`naiveMatcher`, v. 0.2: Naïve Exact Matching

Bernhard Haubold
Max-Planck-Institute for Evolutionary Biology, Plön, Germany

December 6, 2018

1 Introduction

The exact matching problem is to find all occurrences of a pattern P in a text T. The simplest algorithm for solving this problem consists of two nested loops, one iterating over the text, the other over the pattern. Initially the first positions of P and T are compared. Then the second, and so on. Whenever P has been found, or one of its characters mismatches T, P is shifted by one position and the comparisons started again at the first character of P.

The program `naiveMatcher` demonstrates the strengths and weaknesses of this simple strategy. The strengths are simplicity of implementation and speed in many practical situations. The main weakness shows up when long patterns are used that match at many places. Then the run time becomes the product of pattern and text length, $O(|P| \times |T|)$.

2 Getting Started

`naiveMatcher` was written in C on a computer running Linux and should work on any standard UNIX system. However, please contact me at `haubold@evolbio.mpg.de` if you have any problems with the program.

- Unpack the program

  ```
  tar -xvzf naiveMatcher_XXX.tgz
  ```

  where `XXX` indicates the version.
- Change into the newly created directory

  ```
  cd NaiveMatcher_XXX
  ```

  and list its contents

  ```
  ls
  ```
- Generate `naiveMatcher`

  ```
  make
  ```
- List its options

  ```
  ./naiveMatcher -h
  ```
- Test it

  ```
  ./naiveMatcher -p AAC ../Data/testSeq.fasta
  ```

3 Listing

The following listing documents the driver program for `naiveMatcher`.

```
/***** naiveMatcher.c *****************************
 * Description:
 * Author: Bernhard Haubold, haubold@evolbio.mpg.de
 * Date: Thu Jun  8 23:42:53 2017
 **************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include "interface.h"
#include "eprintf.h"
#include "sequenceData.h"

void naive(char *p, char *t);

void scanFile(int fd, Args *args){
  Sequence *seq, *pat;
  int fdp;

  seq = readFasta(fd);
  if(args->P){
    fdp = open(args->P, 0);
    if(fdp < 0){
      printf("ERROR:␣Could␣not␣open␣file␣%s\n", args->P);
      exit(-1);
    }
    pat = readFasta(fdp);
    if(pat == NULL){
      printf("ERROR:␣Could␣not␣read␣sequence␣from␣file␣%s\n", args->P);
      exit(-1);
    }
    pat->seq[pat->len-1] = ’\0’; /* cut off sentinel character */
    args->p = pat->seq;
    close(fdp);
  }
  naive(args->p,seq->seq);
  freeSequence(seq);
}

int main(int argc, char *argv[]){
  int i;
  char *version;
  Args *args;    /* command line arguments */
  int fd;        /* file descriptor */

  version = "0.2";
  setprogname2("naiveMatcher");
  args = getArgs(argc, argv);
  if(args->v)
    printSplash(version);
  if(args->h || args->e)
```

```
    printUsage(version);
  if(args->numInputFiles == 0){
    fd = 0;
    scanFile(fd, args);
  }else{
    for(i=0;i<args->numInputFiles;i++){
      fd = open(args->inputFiles[i],0);
      scanFile(fd, args);
      close(fd);
    }
  }
  free(args);
  free(progname());
  return 0;
}
```

4 Change Log

- Version 0.1 (June 9, 2017)
  - First running version.
- Version 0.2 (December 6, 2018)
  - Fixed two bugs in interface.