给定条件下解决DNA序列的k-mer index问题的简易程序

需求分析说明书

2016年12月27日

徐维嘉 1351371 laoxu1994@gmail.com

王俊杰 1452658 1452658@tongji.edu.cn

吴硕旻 1452675 simonwux@126.com

[一. 项目概述 6](#_Toc470991445)

[1. 编写目的 6](#_Toc470991446)

[2. 项目背景 6](#_Toc470991447)

[3. 参考资料 6](#_Toc470991448)

[4. 项目目标 6](#_Toc470991449)

[二. 可行性研究报告 7](#_Toc470991450)

[1. 可行性研究分析 7](#_Toc470991451)

[1.1 技术可行性 7](#_Toc470991452)

[1.2 经济可行性 7](#_Toc470991453)

[1.3 社会可行性 8](#_Toc470991454)

[2. 结论意见 8](#_Toc470991455)

[三. 需求分析报告 8](#_Toc470991456)

[1. 引言 8](#_Toc470991457)

[1.1 编写目的 8](#_Toc470991458)

[1.2 背景及范围 8](#_Toc470991459)

[1.3 名词定义 9](#_Toc470991460)

[2. 任务概述 9](#_Toc470991461)

[2.1 目标 9](#_Toc470991462)

[3. 具体需求分析 11](#_Toc470991463)

[3.1 系统流程图 11](#_Toc470991464)

[3.2 数据流图 12](#_Toc470991465)

[3.3 数据字典 12](#_Toc470991466)

[3.4 功能性需求 13](#_Toc470991467)

[3.5 非功能性需求 14](#_Toc470991468)

[四. 算法分析 15](#_Toc470991469)

[1. 符号说明 15](#_Toc470991470)

[2. 模型建立 15](#_Toc470991471)

[2.1 问题分析 15](#_Toc470991472)

[2.2 索引算法模型 16](#_Toc470991473)

[2.3 查找算法模型 17](#_Toc470991474)

[2.4 数据分析与算法模型优化 18](#_Toc470991475)

[3. 模型求解 21](#_Toc470991476)

[3.1 建立索引表的复杂度分析 21](#_Toc470991477)

[3.2 使用索引查询的复杂度分析 21](#_Toc470991478)

[3.3 理论内存占用分析 22](#_Toc470991479)

[五. 概要设计说明书 23](#_Toc470991480)

[1. 引言 23](#_Toc470991481)

[1.1 编写目的 23](#_Toc470991482)

[1.2 背景 23](#_Toc470991483)

[1.3 定义 23](#_Toc470991484)

[2. 总体设计 24](#_Toc470991485)

[2.1 需求规定 24](#_Toc470991486)

[2.2 运行环境 25](#_Toc470991487)

[2.3 基本设计概念和处理流程 25](#_Toc470991488)

[2.4 结构 27](#_Toc470991489)

[2.5 功能需求与程序的关系 27](#_Toc470991490)

[2.6 人工处理过程 27](#_Toc470991491)

[3. 接口设计 28](#_Toc470991492)

[3.1 用户接口 28](#_Toc470991493)

[3.2 外部接口 28](#_Toc470991494)

[3.3 内部接口 28](#_Toc470991495)

[4. 运行设计 28](#_Toc470991496)

[4.1 运行模块组合 28](#_Toc470991497)

[4.2运行控制 29](#_Toc470991498)

[5. 系统数据结构设计 29](#_Toc470991499)

[5.1 逻辑结构设计要点 29](#_Toc470991500)

[6. 系统出错处理设计 29](#_Toc470991501)

[6.1 出错信息 29](#_Toc470991502)

[6.2 补救措施 29](#_Toc470991503)

[六. 详细设计说明书 30](#_Toc470991504)

[1. 引言 30](#_Toc470991505)

[1.1 项目说明 30](#_Toc470991506)

[1.2 项目定义 30](#_Toc470991507)

[2. 程序系统的结构 31](#_Toc470991508)

[3. 系统功能的实现 31](#_Toc470991509)

[3.1 读取文件 31](#_Toc470991510)

[3.2 建立索引 31](#_Toc470991511)

[3.3 DNA序列查询 31](#_Toc470991512)

[4. 内部接口说明 32](#_Toc470991513)

[七. 软件测试 32](#_Toc470991514)

[1. 操作方式 32](#_Toc470991515)

[1.1 读取数据 32](#_Toc470991516)

[1.2 建立索引 33](#_Toc470991517)

[1.3 DNA序列查询 34](#_Toc470991518)

[2. 调和测试 35](#_Toc470991519)

[2.1 K=5的情况： 35](#_Toc470991520)

[2.2 K=15的情况 37](#_Toc470991521)

[八. 结论 38](#_Toc470991522)

[1 模型实际效果 38](#_Toc470991523)

[1.1 查询时间 39](#_Toc470991524)

[1.2 支持k值范围 39](#_Toc470991525)

[1.3 建立索引表的时间 39](#_Toc470991526)

[2. 模型的比较与评价 39](#_Toc470991527)

[2.1 顺序查找法 39](#_Toc470991528)

[2.2 模型的评价 40](#_Toc470991529)

## 一. 项目概述

### 1. 编写目的

本需求分析说明书是为同济大学软件工程课程设计项目“给定条件下解决DNA序列的k-mer index问题的简易程序”而撰写的。通过介绍项目背景、目标、功能性需求、非功能性需求进行需求分析，合理计划本项目的进度，论证本项目进行思路的可行性。

### 2. 项目背景

这是一个来自 DNA序列的k-mer index问题。

给定一个DNA序列，这个系列只含有4个字母ATCG，如 S =“CTGTACTGTAT”。给定一个整数值k，从S的第一个位置开始，取一连续k个字母的短串，称之为k-mer（如k= 5，则此短串为CTGTA），然后从S的第二个位置，取另一k-mer（如k= 5，则此短串为TGTAC），这样直至S的末端，就得一个集合，包含全部k-mer。如对序列S来说，所有5-mer为：

｛CTGTA，TGTAC，GTACT，TACTG，ACTGT，TGTAT｝

通常这些k-mer需一种数据索引方法，可被后面的操作快速访问。例如，对5-mer来说，当查询CTGTA，通过这种数据索引方法，可返回其在DNA序列S中的位置为｛1，6｝。

### 3. 参考资料

《实践者的研究方法》 Roger S. Pressman 机械工业出版社

### 4. 项目目标

依照背景进行软件开发，根据相应要求达成以下目标：

以文件形式给定100万个 DNA序列，序列编号为1-1000000，每个基因序列长度为100。

（1）对给定k，实现数据索引，可返回任意一个k-mer所在的DNA序列编号和相应序列中出现的位置。每次建立索引，只需支持一个k值即可，不需要支持全部k值。

（2）索引一旦建立，查询速度尽量快，所用内存尽量小。

（3）在开发过程中给出建立索引所用的计算复杂度，和空间复杂度分析。

（4）在开发过程中给出使用索引查询的计算复杂度，和空间复杂度分析。

（5）假设内存限制为8G，能够分析所设计索引方法所能支持的最大k值和相应数据查询效率。

## 二. 可行性研究报告

### 1. 可行性研究分析

#### 1.1 技术可行性

题目要求为1000000条信息建立索引，实现快速访问。通常建立快速访问索引的方法是建立哈希表，由于不论k的取值为多少，哈希表消耗内存都是固定的，而从demo尝试的结果以及之后会提及的内存占用分析来看，所需内存都不会超过4GB，因而在有8GB的内存的情况下，使用哈希索引的方式可行。之后的算法部分也会具体讲到有关实现的部分。

#### 1.2 经济可行性

研究相关算法并开发对应软件，规模限制在3人团队，工作时间约为半个月业余时间，因而经济支出较小。虽然属于学术范畴无最终收益，但在开发过程中，查找研究算法、进行团队分工、实际开发所带来的锻炼值得此次开发。

#### 1.3 社会可行性

本系统涉及到的DNA文件由使用者提供，所涉及到的隐私权与本系统无关，由文件提供者负责。全部软件购买正版；机器设置通过正当途径购得；所有软件都用正版，技术资料都由提出方保管，数据信息均可保证合法来源。所以，在法律方面是可行的。

### 2. 结论意见

通过对项目的技术、经济、社会三个方面的可行性研究，我们小组一致认为该项目可以实现。

## 三. 需求分析报告

### 1. 引言

#### 1.1 编写目的

需求分析报告的编写目的是为了使用户和软件开发者方对该系统应具有的功能达成共识，该分析报告将指导DNA序列的k-mer index问题的简易程序的开发。

#### 1.2 背景及范围

这是一个来自 DNA序列的k-mer index问题。

给定一个DNA序列，这个系列只含有4个字母ATCG，如 S =“CTGTACTGTAT”。给定一个整数值k，从S的第一个位置开始，取一连续k个字母的短串，称之为k-mer（如k= 5，则此短串为CTGTA），然后从S的第二个位置，取另一k-mer（如k= 5，则此短串为TGTAC），这样直至S的末端，就得一个集合，包含全部k-mer。如对序列S来说，所有5-mer为：

｛CTGTA，TGTAC，GTACT，TACTG，ACTGT，CTGTA，TGTAT｝

通常这些k-mer需一种数据索引方法，可被后面的操作快速访问。例如，对5-mer来说，当查询CTGTA，通过这种数据索引方法，可返回其在DNA序列S中的位置为{1, 6}。

#### 1.3 名词定义

DNA序列：一串只含有“A、T、C、G”4种字母的字符串，在该项目中，DNA序列的字符串长度为100字符。

k-mer：k个字符的短串。

### 2. 任务概述

#### 2.1 目标

以文件形式给定 100万个 DNA序列，序列编号为1-1000000，每个基因序列长度为100 。

（1）对给定的k， 实现一种数据索引方法，可返回任意一个k-mer所在的DNA序列编号和相应序列中出现的位置。每次建立索引，只需支持一个k值，不需要支持全部k值。

（2）要求索引一旦建立，查询速度尽量快，所用内存尽量小。

（3）给出建立索引所用的计算复杂度，和空间复杂度分析。

（4）给出使用索引查询的计算复杂度，和空间复杂度分析。

（5）假设内存限制为8G，分析所设计索引方法所能支持的最大k 值和相应数据

查询效率。

（6）按重要性由高到低排列，将依据以下几点，来评价索引方法性能：

索引查询速度。

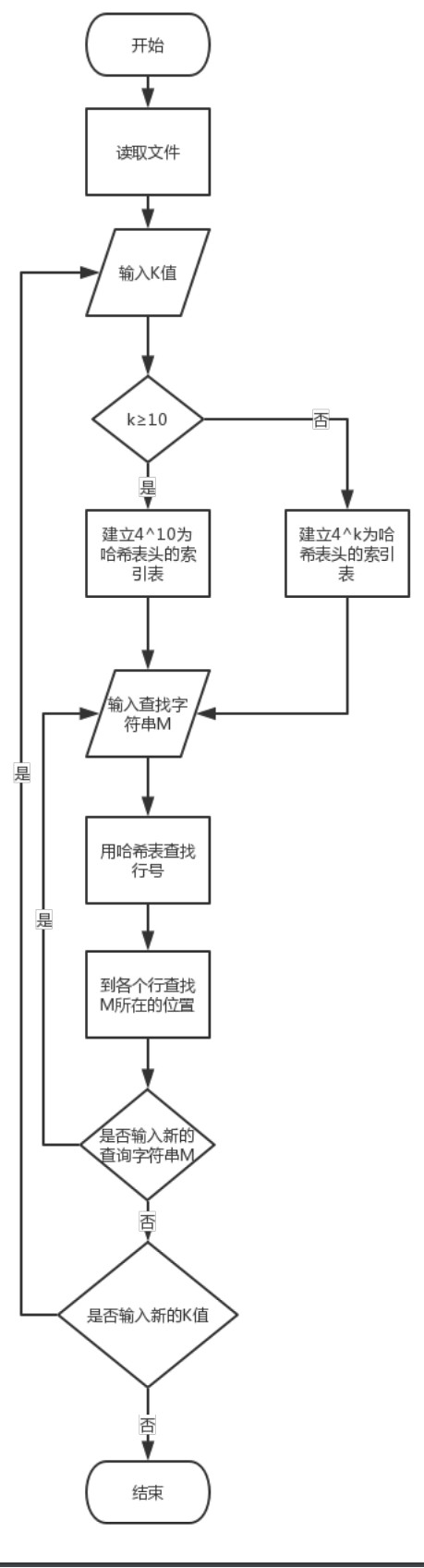
索引内存使用。

8G 内存下，所能支持的k 值范围。

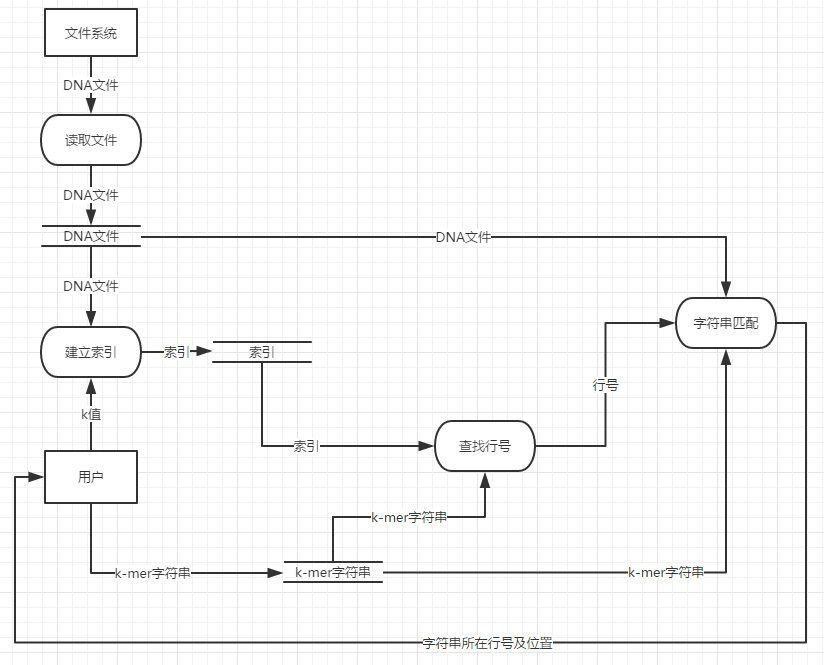
建立索引时间。

### 3. 具体需求分析

#### 3.1 系统流程图



#### 3.2 数据流图



#### 3.3 数据字典

（1）

|  |  |
| --- | --- |
| 名字 | DNA序列 |
| 别名 | DNA字符串 |
| 描述 | 从DNA序列文件中读取的DNA字符串，一共1000000条字符串，每条字符串100字符 |
| 定义 | DNA序列=[‘A’ | ’T’ | ’C’ | ’G’] |
| 位置 | 从DNA序列文件中读取存入内存中的数组中 |

（2）

|  |  |
| --- | --- |
| 名字 | k-mer序列 |
| 别名 | 长度为k的字符串 |
| 描述 | 用户输入的要查询的k位DNA字符串 |
| 定义 | k-mer序列=[‘A’ | ’T’ | ’C’ | ’G’] |
| 位置 | 用户输入的存在内存字符串变量中 |

（3）

|  |  |
| --- | --- |
| 名字 | 索引 |
| 别名 |  |
| 描述 | 根据用户输入的k值建立的索引 |
| 定义 | 索引=二维数组 |
| 位置 | 保存在内存中的二维数组 |

#### 3.4 功能性需求

##### 3.4.1 文件读取模块

（1）读取2个文件，分别为“solexa\_100\_170\_1.fa”和“solexa\_100\_170\_2.fa”，将文件中的1000000条DNA序列读入内存。

##### 3.4.2 索引建立模块

（1）对用户给定的K，通过哈希函数，为读入内存的1000000条DNA序列建立索引。每次建立索引只需支持一个k值即可，不需要支持全部k值。

##### 3.4.3 DNA序列查询模块

（1）对用户给定的K位DNA序列，返回其所在的所有序列编号和相应序列中出现的位置。

#### 3.5 非功能性需求

##### 3.5.1 性能需求

（1）鲁棒性

系统对于故障有一定的容错能力，例如可以检测用户的非法输入等。

（2）易用性

系统提供的命令行界面，用户只需要输入k值和需要查找的k-mer串即可，提高了系统的易操作性。

（3）可扩展性

系统做到高内聚，低耦合，不同的模块管理不同的功能，如果未来要增加新的功能，可以通过新增模块实现。

##### 3.5.2 效率需求

（1）时间效率

建立索引时间越快越好。

索引查询时间越快越好。

（2）空间效率

内存限制为8G。

索引内存使用越少越好。

## 四. 算法分析

### 1. 符号说明

k：k-mer中的k值，即k-mer的长度

M：待查找的k-mer序列

M1：待查找的k-mer序列的前十位（k>10时）

X/X1：k-mer存在的行号

Y：k-mer存在的位置号

(x，y):行号和位置号的二维数组，x为行号，y为位置号

f：哈希函数

S:建表时每次读取的k-mer

L(k)：对k建立索引后，每个索引表头的链表的平均长度

T(n)：时间复杂度

S(n)：空间复杂度

### 2. 模型建立

#### 2.1 问题分析

由于要将查询速度作为第一考量，因此需要较为细致的索引表，采取建立哈希表索引算法模型。采取KMP匹配算法来寻找具体的位置。

问题的另一关键点是：针对一个k建立一次索引。

#### 2.2 索引算法模型

##### 2.2.1 哈希表相关定义

哈希表：

哈希表是根据关键码值(Key value)而直接进行访问的数据结构。也就是说，它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做哈希函数。在本题中，关键码就是每个k-mer。

##### 2.2.2 基本步骤

（A）. 寻找哈希函数

由于DNA序列均有A、C、G、T四种碱基排列组合得到的，所以我们令哈希函数f为：f(A)=00，f(T)=01，f(C)=10，f(G)=11

这样对于每一个k-mer，都有唯一的一组长度为2k位的二进制数与其对应。

例如：k=4时，f(ATCG)=00011011

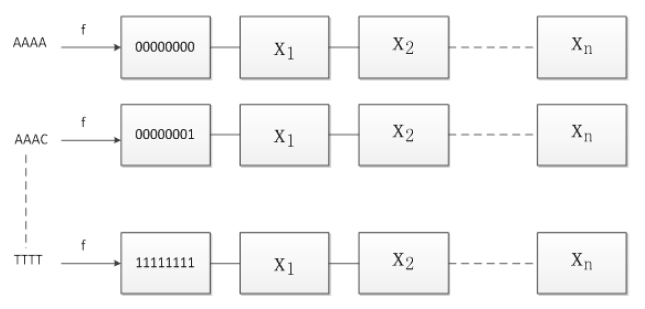
（B）. 建立寻址方法

直接寻址法：

由于当k固定时，每一种k-mer与每一个长度为2k的二进制数是双射的（即一一对应），且均为连续的。因此采用直接寻址法。以k=4为例，建立4^4个表头，编号为从00000000到11111111，每读取一个k-mer 序列S，将其行号存于表头编号为f（S）后面的链表中。

数据结构如下：

K=4的情况



各个链表长度不一定相同

#### 2.3 查找算法模型

为使查找速度最快，采用KMP算法对字符串进行匹配查找。

##### 2.3.1 KMP字符串匹配相关概念

KMP算法是一种改进的字符串匹配算法，关键是利用已经得到的信息，尽量减少模式串与主串的匹配次数以达到快速匹配的目的。

##### 2.3.2 主要步骤示例：

从S中查找T（S和T末尾的“\0”代表字符串的结束符），如图2所示：

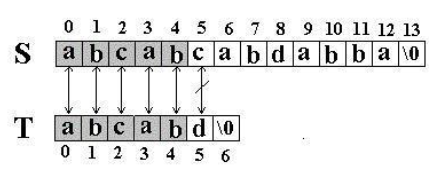


图2 KMP算法的第一趟匹配过程

先从S[0]与T[0]比较，S[0]=T[0]，则比较S[1]与T[1]⋯。当比较当发现S[5]不等于T[5]后，并不从头再寻找，由于在前面的比较中，已经保留了S与T匹配成功位置上的信息，发现T[0]T[1]=T[3]T[4]=S[3]S[4]，则将T后错三位，直接比较T[2]与S[5]，如图3所示。从而快速匹配成功。

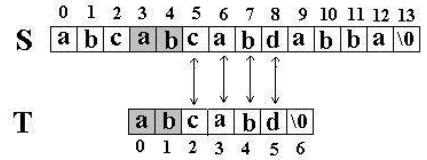


图3 KMP算法的第二趟匹配过程

而常规的匹配算法只会将T后错一位，将T[0]与S[1]进行比较。这使得以及读取的T的信息没有利用起来。而KMP弥补了这一点，高效地利用信息从而达到快速匹配。

#### 2.4 数据分析与算法模型优化

##### 2.4.1 每一种K-mer出现次数分析

对于给定的k值，理论上k个碱基能组成的k-mer种类为4^k。对于一百万条长度均为100的DNA来讲，假设每个k-mer不一样，实际上出现的种类最多为100 0000×(101−k)

设f(k)=4^k−1000000×(101−k) （3）

f(k) 与 k的关系如图4所示（ f(k) 为纵坐标， k为为横坐标）：

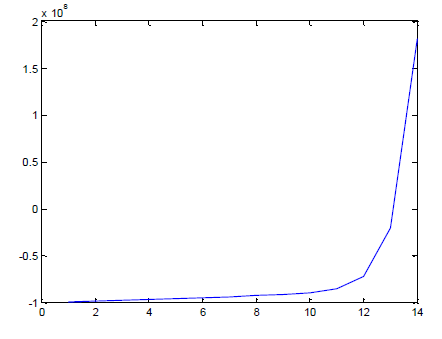


图4 f(k)的函数图像

如图4所示，当k大于13以后，4^k≫1000000×(101−k) ,也就是说，此时实际上出现的k-mer序列的种类远远小于理论上的种类，如果对k大于13建立哈希表的话，将会出现有很多表头后面链表长度为0的情况（因为有些k-mer种类没有出现过），会造成内存空间的极大浪费。

##### 2.4.2 每类K-mer的平均数组长度分析。

（1） 当k≤4，由于k-mer种类过少，假设每条DNA中都含有所有种类的k-mer。则每个k-mer表头对应的链表的长度为

L(k)=1000000 (k≤4)

当k>4时，假设每条DNA产生的k-mer无重复，则平均每种k-mer对应的表头后的链表的长度为L(k)=1000000×(101−k)/4^k (k>4)

L(k)的函数图像如图5所示，具体数据如表1所示：

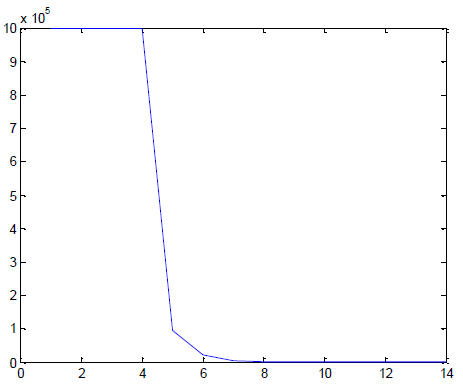


图5 L(k)的函数图像

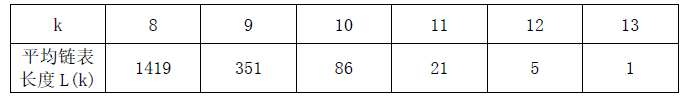


表1 k取不同值时L(k)的大小

通过对程序的实际测试，对于100万条DNA，链表长度大约为90左右

由此可见，当k过大时，链表长度过短，索引表过于稀疏，造成浪费，当k过小时，链表长度过长，不方便后续的具体位置的查找工作。

##### 2.4.3 算法模型优化

为了避免索引表过于稀疏，造成内存浪费，以及索引表过于繁琐，减慢查询速度的情况，通过对表1的分析，发现k取10时数组长度较为合适，选取10作为临界值。

以下为优化后的模型：

①当k<10时，直接建立4^k个表头的哈希表。查找时，读取M（待查找的k-mer），到关键码为f（M）的链表中提取行号，再用KMP算法到该行查找M出现的具体位置。

②当k>=10时，建立4^10个表头的哈希表。查找时，只对M（待查找的K-mer）的前10位即M1进行读取，到关键码为f（M1）的数组中提取行号，再到每个可能的行里用KMP算法匹配查找S并输出具体位置。（由于k大于10后，每个链表长度大概为80，查找并不会耗费太多时间）。

### 3. 模型求解

#### 3.1 建立索引表的复杂度分析

##### 3.1.1 时间复杂度

哈希表实际上是对所有数据进行遍历，所以其时间复杂度和数据量有关，所以时间复杂度为:T(n) = O(n)

##### 3.1.2 空间复杂度

由于不论k的取值为多少，哈希表消耗的内存都是较为固定的常值（后面的内存分析会详细说明），和k没有直接关系，但是和总数据量成线性关系，所以空间复杂度为：S(n)=O(n)

#### 3.2 使用索引查询的复杂度分析

索引查询分为哈希表查询行号和KMP算法查询位置号两部分。

##### 3.2.1 时间复杂度

A. KMP查询位置号

已知主字符串为S，待匹配的字符串串T，由于KMP算法的思想是主串不回溯的简化算法，匹配的时候只会将T不断右移进行比较，所以字符比较最多的情况就是遍历S中所有的元素，则时间复杂度为O（LenthS）,即KMP算法的时间复杂度为O（n）。对于本模型来讲，查找时DNA长度固定为100则比较次数平均为100\*80次，与k的选取以及总数据量无直接关系，所以查询位置号的时间复杂度为O(1)。

B. 哈希表查询行号

哈希表是通过计算关键码值（待查找k-mer对应的哈希函数值）来定位元素位置，所以只需一次即可，复杂度为O(1)。

C. 索引查询总时间复杂度:

T(n)=O(1)+O(1)=O(1)

##### 3.2.2 空间复杂度

A. 哈希查找

对于哈希查询部分，只需提取数据，空间复杂度为:S1(n)=O(1)

B. KMP匹配查找

由于KMP算法在计算时，占用内存用来保存比较串的信息，所以其空间复杂度为O（lenthT）.对于本模型来讲，长度为常值，所以空间复杂度为:S2(n)=O(1)

故索引查询空间复杂度为：S(n)=O(1)+O(1)=O(1)

#### 3.3 理论内存占用分析

由于本模型对于k>10均采用k=10建立索引表，所以以k=10为例分析内存占用空间。

在索引表中，k=10的情况下，一共有4^10个表头，平均每个表头后的链表长度为90，链表的每一个节点占用8B（即8字节）内存，。则：

理论总占用内存=4^10×90×8B=0.703GB

由于程序在运行时会占用一些内存空间，故本数据和实际运行有些许差别。

## 五. 概要设计说明书

### 1. 引言

#### 1.1 编写目的

编写软件概要设计说明是为了详细定义软件总体的功能、系统的接口和数据属性；划分程序基本结构、各程序名称和功能，以便于软件详细设计和编程。

#### 1.2 背景

这是一个来自 DNA序列的k-mer index问题。

给定一个DNA序列，这个系列只含有4个字母ATCG，如 S =“CTGTACTGTAT”。给定一个整数值k，从S的第一个位置开始，取一连续k个字母的短串，称之为k-mer（如k= 5，则此短串为CTGTA），然后从S的第二个位置，取另一k-mer（如k= 5，则此短串为TGTAC），这样直至S的末端，就得一个集合，包含全部k-mer。如对序列S来说，所有5-mer为：

｛CTGTA，TGTAC，GTACT，TACTG，ACTGT，TGTAT｝

通常这些k-mer需一种数据索引方法，可被后面的操作快速访问。例如，对5-mer来说，当查询CTGTA，通过这种数据索引方法，可返回其在DNA序列S中的位置为｛1，6｝。

#### 1.3 定义

DNA序列：一串只含有“A、T、C、G”4种字母的字符串，在该项目中，DNA序列的字符串长度为100字符。

k-mer：k个字符的短串。

### 2. 总体设计

#### 2.1 需求规定

##### 2.1.1 系统功能

（1）读取2个文件，分别为“solexa\_100\_170\_1.fa”和“solexa\_100\_170\_2.fa”，将文件中的1000000条DNA序列读入内存。

（2）对用户给定的K，通过哈希函数，为读入内存的1000000条DNA序列建立索引。每次建立索引只需支持一个k值即可，不需要支持全部k值。

（3）对用户给定的K位DNA序列，返回其所在的所有序列编号和相应序列中出现的位置。

##### 2.1.2 系统性能

（1）时间效率

建立索引时间越快越好。

索引查询时间越快越好。

（2）空间效率

内存限制为8G。

索引内存使用越少越好。

##### 2.1.3 输入输出要求

（1）读入DNA序列文件，保存在内存中

（2）允许用户输入k值，k值范围为：[2,100]

（3）允许用户输入k-mer字符串

（4）在cmd中输出k-mer字符串所在的行号和行中的位置

##### 2.1.5 故障处理要求

（1）若DNA序列文件读取失败，提示用户文件读取失败

（2）若用户输入的k值不符合规定，提示用户重新输入

（3）若用户输入的k-mer字符串长度不为k，提示用户重新输入

#### 2.2 运行环境

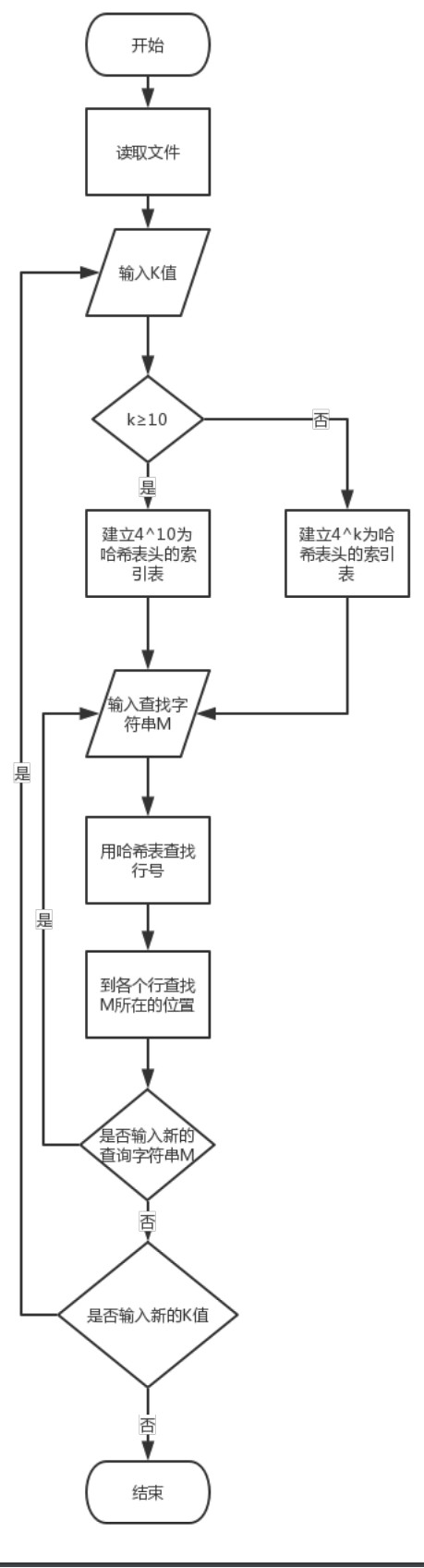
可以编译运行C程序代码的系统，系统的内存限制为8G。

#### 2.3 基本设计概念和处理流程

##### 2.3.1 基本设计概念

系统主要分为4个模块，一是文件读取模块，二是索引建立模块，三是DNA序列查询模块，四是系统主模块。文件读取模块读取存有DNA序列的文件。索引建立模块是在用户输入K值后，根据K值建立一个哈希表索引，通过索引可以快速查找行号。DNA序列查询模块是在用户输入要查询的k-mer字符串后，通过索引查找所在行号，之后查找字符串在行号中的位置。最后，通过系统主模块控制三个模块功能的协作运行。

##### 2.3.2 处理流程



#### 2.4 结构

|  |  |
| --- | --- |
| 模块 | 模块说明 |
| 主模块 | 系统主模块，控制其他模块按顺序执行 |
| 文件读取模块 | 读取文件磁盘中的DNA序列，将其读入内存 |
| 索引建立模块 | 根据k值建立索引 |
| DNA序列查询模块 | 根据k-mer字符串进行DNA序列查询 |

#### 2.5 功能需求与程序的关系

|  |  |  |  |
| --- | --- | --- | --- |
|  | 文件读取模块 | 索引建立模块 | DNA序列查询模块 |
| 读取DNA文件 | √ |  |  |
| 建立索引 |  | √ |  |
| 字符串查询 |  |  | √ |

#### 2.6 人工处理过程

用户需要将DNA序列文件放在指定的目录

用户需要手动输入k值

用户需要手动输入k-mer字符串

### 3. 接口设计

#### 3.1 用户接口

用户界面是cmd命令行

用户输入k：要查询的字符串长度

用户输入长度为k的字符串：要查询的字符串

系统会回复字符串所在的行数和行中的位置

#### 3.2 外部接口

系统是一个封闭的系统，所以不提供对外的接口。

#### 3.3 内部接口

主模块：系统运行时调用，用以控制其余模块按顺序执行

文件读取模块：由主模块调用，用以读取DNA序列文件

索引建立模块：在文件读取模块结束之后，由主模块调用，用以建立索引

DNA序列查询模块：在索引建立模块结束之后，由主模块调用，用以DNA序列查询

### 4. 运行设计

#### 4.1 运行模块组合

系统主模块先运行文件读取模块，文件读取完毕后，运行索引建立模块建立索引，最后运行DNA序列查询模块查询字符串。

#### 4.2运行控制

程序有主模块严格控制各个模块的运行顺序，用户无法随意更改模块的运行顺序。

### 5. 系统数据结构设计

#### 5.1 逻辑结构设计要点

二维数组存储DNA序列，存于内存中

预先分配二维数组用于建立索引表

### 6. 系统出错处理设计

#### 6.1 出错信息

|  |  |
| --- | --- |
| DNA文件读取不正确 | 可能是文件不存在 |
| K值错误 | 用户输入的K值不符合规范 |
| 要查询的k-mer字符串有误 | 用户输入的字符串不符合规范 |

#### 6.2 补救措施

分析上面的出错信息，得到主要的错误有：

（1）文件读取错误：这类错误主要是文件不存在造成的，可以提醒用户检查文件的路径是否正确。

（2）输入错误：这类错误主要是用户输入有误造成的，可以提醒用户输入规范，再让用户重新输入。

## 六. 详细设计说明书

### 1. 引言

#### 1.1 项目说明

这是一个来自 DNA序列的k-mer index问题。

给定一个DNA序列，这个系列只含有4个字母ATCG，如 S =“CTGTACTGTAT”。给定一个整数值k，从S的第一个位置开始，取一连续k个字母的短串，称之为k-mer（如k= 5，则此短串为CTGTA），然后从S的第二个位置，取另一k-mer（如k= 5，则此短串为TGTAC），这样直至S的末端，就得一个集合，包含全部k-mer。如对序列S来说，所有5-mer为：

｛CTGTA，TGTAC，GTACT，TACTG，ACTGT，TGTAT｝

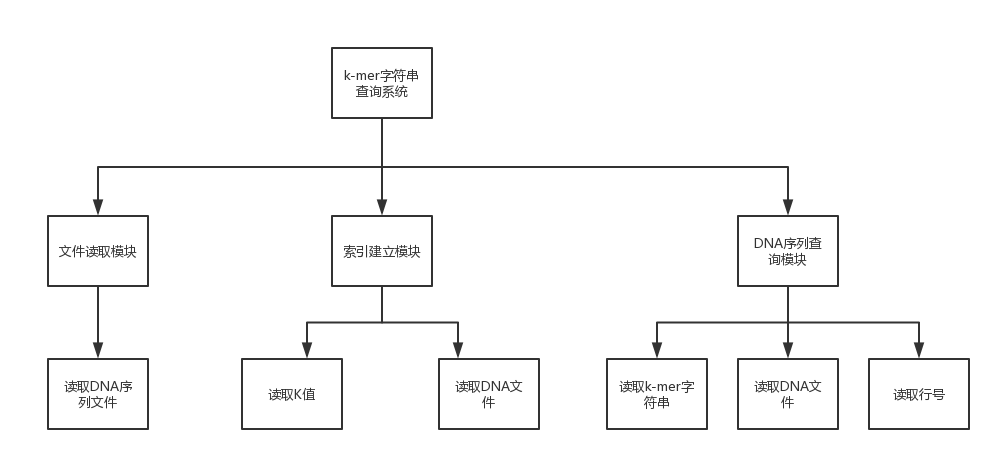
通常这些k-mer需一种数据索引方法，可被后面的操作快速访问。例如，对5-mer来说，当查询CTGTA，通过这种数据索引方法，可返回其在DNA序列S中的位置为｛1，6｝。

#### 1.2 项目定义

DNA序列：一串只含有“A、T、C、G”4种字母的字符串，在该项目中，DNA序列的字符串长度为100字符。

k-mer：k个字符的短串。

### 2. 程序系统的结构



### 3. 系统功能的实现

#### 3.1 读取文件

用C语言的文件输入输出操作，将DNA序列文件读入内存，读入内存后可以加快后续动作的查询速度。

#### 3.2 建立索引

读取用户输入的k值后，通过哈希函数将ATCG四个字符换算为4进制数，再将其转化为10进制数，再将其存入索引表中，索引表是一个二维数组，程序运行开始就创建了这一个二维数组，这样不需要再后续插入的过程中动态分配内存，可以加快插入的速度。

#### 3.3 DNA序列查询

读取用户输入的要查询的k-mer字符串，通过索引表查询出拥有此字符串的行号，之后在相应的行号中用KMP算法查询指定字符串，最后输出结果。

### 4. 内部接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 函数 | 参数 | 返回类型 | 说明 |
| int value(char ch) | char ch | int | 将‘A’、‘T’、‘C’、‘G’分别转化为0、1、2、3的整数 |
| int fun(string stt) | string stt | int | 将4进制数转化为10进制数 |
| void insert(int hash, int row) | int hash, int row | void | 将哈希值和行号插入索引表 |
| void read() | 无 | void | 读取DNA序列文件 |
| void makehash() | 无 | void | 建立索引表 |
| void check() | 无 | void | 查找字符串 |

## 七. 软件测试

### 1. 操作方式

#### 1.1 读取数据

将文件放入正确的路径之后，打开程序即可自动读取数据存入内存

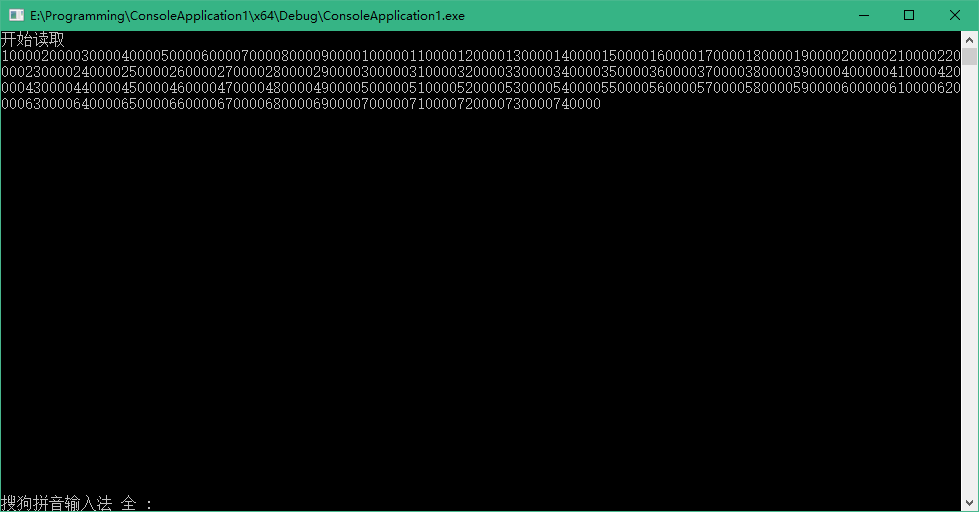


图7-1正在读取文件

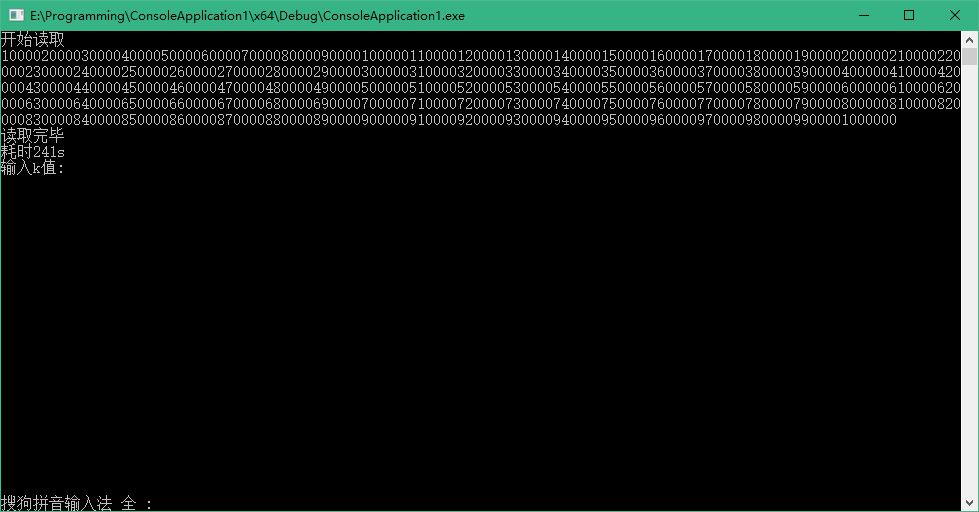


图7-2 读取文件结束

#### 1.2 建立索引

用户输入k值，程序接着会开始建立索引

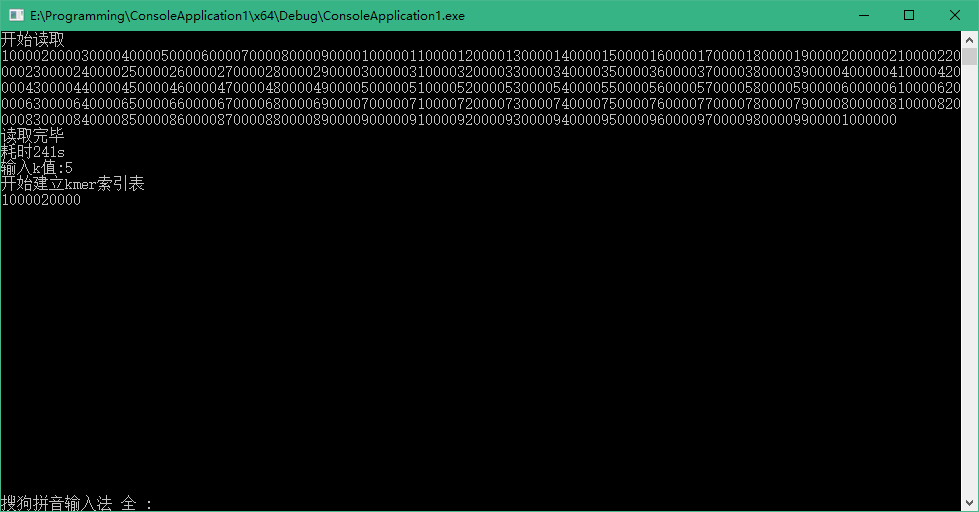


图7-3 输入k值，建立索引表

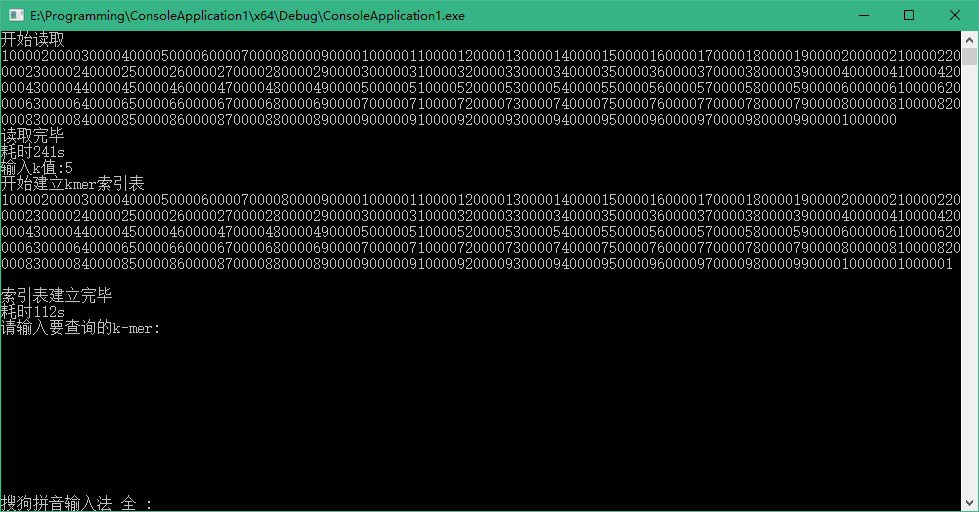


图7-4 索引表建立完毕

#### 1.3 DNA序列查询

用户输入要查询的k-mer字符串，程序开始查找字符串，输出结果

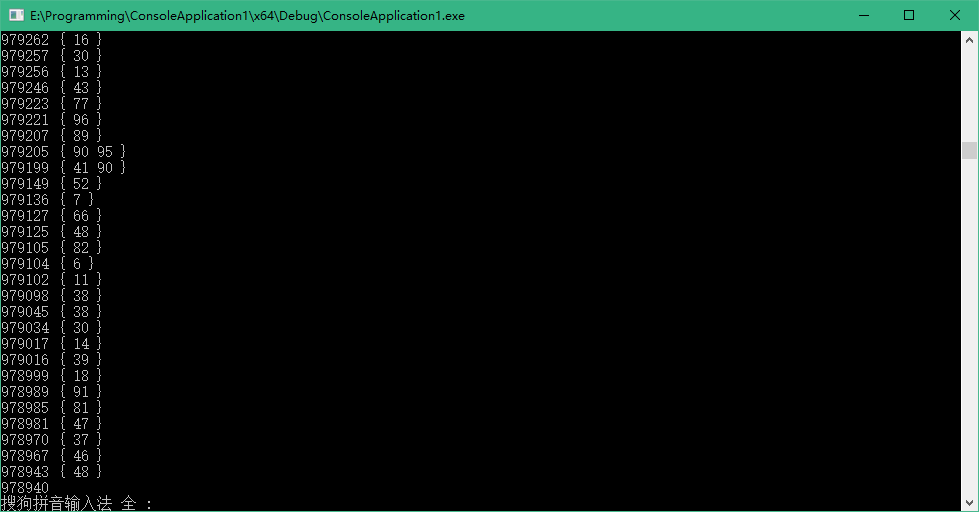


图7-5 正在输出结果

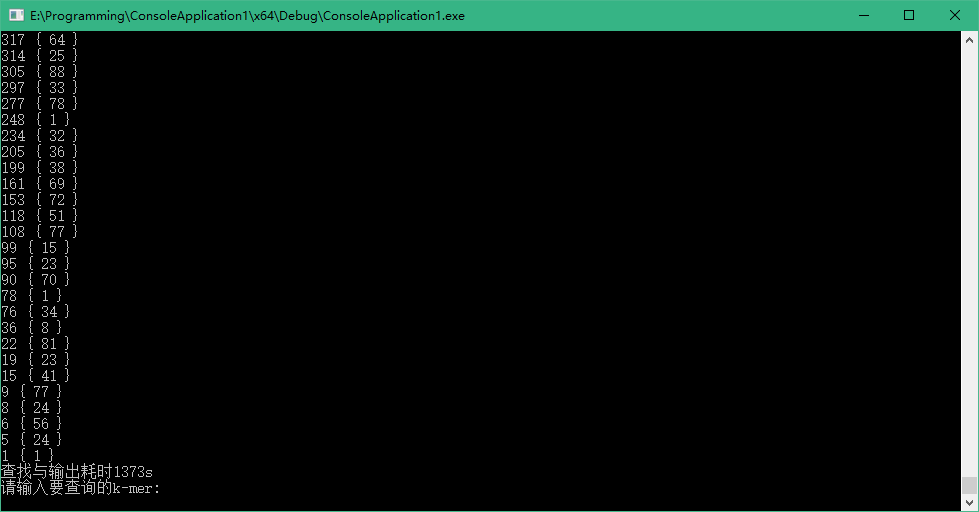
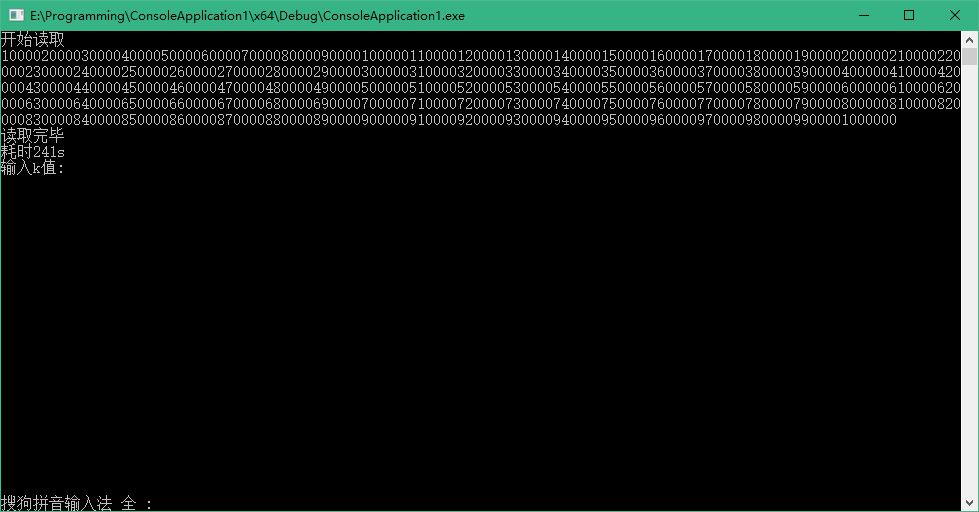


图7-6 结果输出结束

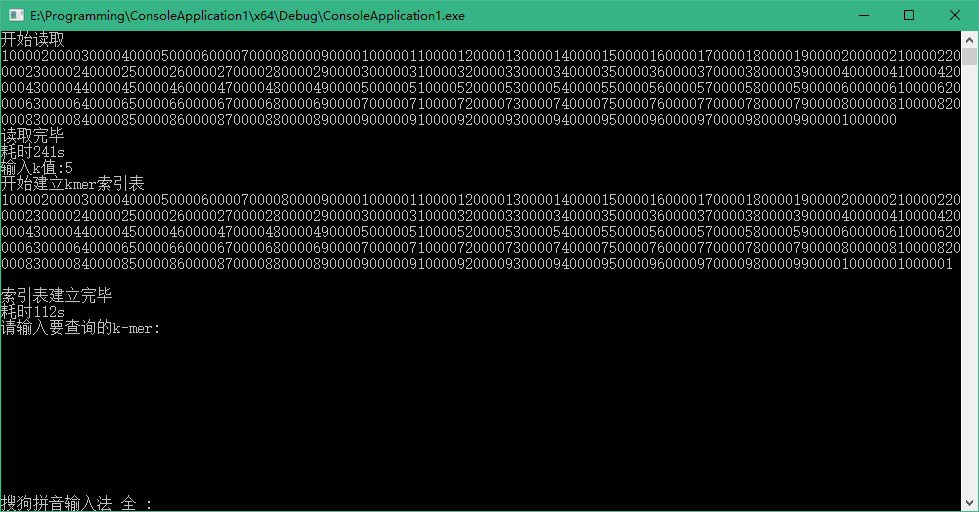
### 2. 调和测试

#### 2.1 K=5的情况：

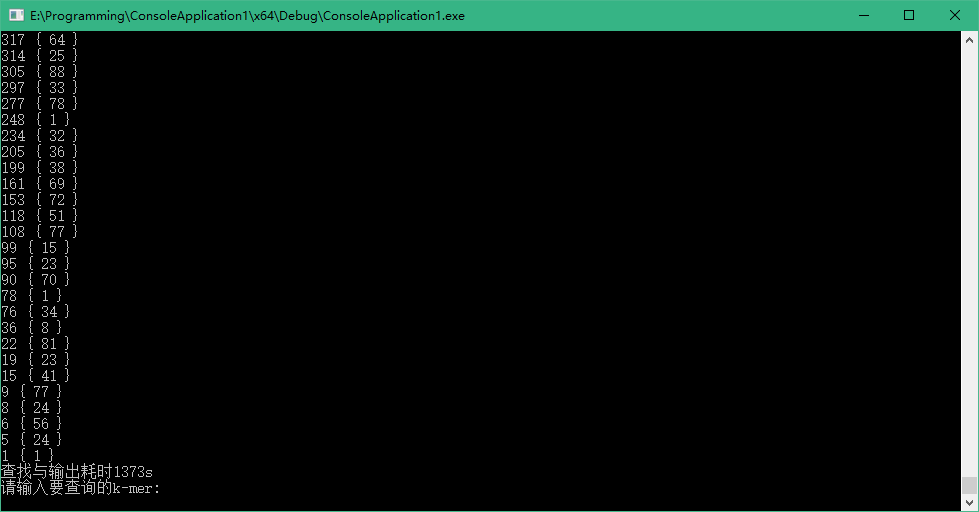
（1）读取文件：



（2）建立索引：

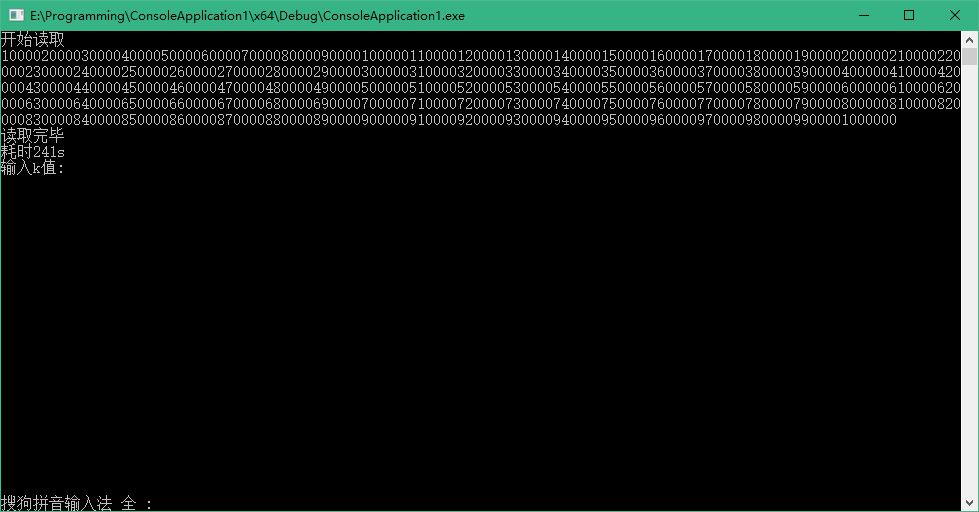


（3）DNA序列查询

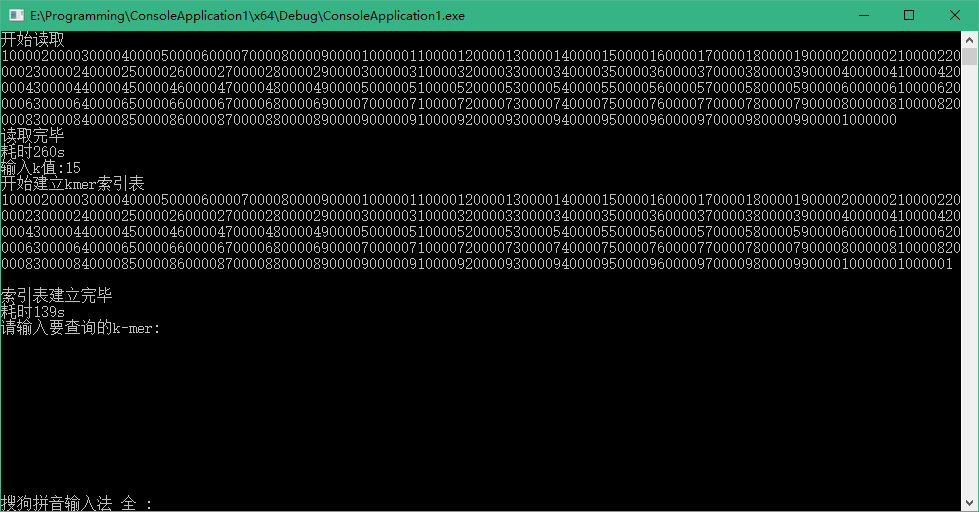


#### 2.2 K=15的情况

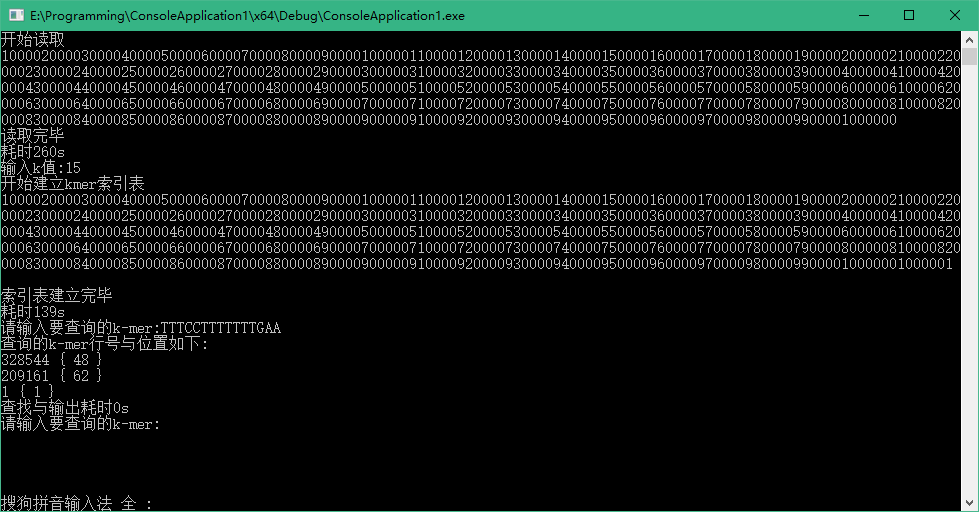
（1）读取文件：



（2）建立索引：



（3）DNA序列查询



## 八. 结论

### 1 模型实际效果

注：以下对100万条DNA进行读取、建表和检索。

运行环境为windows7 64位操作系统下，CPU为Intel Core i5处理器。

#### 1.1 查询时间

k=7的查询时间为0.68s。

k=10的查询时间为0.000s

k=23时的查询时间0.00000s

基本上k大于10之后查询时间都为0.00000

#### 1.2 支持k值范围

k支持2-100。

#### 1.3 建立索引表的时间

k=5 耗时40s建立索引。

k=7 耗时40s建立索引。

k=21，耗时52s建立索引。

k=80，耗时52s建立索引

基本上当k大于10之后建表时间普遍在52s附近，而k较小时建表时间要小于52s。

### 2. 模型的比较与评价

#### 2.1 顺序查找法

顺序查找法是一种很常见的程序设计查找方法，就是在一个已知无(或有序）序队列中找出与给定关键字相同的数的具体位置。

其基本原理就是：从表的一端开始，顺序扫描线性表，依次将扫描到的结点关键宇和给定值K相比较。若当前扫描到的结点关键字与K相等，则查找成功；若扫描结束后，仍未找到关键字等于K的结点，则查找失败

#### 2.2 模型的评价

##### 2.2.1 本模型优点

本模型组合了哈希表与KMP算法，各取所长，和其他算法模型相比较，兼顾了内存占用和查询速度等各个方面，具有查询速度极快，内存占用量较小，支持全部k值，建立索引时间较小的优点。模型的索引时间复杂度、索引空间复杂度为O（n）查找时间复杂度、查找空间复杂度均为O(1)。并在实际检验中有很好的表现。

##### 2.2.2 本模型缺点

在用程序实现算法时，由于代码语句的不规范等不完善的地方，导致程序的实际占用内存量虽然能满足要求，但是和理论内存量比较存在偏差。在内存使用量上还有提高的空间。