

# DNA 计算机中广义表数据结构的设计与实现

李汪根<sup>1),2)</sup> 丁永生<sup>1),3)</sup> 任立红<sup>1)</sup>

<sup>1)</sup>(东华大学信息科学与技术学院 上海 201620)

<sup>2)</sup>(安徽师范大学数学与计算机科学学院 安徽 芜湖 241000)

<sup>3)</sup>(数字化纺织服装技术教育部工程研究中心 上海 201620)

**摘 要** 类似于电子计算机,数据结构能帮助 DNA 计算机合理、高效地组织要处理的信息.文中提出了 DNA 计算机中广义表的一种设计方法.首先,讨论了  $k$ -臂 DNA 分子的结构及其在 DNA 计算中的应用.接着,在讨论了广义表存储结构的同时,给出了广义表两种节点的  $k$ -臂 DNA 编码的形式描述.最后详细描述了 DNA 计算机中广义表主要操作的实现算法.这些操作包括初始化空的广义表,创建包含指定元素的广义表和遍历广义表的元素.文中的方法可推广到 DNA 计算机上其它非线性数据结构.

**关键词** DNA 计算机;广义表;DNA 编码

**中图法分类号** TP301

## Design and Implementation of Generalized List in DNA Computer

LI Wang-Gen<sup>1),2)</sup> DING Yong-Sheng<sup>1),3)</sup> REN Li-Hong<sup>1)</sup>

<sup>1)</sup>(College of Information Sciences and Technology, Donghua University, Shanghai 201620)

<sup>2)</sup>(College of Mathematics and Computer Sciences, Anhui Normal University, Wuhu, Anhui 241000)

<sup>3)</sup>(Engineering Research Center of Digitized Textile & Fashion Technology of Ministry of Education, Donghua University, Shanghai 201620)

**Abstract** Being similar to electronic computer, data structures in DNA computer can help to organize the information processed by DNA computer correctly and efficiently, and make DNA computer for practical applications. This paper proposes a method to construct a generalized list in DNA computer. Firstly, the structures and applications of  $k$ -arms molecules in DNA computer are discussed. Then, the storage structures of generalized list are discussed. At the same time, the DNA encodings with  $k$ -arms molecules for the nodes of generalized list are formally given out. Finally, the algorithm of main bio-operations on a generalized list in DNA computer are described in detail, which include initializing an empty generalized list, creating a generalized list with giving element, and traversing a generalized list. Based on this method, other nonlinear data structures in DNA computer will be developed.

**Keywords** DNA computer; generalized list; DNA encoding

## 1 Introduction

Data processed by DNA computer are stored

in DNA molecules, and the processing is implemented by means of gene engineering such as hybrid, synthesizing, cutting and so on. The best

收稿日期:2008-04-20;最终修改稿收到日期:2008-10-20. 本课题得到国家自然科学基金(60474037)、上海市科学技术委员会重点项目(08JC1400100)、上海市人才发展基金(001)及上海市领军人才后备人选专项资金资助. 李汪根,男,1973年生,博士,副教授,主要研究领域包括生物计算机、生物信息学和智能计算. E-mail: wgli@mail.dhu.edu.cn. 丁永生(通信作者),男,1967年生,博士,教授,博士生导师,主要研究领域包括计算智能、网络智能、DNA 计算、人工免疫系统、生物网络结构、生物信息学、数字化纺织服装技术和智能决策. E-mail: ysding@dhu.edu.cn. 任立红,女,1966年生,博士,副教授,主要研究领域包括计算智能、网络智能、生物计算、生物信息学和生物网络结构.

virtues of DNA computing are the maximum density of storage, the minimum waste of energy and the huge parallel of computing. Many achievements have been obtained in the fields of DNA computing<sup>[1-2]</sup>. For the first time, Dr. Adleman solved the Hamilton Path Problem with seven nodes in the molecular lab in 1994<sup>[3]</sup>. This demonstrated the feasibility of DNA computing with molecules. Lipton proposed a method to solve the Satisfiability Problem by DNA computing in 1995<sup>[4]</sup>. Ouyang et al. settled the Maximum Connected Sub-graph with six nodes by building molecules pools corresponding with total possible connected sub-graph in 1997<sup>[5]</sup>. On the other hand, Guarnieri et al. gave out a system of addition operation in DNA computer in 1996<sup>[6]</sup>. Yurke et al. put forward another model of addition operation by separating input strands from the operator strands in 1999<sup>[7]</sup>. To mimic Turing machine, Shapiro et al. brought forward a programmable and autonomous computing machine made of biomolecules in 2001<sup>[8]</sup>. In this machine, the hardware was biology enzymes and the software was pre-coded DNA molecules. In 2004, Shapiro et al. devised an autonomous molecular computer for logical control of gene expression<sup>[9]</sup>.

As we known, data structure in electronic computer is used to organize the information, usually in memory, for better algorithm efficiency. Some typical data structures are linear list, queue, stack, heap, dictionary, tree, and graph, sometimes a conceptual unit<sup>[10]</sup>. Being similar to electronic computer, data structure can help to organize the information processed by DNA computer correctly and efficiently, and make DNA computer for practical application.

In this paper, we propose a model of generalized list in DNA computer. We discuss the storage structure and DNA encodings for the nodes of generalized list. And, the implementations of all bio-operations on generalized list are described in detail.

2 Design of Generalized List in DNA Computer

2.1 Storage Structure of Generalized List

The generalized list has been widely applied in computer. For example, they are the essential data structure in the programming language LISP. They extend the list and the elements may be alone atoms or sub-list.

For generalized list, it is hard to store the ele-

ments in sequence, because the structures of the elements are different. We usually use linked list as the storage structure of generalized list, and we use one node to represent for each element. It will split into head and tail for a nonempty generalized list. That is to say, a pair of head and tail can make sure a certain generalized list.

Because the element may be atom or list, it needs two types of nodes. One is list node, and the other is atom node. As shown in Fig. 1, the list node contains three parts: *tag*, *hp* and *tp*. While the atom node contains two parts: *tag* and *atom*. If the value of *tag* is one, it means this is a list node. If the value of *tag* is zero, it means this is an atom node. The value of *hp* indicates the pointer of head of the generalized list. The value of *tp* indicates the pointer of tail of the generalized list. The *atom* is the value of the data.

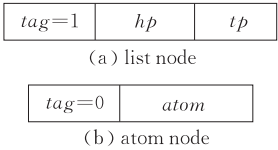


Fig. 1 The structure of the link-list-node of the generalized list

2.2 k-Arms DNA Molecule

Computing with *k*-arms molecule was proposed by Jonoska et al. in 1999<sup>[12]</sup>. The DNA algorithm was used to solve 3-SAT and 3-VCP, and the steps was equal to the number of the variable. In 2002, Jonoska et al. constructed corresponding three dimensional DNA graph with this *k*-arms molecule<sup>[13]</sup>. In this graph, they used *k*-arms molecule to represent the vertex which the degree was *k*. The molecules similar with *k*-arms already exist in the nature, such as Holliday. It is feasible to construct three dimensional DNA graph with *k*-arms molecule in the biological lab. Taking 2-arms, 3-arms and 4-arms as example, the structures of these molecules are shown in Fig. 2. Based on these characters, we use 3-arms molecule to represent the node of generalized list.

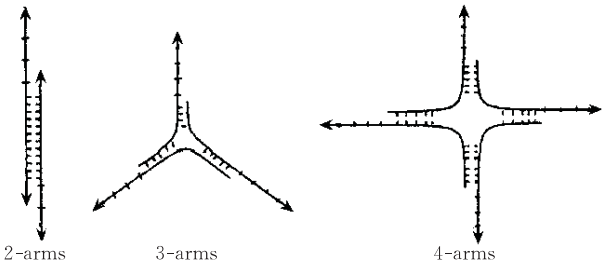


Fig. 2 The structure of the *k*-arms DNA

## 2.3 DNA Encodings for the Node of Generalized List

As mentioned above, there are two types of nodes in generalized list. To intuitively represent the logical structure of generalized list in DNA computer, we use 3-arms molecule to construct the two nodes (see Fig. 3).

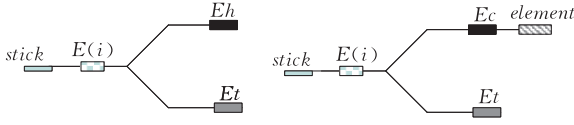


Fig. 3 The node of generalized list.

(1)  $E(i)$ : The recognition site of one restriction enzyme. The aim to use this enzyme is that the node will be cut from the generalized list at the arm of  $E(i)$ . This is very important in the process of generalized list traversal.

(2)  $Eh$ : The recognition site of another restriction enzyme. Under the active condition of this enzyme, the node will be cut and leave a sticky end at the arm of  $Eh$ . This will make preparations to produce a new head node of the generalized list by linking with the node designed in advanced.

(3)  $Et$ : The recognition site of another restriction enzyme. Under the active condition of this enzyme, the node will be cut and leave a sticky end at the arm of  $Et$ . This will make preparations to produce a new tail node of the generalized list by linking with the node designed in advance.

(4) *stick*: A specifically single-stranded DNA sequence. The node designed in advance will link to the generalized list by ligation at this position.

(5) *element*: A specifically double-stranded DNA sequence. This sequence represents the data of the element.

(6)  $Ec$ : The recognition site of another restriction enzyme. Under the active condition of this enzyme, the value of the element will be distinguished from each other.

In order to make the new node link to the generalized list, the sticky ends produced by  $Et$  and  $Eh$  must be the same. The operations of linking a new node and distinguishing the element are not happened at the same time, so the enzyme  $Et$  and

$Eh$  must be different. If they are the same, the one of these two operations will be touched by the other. Otherwise, to complete the traversal on the generalized list, the enzyme  $E(i)$  for  $i$ th node must be different from the others.

## 3 Operations on Generalized List in DNA Computer

In this section, we will discuss the basic operations on generalized list in DNA computer. These operations include initializing an empty list, creating a list and traversing a list. Here, we denote a generalized list in DNA computer with symbol  $L$ .

### 3.1 Initializing an Empty Generalized List

An empty generalized list with none element is shown in Fig. 4. Here, fragment *sign* is special base pairs designed in advance, and they are at the position of cutting site of enzyme  $E0$ . To judge the node is at the head of the generalized list or not, the digestion reaction will be implemented under the active condition of the enzyme  $E0$ . The judgments will be known by the detector molecule corresponding with the fragment *sign*.

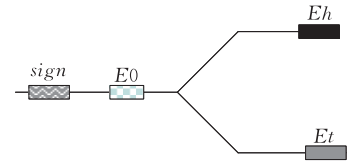


Fig. 4 The empty generalized list with none element

Because DNA molecule can be synthesized in the biological lab by gene engineering, the initializing operation will be completed by synthesizing or obtaining this 3-arms DNA molecule.

### 3.2 Creating a Generalized List

To create a generalized list, we construct an empty list firstly. Then the element will be inserted into the generalized list one by one according to its logical order. In the process of creating a generalized list, we need to prepare the following DNA fragments besides the empty list.

(1) Element node. As shown in Fig. 5(a), all nodes finally inserted into the generalized list are obtained from the corresponding element nodes by

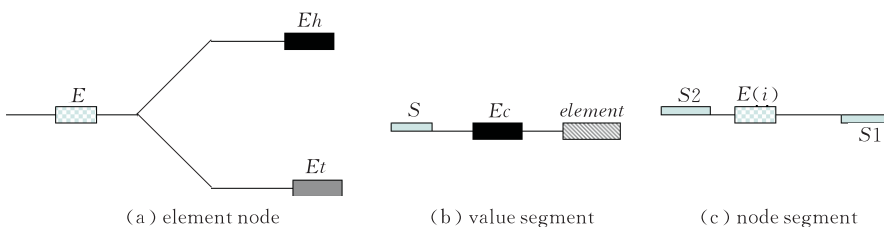


Fig. 5 DNA fragments for a generalized list

the following operations. Here,  $E$  is some restriction enzyme.

(2) Value segment. As shown in Fig. 5(b), it is used to construct atom node with element node, and it is a linear DNA molecule. This segment contains three parts. One is a sticky end named  $S$ . The other is a recognition site of an enzyme named  $E_c$ . The third is to represent the value of the element named  $element$ . To add this  $element$  to the element node, the sticky end of  $S$  must be complemented with the sticky end produced by enzyme  $E_h$  in the element node.

(3) Node segment. As shown in Fig. 5(c), it is used to construct the finally node with element node. This segment contains three parts too. One is a restriction enzyme named  $E(i)$ . The other two are sticky ends named  $S1$  and  $S2$  respectively. To add this enzyme  $E(i)$  to the element node, the sticky end of  $S1$  must be complemented with the sticky end produced by enzyme  $E$  in the element node. The sticky end of  $S2$  must be complemented with the sticky end produced by enzyme  $E(i-1)$  in the  $(i-1)$ th node.

When we construct these three DNA segments, we must design the rest DNA sequences according to the recognition site and cutting site of

these restriction enzymes. For a generalized list with  $n$  data and  $m$  list nodes, the number of value segments and node segments is  $n$  and  $m$  respectively.

We take  $L=(a,b,c)$  as an example to illuminate the steps of creating a generalized list.

Step 1. Synthesize an empty generalized list, three value segments, two node segments and one element node.

Step 2. Digest the generalized list under the active condition of enzyme  $E_h$ ; form a sticky end named  $S'$ ; hybrid and ligate the value segment to the generalized list.

Step 3. Digest the generalized list under the active condition of enzyme  $E$ ; form a sticky end named  $S1'$ ; hybrid and ligate the node segment named node1 with a complemented sticky end to the element node.

Step 4. Digest the generalized list under the active condition of enzyme  $E_t$ ; form a sticky end named  $S2'$ ; hybrid and ligate the element node obtained in step 3 to the generalized list.

Step 5. Repeat step 2 ~ step 4, add data  $(b,c)$  to the generalized list.

The process of adding one data  $a$  to the generalized list is illustrated in Fig. 6.

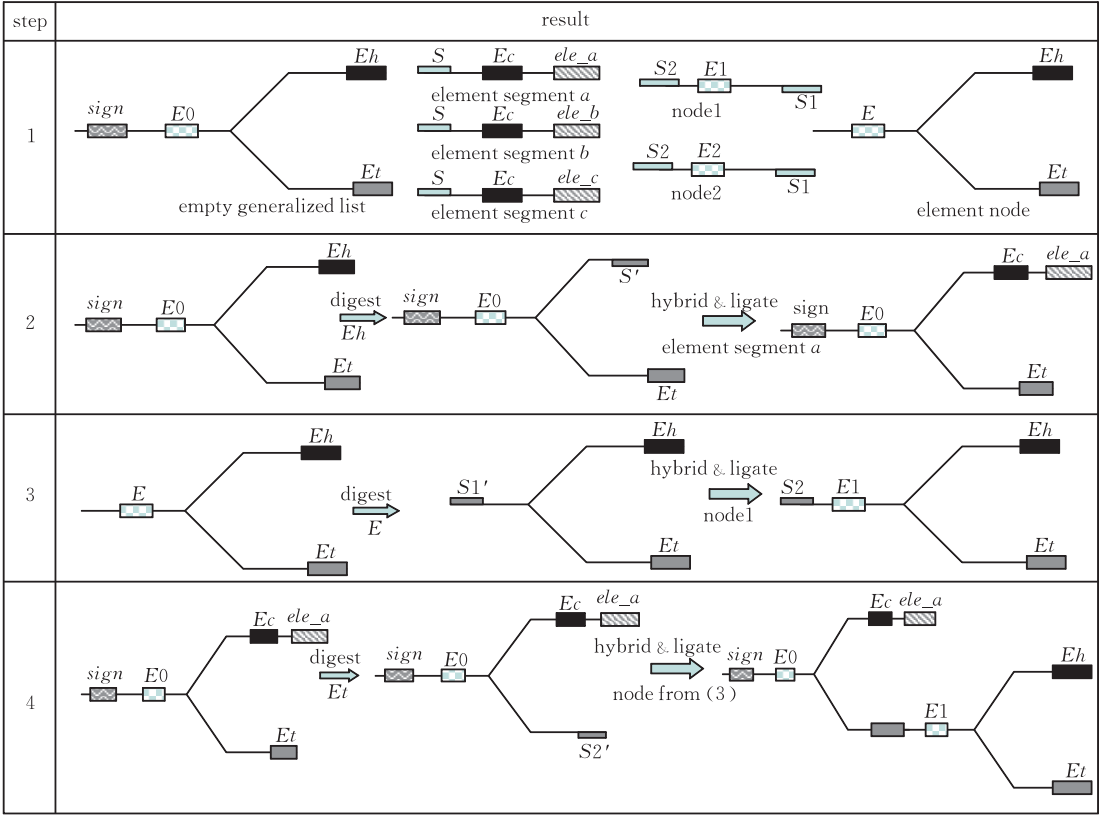


Fig. 6 The procedure of adding element  $a$

### 3.3 Traversing a Generalized List

To traverse a generalized list, we must begin with the last node until the first one. For the head node, we need design appropriate detector molecule to detect the symbol sequence *sign*.

Here, we explain the procedure to traverse the element *c* based on the generalized list  $L = (a, b, c)$  constructed in section 3.2.

Step 1. Digest the generalized list under the

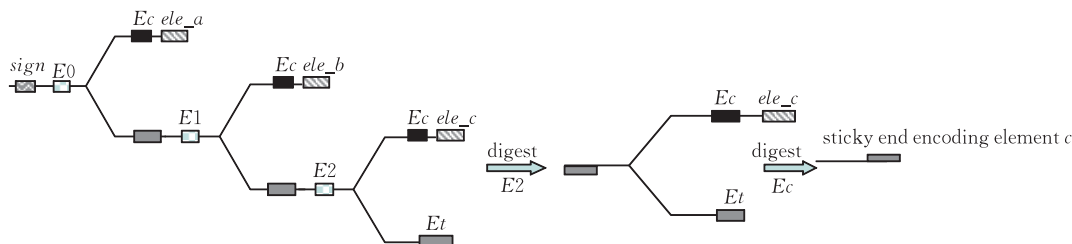


Fig. 7 The procedure of getting element *c*

## 4 Conclusion

This paper studied the implementation of the generalized list in DNA computer. We discussed the storage structure, the DNA encodings of element and the main bio-operations of generalized list in DNA computer. The model proposed in this paper is good only for this generalized list which is linear in logic. The model which is effect to the recursive generalized list is the advanced work.

### References

- [1] Wang Lei, Lin Ya-Ping, Li Zhi-Yong. DNA computation for a category of special integer planning problem. Chinese Journal of Computer Research and Development, 2005, 42(8): 1431-1437(in Chinese)
- [2] Xu Jin, Zhang Lei. DNA computer principle, advances and difficulties (I): Biological computing system and its applications to graph theory. Chinese Journal of Computers, 2003, 26(1): 1-11(in Chinese)  
(许进, 张雷. DNA 计算机原理、进展及难点(I): 生物计算系统及其在图论中的应用. 计算机学报, 2003, 26(1): 1-11)
- [3] Adelman L M. Molecular computation of solutions to combinatorial problems. Science, 1994, 266: 1021-1024

active condition of enzyme *E2*; the last node is cut from the generalized list.

Step 2. Digest the last node under the active condition of enzyme *Ec*; form a sticky end corresponding to the data *c*; detect the sticky end by detector molecule designed in advance.

The process of getting *c* from the generalized list is illustrated in Fig. 7.

- [4] Lipton R J. DNA solution of hard computational problem. Science, 1995, 268: 542-545
- [5] Ouyang Q, Kaplan P D, Liu S, Libchaber A. DNA solution of the maximal clique problem. Science, 1997, 278: 446-449
- [6] Guarnieri F, Fliss M, Bancroft C. Making DNA add. Science, 1996, 273: 220-223
- [7] Yurke et al. DNA implementation of addition in which the input strands are separate from the operator strands. Biosystems, 1999, 52: 165-174
- [8] Benenson Y, Paz-Elizur T, Adar R, Keinan E, Livneh Z, Shapiro E. Programmable and autonomous computing machine made of biomolecules. Nature, 2001, 414: 430-434
- [9] Benenson Y, Gil B, Ben-Dor U, Adar R, Shapiro E. An autonomous molecular computer for logical control of gene expression. Nature, 2004, 429: 423-429
- [10] Yan Wei-Min, Wu Wei-Min. Data Structure, 2nd Edition. Beijing: Tinghua University Press, 1992(in Chinese)
- [11] Ding Yong-Sheng, Shao Shi-Huang, Ren Li-Hong. DNA Computing and Soft Computing. Beijing: Scientific Publishing House, 2002(in Chinese)
- [12] Jonoskan, Stephenak, Saitom. Three dimensional DNA structures in computing. Biosystems, 1999, 52: 143-153
- [13] Sa-Ardyepn, Jonoskan, Seemannc. Self-assembling DNA graphs//Proceedings of the 8th International Workshop on DNA-Based Computers. Sapporo, Japan, 2002: 1-9



**LI Wang-Gen**, born in 1973, Ph. D., associate professor. His research interests include bio-computer, bioinformatics and intelligent computing.

Ph. D. supervisor. His research interests include computational intelligence, network intelligence, DNA computing, artificial immune systems, bio-network architecture, bio-informatics, digitized textile & fashion technology, and intelligent decision-making.

**REN Li-Hong**, born in 1966, Ph. D., associate professor. Her research interests include computational intelligence, network intelligence, bio-computing, bio-informatics, and bio-network architecture.

**DING Yong-Sheng**, born in 1967, Ph. D., professor,