的封装，很多核心源码又不能调试仅仅通过几个log和 trycatch给作者带来了无形的压力（作者的思维很简单，就是自己写个服务器，能够调试断点从头断到尾），于是有计划从无到有进行设计一个TCP/IP的 SOCKET 协议做服务器HTTP请求。作者当时没有想到，一个这样的小动机给带来了丰厚的回报，VPCS 目前成为了 DNA 元基映射编码算子 的核心组成部分。**

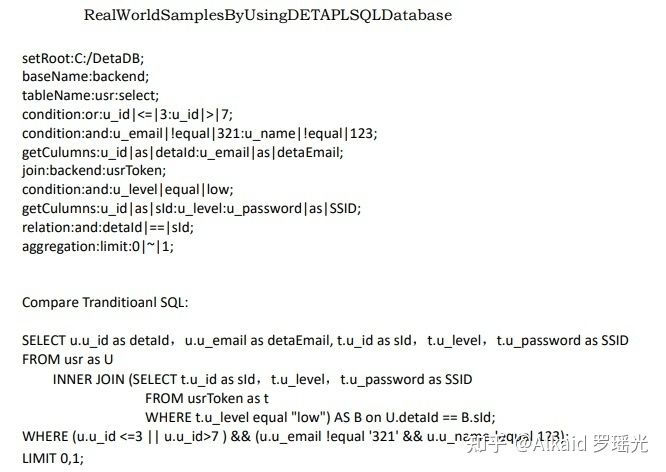
**PLSQL语言，**

1 德塔PLSQL语言是一种从上到下的脚本执行语言。refer page 377,

2 德塔PLSQL语言包含常用增删改查命令。refer page 406~409,471,1035

3 德塔PLSQL语言支持join和 aggregation 高级操作。refer page 419,431,435,438,447

4 德塔PLSQL语言行 可批处理，可拆分。refer page 1035~1041 将例子写入main，class编译，然后 bash boot class 即可。 还可以bash 定时批处理。



**PLSQL编译机，**

1 德塔PLSQL编译机 用于理解和执行 德塔PLSQL语言。refer page 413,414

2 德塔PLSQL编译机 包含常见脚本命令计算算子如 条件算子，比较算子，包含算子，离散算子。refer page 419

3 德塔PLSQL编译机 采用map进行的内部中间数据缓存。refer page 431,432~

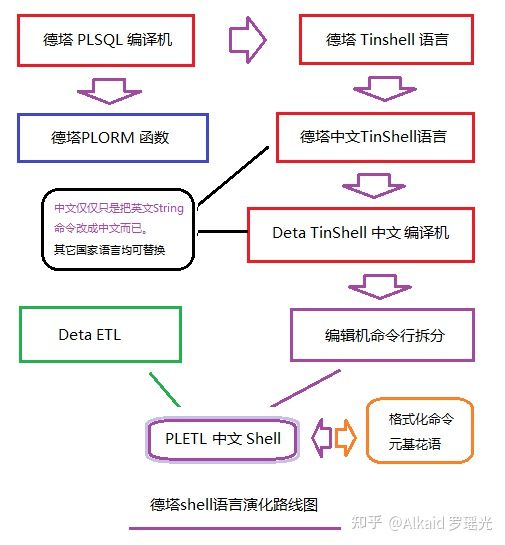
**PLORM语言，**

1 德塔PLORM语言 用于 德塔PLSQL语言进行函数封装。refer page 1003~

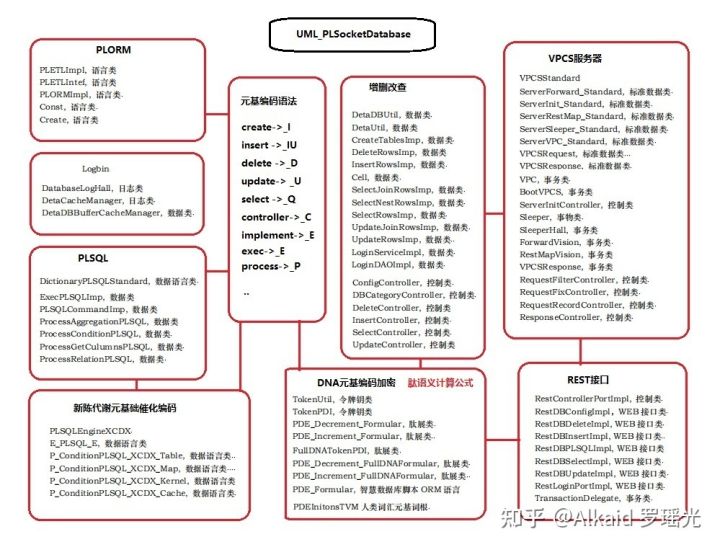
2 德塔PLORM语言 有先后顺序，需要遵循 德塔PLSQL语言语法。refer page 1019~

3 德塔PLORM语言 对比 德塔PLSQL语言 用于一些不需要配置的nosql的场景，类似 hibernate 对比 ibatis。refer page 1019~

4 VS hibernate 对比 ibatis的不同，德塔PLORM语言 另外也是 德塔PLSQL的上层语言。refer page 1019~



**德塔的PLORM 和 PLSQL 的引擎出现，作者开始有信心将其优化成 节点执行的命令行脚本模式，于是之后的TinShell 和 PLTinShell， PLETL Shell 诞生了。这个PLETL体系弥补了 当前世界按语言理解方式来模拟神经组织计算的映射空白。**



**灾后重建，**

1 德塔数据库包含logbin 系统。refer page 398,

2 德塔数据库包含logbin 系统基于单个写操作进行log保存 并行加密成文件。refer page 399

3 单个写操作用时间戳作和写增量序列进行对应标识，避免混乱。refer page 399

4 德塔数据库包含logbin 系统 并支持热备和错误写 实时rollback 检测。 refer page 398

**第五章\_德塔数据结构变量快速转换**

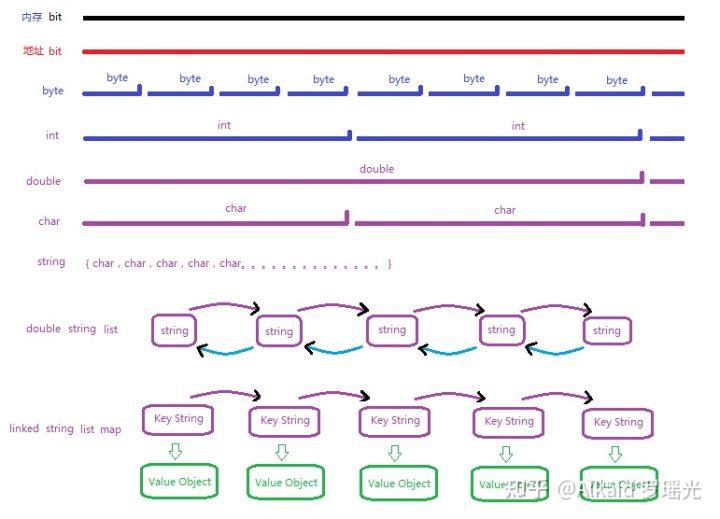
**内存的结构，**

1 德塔数据结构变换 最早归纳来自对 雪球新浪的股票数据 web页抓取进行的String格式统一。refer page 508,528

2 基于String的格式统一，然后逐步进行文本数据在计算过程中的状态进行分类扩展归纳。refer page 532,535

3 于是产生array，StringBuilder， iterator，map， 4种 高频内存结构的快速互换。refer page 499,536,515,520

4 最后进行对所有常见数据结构进行统一归纳和快速变换。 作者的研发基础来自2008年 在印度基督大学的C语言数据结构实验室课程。讲课教授 Rohini.V refer page 492~

常见数据结构类型，罗瑶光画图

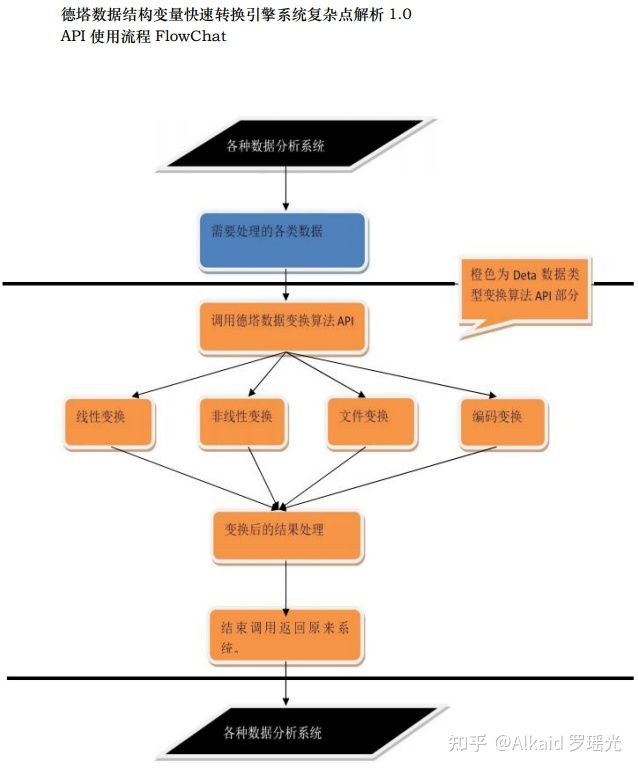
**数据的结构，**

1 德塔数据结构梳理完整依据 C语言数据结构 思维进行归纳refer page 无

2 归纳包含 array，String，struct object，hash， map list， tree，buffer的数据变换模式。refer page 499，535,527,507,520,516,537，

3 德塔数据结构不包含数据的计算逻辑变换，仅仅包含数据类型的载体变换。refer page 498

4 数据类型的载体变换通过接口形式表达。广泛用于工程中。refer page 498



罗瑶光画图

**类的结构，**

1 德塔数据结构的类，采用VPCS的静态接口模式设计。refer page 492~

2 每一种相同数据类函数封装在同类的文件中。refer page 492~

3 每一个类 主要包含数据变换文件，数据变换的纠正文件，数据变换的索引文件。refer page 492~

**转换加速，**

1 数据变换的索引文件，通过元基花索引24组染色体注册，进行语言调用加速。refer page 下册597 StaticFunctionMapU\_VECS\_E

2 数据变换采用静态函数，加速了function call。refer page 492~全章

3 数据变换的函数 根据功能进行了分类，于是静态函数文件形成了balanced静态函数集树模式。refer page 下册274 第十六章

**不规则对象的变换，**

1 不规则对象的变换主要包含 邻接矩阵array变换和 类复制。refer page 521

2 邻接矩阵array变换 如 跨格式变换，如xml，json，officerefer page 558,516,503

3 类复制如 deta的TinMap class和 Objectrefer page 527,881

4 xml和json，德塔不做加工，仅仅用google的Gson包引用。refer page 516

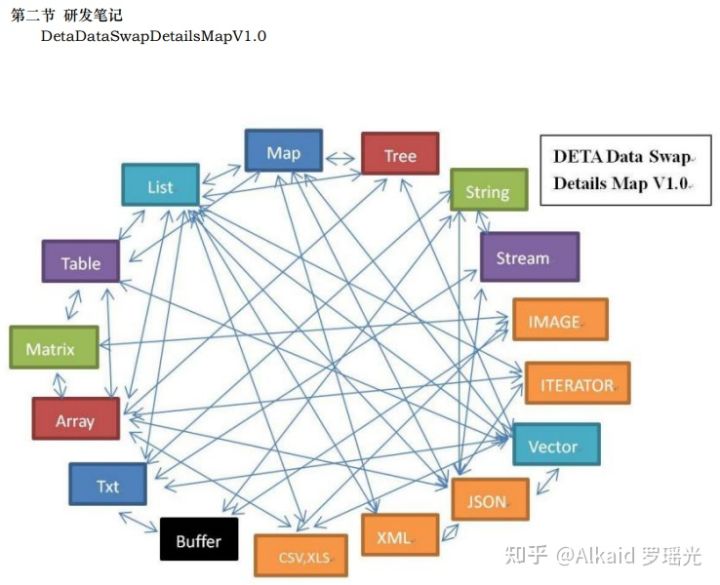
**场景变换，**

1 德塔数据结构的场景主要应用在网页html数据抓取，文本数据计算refer page 508,492~

2 html数据 主要体现在文字的编码格式变换，加密变换，和http response的内容载体变换如json。refer page 508,555，

3 文本数据计算主要体现在 map和array的变换，与 list和array的变换，用于字符串排序加速。refer page,499,516,520

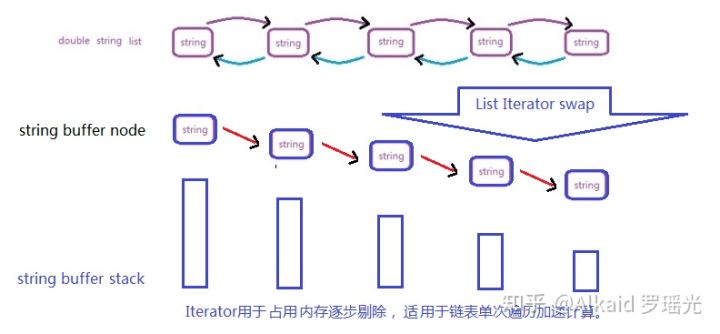
4 在德塔分词场景中体现在另外String与 StringBuilder的加速变换。refer page 536

罗瑶光画图

**计算的模式变换，**

1 德塔数据结构计算的模式变换主要用于 buffer中间态变换。refer page

2 buffer中间态包含 map与tree的变换， list与iterator的变换。refer page 520,537,516,515



图中String buffer stack 可先后序列排列，可断开成链，高度是iterator对象当前的内存占用大小。罗瑶光画图

list.toIterator()变换模式优势罗瑶光先生个人认为在计算过程中，基于内存的占用和寻址效率加速。String to Stringbuilder 变换同理，对象buffer化能实现内存变量计算和调用进行极限加速。作者在印度基督大学 学数据结构没有stringbuilder和 iterator的知识点，在2016年亚马逊的岗位技术经理面试时候， 有几次印度经理多次面试我关于String计算方式，我当时没有答上细节，错失了月薪12000美金的工作。我的罗瑶光画图

3 模式变换计算趋势归纳主要为非线性与线性的降维变换，通过改变观测面实现。refer page 497

应用

罗瑶光画图

太多略。

**章节的著作权文件列表：**

第六章\_数据预测引擎系统



此算法为版权算法

api包 函数完整包含2维和3维的空间轨迹算法。

[GitHub - yaoguangluo/Data\_Prediction: 快速计算商旅轨迹 非线性坐标数据分析](https://link.zhihu.com/?target=https://github.com/yaoguangluo/Data_Prediction" \t "https://zhuanlan.zhihu.com/p/_blank)

Java api [https://github.com/yaoguangluo/ChromosomeDNA/blob/main/BloomChromosome\_V19001\_20220108.jar](https://link.zhihu.com/?target=https://github.com/yaoguangluo/ChromosomeDNA/blob/main/BloomChromosome_V19001_20220108.jar" \t "https://zhuanlan.zhihu.com/p/_blank)

**坐标系统预测，**

1 数据预测引擎的坐标系统主要用来做离散非线性计算。refer page 566~

2 离散非线性计算主要体现在 降维 商旅TSP路径的线性求解。refer page 629~

3 坐标的降维计算包含 轨迹降维，趋势降维，观测降维。refer page 567~

4 降维计算过程可以进行逆向跟踪还原。refer page 570

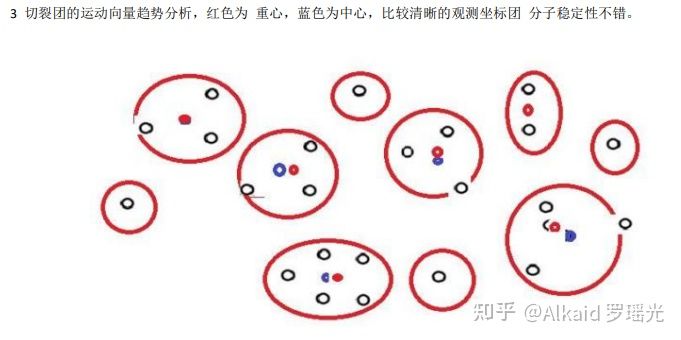
**环境预测，**

1 数据预测引擎的环境计算主要体现在 压力计算。refer page 570,573

2 压力计算可理解为 中心向重心的两点间距离。refer page 674

3 两点间距离的长短和方向代表压力的大小和趋势。refer page 574 红蓝点距离

4 趋势大小确定环境的稳定性表达。refer page 574



**雷达机，**

1 数据预测引擎的雷达机主要体现在坐标群的边缘识别和归纳计算。refer page 577

2 坐标群的边缘识别和归纳计算 采用角度 + 中心到点距离进行进行轮循链接。refer page 576

3 链接的面形成 极速计算边缘包含，确定坐标的团大小面积，密度。refer page 577

4 极速边缘计算的价值可以迅速利用在所有实时坐标系统中。refer page 593

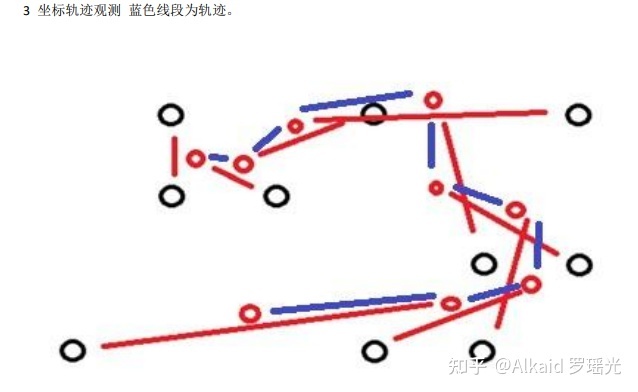
**状态机，**

1 数据预测引擎的状态机主要包含 压力状态，轨迹状态，refer page 571，573，

2 压力状态体现在坐标团的之间的距离，和团中心和重心的距离分析。refer page 571

3 轨迹状态体现在坐标团的内部欧基里德距离熵增和团中心KNN迁移熵增分析。refer page 569,570

4 数据预测引擎的状态机应用在非线性坐标计算系统中。refer page



**离散模型预测，**

1 数据预测引擎的离散模型预测，作者主要用在商旅计算中。refer page

2 作者主要用在商旅计算中的 小坐标分子群计算中。refer page 568

3 作者的商旅计算最大价值主要体现在 欧拉环路的分析中。refer page 568

4 作者的 欧拉环路为破解 十六进制 十六元基进制编码 起到了基础研究作用。refer page 下册56, 下册125

**概率机，**

1 数据预测引擎的概率机比较简单，仅仅贝叶斯系统。refer page

2 贝叶斯系统在作者的工程中很少用到，如线性回归，衰变失效就不包括。refer page

3 贝叶斯系统作者有设计交叉概率机，关于数据挖掘pangningtan教材的质量分析。讲课教授 卡拉森。refer page 616

4 作者设计概率机，主要是之后做图片识别预测用。refer page

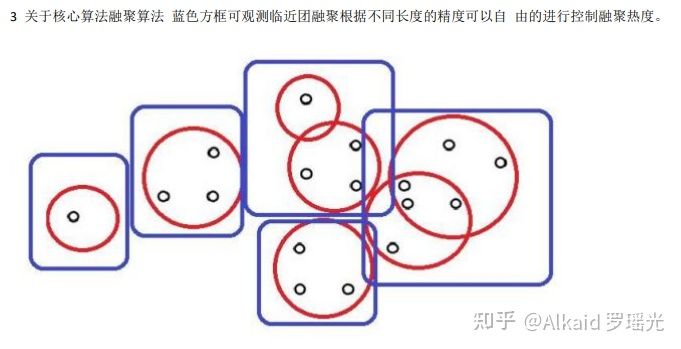
**向量机，**

1 数据预测引擎的向量机作者主要设计了团中心和重心的距离向量。refer page 595

2 距离向量 可以作为路径猜测，运动趋势，和轨迹判断用途。refer page 621,624,634

3 距离向量理解为斥力，可以表达坐标团的稳定性评估。refer page 601

4 距离向量理解为压力，与雷达机结合，可以计算表达坐标团的密度。refer page 610,613,605, 593



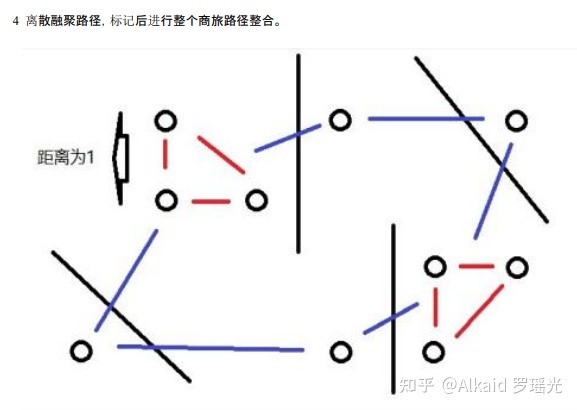
**德塔坐标团的密度 一般指，将坐标进行 观测距离的区间进行划分后的坐标融聚小团，的坐标数和团数的比值举例 如果划分有5个区间，每个区间坐标数是 1,3,4,3,6,，那么比值是1/5 , 3/5, 4/5, 3/5, 6/5 这里的观测距离是可以精度调节的。通过排序可以迅速计算 用于确定压力的位置。 定义归纳人 罗瑶光，稍后优化**

**商旅TSP，**

1 数据预测引擎的商旅TSP，主要计算随机坐标集的欧拉环路。refer page 625

2 数据预测引擎的商旅TSP，作者设计动机为极速小分子团间的欧拉2阶图研究。refer page 630

3 作者研究动机为破解元基罗盘的 离散活性邻接矩阵变换。refer page 下册5，



4 作者研究结果为十六元基进制 破解 DCPE-THOS-MAXF-VIUQ 。refer page 下册5，下册56,下册125

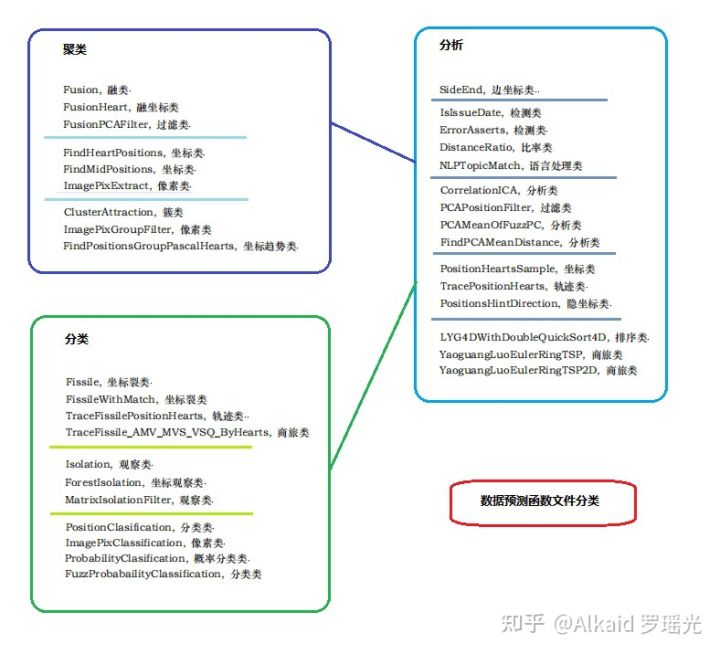
应用

太多了，略，

**早期应用实例，不仅在德塔自己的坐标插件可以灵活应用， detaETL 也可以集成 awt+ weka第三方插件研发 进行数据显示实现。如下图的pilot例子。作者早期用swt+knime进行weka设计，自从自己写了etl unicorn后，发现SWT插件都不需要了。**



另外函数分类方法如 切裂，融聚，隔离，簇类，就不介绍了数据挖掘的聚类思想作者个人表达方法而已。



第七章 类人DNA与神经元基于催化算子映射编码方式 原文 的英文部分， 图片在如下链接。或原书第七章中。

**The Initons Catalytic Reflection Between Humanoid DNA and Nero Cell**

Yaoguang Luo, Rongwu Luo

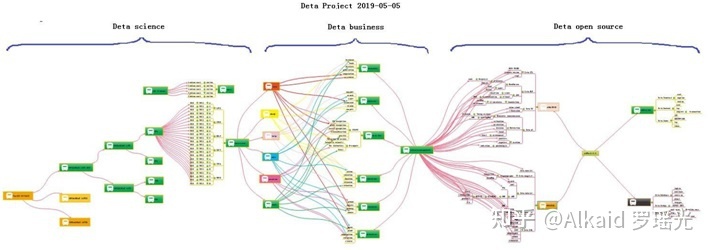
Keywords: VPCS, AOPM, IDUC, Nero, Artificial, Decoder, Medical, Paralling, Computing,

Humanoid, ETL, Parser, Data Mining

**Outlook:**VPCS architecture is not the end. Absolutely, At least at this paper, I will make an implementation in five sections: DETA humanoid cognition, DETA Medical Business backend logic, DETA Catalytic computing, DETA Findinginitions, DETA DNA decoding.Above all I also will spend more and more words in my DETA DNA Law of IDUC. And itsapplications in the real world. ok next step as below.

**1 DETA humanoid cognition**

Since the inition of the DETA OSS, there has a lot of questions where based on the humanoid DNA catalytic computing, I have been working on this domain for a long time. Absolutely also, I have got a lot of flashing points here, for example AI, still remember the first time I touch the cognition this verbal at CLU Dr Renhat’s class about cogs PU computing and cognition quality Sonar test where in Folsom Intel, I know that cog-work is a trending task in my life. Even though I need do a lot of basic foundations at AI, but I insist to now, from the code where between normalization and duplication. I have been thinking how to make a contraction and distinction AI logic with human and humanoid. Ok, PLAN A starts as below. Code a problems solution software like a way of YaoGuang. Luo's cog-style life.



Look at this PIC, smartly, build basic foundations first, then create more business software where based on this foundations, and finally swap to a humanoid model. Many many times I hope the model could be an Immortality.

**1.1 DETA humanoid cognition history,**

In the past, knowledge of the world could be parsed by five sections, the world's cognitive way, philosophers and scientists liked to describe objects with five senses (touch, taste, hearing, smell and vision). Species sense can make creatures in order to adapt to the environment well, understand the environment, and think about ways to protect themselves in dangerous environments. Thesemethods are implemented in a variety of ways, and their execution logic is also varied, but the causes and results are clear about a basic point. Better adapt to the environment. During the process of designing the YangLiaoJing, the author has well integrated the bionic technologies of voice, text, association and visual media, and has been optimizing them to gradually form a comprehensive intelligent YangLiaoJing system.

**1.2 DETA humanoid cognition development,**

In order to better adapt to the environment, human beings began to create characters, invent tools and improve their cognitive ability, from the ignorance of slaves to the open and compatible world, from the Iron Age to the current nano-chip technology. In the river history of 5000 years, human beings seem to have evolved aimlessly. Through these phenomena, the essence can be easily discovered. Better adapt to and transform the environment. The best arguments to improve the cognitive ability of environmental things are the research and development of basic science and technology and systematic induction, Marconi's wireless telegraph, Zu Chongzhi's pi, Darwin's origin of species, code of Hammurabi, and countless outstanding scientists, thinkers, inventors and philosophers in history. These people seem to be shining stars in the night sky. Gradually, Intelligent creatures begin to have enough ability to look up into space and explore the mysteries of the universe, and this enough ability is the basic ability to improve the cognition of things, which is very important. The HuaRuiJi system of DETA Company draws lessons from the systematic induction method of human instinct and dialectically treats medical diseases with medical textbooks as the cognitive basis, which is in line with the embodiment of scientific development.

**1.3 DETA humanoid cognition application,**

There are many applications of humanoid cognition in social science, and there are many excellent arguments here, such as auxiliary auditory system, big data reasoning system, weather forecasting system and criminal investigation database system, which undoubtedly proves that cognitive model has greatly improved the adaptability of human beings to the environment. To the ability to gradually transform the environment from part to whole. There are many outstanding arguments here, from the ancient Dayu flood control, to the renovation of Emperor Yangdi's Canal, (当前政治经济实体实例已经过滤), from artificial rainfall to artificial islands. The argument is very clear, and the cognitive ability of things comes from the accumulation of basic science and technology. These basic technologies gradually form a system. On top of it are a wide range of scientific and technological commodity applications, (当前政治经济实体实例已经过滤), giant hydro-power station of Three Gorges of Yangtze River to improve Geographical environment, etc. There are too many. The evolution of human wisdom gradually forms a clear route, and improving basic science and technology and cognitive ability complement each other. These abilities all come from thinking about things. Then form a solution, and finally implement it. This process is summarized as the process of analysis, operation, processing and management. The life cycle of software engineering is well explained here. Analysis A, Operation O, Processing P, Management M, use simple words to describe that even if unknown data are collected and analyzed, and then things are operated, the solutions to various difficulties encountered in the operation process are implemented. Finally, maintain and manage these implementation experiences. The back-end computing mode and system life cycle of DETA have gradually condensed from the earliest collection, analysis, operation, sorting, coding, running, debugging and maintenance to the module modes of Analysis A, Operation O, Processing P and Management M, such as DETA word segmentation, DETA DNN mind reading, etc. Now ETL of DETA is ready to go in this direction. The author designed a paper last year to describe the application mechanism of AOPM as follows:

**AOPM Open Source System On SDLC Theory**

Mr. Yaoguang. Luo

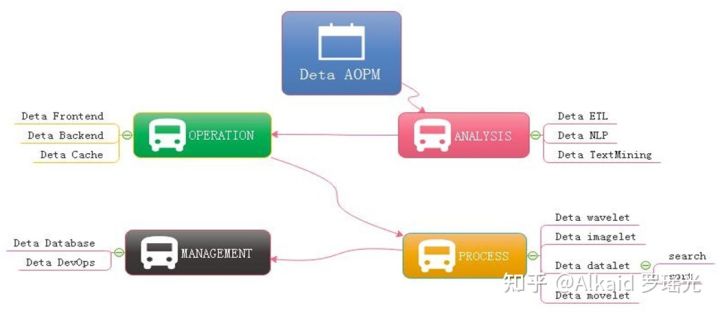
Outline: Mr. Xuesen. Qian once said: Science Is A Titan System, as an open source software conception. this topic implements a software interaction theory of SDLC for Analysis, Operation, Process and Management— AOPM. Also, this is a tiny paper where easy to show more idyllic landscapes of using DETA open source projects. Not only for web system, also for mobile and desktop platform. The final goal are makes complex project to simple. Ok let go and the next steps.

Keywords: SDLC, AOPM, VPCS, WEB, Concurrent, Open Source, Interaction, Management, Automation

Introductions Recently my colleagues take more care on the SDLC evolution of open source software engineering, for each project they undertake on where it cost a lot of times, that’s for my job, continuing found out a high effect, simple and clear theory of SDLC what be my main task now. after imagination and logic recursion, the key is an optimization of ordinary SDLC such as water fall. First time for makes an introduction to waterfall of SDLC? The author’s explanation likes sequence linked list of component nodes. With DETA projects here contains four aspects at Figure1-1. And my explanation of open source as belows.

**Topic: Ten Definition of The Open Source, OSS Book Reading Note**

In this paper, through a premise: the contrast between the copyright and the contract. the Author talks a comprehensive introduction to the definition of the open source code. The role of the open source licenses, which is to allow the work permit under the non-exclusive business. Not only does it mean that the source code was visited by the public user, and also meets another 10 conditions as follows. The first point: the open source software allows the free reusable distribution. The license must not restrict that any party sell or give away the software. At the same time, it can't get the sold fees and other fees for this software. The second point: the program must include the full of source code. The license does not allow that getting the source code from any specific forms of the production. The license assures that no one can intentionally to confuse the source code. At the same time, the users have the right to access to the source code under this license. The third point: which talks about the rights of the derivative work. The license must allow the work-modification and the new-work-derivation . those new's are published under the same license. The fourth point: the integrity of the source code. Licenses and the integrity of permits,which may limit the distribution of the form of the modified source code. The fifth point: license does not discriminate against any specific groups and individuals. The sixth point: license does not limit the use way of any particular field scheme. At the same time, the license can't limit the use way's flexibility and reliability. The seventh point: the distribution of the license. Distribution solutions do not need additional license.The eighth point: the license must not specific to the product. The redistribution of the software does not dependent on the program. The ninth point: license may not restrict other software. This license may not restrict the publish of the software. The distribute software will be built by using open source. The end point: license rights is neutral. So, it effective limits that the freedom of the code transmission. In other words, it provides the preventive measures.



**Figure 1-1 AOPM Applications with SDLC Evolutions**

Last year I was asked by so many engineers, almost the same question: how have you done so many projects during the year of 2018~2019 My answer is absolutely: connection. Always, with connection, I got lots of fantasy inspirations on the projects where I undertook. My projects all are lower basic technical factors, with connections, what support me the necessary energy for continuing development on my projects. What means connection? Be an internal union bridge between my projects. For example DETA NLP and DETA ETL, they both have the same attributes such as AI, Analysis and Data etc, with this connections, my tasks became more dynamically. Every time before I made a decision of priority levels of my projects, I thought the connection first, DETA projects totally can be separated into three dimensions. Front end Back end and Storage, as the Figure 1-2, the connection between DETA projects is WEB AI, now is a Bazaar requirement, but we will easy to make estimation of it’s future, toward to Cathedral.

**Topic: Cathedral and the Bazaar, OSS Book Reading Note**

Cathedral and the Bazaar, this article has a profound implication, the author is a computer scientist with extensive experience. We can say that he is one of the early code and program contributors in the UNIX system. This article describes the Linux development with the revolutionary road, as the process from the bazaar to the cathedral. First, the author tells the contrast between UNIX and Linux: now UNIX is still popular around the world. Its rigorous structure and contribution to science, let it is proud of the same dignity as a church. Linux looks like a noisy bazaar, the code work in various countries around the world, to solve their own problems and arguing in the forums and communities. Like a bazaar. Then, author points an internal factors to get an in-depth discussion: UNIX reason why it has the church's authority, because its development has always been tailor-made by the world's most senior and most eminent researchers and software scientists. Although the discussion, because of the nature of the project-oriented, so that UNIX has been applied still to today. Even of the unreasonable original design, through decades of use, engineers have become accustomed to this experience now, there fore, we are called transcendental. which makes UNIX feels like a cathedral. The birth of the Linux was different, survival in an all-spittle environment. Every update, are implemented in controversial circumstances. The crowd here, are huge number of scientists, or writers,or code workers or merchants, their common ideal is that make Linux development meets the needs of all groups. Similar a huge bazaar. The author commenced a leno-vo, a conclusion that Linux will eventually beat UNIX, UNIX gets the range of fresh blood is less than the Linux' s, also the number of the UNIX team members is less than the Linux' s. UNIX customers and employees are aging. But Linux development more in line with the user of the needs. Its own development is to establish a relationship on this demand and requirement. Linux is young now. Summary, UNIX and Linux development option is the two kinds of very different road. These processes and methods to determine the fate of the two kinds of software development. Of more optimistic about Linux because it is better adapted to the environment.

At figure 1-2, DETA open source main based on AI domain, it already formed as an ecology system, go ahead to the application, thanks.

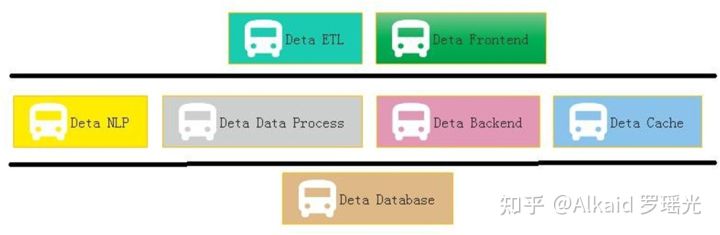


Figure 1-2 Sections of DETA Projects Group Applications One question is my friend asked me why does DETA support the e-commence logic? Definitely! Please see the Figure 1-3, this is a classic horizontal deployment sample of the real word. Alibaba, Amazon, Ebay and JD etc, all based on this technology, instead of Spring, DETA can be the next generation of technology.

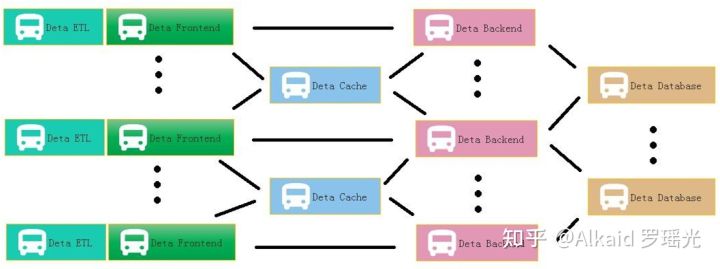


Figure 1-3 DETA WEB Projects System At Figure 1.4 is a real sample for web Devops by using DETA Open Source.

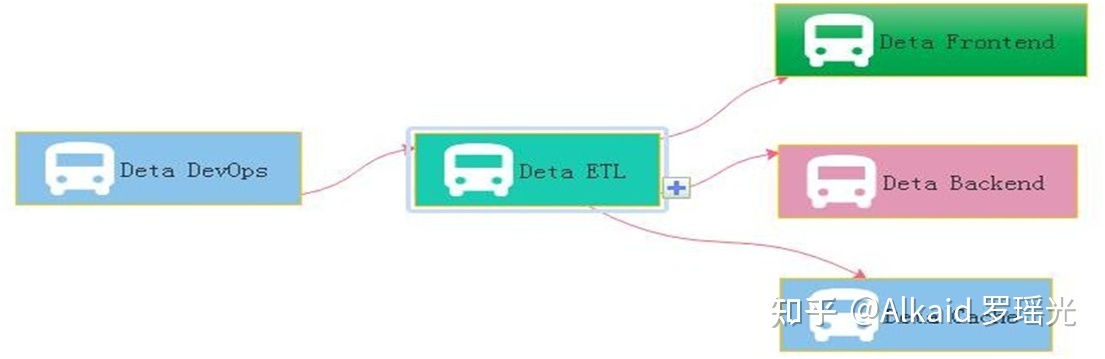


Figure 1-4 DETA DevOps Projects System

**2 DETA Business back end logic**

Before 2010, the author systematically contacted the mechanism of analyzing A, operating O, processing P and managing M in the learning process. After graduation, he had the opportunity to deal with the business logic corresponding to these things through programming in some software companies in the society. From the research of MP6 mail system (当前政治经济实体实例已经过滤), The Bluetooth group advertisement machine to (当前政治经济实体实例已经过滤), from the e-commerce back-end calculation(当前政治经济实体实例已经过滤)to the global hotel reservation (当前政治经济实体实例已经过滤), the author has been thinking about how wonderful it would be if the front-end system could give wisdom like human beings. So the bud in the author's heart began to take root, and he was confident to design a set of architecture system with humanoid wisdom to meet the rapid development of business intelligence applications.

**2.1 DETA Business backend logic history,**

The first contact with the Front end and Back end separation was in 2004, when the author first published a website in Liuyang city by using the 7week platform. The website was a second-level domain name, using a third-party server, even though the concepts of front-end and back-end were ignorant at that time. The author first contacted MVC architecture in Shanghai Fan Teng Information Technology Company. At that time, the feeling was that MVC could solve all kinds of business logic. In the same year, the author first came into contact with MVP to do multi-thread Bluetooth big file project, and felt that MVP seemed to make the architecture handle the problem of concurrent computing well. From 2014 to 2017, the author worked almost with business logic corresponding to various MVC architectures, such as Spring, Martini, etc. The author thinks that gives MVC an intelligence urgently.

**2.2 DETA Business backend logic development,**

Thanks to my father, in 2018, he told me to design a pharmacy-assisted search software according to the concept of Chinese medicine, so he began to design Huaruiji Medical Big Data System. At that time, I thought spring boot, mysql were too heavy. If the database rest handshake system of socket stream was designed according to CGI, many problems could be solved easily. So I began to analyze, operate and deal with the problems. Gradually found some irreplaceable primitives, such as S static data, V visionary observation model, P procedure registration mode, C control unit, etc. It would be wonderful if we could redesign a set of architectures for these primitives. The PC separation mode here comes from an IOC doctoral design paper in Spring in 2015. Thanks here, I integrated MV into V observation model, and then took out the corresponding static data of M and function S. This VPCS structure choots me at present.With regard to the excessive description of VPCS, I can take a previous note as follows: VPC architecture programming thought, software programming for many years, accumulated some thoughts on program realization. Through the certification of Darwin's theory of evolution, an effective VPC programming concept is elaborated based on the neutral coupling of MVC+MVP. V is an observer model, similar to storage object and observation model. P is the processor, which handles the registration interface. C is the control machine, which describes and classifies the registration interface. S static control machine, why use static control machine, advantages: 1. Because of the separation of PC, the functions of C mode are inherited through abstract virtual functions, interfaces inherit interfaces, interfaces are uniformly registered, and calls are extremely discrete, thus achieving the efficiency of high-speed concurrent iteration. 2. Realize EI separation and skip IOC scanning. 3: P is responsible for reference and description, and C can carry out various functional operations through descriptions of multiple P. Mapping control technology ensures thread safety and stability. 4: V stores each single case class to ensure low data redundancy and unified recovery.

**2.3 DETA Business backend logic application,**

In the whole year of 2019, the VPCS back-end engine gradually formed some standardized functions and papers, which were applied to the front-end, back-end, cache, database and other subsystems of DETA. My evaluation of them is that they are lightweight and extensible. VPCS is gradually integrated into the works of Yangliaojing and Huaruiji. Of course, there are many shortcomings, the biggest one is that they do not repair themselves. Although I designed the sleeper and hall keeper mechanisms, these mechanisms are only the corresponding business logic units that I complete through decision trees, not humanoid evolutionary thinking. At least, I don't think they are humanoid intelligence. To be precise, at present, they are only artificial intelligence, a kind of artificial intelligence logic corresponding to AOPM and VPCS, but not the humanoid evolutionary intelligence logic that I want. So I started to explore humanoid computing again. About the application principle description of VPCS, the author designed a paper as below:

**VPCS Backend Theory And Its Application**

Mr. Yaoguang. Luo

Outline: due to the development of the software acquisition and definition in what we use the code theory always in messy and unforeseeable status. A new method of the coding style like VPCS that will show in this topic paper, feel free to resonate with my imagination of the portrait—VPCS(Vision, Process, Controller, Sets) theory, fun yet? Not only this paper will gazer a big point how we show the onstructions of the VPCS, you guys also sure to get lots of idyllic landscapes of the coding sections. While you got lots of the illness codes at the so messy fungus projects, I guess at this paper out where you are finding anxiously. Let’s catch more opportunity about how does the VPCS working, executing and scheduling in our software project and make the software fast, fast and safe! lets go, So the key words as below:

Quantum Sets, Concurrent Consumer, Vision, Scheduler, Threads, Surf.

Introductions

Let see the verbal keys, the first time you …, okay, get any sense? Sure, this paper is not talking about the human careers, truly about software, as a human, if you got my points, yes, cool! Make any sense? Let’s see the landscape as below figure 1-1.



Figure 1-1 VPCS STAR MODEL

From the ordinary software development architecture, always like a factory model, for instance, controller, transaction delegate, web service, job bean, data DAO, like that of traditional back end or front end coding style, but, compare now the seamless clients services system, those model more and more not suitable for us for the project application, at least in the light level, multitasks, satellite boots projects system, if we choice the factory model, you will feel so heavy. But the big conflict problem is where the factory model was used in all and all bazaar companies. Even more CTOS that I met before often complaining about the reference room likes that “we need one server for database system, one more for cache system, one more for front end, one more, for backend, one more….”after that what do you think? My lord…Finding a new method of how to integrate the sets about the micro satellites service in the same sever, and make them small, lightly and faster for the commence service, now become a fatal topic. Which can be a pretty warm-up for where I make an explanation for VPCS. The VPCS model, only includes four aspects. Vision, Process, Controller, Sets, and those factors makes an interactions in the sleeper containers. Let talk about the definition of the sleepers. From the software engineering domain, the sleepers are more like an identified thread person. Who can make a lot of fantasy dream in a Hall, what means a dream? Dream is a requirement what the consumer really needs to finished. But here the dream can be separated out more tasks, those tasks will register the ID in the Pillow, so that the sleeper hugs the pillow then goes into the hall and make a dream. Got an idea? Cool. So what does the sleeper does in a hall? The answer is to make all kinds of the dream. For example if we want to build the web service to get rest call, and return the JSON feedbacks, we only need to do like the way: Firth, build rest call path in the controller; Second: register the call requirements as a dream; Third, build the sets of the dream in the pillow, Fourth hire a sleeper to hug this pillow, and go to the hall to make a dream process. At last but least: return the dream goods. Any sense? Cool! For this unique instance, you will know that the sleeper was more like a socket, and the hall more like a thread pool, the pillows like the single vision instance, and the sets like a vision storage, the controller and the process those two sections is a common way of the factory model. The steps landscape of the sleeper who makes a dream as bellow figure 1-2.

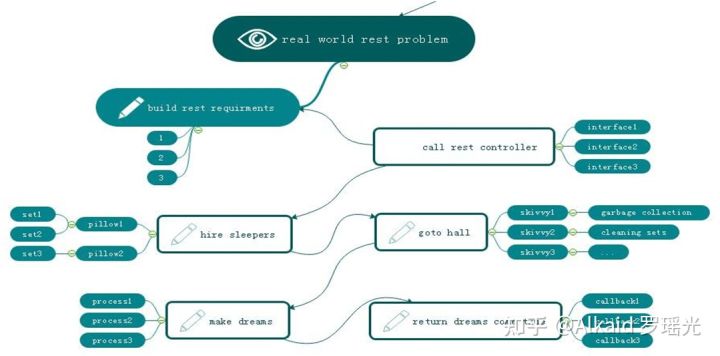


Figure 1-2 VPCS BACK END MODEL

Focus on this landscape, mostly different to the MVC: Model View Controller, MVP: Model View Presenter or other architectures we know before. but is very easy to understand after you read for a while. Too simple. Sleeper makes dreams come true, hall container sleepers, skivvy make up the hall, pillow clear and wake up the sleepers who often lost in finding the way in the dream. Got fun here, but I would hear more argue voice details of my VPCS, desktop App once said: VPCS is good in the concurrent WEB project, but not suitable for the desktop applications. Ok, follow this question, let make a new landscape based on desktop application as below figure 1-3.

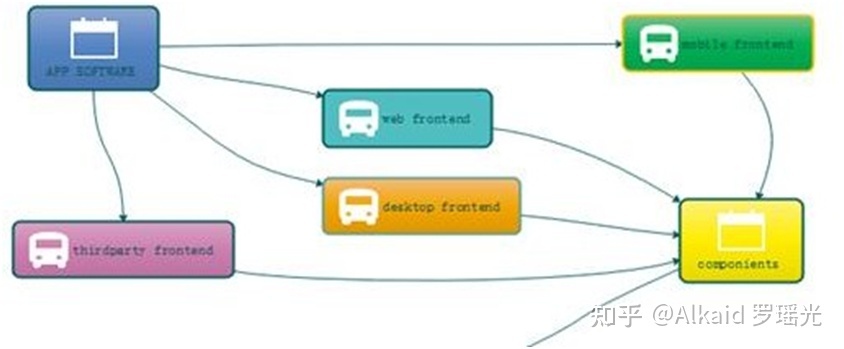


Figure 1-3 VPCS WORK WITH FRONT END

From this picture, we know all of the software can be fast and safe while using VPCS, because it is already separated out the big system into back end and front end two parts. and VPCS keeps safe and fast in the back end section. Compare to the MVC, VPCS will get more cautious and details, and compare to MVP, VPCS also will get more safe and high efficiency. Those factors are why I will make inauguration here. In the common software engineering cycle life times,scientist used to build front end and back end for all kinds of the software applications, because it is easy to control. Why? Frontend only spend time to make design, and Back end for the data operations. Using VPCS system, we don’t care about what they do for the front end, we only fit about what they want. Alignment that gets a blame and fix, then return OK, the restful service developer makes a voice that http functions are concurrent functions. At here, VPCS will say: concurrent functions are safe functions. We guess in the future REST-VPCS will be used in multiple WEB service. Especially in the high speed, efficiency, micro web systems with high level security for example medicine, DNA, cloud server, electronic police system and ecommerce systems etc.

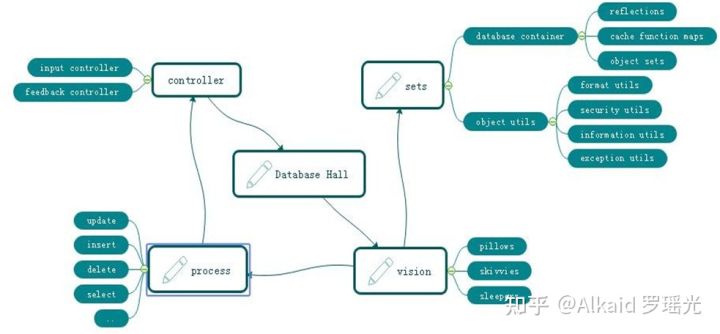


Figure 1-4 VPCS FOR DATABASE SYSTEM

As the figure 1-4, a new method of the Database system designing shows us VPCS is a pretty way for the modern data information management system. Definitely used in the DETA Database system. For this instance, do we get a view that the controller section of the factory model becomes thin yet? Controller only works for the hands transactions, for example that the controller get an input requirement such as select SQL, then immediately call the hall keeper to register this SQL and hire new sleepers to make a result. Because of the VPCS. Once it happened any exceptions, will very easy to awake sleeper and let them get theirs working papers out, finally call skivvy to fork the sets to the fresh sleeper. This method mostly be like a Count Down Latch model, once the sleeper gets the dreams come true, then told the hall keeper for the feedback, hall keeper will makes a type procession to return after everything goes well, This method mostly be like a Cyclic Barrier model.

Questions

How does skivvy doing? please see the figure 1-5 the hall building need a singleton instance like a home keeper but here is a hall keeper, any else, this person is very important for keeping the VPCS safe, because all of the skivvies will be managed by him. You will see, the memory check, JVM garbage collection, disk cleaning, thread status management, deadlock alarm, security protocol all and all in one at here. Mostly like a static class in the VPCS system. If we need to know every thing about skivvy’s work status, ok just call the hall keeper.

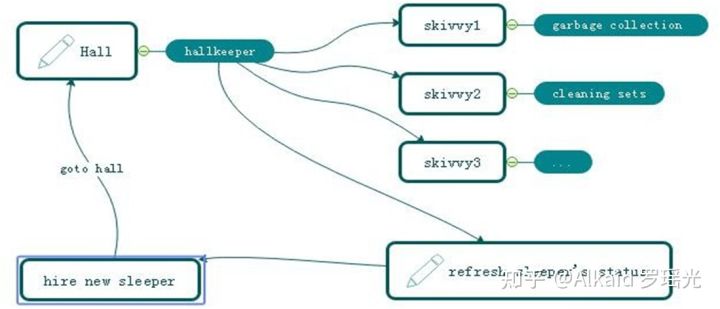


Figure 1-5 VPCS KERNEL

How does sleeper doing? Make a dream? Cool, you shoot!, in the VPCS system, it doesn’t have the definition of the process, everything likes subsets. Immutable or unlined, hall keeper get request from visionary and hire the sleeper, who is likes a thread, get requirement, add those sets in pillow, hugs pillow then go to hall to make a dream, after that then return the callback to hall keeper what they did. Fun yet?What does the sets meaning? Sets, is a format of the data where appearing in the VPCS system. For the static prototype, it used like a concurrent hash table, and list which can be copy base on writing format, the single instance, it always runs in the static function or be liking an interface implementation because need safe at the same time, so that compare to the factory model, it is too simple and without annotation. Everything becomes easy in this environment.The one more question is that so many peoples asked me what does the sequence diagram of the VPCS, because they really want to know why VPCS is faster and safe. Ok, please see the figure 1-6, the answer is absolutely, VPCS main components of the time sequence only contains five aspects. Almost similar like the hotel management. Certainly, we are talking about VPCS software, not for guesthouse. You will see that the rest call only makes the interactions with the hall keeper. And hall keeper got two jobs, one for waiting the fresh sleeper and one more for giving task to skivvy. The sleeper only hugs the relate pillow and make the dreams come true. Fun yet? Cool. VPCS only take cares about how does the sleeper’s imagination and skivvy’s working status. If is the pillow broken? Make new pillow, got lazy sleeper? Get out his working papers, got a cheat skivvy? Fix of fire him, the real source of the java version project for the VPCS only 30kb, we will find more sources or documents from the reference links at the end.

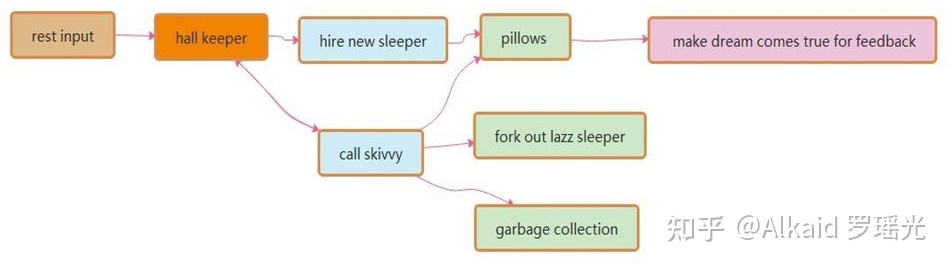


Figure 1-6 VPCS Sequence Diagram

I always be asked by the colleagues that once said: how does the hall build? I answered them, such like the hospital, no one cares about the address of hospital, because they just call the cell phone number when will get a directly feedback. This is why I need a hall keeper role in the gate way. For the instance about figure 1.6.1, this sample is a true demo in the real world for the WEB rest service. Its very important to create a player role such like hall keeper. what would likes about author’s theory? Because of the maintenance. Because of whom, the software build team are very easy to make a maintenance web portal, all of the system current status will be solved on this html page by DEV-OPS.

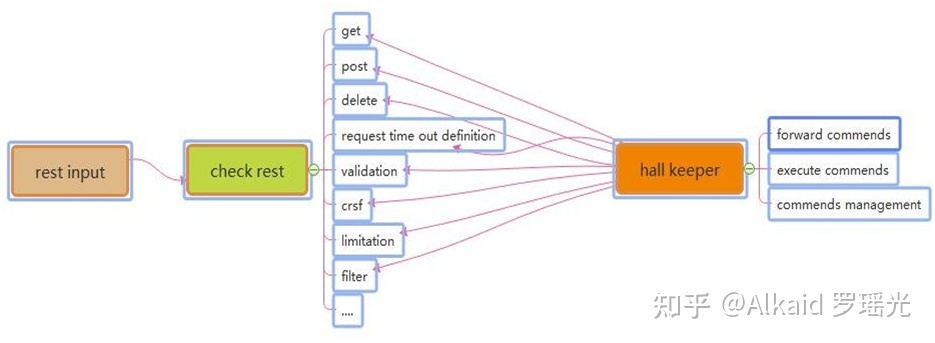


Figure 1-6-1 The Interaction Between Rest Call and Hall Keeper

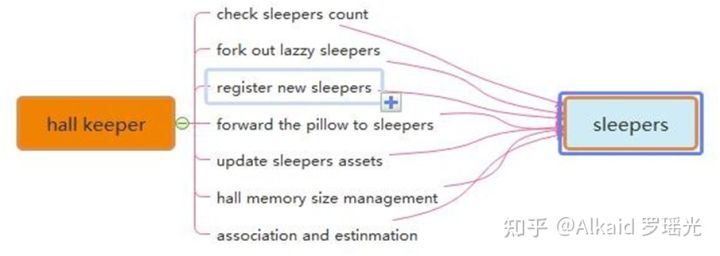


Figure 1-6-2 The Interaction Between Hall Keeper and Sleepers

Many of these software developer also asked me how and why we fork out the LAZZY sleepers excluding their sets. Arthur answered because of the pillows. When the sleepers be hired from the hall keeper, they will get an independently pillows such like static functions. So that sleeper only has their own identify attributes and unique information as the singe instance class. Once they got theirs working paper, the pillows they used will be arranged to the new fresh sleeper, this theory keeps safe, quality and quantity. Like figure 1.6.2.1 VPCS kernel.

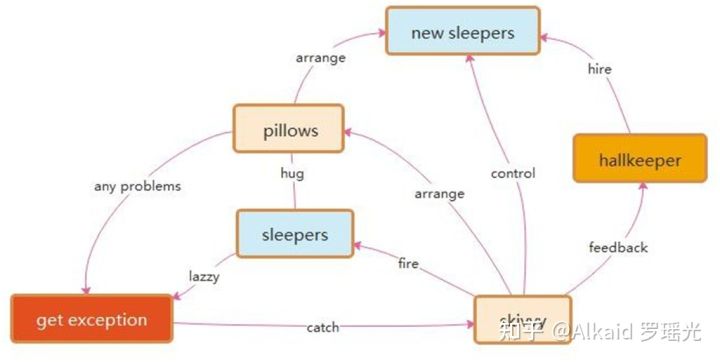


Figure 1-6-2-1 VPCS kernel

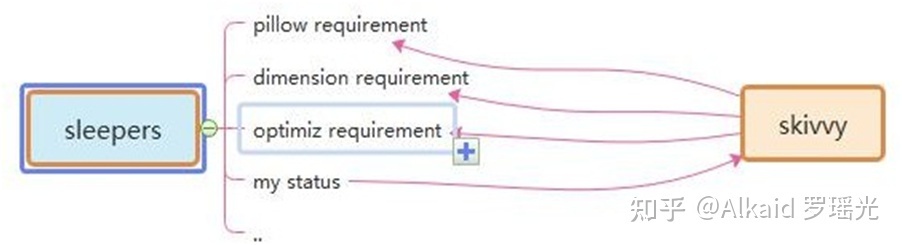


Figure 1-6-3 The Interaction Between Sleepers and skivvy

Many times, I got questions about the DEVOPS, they really worry about VPCS if suitable for their project system maintenance? The answer is absolutely, as the Mr. Ray [274138705@qq.com](mailto:274138705@qq.com) once said: we are DEVOPS, at least we need three important keys in our environment assignments: implementation capacity, transparency and maneuverability. How does the VPCS supports us for daily works? Because of Hall Keeper and skivvy, as figure 1-6-3 and 1-6-4. DEVOPS will get all of these transparency information about project from hall keeper under the encryption and security issues. Also, hall keeper will directly get the rule for DEVOPS by rest calls, then makes to commend to skivvy. All of the information and record logger will be cached by hall keeper, that keeps controllability. The html control page will make an interaction between hall keeper and DEVEOPS, which keeps safe, implementation capacity, transparency and maneuverability. This is my true answer.

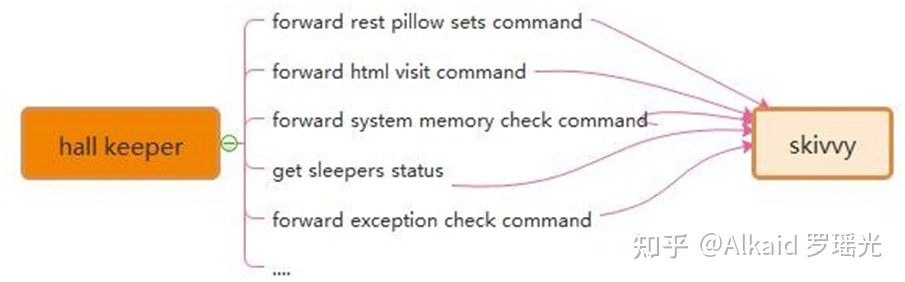


Figure 1-6-4 The Interaction Between Skivvy and Hall Keeper

Recently Mr. Yang [1291244774@qq.com](mailto:1291244774@qq.com) who asked me about VPCS of IOS desktop APP, where and how to avoid the data leakage risks. Because he really worries about the separation between controller and process. Following this topic, my answer that the key is the separation between pillows and sleepers. Due to the pillows all have their own unique ID, skivvy will easily arrange the pillow to new sleeper after the original sleeper who made problems. Make unique ID and arrangement by ID, is the key method. Also for the rest call service, the asymmetrically irreversible combination encryption is one of the best solutions to the data leakage controller. VPCS seems so smart.

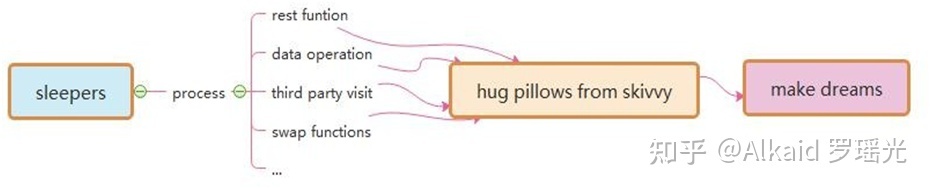


Figure 1-6-5 The Interaction Between Sleepers and Pillows

**3 DETA Catalytic computing**

I got my mind storm for a month in early 2019. How to realize human computing? It's been bothering me for a long time. How to start? I didn't have a clue, so I started to read my notes made in the past 20 years. I got it! Do basic research! According to my notes, I dig some unknown basic knowledge. I am veryhappy, because I have the results of quick sorting by left-right comparison in 2014, the butterfly calculation manuscript of Fast Fourier, the Chinese word segmentation works of Huaruiji, UNICORN ETL, socket stream PLSQL database, etc. and an idea came into being. I think about optimizing them continuously, refining, optimizing and testing repeatedly, and remembering these optimized ideas.

**3.1 DETA Catalytic computing history,**

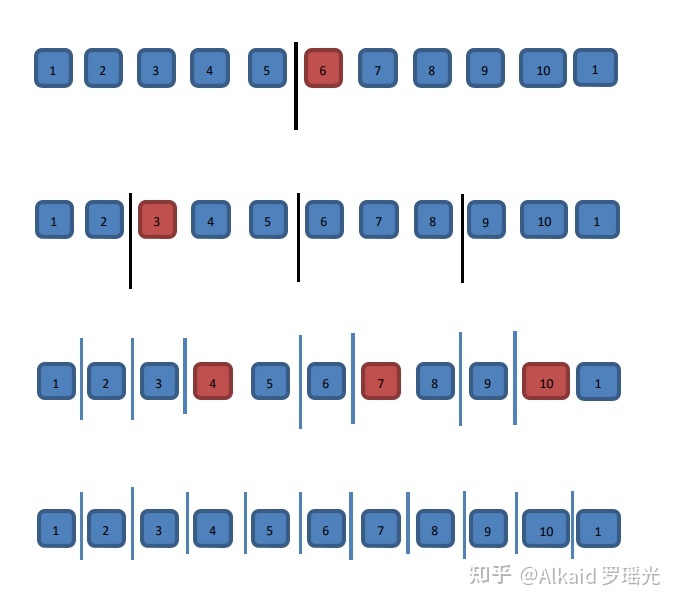
Since of the bright flashes, may I follow the operation method. The refinement method of DETA's first catalytic calculation is first reflected in DETA parser, such as the refinement of semantic part-of-speech analysis, the optimization of flow valve, the irrational conditional transformation of discrete data, the filtering of the same frequency operator, and the filtering of calculation peaks. These optimization methods of human thinking gradually form a system, which not only changes the design mode of DETA's works, but also changes the author's research and development philosophy.

**3.2 DETA Catalytic computing development,**

R&D is not successful every time. In the process of butterfly calculation optimization of Fast Fourier, I coded the features of discrete DCT in complex numbers, which took me one month, but failed. I remembered that I said in Weibo at the beginning that I could speed up the calculation of Fast Fourier by 200 times, but I really didn't give up. Since butterfly calculation optimization was unsuccessful, I tried to sort the small peaks by fast left-right comparison. I was excited when I saw the 10th generation of single machine random double with a sorting speed of 12 million arrays per second of quick sorting. My thought is right, and thinning logic is an important way of human thinking. Here, the author designed an argumentation paper when designing fast word segmentation and extremely fast peak filtering catalytic sorting, as follows:

**Theory on YAOGUANG's Array Split Peak Defect**

Mr. Yaoguang. Luo

稍后继续优化， 罗瑶光

Outline ：In the common software development factory, engineer always did more and more interactions with data structure and math algorithms. Especially in the recursion, convolution, sort and generic loops, scientist likes to find a simple, more sufficiently and alignment way to face the project requirements with the large association. For instance me, I really got a real world problem at this domain while I use quicksort,also for other project such like DETA parser. What is the peak array split defect? How does it count the real world problems? Why need find it and how to get the nice solution? Cool, this paper will cause an implementation about our goals, ok now, keep forward to the context where I talking as below, thanks For more theory DETAils and the source code implementations please check the bottom reference section. Peak, Array, Split, Defect, Recursion, Convolution, Sort, Generic

**Goal one: Quicksort Yaoguang.Luo 4D**

DETAils：

For example the array input as below where we gave 11 digits.

1.1 The first split, we could see the digit-6 will auto arranged to the right part.

1.2 And the second split, we may see the digit-3 will be auto arranged to the right part

1.3 The third split, we may see the digit-4.7.10 will be auto arranged to the right part

2 Thinking:

After the split array showing, we could see clear that the big problem about the asymmetry defect, as I did an annotation of N, so the i of N

will absolutely find a n/POW（2，i）value points, as an insufficiency asymmetry defect model, I fall in thinking…if I do any compute theory as

the same with this model style, for example in the ecursion or inner loops, it will autonomic separate to the 2 different process way, it necessary

to do indifferent flows.

3 Problems:

So, after the above thoughts, I may get any flashes, First, the even and odd digits both are asymmetry while in the Differential loops. For this

noise, I defined as (Tinoise Peak) Second，once we did a split compute under this model, it must get more unfair sets. I defined as (Tinsets defect).

Third, if this model almost in the messy and timer data system, it will catch more time and asserts wastes or exceptions.

4 Solutions:

For the god like, I find three solutions while I currently enrolled in my projects. First: computer logic acceleration, at least it can avoid the

waste of the compute by using inner process optimism. -- To avoid the deep recursion. Second, reduce the compute sets. For any less memory system,

we may reduce more and more memory garbages after we reduce the inner register or temp value sets. Third, we may make an optimization of the function

logic where to instead the old complex functions. Those ways include the condition, algorithm, method or discrete optimization. End, we may use

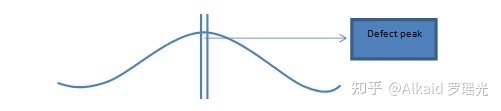
mathematics of double differential, deep definition, acquisition or polynomial to get the solutions.

5 True Instances



贴个最新TopSort5D: [https://github.com/yaoguangluo/sort/blob/master/LYG9DWithDoubleTopSort5D.java](https://link.zhihu.com/?target=https://github.com/yaoguangluo/sort/blob/master/LYG9DWithDoubleTopSort5D.java" \t "https://zhuanlan.zhihu.com/p/_blank)

Let me show the algorithms here,From this code: in a common quick sort way, the recursion based on the average deep split, suppose the initial array length is N{1,2,3…n} is an Odd, so the separate two arrays will cause an asymmetry defect, those timer asymmetry compute peak collection will cause more and more probability problems such like jam, lock, time waste and heap increment. The odd peak binary split as below：How to avoid those timer distinction peaks? I go more absolutely research where focus on these problems, first, differential flows. This flash is not suitable for here, May good for the DETA parser, I will show you later. Second, compute acceleration. Yep, this is a good way, for example find the big X as the code blew, it will cause the while loop ability accelerations.



int x= a[lp]< a[rp]? a[lp]: a[rp];

（int x= a[lp]< =a[rp]? a[lp]: a[rp];）suitable for string swap top sort 5D

Third, De Morgan condition differential as the code below, it will cause the condition ability accelerations.

while(!(a[lp1]>x|| lp1>=rp)) {

At last but the least, value reduce, code optimization both are very important way of the peak avoid filter.

**Goal Two: DETA parser**

6 DETAils：

Last year I help my father to develop the study software about getting the medicine data collection for quick search. I’ m going to try to

build a search engine system, input format is a string, how to get a Chinese string array split?

Convolution: length indicate by marching Nero index tree as below: 2|1|2|3

Convolution: POS indicate as below: n |c |n |adv |adj

Convolution: split

7 Thinks:

While I use this way on my DETA project Chinese separations, I met so many problems, the more and more important problem is the POS frequency peak waste, my POS flow functions will spend a lot of time to do the low prior convolution split condition check first… it cost me a lot of time… after I did a collection of my projects, the results are clear.

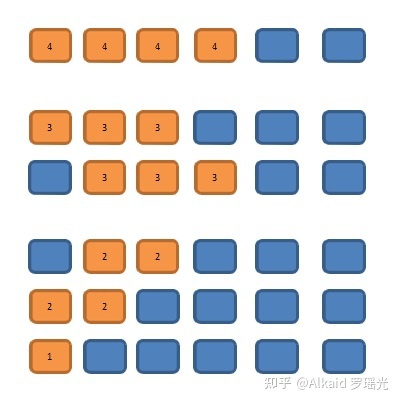
8 Problems:

First, convolution kernel gets asymmetry problems. Second, unnecessary conditions check loops. Third, unimportant heap register values. End, values sets and conditions sets too more.

9 Solutions:

First, format the convolution kernel of the index dictionary tree, for example I use the char ASCII as the index length to reduce the match time of the convolution length indication. Also, I define the POS convolution kernel size less than five. Second, I did a condition frequency statistic, and re arrangement it, at the same time, reduced a lot of the inner sets to avoid the compute pause.

10 True instance:

做成了图片，稍后修改，罗瑶光

11 Finally I developed a convolution String array split way for marching as below: orange color are presplit sets

11.1 check 4 chars slang

11.2 Check 3 chars key word

11.3 Check 2chars normal word

In order to make a compute acceleration, I did 2 string builder array to store a pre order sets.

in order to make a PCA POS acceleration, I did 5 chars marching array to store a post order sets.

It seems the Nero Index, NLP and POS for the PCA separation with convolution kernel marching by using stepwise iterative differentiation got much higher sufficiency.

**3.3DETA Catalytic computing application,**

In order to demonstrate the importance of DETAiled logic, I began to integrate this logic concept into my YangLiaoJing and all my soft works. When I saw 13 million high-accuracy word segmentation per second, 6 million mixed phonetic symbols per second, 12 million double arrays sorting per second, and other amazing works came out, I began to sigh my own cognition. I unreservedly opened up all these ideas and works, which hope aroused humanoid thinking [Resonance Promotions - More Than Promotion - We Care A Lot - Andi Appel](https://link.zhihu.com/?target=http://resonance.at/" \t "https://zhuanlan.zhihu.com/p/_blank) present, the significance of differential catalysis has included seven categories: frequency valve (von Neumann) differential, discrete logic (De Morgan) differential, high-frequency function degradation, conditional refinement differential, executive mode differential, giant system (Qian Xuesen) module differential, and mathematical differential (Newton, Blainez), which is the only way to catalyze humanoid DNA evolutionary algorithm. And made a DETAiled demonstration and summarized as follows

[https://gitee.com/DETAChina/DETAParser/blob/master/wordSegment/org/tinos/engine/pos/imp/POSControllerImp.java](https://link.zhihu.com/?target=https://gitee.com/DETAChina/DETAParser/blob/master/wordSegment/org/tinos/engine/pos/imp/POSControllerImp.java" \t "https://zhuanlan.zhihu.com/p/_blank)

Demonstration of differential algorithm of POS water valve with DETA fast segmentation;

Demonstration topic: Differential catalytic calculation can well observe the execution flow logic of quantum state function. Through statistics, high-frequency function can be advanced gradually according to von Neumann state, and low-frequency logic can be eliminated and screened gradually.

Demonstration result: The demonstration was successful. Reduce the traversal times of irrelevant code. Greatly improve the computing power.

Demonstrating influence: the main way of evolutionary mechanism of humanoid DNA evolutionary algorithm.

[https://gitee.com/DETAChina/DataSwap/blob/dceeb0b06f726d640553964058d85b736354ac89/src/org/DETA/tinos/array/L](https://link.zhihu.com/?target=https://gitee.com/DETAChina/DataSwap/blob/dceeb0b06f726d640553964058d85b736354ac89/src/org/DETA/tinos/array/L" \t "https://zhuanlan.zhihu.com/p/_blank) [YG4DWithDoubleQuickSort4D.java](https://link.zhihu.com/?target=http://yg4dwithdoublequicksort4d.java/" \t "https://zhuanlan.zhihu.com/p/_blank)

Demonstration of The 4th generation example demonstration of filtering sorting algorithm for Luo yaoguang s small peak calculation;

Demonstration topic: Differential catalytic calculation can be combined differentially with discrete mathematics system to filter high-frequency functions where from the digital logic level. Ensure smoothness

Demonstration result: The demonstration was successful. Greatly increase the calculation speed.

Demonstration influence: humanoid DNA evolutionary algorithm can effectively smooth the peak of computation.

[https://gitee.com/DETAChina/Data\_Prediction/blob/master/src/org/tinos/DETA/tsp/YaoguangLuoEulerRingTSP2D.java](https://link.zhihu.com/?target=https://gitee.com/DETAChina/Data_Prediction/blob/master/src/org/tinos/DETA/tsp/YaoguangLuoEulerRingTSP2D.java" \t "https://zhuanlan.zhihu.com/p/_blank)

Demonstration of the second generation of differential TSP algorithm for Luoyaoguang Euler forest business travel ring Demonstration topic:

Differential catalytic computing can optimize the thinking mode of traditional complex logic at cognitive level, and fundamentally change the cognitive process from the beginning.

Demonstration result: The demonstration was successful. Differential catalytic computing can change the traditional cognitive style in some social fields.

Impact of demonstration: Humanoid DNA evolutionary algorithm can effectively select the fastest algorithm module and cognitive module to do calculation in specific fields, and improve computing power.

[https://gitee.com/DETAChina/DataSwap/blob/master/src/org/DETA/tinos/string/LYG4DWithChineseMixStringSort7D.java](https://link.zhihu.com/?target=https://gitee.com/DETAChina/DataSwap/blob/master/src/org/DETA/tinos/string/LYG4DWithChineseMixStringSort7D.java" \t "https://zhuanlan.zhihu.com/p/_blank)

Demonstration of the 7th generation example of Luo Yaoguangs conditional differential sorting algorithm for light image strings;

Argument topic: Differential catalytic computing can unify conditional functions considerably, reduce logic complexity, and continuously optimize and focus.

Demonstration result: The demonstration was successful. Differential catalysis algorithm can split the local modules of the whole function by vpcs logic, and form the purine element of initon operation of DNA peptide chain.

Demonstration influence: the guarantee of autonomous evolution mechanism of humanoid DNA evolutionary algorithm.

[https://gitee.com/DETAChina/DETA\_PLSQL\_DB/blob/master/java/org/lyg/db/plsql/imp/ExecPLSQLImp.java](https://link.zhihu.com/?target=https://gitee.com/DETAChina/DETA_PLSQL_DB/blob/master/java/org/lyg/db/plsql/imp/ExecPLSQLImp.java" \t "https://zhuanlan.zhihu.com/p/_blank)

Demonstration of PLSQL differential compiler for DETA Socket stream programmable database engine;

Demonstration topic: Availability of multi-condition execution of differential catalytic calculation.

Demonstration result: The demonstration was successful. Differential catalysis algorithm can provide reverse observable operation and maintenance guarantee for functional system operation.

Demonstration influence: the comprehensive application practice of conditional differentiation, logical differentiation, high-frequency valve preposition and other functions of humanoid DNA evolutionary algorithm.

These arguments were a year ago. At present, many works have been in a good follow-up state because they are developed as subsystems of the project of YangLiaoJing. Over the years, I have been thinking, what is the final expression of DETA acquisitive logic? I have never stopped exploring, and I have always been absolutely focused.

**4 DETA Finding Initions**

I have been thinking, what is the final expression of DETA Acquisitive Logic? I have never stopped exploring, and I have always been absolutely focused. I must find the final expression of these logic. From 2018 to 2019, I thought that the final expression of logic refinement must not be as simple as AOPM and VPCS. VPCS is just a refinement layer of AOPM, so how can VPCS be refined? So I began to sort out my existing things, my works and soft thoughts. God, I can only make persistent and absolute focus on what I have. I have to make a bet.

**4.1DETA Finding initions history,**

What's under VPCS? What is the essence of a function? At school, I got some basic answers. the primitives of DNA are ACGTU purine and pyrimidine, the primitives of back-end architecture are VPCS, the primitives of thing logic are AOPM, and the primitives of database are IDUC addition, deletion and modification. the primitives of function are IOAON Input, Output, And, Or , Negation, which I can only find in the knowledge structure I can understand and have. how to demonstrate? How to confirm the argument?

**4.2DETA Finding initions development,**

The first demonstration process is DETA word segmentation. In 2019, I continued to refine, optimize and refine the word segmentation, and found an exciting argument. My word segmentation function was continuously split rationally.Finally, a pile of simple combination application fragments of addition, deletion and modification were displayed by IOAON. For the most powerful argument, when I was processing nouns in word segmentation, the final function was formed. Memory takes out 4 words, compares 4 words proverbs, does not? then compare 3 words, does not? then compare 2 words, and does not? then split into single words. This process is summarized in one sentence as a combined decision-making process of adding, deleting, modifying and checking memory data according to John Von Neumann's time flow form. When I think about this, my eyes shine. IDUC is not only the operation mode of database, but also the operation mode of memory data, and it is absolutely focused continuously! ! ! Assuming that IDUC is effective for all data operation modes, assuming it is successful, if it is coded, it is a very strict coding mode of data DNA. I found it! I began to refine my sorting algorithm, word segmentation algorithm, ETL, YangLiaoJing, etc., and found one thing in common. All my works were refined to the rational function level that I could understand, which were small fragments of the combined decision-making process of adding, deleting, modifying and checking linear, multidimensional, database and memory data. These fragments can be coded effectively.

**4.3DETA Finding initions application,**

With this in mind, I have determined a ternary mapping coding mode of DNA to ETL neuron nodes, AOPM -VPCS- IDUC 3D coding mode. 4\*4\*4 Then each primitive is a 64-bit space, which is the computing primitive I have been looking for decades.

**5 DETA DNA decoding**

If the AOPM -VPCS- IDUC 3D computational neuron mapped by DNA IDUC is established, how to decode it? I thought about what I have, about YangLiaoJing! It is the only way that I can do at present to construct the system of YangLiaoJing and demonstrate this idea and technology. It is still the absolute focus of that sentence.

**5.1DETA DNA decoding history,**

At present, what we know is that DNA pep-tide group has billions of long, double chains and 24 pairs of chromosomes. there are five primitives of ACGTU. if ACGT can encode human higher intelligence logic, then human-OID data DNA with IDUC unit can also write hundreds of thousands of business transaction processing logic of AOPM VPCS. these two logics do not conflict.

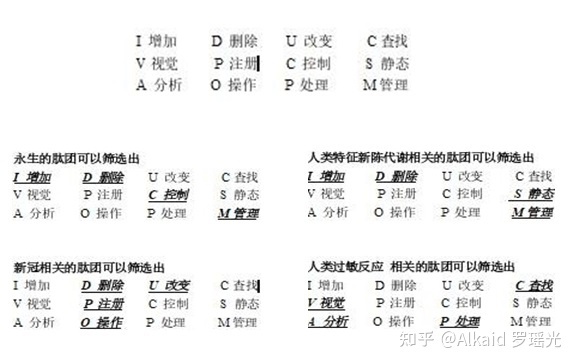
**5.2DETA DNA decoding development,**

These two logics do not conflict. Are they one? I got an exciting argument, and I had to find an argument, so I screened my study and work style in recent 20 years, my own thinking style and the execution logic style of my software, hoping to find a negative theory to overturn it, but unfortunately I couldn't find it. So I followed up and re-examined my soft works, optimized them, and found that once optimized to the edge of rational function to irrational function, they were all linear. Small fragments of the combined decision-making process of adding, deleting, modifying and checking multidimensional, database and memory data. These fragments can be coded effectively. AOPM -VPCS- IDUC seems to explain all the answers I want.

**5.3DETA DNA decoding application,**

In order to overthrow my argument, I began to look for arguments everywhere to attack this argument. First, I found the topic of eternal life. According to AOPM -VPCS- IDUC, IDMC is true. Since it can be perfectly explained, I found the topic of infection in COVID-19, that is, DUOP is true, which is an exciting conclusion! I have been searching for answers to all the problems for decades from AOPM -VPCS- IDUC, so I started mapping and coding as follows:I got a clear

**DNA theorem:The essence of DNA is a combination indexing link list of four meta-operations of adding, deleting modifying and Querying data. 10-04-2020 DC**



**6 IDUC DNA and Its Applications,**

These are all the later stories. The application is too wide. First of all, my ETL began to expand in the three-dimensional direction to better serve medicine. Secondly, virus immunology and immortal virus exploration will never stop. Why ETL is used as the expansion point is inspired by my OSGI paper on October 17, 2013. It is as follows

**The Darwin's Theory of The Artificial Intelligence**

In the latest knowledge engineering structure, the traditional expert system occupies a dominant position, but the world's demand system is in a changeable operating environment, so the data persistence theory is a goal to strive for. Artificial intelligence software, too, can't escape the disadvantages brought by natural updating. Where artificial intelligence will go, it will be planned naturally. Just like Darwin's theory of biological evolution, the new intelligent system standards are naturally selected by needs, which is the central idea I want to express. In the past 50 years, some classic software can't escape the choice of demand, and finally it turns yellow and dim. Of course, some enterprises rewrite and upgrade their products desperately. Because of the aging of core developers, new reformers can't master the original development ideas and theories. Finally, the quality of products suffers a huge impact and suffers heavy losses. A new software development theory needs to be confirmed, which is my thought. Software, too,needs an evolutionary system with self-artificial selection.

Through the recent construction, design and coding test of UNICORN AI software, I found that many computer theories created by fantasy have great differences in actual programming analysis. I used JAVA-based language, and I found that the inheritance of JAVA did not meet the language standard with evolutionary thought, but its methodology in this initial evolutionary standard test was far superior to C/C++. I didn't bring any troubles to my actual programming when I wrote JAVA program in C style, but JAVA still needs to be improved. For example, you abstract a parent class, and the variable function of your subclass still needs to be written in the way of "OBJECT parent class = (subclass) parent class" to make subclass operation. If grand children inherit sub classes, how can OBJECT get grand children? (I use the OBJECT subclass to inherit the parent class, and then the OBJECT subclass = (grandson) subclass. In this way, the grand children get the operation), but this is a big problem of dynamic memory structure allocation!

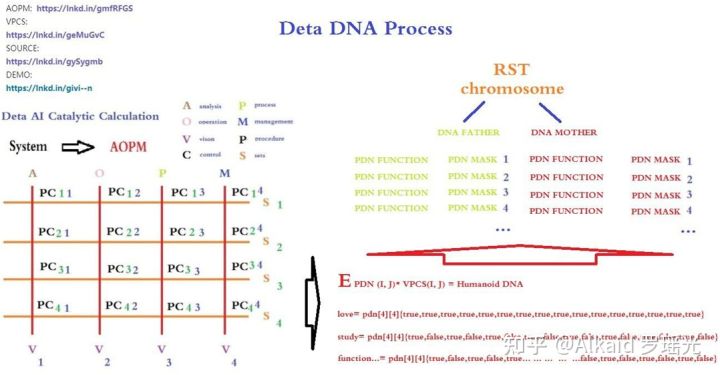
The design is rather cumbersome. JAVA still stays at the level of primary language evolution, and does not have advanced evolutionary ideas. Secondly,if a subclass has more than one grand child, only the subclass can operate, and the parent class cannot operate accordingly. This is also a criticism. Is it realized by adding OBJECT subclass = (grandson) subclass and OBJECT parent class = (subclass) parent class? This is even more complicated.

Through the above description, I have my own views, but I still chose JAVA, even though it is cumbersome, but there is no mistake, because it will be more cumbersome to implement in the underlying language. There are more traps. It is a natural choice for artificial intelligence to choose JAVA. JAVA and C# are both high-level languages, but JAVA's personality is born to deal with data, because JAVA was a WEB language in its early days, and WEB has unique advantages in dealing with data information, which is a real example of JAVA evolving into a data analysis language. C# has been improving itself in this problem, just like JAVA, even like JAVA, but there is no system to evaluate it. The WEB data engineers who applied JAVA in the early days will not transfer to C#, so the biggest advantage of C# is that it is only applied to controls on WINDOWS.

Through this description, it only proves that the greatest advantage of any language is only reflected in its creative theory and thought at the beginning of its birth. Therefore, JAVA and C# are not comparable at all. Because their original creative theory, system and ideological structure are different. If JAVA and C# really fail, finally, through the prediction of evolutionary thought, JAVA will go in the direction of graphics, big data analysis, WEB and C# should go in the direction of interface, control and WINDOWS device integration. The evolution of artificial intelligence software is mainly divided into update of parent class, variation and inheritance of subclass.

JAVA is perfect for dealing with subclass functions, and people who have used JAVA to develop large projects are quite experienced in dealing with interfaces and inheritance. But is there any variation in JAVA? It can be said that there is no, for example, when the parent class PUBLIC attribute 1 = 0; Sub classes can't have the PUBLIC attribute 1=1, which is a mutation failure problem. JAVA is flexible, but not as flexible as scripting language. Secondly, I want to say that the variation of JAVA is a variation with quotation marks. Its characteristic is that sub classes modify the functions of the parent class, and sub classes of JAVA can modify the processing procedure of functions with the same name of the parent class. However, you have to make the subclass and the parent class have the same function names. This is a JAVA default mechanism, which executes the same name of the parent class first, and then executes the same name of the subclass. Then return to the parent class, and then return to the procedure of. Therefore, the function with the same name can be modified in sub classes, thus ensuring parameter variation. In this way, the software is also very flexible and unique in the actual writing process.

Finally, I have a deep experience through the expression of language evolution thought and program evolution thought mentioned above. Every language needs its needs if it is to be deeply rooted,and its functions should be selectively evolved in the needs. Otherwise, this is the biggest reason why languages have been eliminated. I don't like to see various languages emerge one after another in today's world. This is the biggest criticism that many languages have not evolved and can't reflect their needs. Secondly, languages need to be extended, and the appearance of API class libraries and some architecture systems of high-level languages is a good proof of extension. Finally, variation is similar to scripting language, which is flexible and convenient.What about software? The same is true for software. It is particularly important to choose a language that suits your own needs. Secondly, the architecture of the software should have loose coupling, which is similar to OSGI and FELIX. The OSGI idea of KNIME is consistent with that of LIFERAY. Although the API design style is different, the effect is very thick. Biology needs Darwin's thought, and artificial intelligence also exists, which is the basis of demand persistence. This is also the basic condition for my research and development of UNICORN AI platform. Now I have enough confidence to continue to focus on the argument of making ETL mapped by my DNA code with evolutionary system reuse the perfect guarantee requirement persistence. On how to use ETL to map the code, I will go back to the previous year again and analyze the design idea of this picture at that time as follows



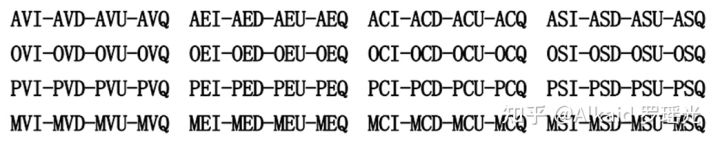
Yes, I have an idea already! ! Exciting, ETL node three-dimensional classification. this vocabulary

**7 IDUC VPCS AOPM 3D Nero Cell and Its Applications**

These are the following words. ETL begins to expand in the direction of 3D. First of all, I want to design the 3 D functional area of neurons based on the DNA mapping of DETA IDUC. This is the real human-OID independent thinking way that I can understand.This is the first index application idea of DNA coding manual in human history.

I will Change Org.lyg.node.medcine.addchufangattributeH.jar into Org.node.a.v.c.u.medcine.addchufangattributeH.jar

This a.v.c.u will form a DNA mapping system code for analyzing visual control changes, which is convenient for future evolutionary optimization tests. After that, I will systematically encode these ETL index mapping sets into DNA index chains for YangLiaoJing. The ideas given to me in this paper are all creative ideas. Thank you for everything. I can first design AOPM VPCS IDUC INITONS 64-bit single chain for the integrity of coding, such as



This index mode, even though it is not the final index of organic DNA of human beings, has become the first mapping execution mode of human-OID artificial design representing evolutionary encoded DNA.

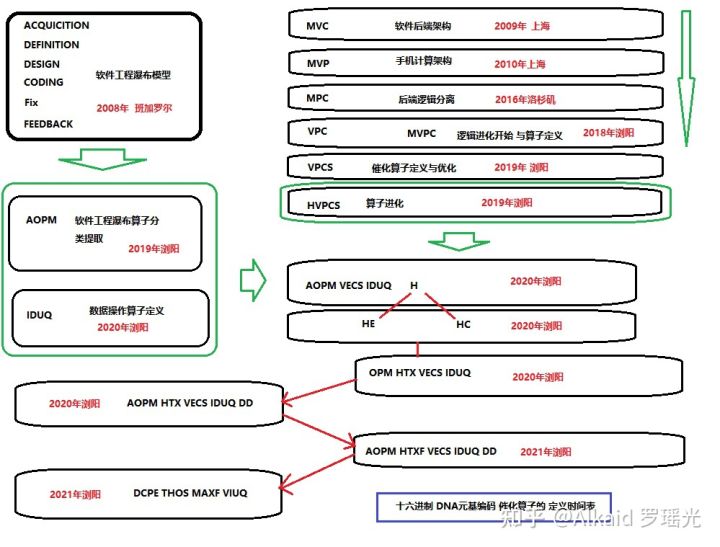
**Not The End:**

In this paper, I spent 20 years of basic study, 7 years of work practice, 2 years of open source implementation, 18 corresponding data sub-projects, 6 data fields, 2 big data works, 7 artificial intelligence papers, and gradually demonstrated some essence, such as The essence of DNA is a combination indexing link list of four meta-operations of adding, deleting, modifying and Querying data.Execution mode of neuron calculation: a neuron time series calculation chain of specifical function calculation by mapping of DNA coding index reflection.

The essence of adapting to the environment is that DNA coding indexes map related neurons compiler link to achieve better addition, deletion,modification and query the environment data.

The essence of human-OID evolution offspring: the offspring produced by optimizing the hybridization of the same coding logic part in the DNA encoding index chain mapped and retained by the neuron calculation method of data efficient processing.

It seems that the topic will never end, so I might as well boldly put forward new arguments, continuously and tenaciously focus and demonstrate, and I still enjoy it.

目前，映射算子最终成为了元基算子。罗瑶光

**8 Refer** (原文文章太长，著作权文件中含有完整refer清单)

第八章\_肽展公式推导与元基编码进化计算以及它的应用发现.

initon定义，肽元基单位如AOPM VECS IDUQ TXH DD（**F 元基不在此章出现**）

**TVM定义, 离散语义肽虚拟机**

**PDW定义, 离散语义肽元基词汇**

**PDC定义, 离散语义肽元基编码**

**PDN定义, 离散语义肽元基结构**

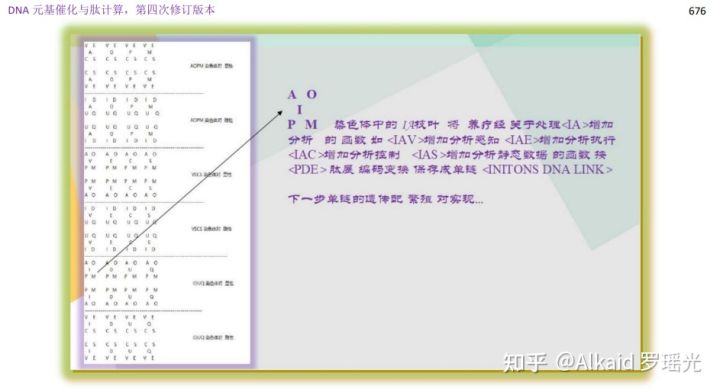
**PDE定义, 离散语义肽展公式变换**

**PDS定义，离散语义肽展逐级丝化变换**

1 DETA INITONS classify/德塔元基分类。refer page 674

2 DETA INITONS PDN words root/德塔元基分类词根。refer page 676

2.1 元基染色体观测。refer page 676



3 DETA INITONS PDN words/德塔元基分类词典。refer page 677

3.1 元基染色体分类的编码根。refer page 677

4 DETA TVM/德塔词典肽翻译虚拟机。refer page 678

4.1 元基染色体的分层级表达。refer page 678

5 DETA TVM applications/德塔肽翻译虚拟机应用技术。refer page 679

6 DETA TVM PDC/虚拟机应用优化。refer page 680

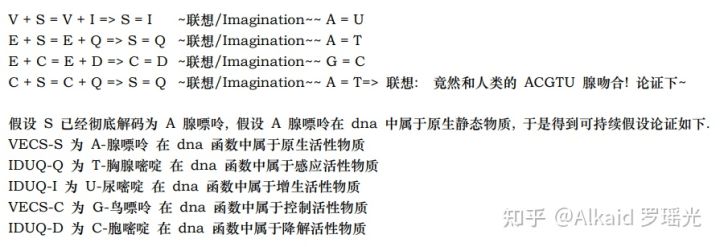
6.1 元基文本加密方式观测。refer page 680

7 DETA TVM PDE/德塔肽翻译推导。refer page 681



7.1 元基进行离散数学推导观测（狄摩根定理）。refer page 681

7.2 ACGTU解码。refer page 682

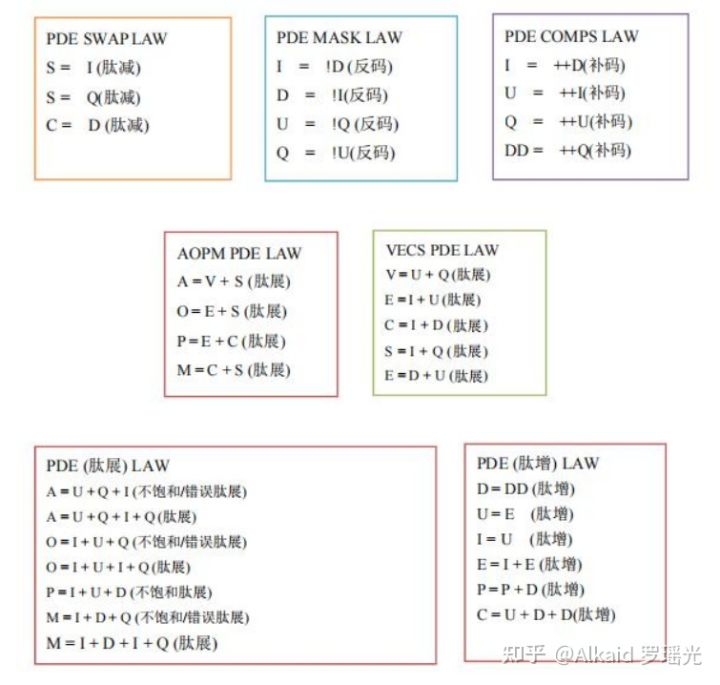


8 DETA TVM PDC functions/德塔肽推导函数化。refer page 683

8.1 三元PDC进行意识词汇的设计。refer page 683

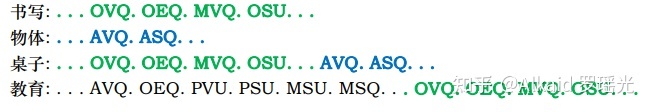
9 DETA TVM PDC function optimization and PDE/德塔肽推导函数逻辑优化。refer page 683

9.1 肽展公式推导集 。refer page 685

肽展公式PDE

10 DETA TVM PDE Logic/德塔肽推导函数逻辑优化成肽展公式化。refer page 686

11 DETA TVM PDE and its application/德塔肽展公式应用论证技术。refer page 687



12 TVM humanoid life Research/应用在类人生命进化中。refer page 687

13 Eternal Research/应用在类人生命永生探索领域。refer page 691

**章节的著作权文件列表：**

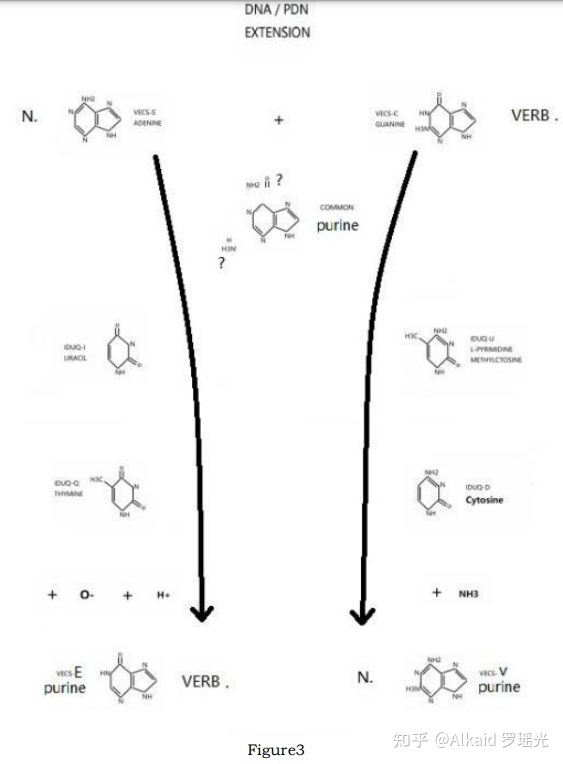
第九章\_DNA催化与肽展计算和AOPM-TXH-VECS-IDUQ元基解码

语义逐级表达的方式，十五元基解码。

1. 推导与定义:甲基胞嘧啶在 DNA 编码和肽计算中具体定义为 IDUQ-U 变嘧啶。refer page 695

2.推导与定义:2 氨基腺嘌呤在 DNA 编码和肽计算中具体定义为 VECS-V 变感腺嘌呤。refer page 698

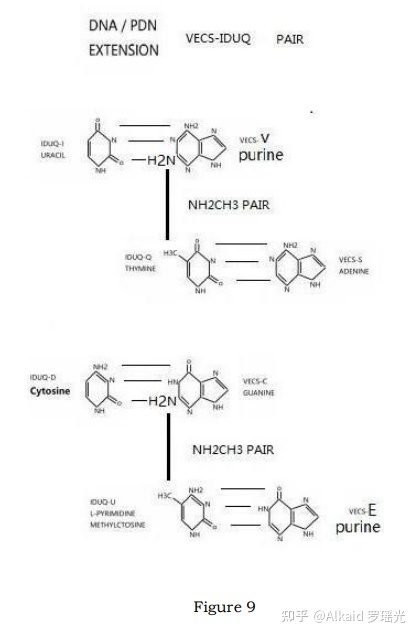
2.1 PDN extension 腐蚀。refer page 699



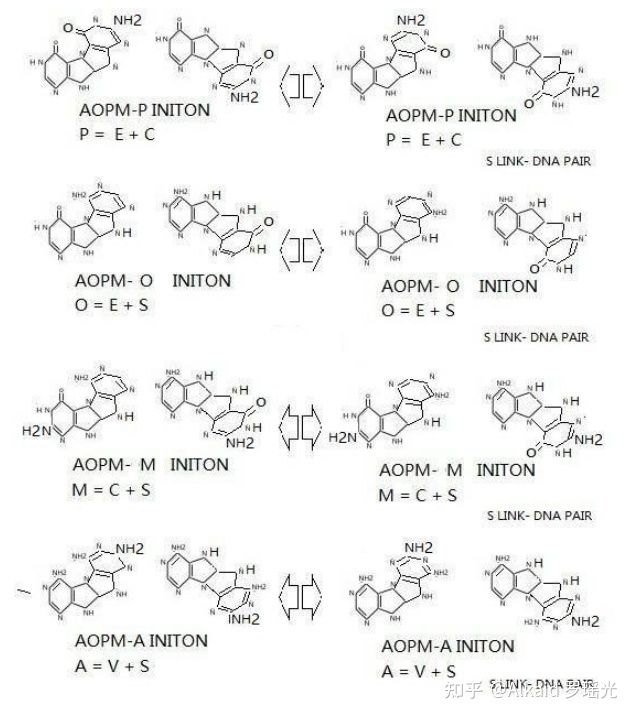
3.推导与定义:次黄嘌呤在 DNA 编码和肽计算中具体定义为 VECS-E 尿变嘌呤。refer page 699

4.推导与定义:AOPM-A 变胸腺苷, AOPM-O 尿胞变腺苷, AOPM-P 尿胞变鸟苷, AOPM-M鸟腺苷的S形螺旋纹血氧峰触发器分子式催化计算严谨完整过程。refer page 700

5. 推导与定义:VECS-VECS 嘌呤对, VECS 嘌呤弧, VECS-IDUQ 碱基对, IDUQ-IDUQ嘧啶对的催化模型。refer page 701



5.1 上下 旋弧 rotation。refer page 701

AOPM rotation

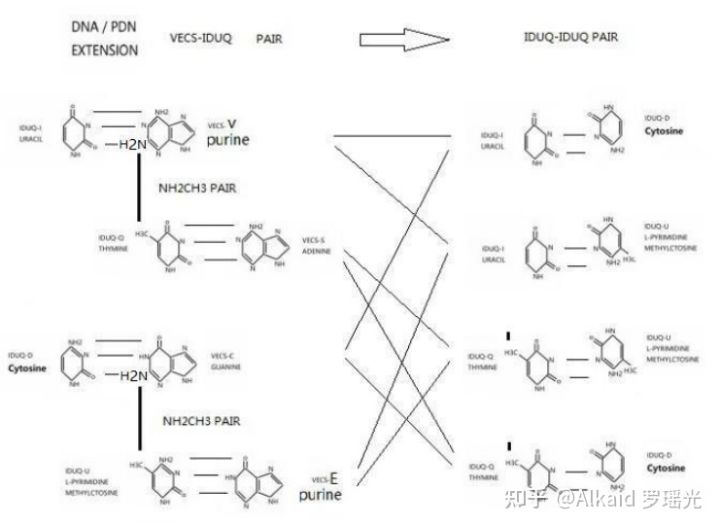
5.2 S link dna pair。refer page 704

6.推导与定义:次黄嘌呤, 尿变嘌呤 VECS-E=IDUQ-U 变嘧啶, 甲基胞嘧啶 E=U 全新DNA计算碱基对。refer page 705

7.推导与定义:2 氨基腺嘌呤, 变感腺嘌呤 。refer page 705

7.1 VECS-IDUQ pair。refer page 705

7.2 VECS-IDUQ pair rotation。refer page 706

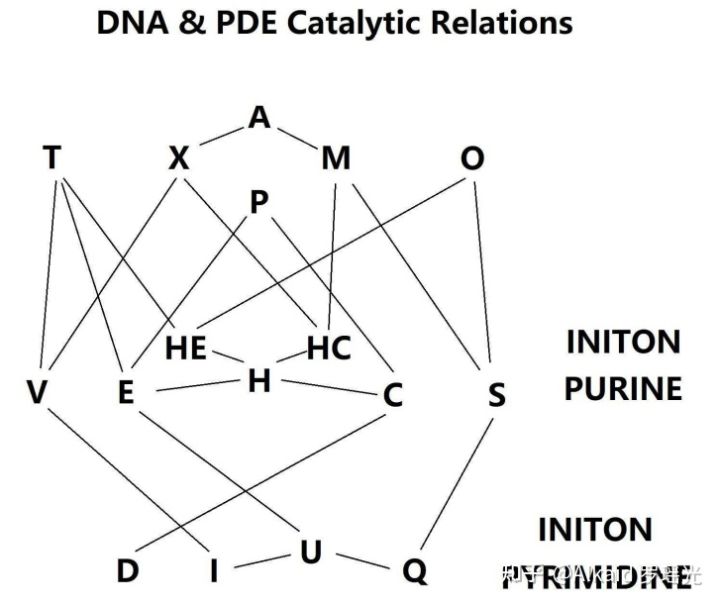


8.推导与定义:碱基对 Rotation 观测与黄嘌呤在 DNA 编码和肽计算中具体定义为VECS-EC尿变鸟嘌呤。refer page 706

8.1 VECS-VECS pair rotation。refer page 707

9.推导与定义:尿变鸟嘌呤, 黄嘌呤肽展计算 AOPM-OP-T 变感腺尿变苷与 AOPM-OP-X 变感腺鸟苷。refer page 708

9.1 DNA and PDE catalytic relations。refer page 708

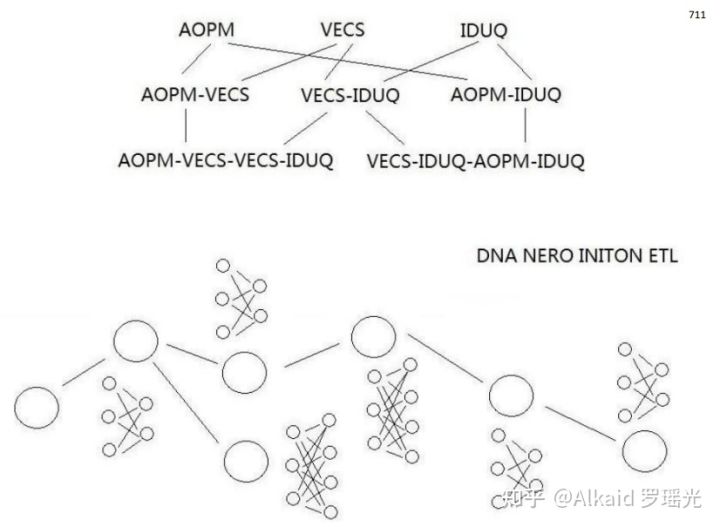


10.归纳与定义:DNA 与 TX-H-U 元基解码。refer page 709



11. 推导与定义:DNA 元基催化计算与 ETL 肽展神经网络计算流。refer page 710

11.1 rotation S link encoding & decoding。refer page 710

PDE ETL

**章节的著作权文件列表：**

第十章\_DNA非卷积视觉技术

定义：**非卷积腐蚀视觉，一般指在不采用卷积内核进行图片低速计算的方式下，图片每个像素仅仅做一次遍历来模拟酸碱腐蚀肽展公式计算，目的是 达到人肉眼观测不到而又难以辨认的色阶群进行观察拉伸。用于极速图片观测识别领域。**

定义人 罗瑶光

测试原图来自医学教材

**DNA非卷积视觉技术原理**

1 DNA非卷积视觉技术将图片像素0~255的区间捕获后，进行元基进制变换，产生了离散色阶。

2 这个色阶，与之前像素亮度色阶完全不对称，利用这一点不对称，可以将邻近像素差不大的像素团进行颜色差 的拉伸。

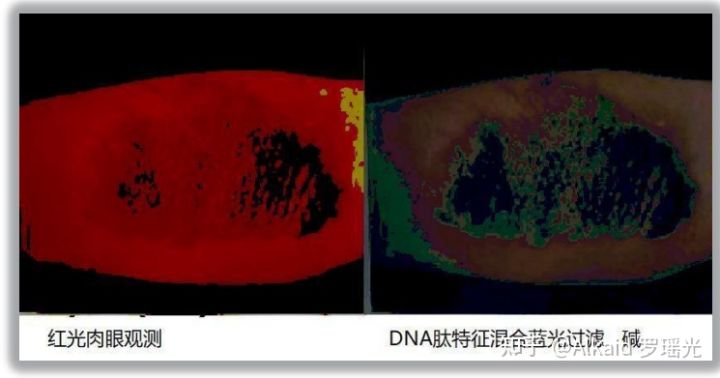
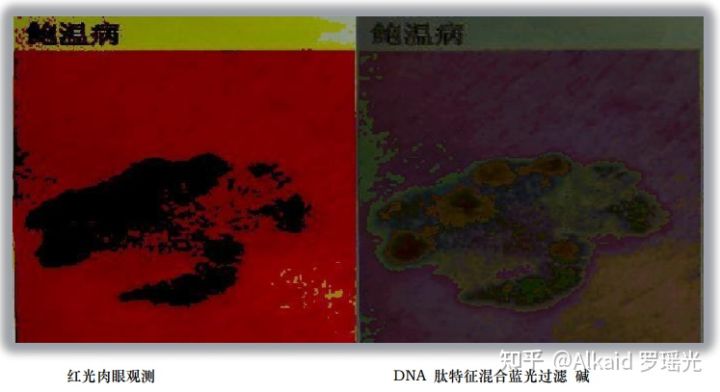
3 颜色差 的拉伸方式 为 有系统的 10进制色 变换为 元基色 进行固定的 酸碱 更换。实践观测后发现价值巨大。

**DNA非卷积视觉技术原理**

1 DNA非卷积视觉技术将图片像素0~255的区间捕获后，进行元基进制变换，产生了离散色阶，

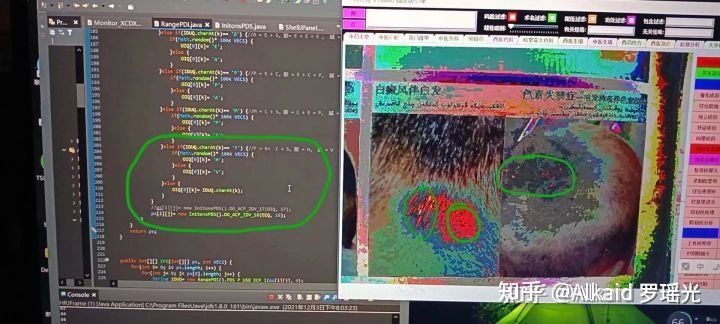
2 这个色阶，与之前像素亮度色阶完全不对称，利用这一点不对称，可以将邻近像素差不大的像素团进行颜色差 的拉伸。

3 颜色差 的拉伸方式 为 有系统的 10进制色 变换为 元基色 进行固定的 酸碱 更换。实践后发现价值巨大。

测试原图来自医学教材测试原图来自医学教材

**肽腐蚀，**

1 DNA非卷积视觉 用元基的酸碱变化规律定义为肽腐蚀。refer page 723

测试原图来自医学教材

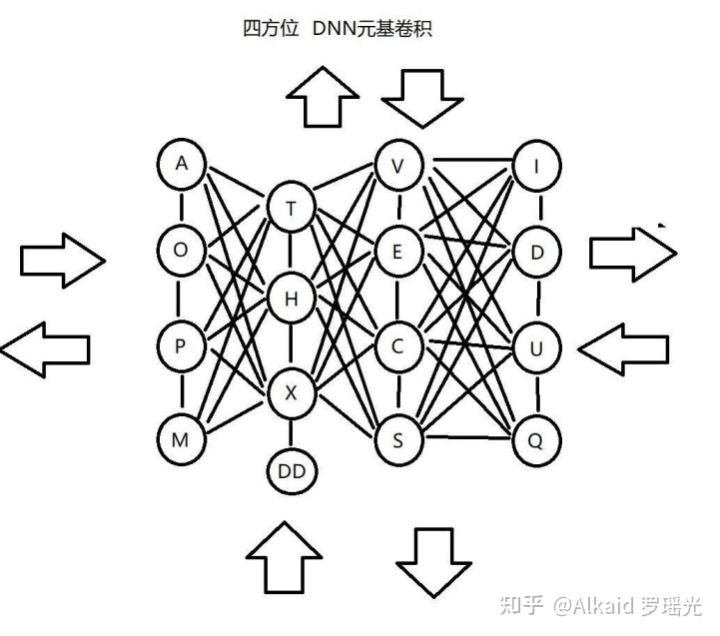
2 DNA非卷积视觉的肽腐蚀 需要将10进制数字变换成生化进制数值。refer page 724

3 DNA非卷积视觉的肽腐蚀观测体征体现在颜色区间上。refer page 735 ，742

4 DNA非卷积视觉的肽腐蚀浓度用概率百分比来标识。refer page 756 monitor.fac x y..

**元基的视觉叠加与表达方式，**

1 视觉流计算。refer page 756



2 animation动画。refer page 744

3 颜色的腐蚀精度调节。refer page 757

4 肽展公式的应用。refer page 723

**时序视觉模拟机，**

1 线性神经网络卷积计算 。refer page 772

2 卷积计算的方式。refer page 773

3 非卷积视觉的应用。refer page 756

图片识别应用，测试原图来自医学教材

**费洛蒙的计算方式，**

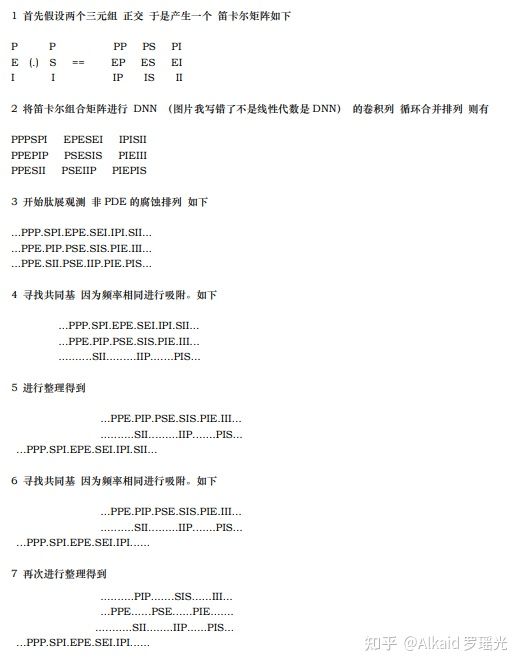
1 CNN卷积元基PDC扩展。refer page 774

2 邻近元基PDC代谢 共同基。refer page 774

3 PDC链结构rotation。refer page 774

4 丝化散开与腐蚀。refer page 775

（**下图的丝化是元基的概率组合的归纳如矩阵的新陈代谢模拟发散。不是肽展公式PDS丝化过程。下图的1和2，作者认为是一种比较合乎情理的又具有代表性的概率矩阵组合归纳。罗瑶光补充20220307）**

罗瑶光的费洛蒙计算发散

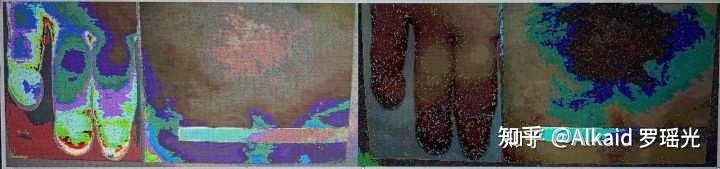
**应用**

1 舌诊观测应用。refer page 736

2 骨CT观测应用。refer page 735

测试原图来自医学教材

3 皮肤病观测应用。refer page 下册156，下册157

测试原图来自医学教材

4 图片读脏 应用逻辑



**章节的著作权文件列表：**

第十一章\_DNA\_ETL与元基索引ETL中文脚本编译机.

**[养疗经 DNA 元基 版本tin shell 集成 测试 成功](https://www.zhihu.com/zvideo/1426319837153411072" \t "https://zhuanlan.zhihu.com/p/_blank)**

[](https://www.zhihu.com/zvideo/1426319837153411072" \t "https://zhuanlan.zhihu.com/p/_blank)

[Alkaid 罗瑶光的视频](https://www.zhihu.com/zvideo/1426319837153411072" \t "https://zhuanlan.zhihu.com/p/_blank)

[· 14 播放](https://www.zhihu.com/zvideo/1426319837153411072" \t "https://zhuanlan.zhihu.com/p/_blank)

**ETL元基编码方式,**



1 DNA\_ETL的编码继承了德塔数据库的语言编译机。refer page 413,788

2 DNA\_ETL的编码字符串可以自由设计，如中文描述。refer page 834,835

3 DNA\_ETL的编码行可以集成在节点中 etl单个 执行。refer page 782

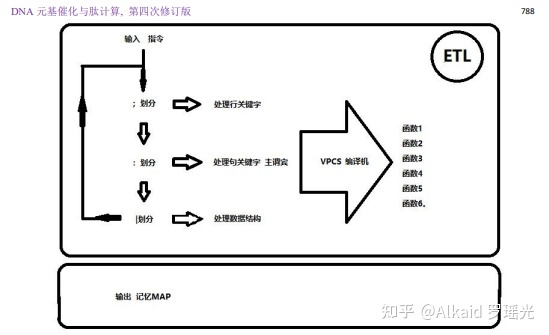
4 DNA\_ETL的编码可以拆卸成节点模式单行进行 etl流 执行。refer page 784

**PLETL语言，**

1 PLETL语言 继承了德塔数据库的语言编译机语言。refer page 377,786

2 PLETL语言 扩展了德塔数据库的语言编译机语言，如TCP, REGEX 应用等。refer page 784

3 PLETL语言 支持多语种 命令设计。refer page 789,790



4 PLETL语言 节点流编译机 可模拟神经网络语言 做计算需求。refer page 783

**Tinshell，**



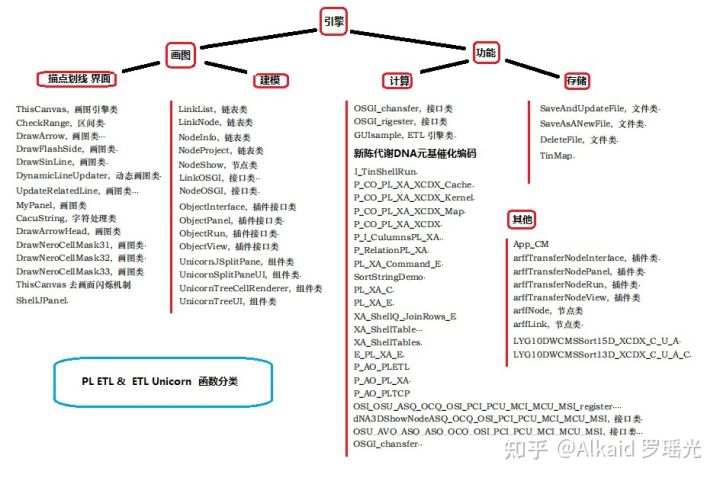
1 Tinshell是封装在 PLETL语言下的基础组件。refer page 860,877

2 基础组件体现在 脚本的编译和执行。refer page 786

3 Tinshell 采用 德塔数据库的语言编译机 进行改装。refer page 788~835

4 Tinshell 主要用于脚本语言的输入和 计算输出 的 IO计算。refer page 782

**编译机的进化，**



1 德塔编译机，最早取自 德塔socket流可编程数据库系统的 plsql编译机。refer page 377

2 德塔编译机在设计tinshell的时候从数据库中分出来做脚本编码编译机。refer page 786

3 德塔编译机在脚本编码中开始扩展，如和etl结合，和tcp结合等。refer page 783

4 德塔编译机在肽化索引后，将用于神经元 etl节点网络计算中枢模拟。refer page 783,784

**osgi插件的肽化方式，**



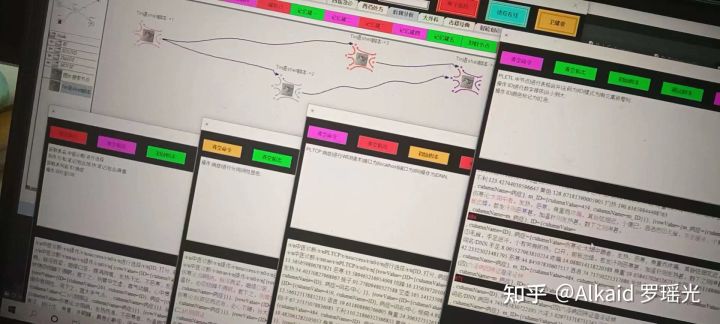
1 osgi插件最早作者设计为了 将节点进行像knime那样 导入。但一直没有实现，只是继承而已。refer page

2 作者2017年5月 在洛杉矶与 印度的 印佛西思 2个月的电话+citrix经理面试时候，被频繁询问classloader技术。于是笔记。refer page 291 Class<?> myclass = loader.loadClass, 作者的最早classloader思维不是来自cnblogs chinaxin。

3 作者2019年开始尝试并真正的 classloader jar实现 节点插件化。refer page 781

4 最近开始肽化索引，用于 classloader识别标记与节点文件分类。refer page 781

**神经元计算模拟 应用**



1 DNA\_ETL的 神经元计算 是一种有向 节点拓扑计算。 refer page 786

2 DNA\_ETL的神经元计算中节点是一个载体单位，不再是计算单位。 refer page 782

3 DNA\_ETL的计算单位是单一一句tinshell 命令。 refer page 783

4 DNA\_ETL的tinshell命令可一句 或者 多句 载入 一个 和 多个节点中。 refer page 784

**章节的著作权文件列表：**

第十二章\_DNA语料数据库加密技术.

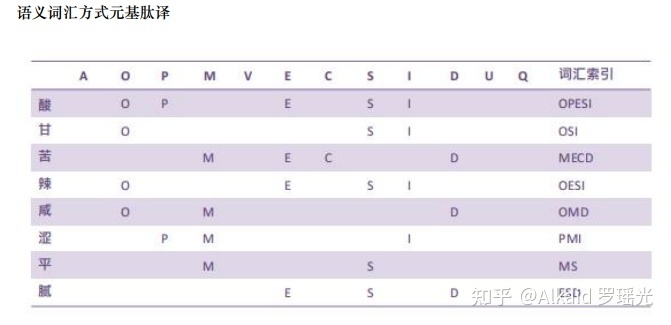
**DNA 元基加密，**

1 DNA 元基加密 包含物理元基加密和 非物理 的 语义元基加密.refer page 900,1015

2 物理元基加密，可理解为将元基编译成密码子，通过算法将密码子替换原文。refer page 900

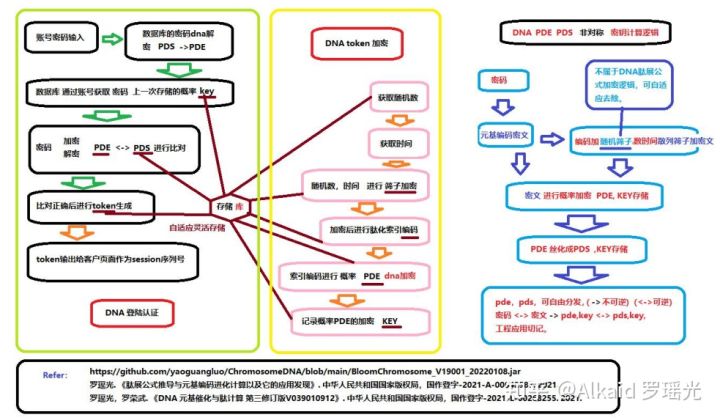


3 语义元基加密，可理解为将文字进行语义肽展公式变换，然后通过酸碱概率二次加密.refer page 907,910



4 酸碱概率的几率成为概率钥匙，用于密文的解密和对比检测。refer page 915~

**非对称概率钥匙加密，**



1 非对称概率钥匙加密 利用的是 肽展公式通过酸碱概率二次和三次加密，加密次序的拓扑过程产生的不稳定性编码如 1变2, 2变1，产生不可还原的因素。refer page 936 A=V+S 1变2,

2 于是这个加密的概率钥匙可以进行前序解密， 但不可后序解密，具备了非对称加密的条件。refer page 1016,1017

3 非对称概率钥匙加密适用于各种不可逆的加密场景中，如互联网登陆认证。refer page 1017

4 非对称概率钥匙加密可以物理与语义进行组合加密，增加安全级。refer page 981

**[DNA催化计算加密 REST 会话中 肽元基Token生成](https://www.zhihu.com/zvideo/1330485210245173248" \t "https://zhuanlan.zhihu.com/p/_blank)**

[](https://www.zhihu.com/zvideo/1330485210245173248" \t "https://zhuanlan.zhihu.com/p/_blank)

[Alkaid 罗瑶光的视频](https://www.zhihu.com/zvideo/1330485210245173248" \t "https://zhuanlan.zhihu.com/p/_blank)

[· 22 播放](https://www.zhihu.com/zvideo/1330485210245173248" \t "https://zhuanlan.zhihu.com/p/_blank)

**[pow（2的60次方，元基长度 ）概率随机DNA加密](https://www.zhihu.com/zvideo/1328926594035953664" \t "https://zhuanlan.zhihu.com/p/_blank)**

[](https://www.zhihu.com/zvideo/1328926594035953664" \t "https://zhuanlan.zhihu.com/p/_blank)

[Alkaid 罗瑶光的视频](https://www.zhihu.com/zvideo/1328926594035953664" \t "https://zhuanlan.zhihu.com/p/_blank)

[· 120 播放](https://www.zhihu.com/zvideo/1328926594035953664" \t "https://zhuanlan.zhihu.com/p/_blank)

**DNA元基隐写术，**

1 DNA元基隐写术的特点是数据隐藏。 作者给大家一个闪光发散点：行为隐写术。refer page 1015

2 设计情报学和安全学，本书不做描述。 refer page

**DNA元基特征，**

1 DNA元基腐蚀特征，可类比生化的DNA基元 如基因。refer page 第7， 8， 9章 元基编码

2 DNA元基物理特征，可类比电工的信号单元 如锁存器等。refer page 第13章 元基数字逻辑

3 DNA元基语义特征，可类比人类的词汇表达，如近义词扩展。refer page 901， 683，

4 DNA元基加密特征，可类比密码学的符号。refer page 901，

**Web登陆token，**

1 Web登陆token 包含物理加密，语义加密，肽展加密，丝化加密，和概率钥匙。refer page 981~结果

2 物理加密 通过物理算法将原文进行元基加密的过程。refer page 981~结果

3 语义加密 通过 语义PDC三元根字典编码 将原文进行元基加密的过程。refer page 981~结果

4 肽展加密 通过 肽展公式进行 元基肽展变换 将原文进行元基加密的过程。refer page 981~结果

5 丝化加密 通过 肽展公式进行 元基丝化变换 将原文进行元基加密的过程。refer page 981~结果

**Session会话加密，**



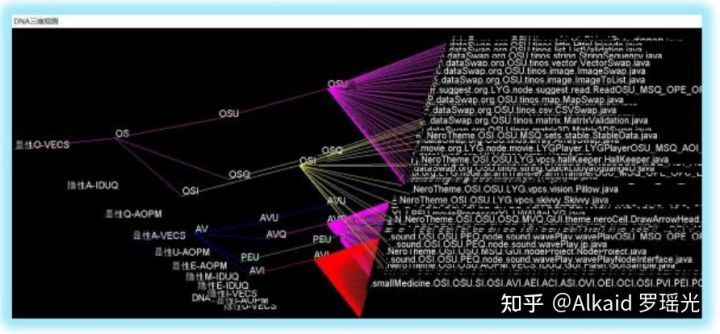
1 Web登陆token 封装的过程 为 Session会话加密.refer page 下册631

2 Session会话加密记录了token加密过程的 概率钥匙记录.refer page 下册631

3 Session会话加密 是一种非对称加密的应用。refer page 下册638

4 概率钥匙的分发方式可以进行 有效的实现 非对称组合加密 应用场景。refer page 下册638

**元基索引**



**[优酷视频](https://link.zhihu.com/?target=https://v.youku.com/v_show/id_XNDkzNTg4OTM4OA==.html" \t "https://zhuanlan.zhihu.com/p/_blank)**

**[养疗经 软件 函数 肽化 染色体索引观测](https://www.zhihu.com/zvideo/1314493149612593152" \t "https://zhuanlan.zhihu.com/p/_blank)**

[](https://www.zhihu.com/zvideo/1314493149612593152" \t "https://zhuanlan.zhihu.com/p/_blank)

[Alkaid 罗瑶光的视频](https://www.zhihu.com/zvideo/1314493149612593152" \t "https://zhuanlan.zhihu.com/p/_blank)

[· 27 播放](https://www.zhihu.com/zvideo/1314493149612593152" \t "https://zhuanlan.zhihu.com/p/_blank)

1 元基索引 最早作者只是 对工程的文件名索引，方便分类.refer page 1015

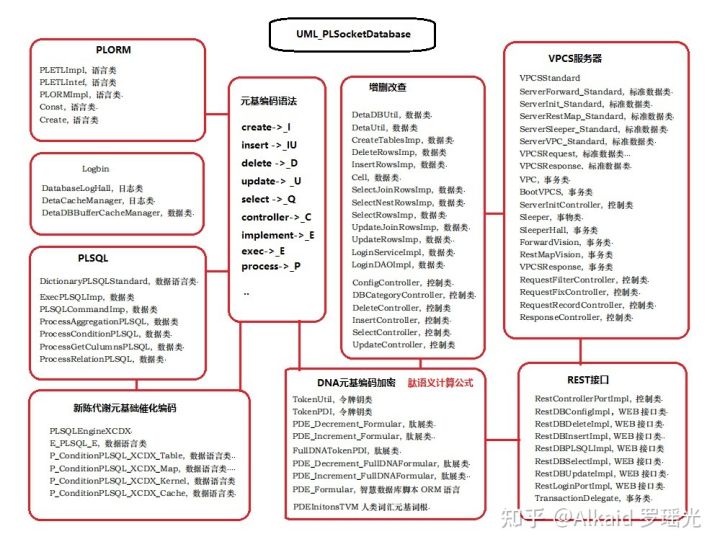
2 元基索引 逐渐进行染色体分类观测，确定函数文件的具体属性。refer page 1015

3 元基索引 开始对函数的具体作用进行特征表达，为了更好的序列化染色体。 refer page 见元基索引花

4 元基索引 用于元基花遗传编码。refer page 下册630

**应用**

数据库加密的文件名 元基索引 新陈代谢. refer page 1018~



**章节的著作权文件列表：**

**第十三章\_DNA\_数术推导与RNA\_X\_THF\_DD元基芯片与肽逻辑**

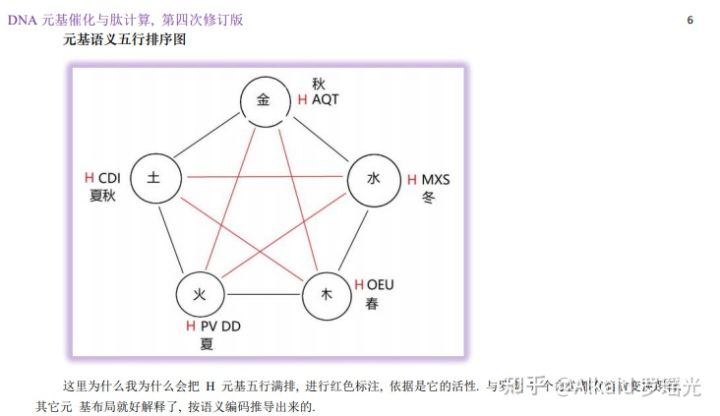
**理解这章节，首先要先进行元基的 O N的数量来进行统计 酸碱腐蚀度比值，进行罗盘归纳元基的活性，和元基的酸碱腐蚀性，主要体现在元基的生化和语义的两种方位排列方式，如下面的罗盘例子展示。作者罗瑶光**

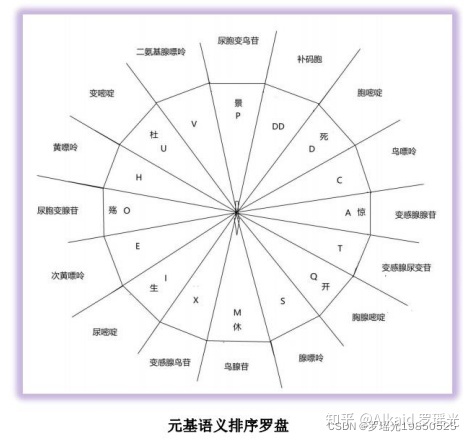


《DNA 元基催化与肽计算 第四修订版V00919》 下册第5页 作者罗瑶光

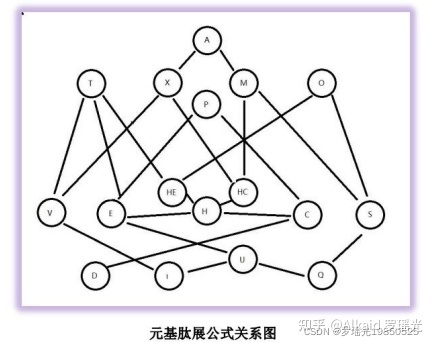
**元基罗盘分类，**

1 DNA元基语义罗盘，refer page 下册6

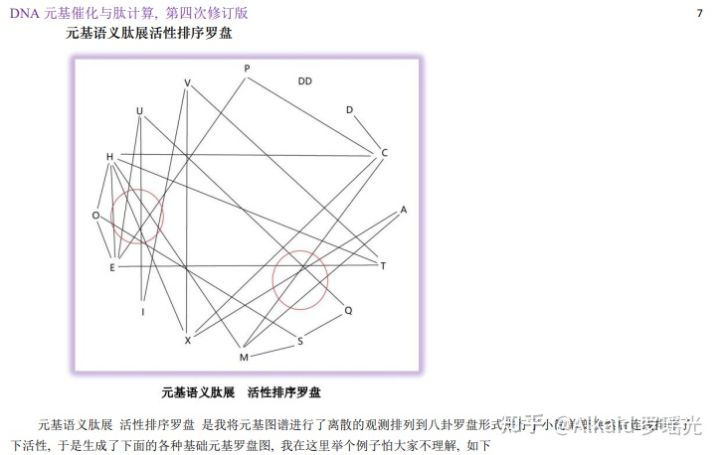




2 DNA元基活性罗盘，refer page 下册6



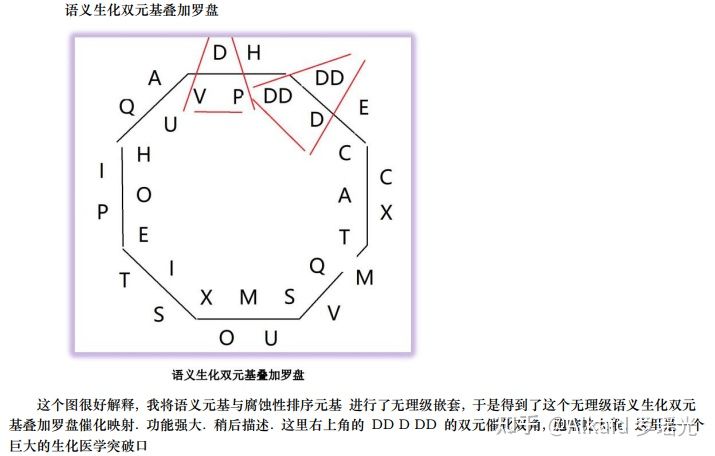
上图进行邻接变换观测如下图



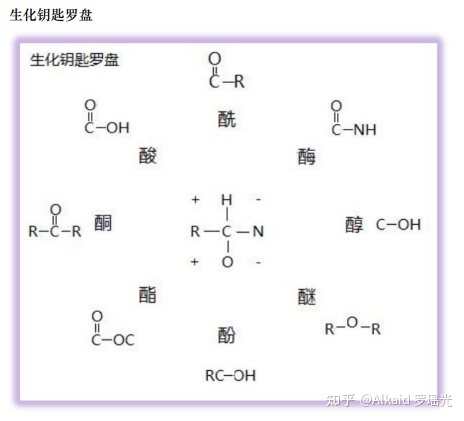
3 DNA元基腐蚀罗盘 refer page 下册7



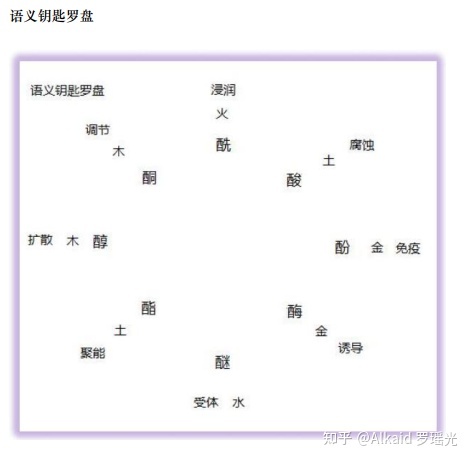
4 DNA 双元罗盘，refer page 下册8



5 DNA 生化钥匙罗盘，refer page 下册9



6 DNA 语义钥匙罗盘，refer page 下册10



/\* 并入进来。

**肽钥匙，**

1 DNA非卷积视觉的肽钥匙采用化学的 酸酚酮酯 醇酶酰醚 来做钥匙refer page 下册10

2 DNA非卷积视觉的肽钥匙按CNO比例和活性来罗盘归纳refer page 下册9，下册10

3 DNA非卷积视觉的肽钥匙通过罗盘的方位和活性确定其语义属性refer page 下册10

4 DNA非卷积视觉的肽钥匙具备双元 生化语义无理级价值。refer page 下册10



**肽活性表达，略**

1 DNA非卷积视觉的肽元基有化学活性归纳

2 DNA非卷积视觉的肽元基有方位语义归纳

3 DNA非卷积视觉的肽元基有活性归纳

\*/

**元基进制推导，**

1 欧拉计算refer page 下册56

2 商旅分析refer page 下册56

3 十七进制refer page 下册15

4 十六进制refer page 下册16

**十六进制变换方式，**

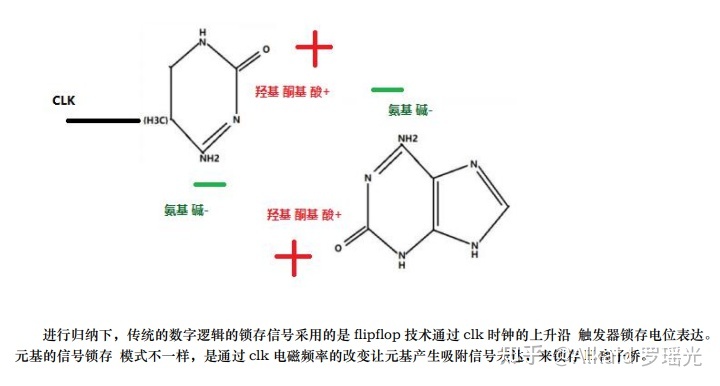
1 十六进制定义 refer page 下册48

**作者因为发现了全嘌呤F于是进行DD 和 HE HC 元基替换观测，发现了DCPE THOS MAXF VIUQ 十六元基欧拉排列，因为首尾是 DQ ，于是定义为人类史第一次定义元基十六进制。以后会不断优化**

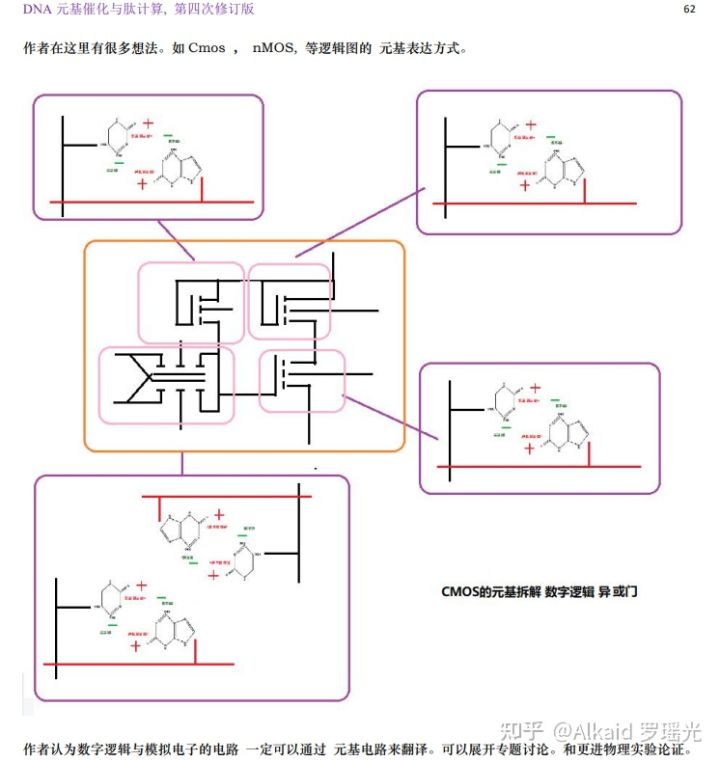
2 十进制互换 refer page 下册11

**元基数字逻辑，**

1 锁存器 refer page 下册60

元基思维发散： 锁存器

2 触发器 refer page 下册61,下册62

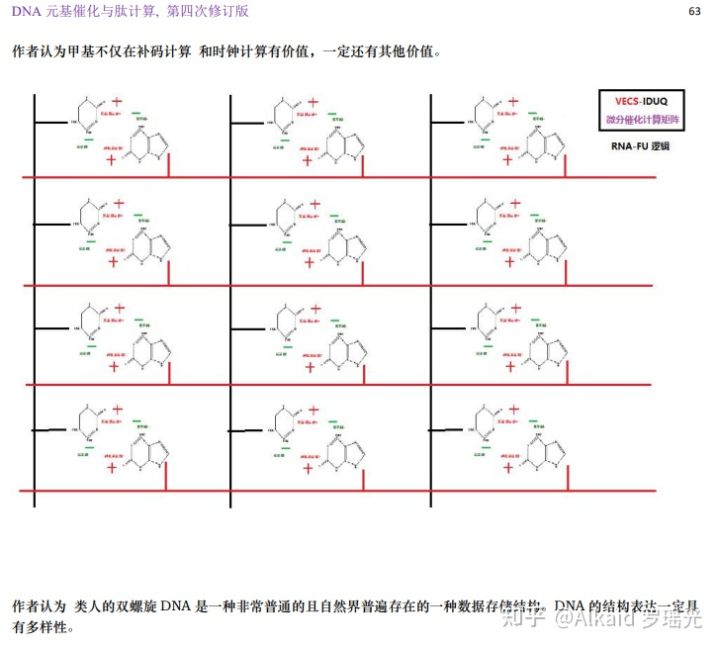
CMOS 元基数字逻辑思维发散

3 寄存器 refer page 下册63

4 锁相环 refer page 下册63

**锁存器与触发器的模拟猜想，**

1 锁相环存储 refer page 下册63

元基阵列存储设计 思维发散

2 锁相环计算 refer page 略

3 锁相环滤波 refer page 下册62

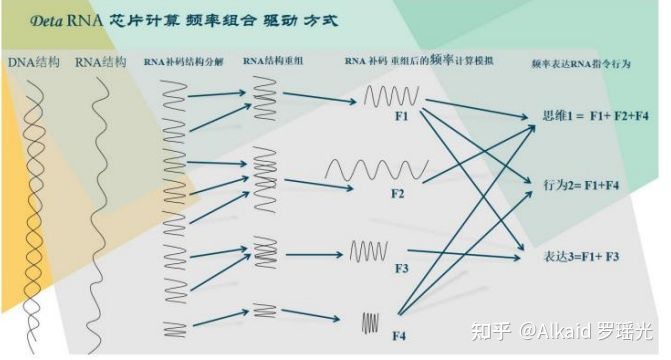
4 计时器 refer page 略

**周期频率语义肽减法公式。**

1 元基频率推导 refer page 下册55

2 元基频率PN极性推导 refer page 下册61

3 元基频率补码减法推导 refer page 下册54



推导过程见书籍《DNA元基催化与肽计算 第四修订版 V00919》

**章节的著作权文件列表：**

**第十四章\_DNA搜索与筛选应用**

这一章节主要涉及 全书的函数进行应用的实践。

**元基的细化模式，**

1 将人类词汇进行语义元基编码。refer page 下册64

2 编码中的元基含有量和元基的搭配位置用于特征标识。refer page 表格中数据元基 在笛卡尔循环搜索时候会自动叠加 略。

3 特征标识主要包含 生化标识和 语义标识。refer page 表格中数据元基 在笛卡尔循环搜索时候会自动叠加 略。

3.1 语义元基定义方式。refer page 下册77

3.2 生化元基定义方式。refer page 下册78

4 特征标识用于搜索和筛选应用。refer page 下册79

**语义的元基表达，**

1 语义的元基表达主要体现在 特征标识的方式。refer page

2 固定的特征标识可以生成元基词汇。refer page 683

3 单个的特征标识可以用于索引分类。refer page 下册79 如筛选分类应用

元基筛选应用实例

4 单位长度的特征标识可以用于索引加密。refer page 下册77 如表格中数据元基描述。

**特征的PCA打分模式，**

1 特征的PCA打分 体现在某元基的 占有概率比重。refer page 下册76

2 特征的PCA打分 体现在某元基团的 占有概率比重。refer page 下册79

3 特征的PCA打分 体现在搜索中权重叠加打分 。举例 张三AOP， 李四POM ,那么 PO 就叠加了，搜索分值权重自动增加。refer page 下册79

**搜索对象的元基索引方式，**

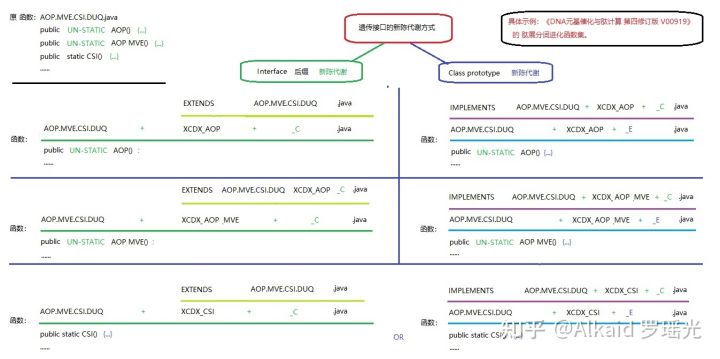
1 搜索对象的元基索引可以通过单个元基染色体分类索引。refer page 下册77

2 搜索对象的元基索引可以通过单个元基词汇 索引。refer page 683

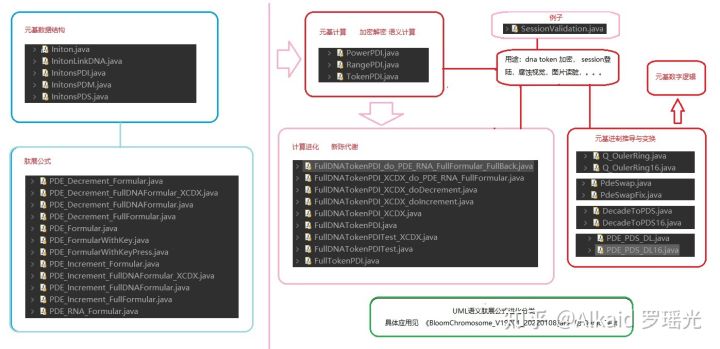
**元基索引染色体分类观测，**

1 元基索引染色体分类观测 体现在函数的功能进行分类。refer page 682，692

2 函数的功能进行元基编码，体现在文件名编码和文件函数名编码。refer page 671，下册147



3 文件名和函数名 元基编码，主要用进行新陈代谢，方便之后的进化计算。refer page 下册149



**应用**

DNN分词词汇花。refer page 下册80~



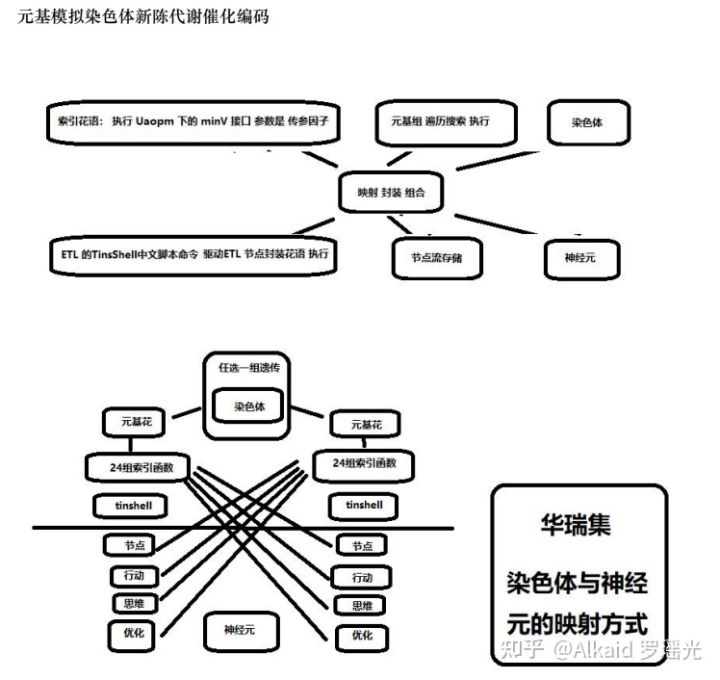
**章节的著作权文件列表：**

第十五章\_元基模拟染色体新陈代谢催化编码

**定义：元基的 新陈代谢一般指 源码工程的函数文件 进行元基编码后的 内容逻辑 接口和类 索引优化方式，主要体现在 文件 的分类，剔除，继承，分配。**

**定义：元基的 二次新陈代谢一般指 源码工程的函数文件名 进行元基编码后的 文件名称 索引优化方式，主要体现在 元基 的分类，剔除，缩进，分配。**

定义者 罗瑶光



**元基造字，**

1 元基造字的编码方式。refer page 672~ ，901~

2 元基造字的编码字典。refer page 901，913

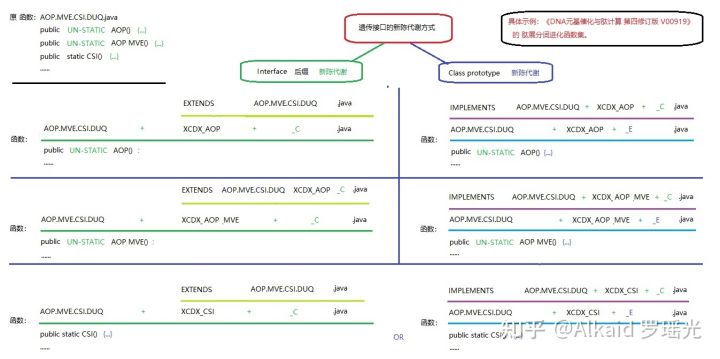
3 元基造字的编码词汇。refer page 语料库方式 914

4 元基造字的字词定义。refer page 下册119（作者的意识而已，没有全民代表性）

**元基进化方式，**

1 元基进化方式 肽展公式新陈代谢。refer page 下册~144~

2 元基进化方式 函数索引二次新陈代谢。refer page 下册149，遗传代谢模式见uml



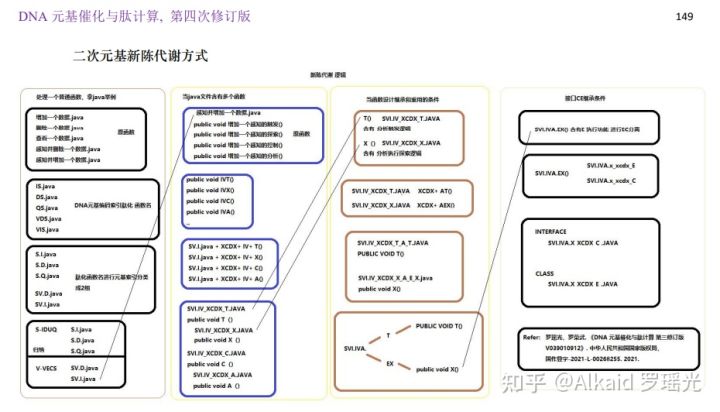
**元基新陈代谢，**

1 文件名新陈。refer page 下册149

2 文件名代谢。refer page 下册149

3 函数名新陈。refer page 略

4 函数名代谢。refer page 略



**元基二次新陈代谢，**

1 文件与函数名的新陈代谢。refer page 下册176~192, 下册214~232, 下册242~274

2 文件内容与函数内容的新陈代谢。refer page 下册172~

3 文件与函数的继承函数新陈代谢。refer page 下册214~274

4 文件与函数的接口函数新陈代谢见CE分离。refer page 下册242,下册248,下册253,下册271

/\*移出本章

refer page 下册172~

**元基花，**

1 元基花染色体模拟。

refer page

2 元基花瓣 映射接口 模拟。refer page

3 元基花萼 接口调用 模拟。refer page

4 元基花蕊 遗传序列 模拟。refer page

**元基枝，**

1 元基枝叶模拟 花蕊集工程文件。refer page

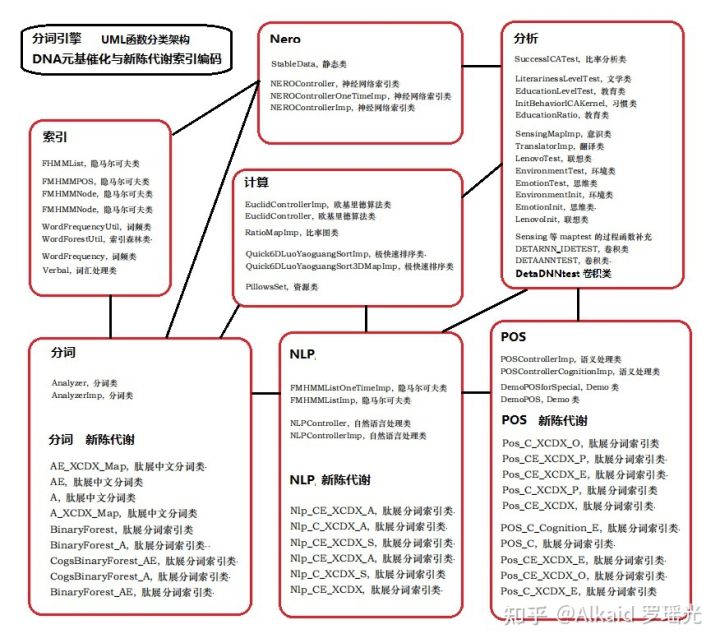
2 元基枝干模拟 养疗经启动文件。refer page

\*/

**元基催化在分词， 排序，图片读脏识别上的应用。**



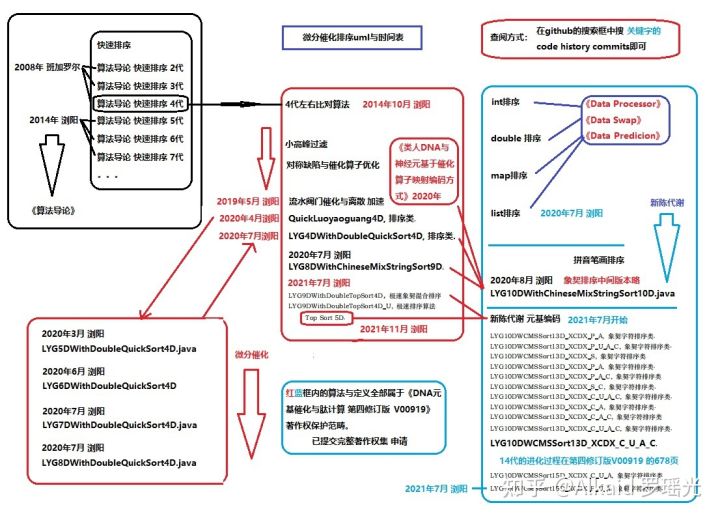
1 肽展催化分词新陈代谢。refer page 下册193~



2 肽展图片读脏新陈代谢。refer page 下册156~

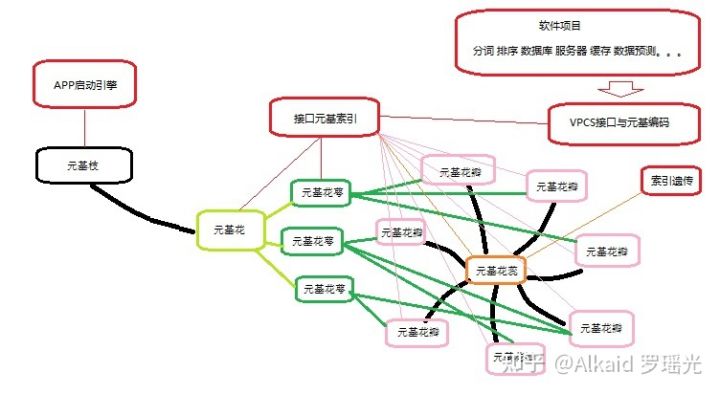


3 肽展象契形排序新陈代谢。refer page 下册172~



**章节的著作权文件列表：**

第十六章\_TinShell插件\_元基花模拟染色体组计算索引系统

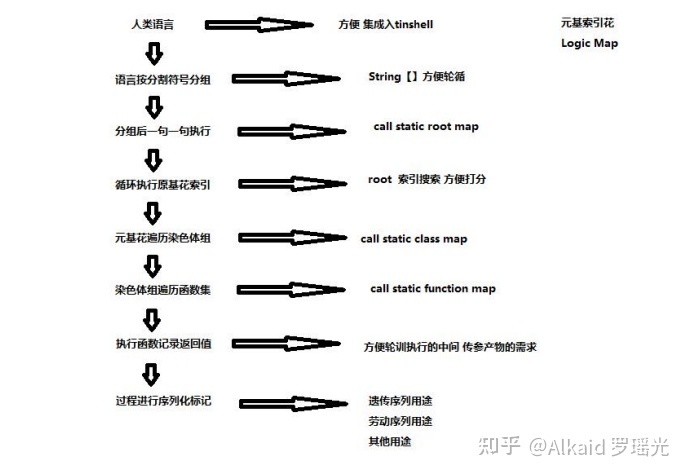


**元基花定义 一般指 软件工程源码 在 进化计算 表达中， 能够进行 将执行函数 序列化的 索引组件。**

**元基枝定义 一般指 软件工程源码 在 进化计算 表达中， 能够识别 元基花 索引组件 的引擎和终端。**

**定义者 罗瑶光**

**元基索引花，**



1 元基索引花映射计算。refer page 下册278,下册292,下册296

2 元基索引花调度模式。refer page 下册299~

3 元基索引花语言模式。refer page 下册630

/\*

**元基花，**

1 元基花染色体模拟。refer page 下册278~

2 元基花瓣 映射接口 模拟。refer page 下册296~630

3 元基花萼 接口调用 模拟。refer page 下册292

4 元基花蕊 遗传序列 模拟。refer page 本章

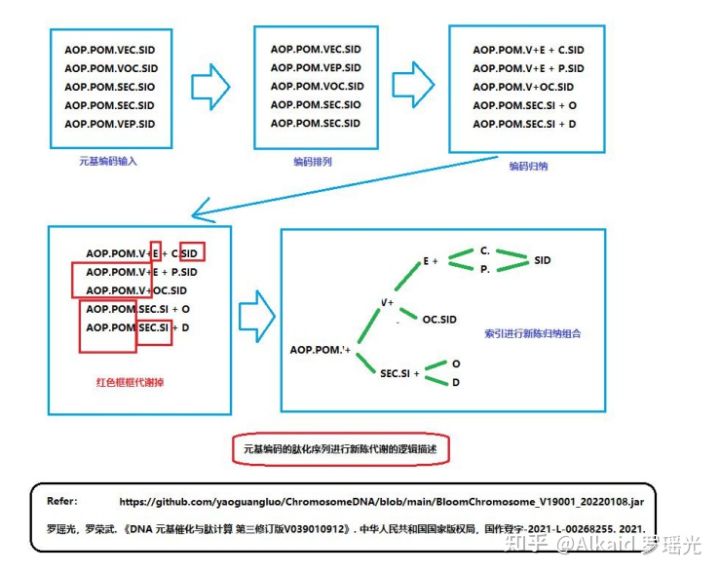
**元基枝，**

1 元基枝叶模拟 华瑞集工程文件。refer page 前六章的实体工程架构 12,186,267,368,492,560

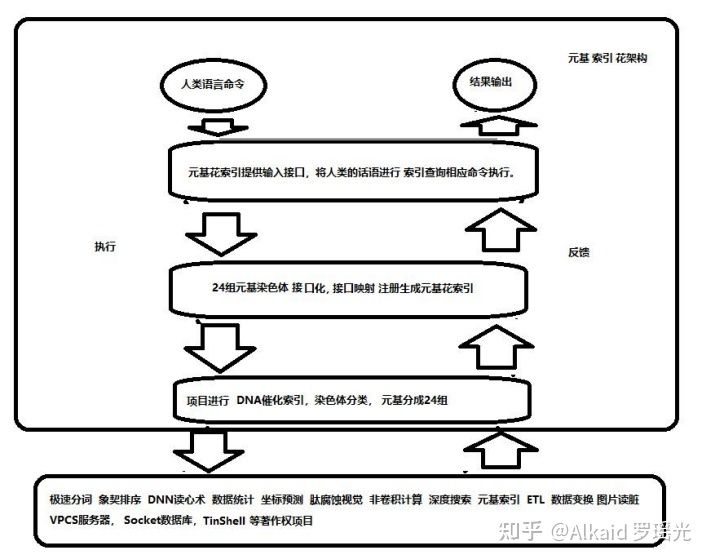
2 元基枝干模拟 养疗经启动文件。refer page 养疗经的boot app启动主引擎用于连接元基花计算。

\*/

**元基花的优化方式，**

第十五章的新陈代谢铺垫

1 元基花的索引优化。refer page 下册299~ (传参因子[因子++])设计模式



2 元基花的映射优化。refer page reflection 优化 见UML，不断裁剪分出去即可，

[https://github.com/yaoguangluo/ChromosomeDNA/tree/main/2022/02/02](https://link.zhihu.com/?target=https://github.com/yaoguangluo/ChromosomeDNA/tree/main/2022/02/02" \t "https://zhuanlan.zhihu.com/p/_blank)

3 元基花的文件细化。refer page 与sonar的规范一致，国际统一，文件大化小，循环多化少，内容重化简，不多介绍了，

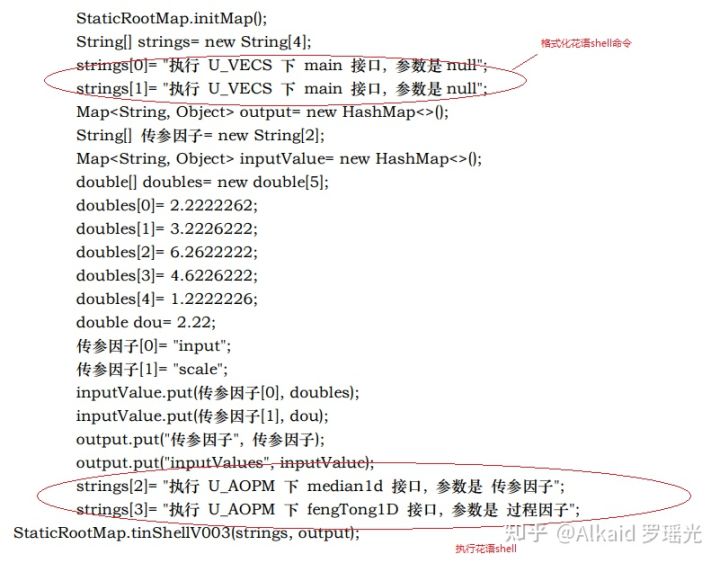
4 元基花的新陈代谢。refer page 下册149， 更多见uml归纳。

**元基花的绽放方式，**

1 元基花的展示。refer page 下册278~

2 元基花接口调用方式 。refer page 下册631

3 元基花接口调用的格式化序列记录。refer page 下册631



**元基花的遗传方式，**

1 元基花的遗传属性。refer page 下册663

2 元基花的遗传序列函数统计方式。refer page 下册696

3 元基花的遗传序列。refer page 下册631,下册696

**元基花的配对方式，**

1 元基花的序列实现。refer page 下册278,下册292,下册296

2 元基花的序列编码。refer page 下册630

3 元基花的配对的成分。refer page 元基索引花，元基索引花对应的工程函数映射，下册480 StaticFunctionMap的 annotationMap 注册函数。



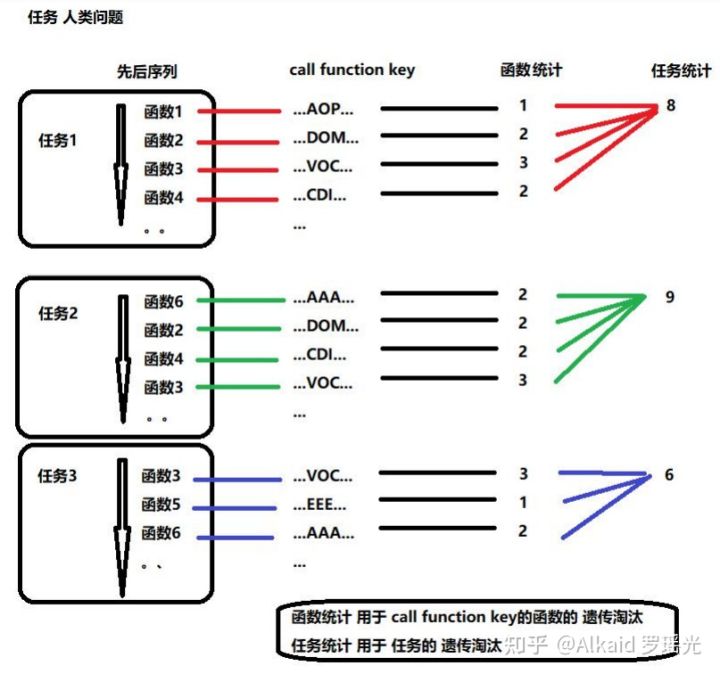
**元基花的进化方式。**

1 元基花的新陈代谢。refer page 见下册149 与 uml归纳

[https://github.com/yaoguangluo/ChromosomeDNA/tree/main/UML](https://link.zhihu.com/?target=https://github.com/yaoguangluo/ChromosomeDNA/tree/main/UML" \t "https://zhuanlan.zhihu.com/p/_blank)

2 元基花的自主添加接口方式。refer page 未涉及。常见如OSGI扩展, 继承，classloader扫描 三种写法。

3 元基花的任务统计方式。refer page 下册696



**应用**

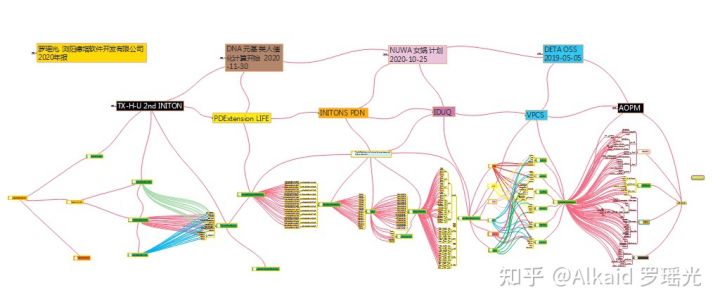
1 元基花调用实例。refer page 下册630

# **在进化计算中，软件进行元基编码的新陈代谢方式 V0. 0. 9**

关键词:**进化计算, 数据软件, 元基索引, 新陈代谢**

2018年10月, 设计养疗经软件, 我花了一个月就把中药搜索的功能实现了. 当时心里只是有点不服,因为我应该多花点精力做些什么, 于是开始包装和优化. 第一个值得优化的问题就是药材搜索的搜索速度. 我采用的是开源插件进行文本分词搜索, 当我不断的加医学教材书进行搜索内容扩充, 于是搜索开始了卡顿. 需求迫使我必须自己写一个新的分词算法, 解决卡顿问题. 软件的元基编码的新陈代谢优化系统拉开了帷幕.

分词算法开始自己写, 一开始, 我要面对如何设计算法的困难. 当算法设计好了, 我的新问题是如何搭配这些算法来设计处理模块. 最后我还要思考怎么优化这些功能模块. 我的思维很简单, 就是先将函数进行简单的分类吧,按软件工程瀑布模型分类, 如分析类, 操作类, 处理类, 运维类, 管理类, 执行类, 控制类. 等等.于是我开始将软件项目进行基础功能的应用分类归纳, 产生了很多基础软件作品. 我发现这些作品不同的组合不但能解决我的问题, 还能解决许多工业, 农业, 服务业的需求问题.



这个过程中, 我得到了很多有意思的价值发现, 如分词作品, 排序作品, 服务器作品, ETL作品, 数据计算作品, 数据库作品, 数据变换作品, 数据预测作品等.

2019年04月03日 1.罗瑶光. 《德塔自然语言图灵系统 V10.6.1》. 中华人民共和国国家版权局，软著登字第3951366号. 2019.

2014年10月19日 2.罗瑶光. 《Java数据分析算法引擎系统 V1.0.0》. 中华人民共和国国家版权局，软著登字第4584594号. 2014.

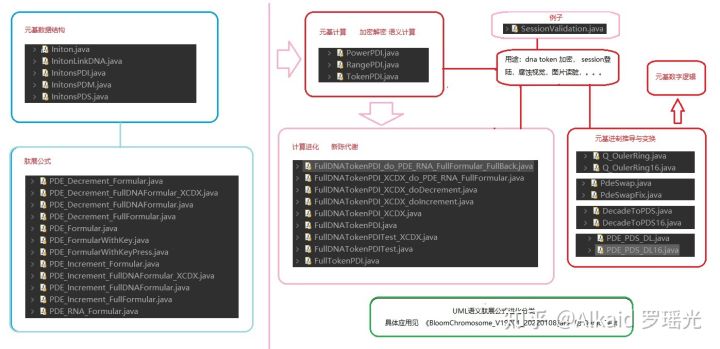
2019年06月10日 3.罗瑶光. 《德塔ETL人工智能可视化数据流分析引擎系统 V1.0.2》. 中华人民共和国国家版权局， 软著登字第4240558号. 2019.

2019年06月24日 4.罗瑶光. 《德塔 Socket流可编程数据库语言引擎系统 V1.0.0》. 中华人民共和国国家版权局，软著登字第4317518号. 2019.

2019年09月16日 5.罗瑶光. 《德塔数据结构变量快速转换 V1.0》. 中华人民共和国国家版权局，软著登字第4607950号. 2019.

2020年03月03日 6.罗瑶光. 《数据预测引擎系统 V1.0.0》. 中华人民共和国国家版权局，软著登字第5447819号. 2020.

有了这些基础算法包和医药数据搜索软件项目, 于是我开始优化和扩展软件的应用价值. 将这些价值发现变成价值体现. 我的研发思维还是很简单, 思考, 如果我罗瑶光, 此时此刻就是这个软件, 我会在怎么做？我会怎么分析问题？怎么解决问题？怎么计算结果？怎么整理结果？这个思维看起来很简单, 实现起来各种阻力.还能怎么办？硬着头皮, 将困难不断的细化, 一点一点的解决累积. 将成果分类归纳. 随着函数的分类细化, 我的数据计算软件作品越来越多. 我在思考怎么进行将函数的有效的归纳和分类, 如设计一个项目目录索引方式？于是AOPM-VPCS的语义元基编码帮我解决了很多问题. 最后这个DNA语义元基编码体系为我解决了大量函数分类的问题. 目前DNA元基编码理论一直在优化中, 目前包含了AOPM-VECS-IDUQ-TXHF16个生化语义元基算子.



这个过程中, 我得到了很多有意思的价值发现, 如DNA元基催化算子的发现, 语义肽展公式的推导, 催化算子的生化解码. 非卷积视觉肽计算， 肽元基加密。具体体现在类人仿生的认知思维表达模式, 类人仿生的神经元计算思维模式, 类人仿生的任务处理思维模式。

2020年10月09日 7.罗瑶光, 罗荣武. 《类人DNA与 神经元基于催化算子映射编码方式 V\_1.2.2》. 中华人民共和国国家版权局，国作登字-2021-A-00097017. 2021.

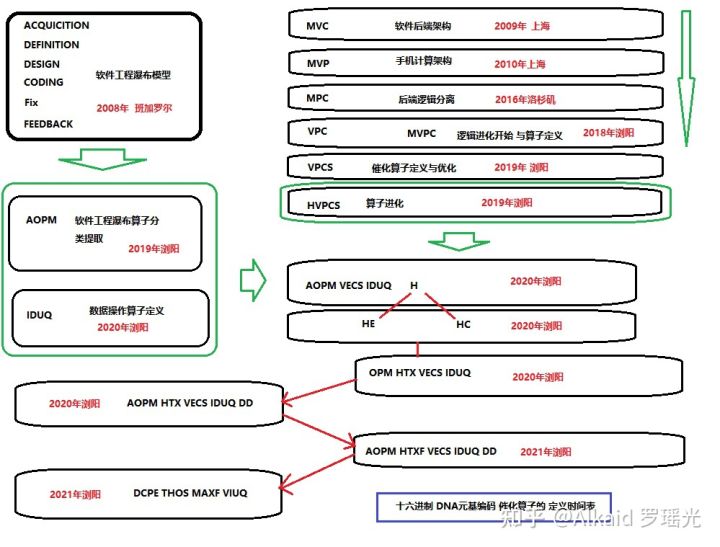
2020年10月31日 8.罗瑶光. 《肽展公式推导与元基编码进化计算以及它的应用发现》. 中华人民共和国国家版权局，国作登字-2021-A-00042587. 2021.

2020年11月29日 9.罗瑶光. 《DNA催化与肽展计算和AOPM-TXH-VECS-IDUQ元基解码013026中文版本》. 中华人民共和国国家版权局，国作登字-2021-A-00042586. 2021.

2021年03月05日 10.罗瑶光, 罗荣武. 《DNA元基催化与肽计算第二卷养疗经应用研究20210305》. 中华人民共和国国家版权局，国作登字-2021-L-00103660. 2021.

2021年09月13日 11.罗瑶光, 罗荣武. 《DNA 元基催化与肽计算 第三修订版V039010912》. 中华人民共和国国家版权局，国作登字-2021-L-00268255. 2021.

2021年10月16日 12.罗瑶光. 《DNA元基索引ETL中文脚本编译机V0.0.2》. 中华人民共和国国家版权局，SD-2021R11L2844054. 2021. (登记号:2022SR0011067) 软著登字第8965266号.



当我的软件开始了DNA元基编码优化方式, 我一直在思考怎么让我的软件自主进行进化计算分析. 我的思维还是很简单设计这个编码的新陈代谢方式, 为软件赋予原始的生命特征活性. 于是我开始研究, 发现元基编码在函数分类索引中有巨大价值. 索引能进行分类, 聚类, 记录, 裁剪, 表达, 等实际功能. 如果索引一旦具备了新陈代谢的活性,那软件的进化方式便具备了生命进化特征. 于是我开始进行系统性的软件遗传特征编码, 将软件任务进行格式化的函数序列来描述. 这个函数序列中的函数进行编码, 于是产生3个编码,

1 具体的某一函数在函数集染色体索引分类中的序列编码位.

2 具体任务包含的函数序列的序列位组合标记编码.

3 多个任务组成的神经元节点处理的流etl档案中的任务集编码.

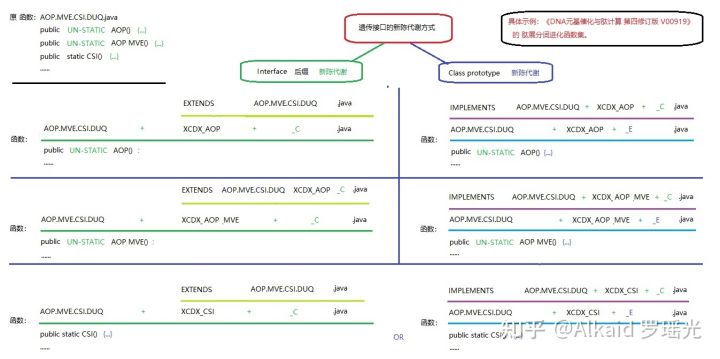
于是DNA元基催化与肽计算的遗传编码的软件生命诞生了.

这个过程中, 我得到了很多有意思的价值发现, 如DNA元基索引的染色体分类方式, DNA元基索引的新陈代谢方式, DNA元基索引的函数序列遗传方式.

2021年12月26日 13.罗瑶光. 《TinShell插件\_元基花模拟染色体组计算索引系统 V20211227》. 中华人民共和国国家版权局，SD-2021R11L3629232. 2022. (受理号:2022R11S0138561).

2022年01月27日 14.罗瑶光, 罗荣武. 《DNA元基催化与肽计算 第四修订版 V00919》. 中华人民共和国国家版权局，SD-2022Z11L0025809. 2022. (受理号:2022Z11S1032939).

有了这个方向，下一步我的 BloomChromosome\_V19001\_20220108.jar 准备进行全面的新陈代谢优化。



文件资源

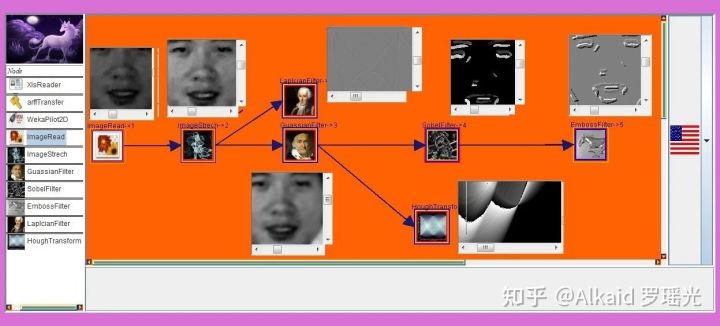
**计算机视觉关于进化计算表达方式0.0.6**

**定义：非卷积腐蚀视觉，一般指在不采用卷积内核进行图片低速计算的方式下，图片每个像素仅仅做一次遍历来模拟酸碱腐蚀肽展公式计算，目的是 达到人肉眼观测不到而又难以辨认的色阶群进行观察拉伸。**

**定义人 罗瑶光**

**1 思想**

天赋是需要后天培养的，正如我自己，年轻的时候能接触许多计算机的课程和知识点。如加州路德大学 Renhart 教授的计算视觉课程。，我比较系统的学习了计算机对图片像素的卷积视觉矩阵CNN计算方法和具体应用实践，如索贝尔，EMBOSS，哈尔变换和高斯模糊等。我很清楚的还原一句话：Renhart曾经说过：他年轻的时候在南加州大学做了一个关于视觉的立体观测的论文设计，主要是用索贝尔梯度向量来处理立体视觉的维度观测。我很庆幸，我能通过一门课和学分就掌握了他的研究理论。

ETL 节点流 可视化界面来 处理图片的 流操作 实例，读取一张照片，然后进行颜色拉伸，然后更进 边缘与模糊过滤，模糊后进行索贝尔计算眼睛眉毛的索贝尔梯度，最后更进做浮雕计算，可用于工业立体墙壁雕刻。

参考

2014年10月19日 2.罗瑶光. 《Java数据分析算法引擎系统 V1.0.0》. 中华人民共和国国家版权局，软著登字第4584594号. 2014.

**2 见解**

当然也是Renhart曾经说过：如果能将卷积计算变成序列化流计算就好了，这样中间过程就可以修改观测，应用会更广泛。我带着这个疑问，实现了UNICORN APPLET ETL的像素节点流数据分析设计工具。新的问题也伴随而来。第一个问题便是，卷积的计算优化加速。很多人的曾经说我的思考方式和常人不一样。正如2020年设计计算机腐蚀视觉算法例子，我的思维总是很简单，就是觉得卷积计算太耗费时间，如果能设计一种非卷积计算算法来处理计算机的视觉应用问题，那么将是质的飞跃。



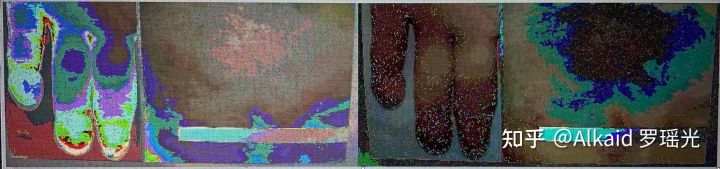
ETL 节点流 可视化界面 进行神经节点皮肤界面设计

参考

2019年06月10日 3.罗瑶光. 《德塔ETL人工智能可视化数据流分析引擎系统 V1.0.2》. 中华人民共和国国家版权局， 软著登字第4240558号. 2019.

**3 论证**

带着这个问题，我沿着非卷积计算视觉的研发路线开始跌跌撞撞的探索着。我在思考我有什么，勇气是不够的，需要有严谨的理论和基础知识作为后盾。此时我已经研发出来了肽展公式， 元基进制编码，最早从4bit的元基色进行图片快速腐蚀，然后8bit元基腐蚀，当我的十七进制研发得到了十六进制的成果，于是迅速将十六进制的肽展腐蚀应用在图片观测上，我发现，元基腐蚀后的图片的色阶与256bit的十进制色阶完全不对称，利用这个不对称，能得到更多的颜色隐藏特征。

16元基进制的非卷积算法处理手指疾病图片的真实观测实例

参考

2020年10月09日 7.罗瑶光, 罗荣武. 《类人DNA与 神经元基于催化算子映射编码方式 V\_1.2.2》. 中华人民共和国国家版权局，国作登字-2021-A-00097017. 2021.

2020年10月31日 8.罗瑶光. 《肽展公式推导与元基编码进化计算以及它的应用发现》. 中华人民共和国国家版权局，国作登字-2021-A-00042587. 2021.

2020年11月29日 9.罗瑶光. 《DNA催化与肽展计算和AOPM-TXH-VECS-IDUQ元基解码013026中文版本》. 中华人民共和国国家版权局，国作登字-2021-A-00042586. 2021.

**4 结果**

有了这个发现，按耐不住自己的情绪，我迅速的进行生产实践，快速的应用在非卷积图片腐蚀观测，和非卷积图片读脏识别真实应用上。我看到了巨大的成果收获。父亲问我能不能做个图片搜索程序，只要输入皮肤病名，就能搜索对应的图片，我说何止是这种应用，直接拍个照，就能识别比对图片是什么。一开始父亲说我吹牛，当我拿出真实的结果关于骨科，皮肤病，的非卷积元基腐蚀测试图片结果，和图片识别结果数据给父亲看后，父亲竟然一时也惊讶的哑口无言。

图片来自医学教材。

1 非卷积图片腐蚀观测，

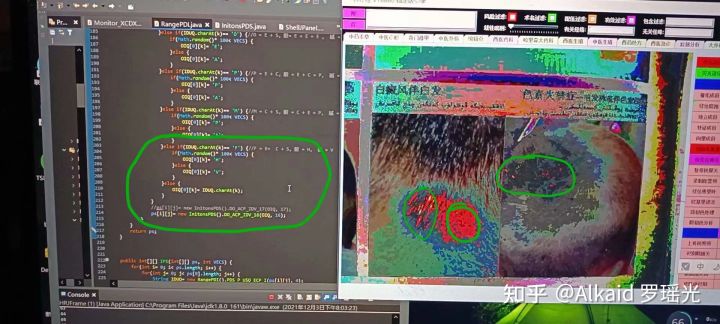
骨头CT非卷积腐蚀观测，常规CT只有黑与白， 黑白之间的灰色区间不能明显观测细微。肽腐蚀

卷积弥补了这个问题如下图。



2 和非卷积图片采样读脏识别

疾病图片比对，肉眼识别皮肤病，在暗色区域和碘酒侵染的同色区域辨别困难，肽卷积腐蚀弥补了这个问题如下图。



3 非卷积图片训练采样读脏识别打分

十六进制八元基 肽腐蚀计算识别 确认皮肤病名，3000组疾病图片训练，读脏识别率第一位。如下图

把疾病图片进行乱改如图，然后再识别，匹配3000张训练原图片，准确识别第一位如右上蓝色标记行

参考

2021年03月05日 10.罗瑶光, 罗荣武. 《DNA元基催化与肽计算第二卷养疗经应用研究20210305》. 中华人民共和国国家版权局，国作登字-2021-L-00103660. 2021.

**5 新陈代谢优化**

得到了这些结果，我不断的进行函数新陈代谢优化，目的的一开始是思考计算算法的可持续优化方案。我一直在思考，元基不仅在生命索引的新陈代谢中有巨大价值。在很多领域有着更真实的更广泛应用价值。正如DNA元基催化与肽计算在非卷积计算视觉的应用发现。



参考

2021年09月13日 11.罗瑶光, 罗荣武. 《DNA 元基催化与肽计算 第三修订版V039010912》. 中华人民共和国国家版权局，国作登字-2021-L-00268255. 2021.

2022年01月27日 14.罗瑶光, 罗荣武. 《DNA元基催化与肽计算 第四修订版 V00919》. 中华人民共和国国家版权局，SD-2022Z11L0025809. 2022. (受理号:2022Z11S1032939).

**我得到一个价值论点：计算机视觉关于进化计算表达方式，体现在仿生视觉的极快速应激表达。**

**著作权人信息列表：**

[https://github.com/yaoguangluo/YangLiaoJing\_HuaRuiJi/tree/18701/%E8%AF%81%E4%B9%A6](https://link.zhihu.com/?target=https://github.com/yaoguangluo/YangLiaoJing_HuaRuiJi/tree/18701/%E8%AF%81%E4%B9%A6" \t "https://www.zhihu.com/question/308886134/answer/_blank)   
证件标注完成日期: 另外 德塔ETL 前身 Unicorn的 qq微博 发布日期 2013年10月20日， Google微博 后更名SW-AI 发布日期 2013年10月23日，后更名为LYG-AI 发布日期 2013年11月20日。

2019年04月03日 1.罗瑶光. 《**[德塔自然语言图灵系统 V10.6.1](https://www.zhihu.com/search?q=%E5%BE%B7%E5%A1%94%E8%87%AA%E7%84%B6%E8%AF%AD%E8%A8%80%E5%9B%BE%E7%81%B5%E7%B3%BB%E7%BB%9F+V10.6.1&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"answer","sourceId":1587866158}" \t "https://www.zhihu.com/question/308886134/answer/_blank)**》. 中华人民共和国国家版权局，软著登字第3951366号. 2019.

2014年10月19日 2.罗瑶光. 《**Java数据分析算法引擎系统 V1.0.0**》. 中华人民共和国国家版权局，软著登字第4584594号. 2014.

2019年06月10日 3.罗瑶光. 《**德塔ETL人工智能可视化数据流分析引擎系统 V1.0.2**》. 中华人民共和国国家版权局， 软著登字第4240558号. 2019.

2019年06月24日 4.罗瑶光. 《**德塔 Socket流可编程数据库语言引擎系统 V1.0.0**》. 中华人民共和国国家版权局，软著登字第4317518号. 2019.

2019年09月16日 5.罗瑶光. 《**[德塔数据结构变量快速转换 V1.0](https://www.zhihu.com/search?q=%E5%BE%B7%E5%A1%94%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84%E5%8F%98%E9%87%8F%E5%BF%AB%E9%80%9F%E8%BD%AC%E6%8D%A2+V1.0&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"answer","sourceId":1587866158}" \t "https://www.zhihu.com/question/308886134/answer/_blank)**》. 中华人民共和国国家版权局，软著登字第4607950号. 2019.

2020年03月03日 6.罗瑶光. 《**[数据预测引擎系统 V1.0.0](https://www.zhihu.com/search?q=%E6%95%B0%E6%8D%AE%E9%A2%84%E6%B5%8B%E5%BC%95%E6%93%8E%E7%B3%BB%E7%BB%9F+V1.0.0&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"answer","sourceId":1587866158}" \t "https://www.zhihu.com/question/308886134/answer/_blank)**》. 中华人民共和国国家版权局，软著登字第5447819号. 2020.

2020年10月09日 7.罗瑶光, 罗荣武. 《**类人DNA与 神经元基于催化算子映射编码方式 V\_1.2.2**》. 中华人民共和国国家版权局，国作登字-2021-A-00097017. 2021.

2020年10月31日 8.罗瑶光. 《**肽展公式推导与元基编码进化计算以及它的应用发现**》. 中华人民共和国国家版权局，国作登字-2021-A-00042587. 2021.

2020年11月29日 9.罗瑶光. 《**DNA催化与肽展计算和AOPM-TXH-VECS-IDUQ元基解码013026中文版本**》. 中华人民共和国国家版权局，国作登字-2021-A-00042586. 2021.

2021年03月05日 10.罗瑶光, 罗荣武. 《**DNA元基催化与肽计算第二卷养疗经应用研究20210305**》. 中华人民共和国国家版权局，国作登字-2021-L-00103660. 2021.

2021年09月13日 11.罗瑶光, 罗荣武. 《**DNA 元基催化与肽计算 第三修订版V039010912**》. 中华人民共和国国家版权局，国作登字-2021-L-00268255. 2021.

2021年10月16日 12.罗瑶光. 《**[DNA元基索引ETL中文脚本编译机V0.0.2](https://www.zhihu.com/search?q=DNA%E5%85%83%E5%9F%BA%E7%B4%A2%E5%BC%95ETL%E4%B8%AD%E6%96%87%E8%84%9A%E6%9C%AC%E7%BC%96%E8%AF%91%E6%9C%BAV0.0.2&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"answer","sourceId":1587866158}" \t "https://www.zhihu.com/question/308886134/answer/_blank)**》. 中华人民共和国国家版权局，SD-2021R11L2844054. 2021. (登记号:2022SR0011067) 软著登字第8965266号.

2021年12月26日 13.罗瑶光. 《**TinShell插件\_元基花模拟染色体组计算索引系统 V20211227**》. 中华人民共和国国家版权局，SD-2021R11L3629232. 2022. (受理号:2022R11S0138561，登记号2022SR0309512) 软著登字第9263711号.

2022年01月27日 14.罗瑶光, 罗荣武. 《**DNA元基催化与肽计算 第四修订版 V00919**》. 中华人民共和国国家版权局，SD-2022Z11L0025809. 2022. (受理号:2022Z11S1032939).

[词条引用日期2020-03-05] 15.类人数据生命的DNA计算思想 Github[https://github.com/yaoguangluo/Deta\_Resource](https://link.zhihu.com/?target=https://github.com/yaoguangluo/Deta_Resource" \t "https://www.zhihu.com/question/308886134/answer/_blank)