**C Project Report**

袁恩铭 515111910107

1. Write a program to find differences between two files containing bioinformatics data.

Synopsis:

Biodiff [options] from-file to-file

If you have two files A (from-file) and B (to-file), you are expected to generated all lines in A-B, A & B, and B-A in terms of the criteria you set. The file format of file A and B can be different. There will be two styles for comparison: one is coordinate based (option –c )and the other is name based (option –n).

1. . Coordinate-based diff. Two or more columns from file A and B will be selected and compared to check if the two regions overlap. If two regions from the two files overlap, then these two regions will be put into to A&B\_A and A&B\_B; those regions in A but not in A&B will be put into A-B; and those in B but not in A&B will be put into B-A. Note, the comparison is based on the coordinates specified by two or more columns set by the user, but the output result contains whole lines in the original files.

For example, we have two example files A\_ucsc\_genes.txt and B\_ucsc\_gene.gtf. If you run

Biodiff –c –a 3,4 –b 3,4 A\_ucsc\_genes.txt B\_ucsc\_gene.gtf

Column 3 and 4 from A\_uscs\_genes.txt file will be selected to represent a region and column 3 and 4 from B\_ucsc\_gene.gtf file will be selected to represent a region, then they are compared. If these two regions overlap, it should generate 4 result files corresponding to A&B\_A, A&B\_B, A-B, and B-A, where A&B\_A contains those lines from file A and overlap with some entries in file B; A&B\_B contains lines from file B and overlap with entries in file A; A-B contains those lines from file A and have no overlapping entries in B; and B-A stands for those lines from file B but have no overlapping entries in A.

1. Name-based diff. Two columns from file A and B will be selected and compared in terms of string comparison. Users need to specify the column numbers in two files to be compared. For example, two example track files (A\_ucsc\_genes.txt and B\_ucsc\_gene.gtf) were downloaded from the WashU Genome Browser website (http://genomebrowser.wustl.edu ). Both Files A\_ucsc\_genes.txt and B\_ucsc\_gene.gtf contain some UCSC genes with different file formats. If you run

Biodiff –n –a 1 –b 9 A\_ucsc\_genes.txt B\_ucsc\_gene.gtf

The first column from A\_uscs\_genes.txt file and the 9th column from B\_ucsc\_gene.gtf file will be selected and compared. If their names “overlap”, it should generate 4 result files corresponding to A&B\_A, A&B\_B, A-B, and B-A, where A&B\_A contains those lines from file A and overlap with some entries in file B; A&B\_B contains lines from file B and overlap with

entries in file A; A-B contains those lines from file A and have no overlapping entries in B; and B-A stands for those lines from file B but have no overlapping entries in A. Here, we call a string s “overlaps” with another string t, if s contains t or t contains s.

**Please write your program in C and test it thoroughly. Your program is expected to deal with very large size files (the test files may be of hundred MBs). Both the accuracy and speed will be evaluated for your program. (Hint: when you compare two files, first sort the entries in each file based on the column of your pick; then compare them.)**

**In addition, please write your code as pretty as you can and put as much explanation as you can.**

You report should include at least 4 parts: 1). Design of the program; 2) implementation of the program; 3). Usage of your program and test examples together with results. 4) Conclusions and discussions. Part 2 should include the source code as the appendix.

Turning in your homework Please hand in a hard copy of your project report before the class start on June 8th, 2017.

设计思路：具体的实施过程，各个mode的设计，细节处理，模块化设计，

1. **设计思路**

要实现题目中要求的比对并不困难，但是要实现对大文件数据高效的对比，和处理各种复杂情况是很困难的，要考虑时间的优化，空间的优化，异常处理等多个方面的问题。在仔细思索了这些问题之后，初步计划通过尝试两种数据结构：线段树、后缀树，分别实现对坐标和名称的overlap查询。

但是在具体的实施过程中遇到了几个棘手的问题：首先使用线段树处理坐标的的数字是因文件而异的，不容易确定下线段树规模的大小；其次后缀树的Ukknoen算法很复杂，C语言实现比较困难；而且考虑实际问题：对生物信息学文件来说后缀树意义并不大（考虑到实际的文件的每一列都是少量字符，相同后缀并没有太大意义）。因此采用了字典树的方法处理相同字符串以及前缀字符串；动态数组实现坐标排序的方法。根据不同情况定义了四种不同的模式，下面分别说明。

1. **[-ne] mode**

定义了一种指定列字符串相等的name-based equivalent，必须指定列完全相同才为overlap，采用字典树的数据结构。字典树是一种典型的根数，定义字典树结构TrieNode，其中有两个成员变量：int类型的exist，用来标记插入字符串的终点；儿子指针数组next[128]，指向他的128个儿子（没有指向NULL），字典树就像所有普通的查询树一样，有3个基本的函数：创建节点函数Create\_tire（），插入函数insert\_tire（），查找函数Search\_trie1（）（还定义了一个允许查询树中字符串前缀的函数Search\_trie2（））。

在此种模式下，首先读取A文件，遍历每一行，并得到指定列插入到字典树中，A文件遍历完成后，A的字典树就建立完成了；接下来读取B文件，遍历每一行，并将指定列的字符串在A字典树中查询，若存在，则将整行字符输出的A&B\_B文件中，若不存在，则将整行字符输出到B-A文件中，然后用B的指定列建立字典树。这样B文件输出结束后，我们就得到了两个文件A&B\_B、B-A，并且建立好了B的字典树。接下来就是释放A树的空间，并像上述一样查询A，输出A。

这种方法读取文件两次，但是速度非常快，因为生物信息学的数据每一列的字符串长度都很短，也就是建立的字典树的层数很小，占用的内存很小，当数据量庞大的时候，更能体现出优越性。

1. **[-no] mode**

定义了一种指定字符串prefix overlap的方法，最终得到的文件中A&B\_A中是A中指定列是B前缀（或相等）的情况（比如A的列abc，B的列abcde，则A的这一行被输出到A&B\_A中），A-B中是不满足这种这种情况的行。同理，A&B\_B中是B中指定列是A的前缀（或相等）的情况，B-A中是不满足这种情况的行。

此模式可以继续使用字典树的框架，但是在查询过程中不要求是完全相等的，查找树的函数中，只要遍历完被查找字符串，就返回true，而不考虑树中节点的exist值。因此重写了一个Search\_tire2（）函数，查询前缀。

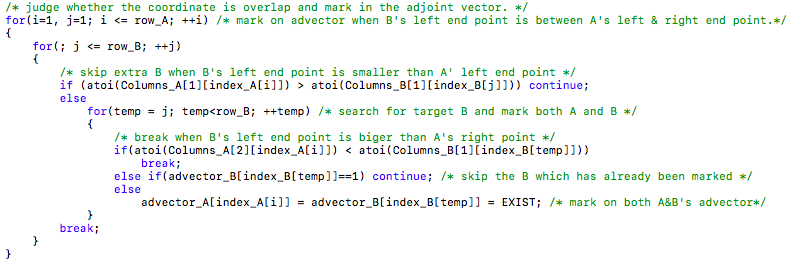
1. **[-co] mode**

定义了一种坐标overlap的方法。主要的思想是读取A、B文件，并将其指定列存储在指针数组中，然后将坐标按左端点排序，然后以一种优化遍历的方式进行比较，所有的overlap可以分为两种情况：A的左端点在B的左右端点之间；B的左端点在A的左右端点之间。

如果把左端点，右端点，行都存储下来的话，考虑到左端点，右端点，行号都要同时排序，非常复杂，而且容易出现segmentation fault，所以在此根据行数建立了一个索引index，并建立了一个作为标记的伴随向量advector（用来标记指定行号是不是overlap），这样就能把左端点、右端点、行号、advector，通过索引index的下标联系起来。而且考虑实际把整行都存储起来是不可取的（因为文件太大时会导致系统空间不足），因此建立索引是有必要的。

主要的过程是：首先读取A、B文件的行数（处理空行的情况），便于根据函数动态申请空间。然后根据行数初始化索引index 与伴随向量 advector。然后申请动态空间将A、B文件的指定两列存储到一个三维指针中（字符指针数组的指针）。然后通过内置函数qsort根据左端点（其次是右端点）实现对index的排序。接下来是判断是否overlap。首先没有采用具体的数据结构所以遍历是必须的，其次所要比较的是一段区间，是两个值，因此二分查找之类快速查找方法并不实际（根据左端点二分查找，对于右端点还是要遍历，而且对于左端点也并不是查找相等，而是查找大于或小于，二分并不实际）。合理优化遍历的方法也能起到很好地作用。

以找出B的左端点在A的左右端点之间为例：

首先判断过程是一个嵌套循环，最外层从小到大遍历A的左端点，内一层遍历B的左端点，是从上次处理到的B向后遍历，如果B的左端点比A的左端点小，continue继续循环，如果B的左端点大于或等于A的左端点的话，记录下此时的B的索引，并继续向后遍历，只要B的左端点小于或等于A的右端点，就在此时的A和B的伴随向量advector上做标记，直到B的左端点大于A的右端点break。同理找出A的左端点在B的左右端点之间的情况，这样就完成了overlap的查询。然后就可以根据伴随向量里的标记，通过索引将行输出到目标文件。

此方法虽然是采用了遍历的方法，但是在实际数据中是效率很高的，因为即使是很大的数据文件，两个坐标之间的差值很小，因此第二层循环实际进行的次数很少，而且第二层的遍历是从上次break的地方继续的，因此也会减少很多的循环，而且判断已经被标记过的直接跳过，也会减少很多循环。因此在处理坐标分布比较均匀，而且左右端点只差很小（序列很短）的情况下比较高效。但是也有一点不足，就是在处理左右端点差大的时候，内层循环的次数会比较多，导致低效。

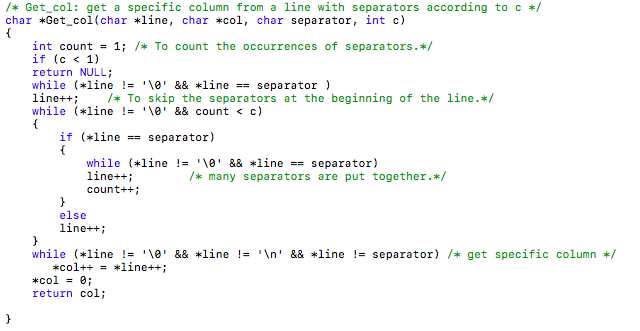
1. **[-ce] mode**

定义了一种坐标全等的方法。主要的思想还是利用字典树比较全等。因为考虑实际数据，坐标的数字位数一般是是相同的，因此完全可以将左右端点坐标的字符串连接起来，利用字典树实现查询比较。

基于这种思想，根据[-ne]的方法很容易实现过程，只是每次插入查询字符串，都要把得到的两列通过strcat连接起来。

1. **模块化设计**

在解决一个实际的大项目的时候，模块化设计的思想是很重要的，在本次大作业中，也使用了模块化设计的思想，对于程序中使用次数较多的功能，可以写一个通用的函数，并提高其健壮性来解决多种实际问题。

project中有很多模块化设计的例子，在处理命令行参数时、在文件行中获得特定列时、在读取文件特定列存储到指针中时，我们频繁的使用同一个功能，就是从一个指定的字符串中，根据分隔符和列数获得指定的列。程序中就设计了一个Get\_col（）函数。它有4个参数：要处理的行，列的储存指针，分隔符，列数，返回一个指向列的储存单元的指针。为提高其健壮性，其参数中有列指针，返回值也是列指针，既能是操作函数，又可以做右值给字符串初始化。类似的还有获得行数的函数get\_row（），树操作函数等等。

模块化的设计增强了程序的易读性，和可移植性；更是一种重要的编程思想。

1. **细节处理**

为增强程序的健壮性，程序设计与调试的过程中考虑了很多具体的实现细节。

1. 写了一个Info（）函数，根据用户的输入异常输出不同的报错信息，增强用户体验。
2. 在主函数中设置输入输出流缓冲区，加快文件读取写入速度。
3. 在读取文件时跳过文件中的空行，以免读取列时出现段错误。
4. 获得指定列数的列时充分考虑一个分隔有多个’\t’的情况。
5. fgets获得的行最后有换行符，比较最后一列时，去掉换行符。
6. 在建立好第一个文件的字典树用第二个文件查询是，同时用第二个文件的列建立第二个文件的字典树，减少I/O操作。
7. **思考讨论**

本程序大体实现了作业设计的要求，但我认为还存在的问题有以下几个：

1. 首先对于实际的生物信息学文件，都存在一个head介绍文件的基本信息，可以更进一步的实现关于head的操作；例如命令行参数加一个option –ah n 去掉A文件的前n行head。
2. 其次对于实际文件中可能会有注释行，以‘#’或‘@’等特殊字符开通，程序可以忽略以这些特殊字符开头的行。
3. 程序未能完整的实现名字的overlap，基于Ukknoen后缀树算法的C语言实现比较复杂，但是值得一试。
4. 基于坐标的overlap中涉及的文件I/O太多，而且比较算法在最坏的情况下时间复杂度很大，可以考虑基于线段树或其他数据结构的算法实现。
5. **心得体会**

我觉得这次大作业是个艰巨的任务。首先程序的构思花费了很久，但是起初的构思在实现过程中都遇到了困难，只能退而求其次，采用容易实现的方法。程序设计过程中充分体会到了C语言自顶向下基于过程的模块化设计的思想；为实现这个project的要求，首先把他差分成几个函数，我有一个觉得比较好的方法，当你程序构思好了，不着急写程序，先把你写的具体的函数的函数原型写出来，写函数原型的时候就能充分考虑到需要的参数、变量等等一系列问题，就能在心目中勾画出大体的蓝图。

还有很痛苦的一点就是不会使用debug工具，所以在写大作业的时候debug用了非常久的时间，只能靠printf；exit来debug非常低效，而且随着代码行数的增加，很多时候也是无从下手。

这次C语言大作业自己花费了至少40个小时，但是自己的水平有了很大的提高。相信掌握了C语言对今后的学习会有很大的帮助。