Genome 540
Discussion Section Week 1

Jan 4, 2021

Chengxiang (CX) Qiu

Agenda (Tu/Th)

- Introductions
- C++/general programming tips
- HW1 context/advice/questions
- Input on future topics

Introductions

- Who am I?
  - 4th year Genome Sciences student
  - Shendure Lab
  - Single-cell genomics
  - Mouse embryogenesis
  - Python/R
- Who are you?
  - Department
  - Programming experience

Programming style

- The more readable your code is
  - The more I'll be able to help you if something's wrong
  - The more useful it will be to you later
  - Try to use Github (version control and easy sharing)

Programming style

- Tips for readability
  - Intuitive variable/function names, (a, b, c vs. seq_name, seq_length)
  - Comments
    - Outlining general structure of program/key points of implemented algorithm
    - Clarifying any tricky/unintuitive lines of code
  - Simplicity over performance optimization (until it becomes necessary)
  - Write more functions
  - Write less “for” loops

General tips for homework

- C/C++ or Java for some homework are necessary
- Python works for most homework
- R and MATLAB are not suitable

General tips for homework

- It is important to understand biological questions
- “Gap” between understanding the algorithm and writing it out
- Your program might work on the test data but not work on the real data (much bigger and some unexpected issues)

General tips for homework

- Print intermediate output
- It's not necessary to write every basic function (for example, sort)
- Please make an effort to match the template!

For a given DNA sequence

ACCTGCACTAAACCGTACACTGGGTTCAAGAGATTTCCC

The longest match substring?

Substring?

Longest match substring?

In assignment 1, you need to compare two DNA sequences

TTTTTATTATTAACCATGATAATTGTTGATAAATCTGTGGATAACTCTAAAAAAATTCCGGATTTATAAA

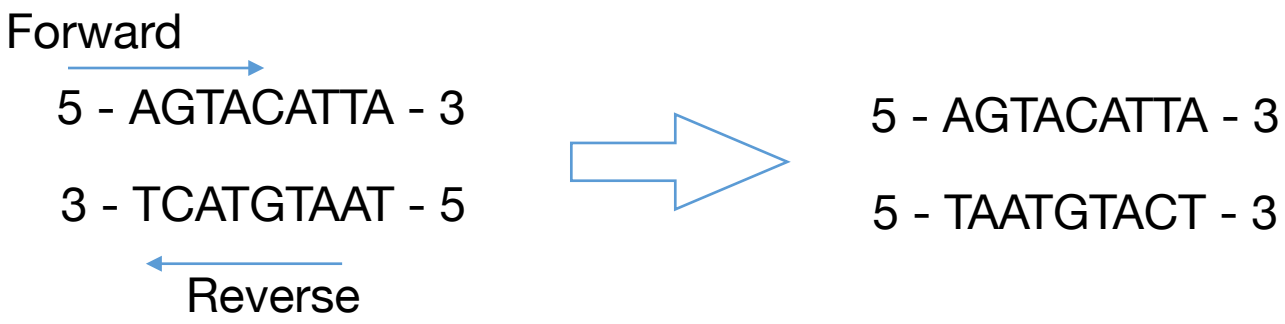
and

AGTACATTAAAATATTTATATTTTAATGTAAATATTATATCACTTTTTCACAAAAACGTGTATTATATAT

- What’s the longest match substring between them?
- Sequence 1 + Sequence 2 = A new Sequence

- Only comparing “substring” from different sequences

- Remember to consider both strands for sequence 2



- Do not store the substrings! (the real data is 10 Mb)

Difference between Python and C/C++

Python

- “interpreted”
- dynamic-typed
- data type is not required while declaring variable

```
A = 5
B = “x”
C = [‘x’, ‘y’, ‘z’]
```

C/C++

- “compiled”
- statically typed
- data type is required while declaring variable
- *debugging is challenging*

```
int A = 5;
char B = ‘x’;
char C[3] = {‘x’, ‘y’, ‘z’};
```

Dynamic arrays in C++ (Vectors)

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector> // must have this in order to use vector
#include <algorithm>
using namespace std;

int main(int argc, char* argv[] ){
	vector<string> vec; // make an empty vector that will be made up of strings
	vec.push_back("ZABC"); // adds “ZABC” to the vector, can be accesses with strings[0]
	vec.push_back("DEF"); // adds ”DEF”
	for(auto x : vec){
		cout << x << endl;
	}
	vec.clear(); // empty the vec of all elements
	// documentation http://www.cplusplus.com/reference/vector/vector/
}
```

Custom data types in C++ (Structs)

```
struct complex{
        int real;
        int img;
};

int main(int argc, char* argv[] ){
        complex a = { 10 , 1} ;
        cout << “Real: ” << a.real << “    Img: ” << a.img << endl;

        complex * p = &a;
        cout << “Real: ” << p->real << “    Img: ” << p->img << endl;
}
```

Pointers in C++

- Pointers are memory addresses, which point to variables

- There are two operators essential for handling pointers and memory addresses in C/C++: `*` and `&`

* as a suffix to a type is a “pointer”

```
char* p;  // p is a memory location that stores a “char”
```

& as a unary prefix is the address-of operator and obtains the memory address of a variable

```
char x = ‘h’;

char* p = &x;
```

& as a suffix to a type means “pass by reference”

Example

```
void swapNums(int &x, int &y) {
  int z = x;
  x = y;
  y = z;
}

int main() {
  int firstNum = 10;
  int secondNum = 20;

  cout << "Before swap: " << "\n";
  cout << firstNum << secondNum << "\n";

  // Call the function, which will change the values of firstNum and secondNum
  swapNums(firstNum, secondNum);

  cout << "After swap: " << "\n";
  cout << firstNum << secondNum << "\n";

  return 0;
}
```

* as a unary prefix means content of that memory location

Example

```
string food = "Pizza";  // A food variable of type string
string* ptr = &food;    // A pointer variable, with the name ptr, that stores the address of food

// Output the value of food (Pizza)
cout << food << "\n";

// Output the memory address of food (0x6dfed4)
cout << &food << "\n";

// Output the memory address of food with the pointer (0x6dfed4)
cout << ptr << "\n";
```

Difference between pointer to an array and array of pointers

- Pointer to an array - we are using the pointer to access the components of the array

```
// pointer to an array of five numbers;
int (*ptr)[5];
```

- Array of pointers is an array of the pointer variables

```
// declaration of an array of pointers;
int *ptr[5];
```

Arrays point to blocks of memory

- Arrays are just pointers to continuous blocks of memory

```
char word[6];  // an array of 6
characters called word
word[0] = 'a';
```

- Array indices are just pointer arithmetic and dereferencing combined
  - `a[12]` is the same as `*(a + 12)`
  - `&a[3]` is the same as `a + 3`

```
const char *word =
"hi";
```

word[0]	word[1]	word[2]
'h'	'i'	'\0'
0x80	0x88	0x90
word	word+1	word+2

- Large arrays should be dynamically allocated (on the heap)
  - Make sure you delete them

```
int n = some_large_number;
double * d = new double[n];
```

Sorting in C++ with sort

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <algorithm> // must have this in order to use sort
using namespace std;

bool my_cmp(const string & a, const string & b ){
	return( a < b );
}
int main(int argc, char* argv[] ){
	vector<string> vec;
	vec.push_back("ZABC");
	vec.push_back("DEF");
	sort(vec.begin(), vec.end(), my_cmp );
	for(auto x : vec){
		cout << x << endl;
	}
}
```

Other sorting options in C++

function

qsort

<cstdlib>

```
void qsort (void* base, size_t num, size_t size,
            int (*compar)(const void*,const void*));
```

Sort elements of array

Sorts the *num* elements of the array pointed to by *base*, each element *size* bytes long, using the *compar* function to determine the order.

The sorting algorithm used by this function compares pairs of elements by calling the specified *compar* function with pointers to them as argument.

The function does not return any value, but modifies the content of the array pointed to by *base* reordering its elements as defined by *compar*.

The order of equivalent elements is undefined.

Other sorting options in C++

function

qsort

<cstdlib>

```
void qsort (void* base, size_t num, size_t size,
            int (*compar)(const void*,const void*));
```

Sort elements of array

Sorts the *num* elements of the array pointed to by *base*, each element *size* bytes long, using the *compar* function to determine the order.

The sorting algorithm used by this function compares pairs of elements by ca
pointers to them as argument.

The function does not return any value, but modifies the content of the array
elements as defined by *compar*.

The order of equivalent elements is undefined.

http://www.cplusplus.com/reference/

```
/* qsort example */
#include <stdio.h>      /* printf */
#include <stdlib.h>     /* qsort */

int values[] = { 40, 10, 100, 90, 20, 25 };

int compare (const void * a, const void * b)
{
  return ( *(int*)a - *(int*)b );
}

int main ()
{
  int n;
  qsort (values, 6, sizeof(int), compare);
  for (n=0; n<6; n++)
     printf ("%d ",values[n]);
  return 0;
}
```

Homework 1

- Program a suffix array
- Test it on the files provided and compare results to the test output
- Run your program on the orthologous 10-megabase regions in the human and mouse genomes.
- Use the UCSC genome browser (hg38 and mm10) to figure out what biological feature they (the longest match) correspond to
- Submit your program output and your code

UCSC Genome Browser on Human Dec. 2013 (GRCh38/hg38) Assembly

move <<< << < > >> >>> zoom in 1.5x 3x 10x base zoom out 1.5x 3x 10x 100x

multi-region chr17:3,000,000-13,000,000 10,000,001 bp. gene, chromosome range, or other position, see examples go examples

chr17 (p13.3-p12) 13.3 13.2 p13.1 17p12 17p11.2 17q11.2 17q12 17q22 24.3 25.1 17q25.3

Scale 5 Mb hg38

chr17: 4,000,000 5,000,000 6,000,000 7,000,000 8,000,000 9,000,000 10,000,000 11,000,000 12,000,000

Reference Assembly Fix Patch Sequence Alignments

chr17_ML143374v1_fix

chr17_KV575245v1_fix

Reference Assembly Alternate Haplotype Sequence Alignments

GENCODE V38 (1058 items filtered out)

RAP1GAP2 P2RX1 VMO1 WSCD1 MIR195 NAA38 MFSD6L USP43 MYH1 DNAH9 MYOCD

RAP1GAP2 ZZEF1 PFN1 WSCD1 MIR497 NAA38 PIK3R6 GSG1L2 MYH3 DNAH9 ELAC

RAP1GAP2 CYB5D2 PFN1 WSCD1 BCL6B NAA38 PIK3R5 GLP2R SCO1 DNAH9 ELAC

RAP1GAP2 CYB5D2 ENO3 WSCD1 ASGR2 CHD3 PIK3R5 GLP2R SCO1 DNAH9 ELAC

RAP1GAP2 CYB5D2 ENO3 WSCD1 ASGR2 CHD3 PIK3R5 RCVRN PIRT ZNF18

OR1D5 ANKFY1 ZFP3 AIPL1 DLG4 HES7 NTN1 MYH13 RP11-628O18.1

OR1D2 ANKFY1 USP6 AIPL1 DLG4 HES7 STX8 MYH8 ZNF18

OR1D2 ANKFY1 USP6 AIPL1 DLG4 PER1 STX8 MYH4 ZNF18

OR1G1 Y_RNA ENO3 RNU6-1264P ASGR2 CHD3 PIK3R5 GAS7 SHISA6 MYOCD

CTD-2309O5.3 UBE2G1 SCIMP AIPL1 DLG4 PER1 AC087501.1 MYH13 RP11-1096G20.5

OR1P1 UBE2G1 SCIMP AIPL1 DLG4 PER1 CFAP52 MYH2 ZNF18

CTD-2309O5.3 RN7SL774P SCIMP AIPL1 DLG4 VAMP2 CFAP52 MYH2 MAP2K4

OR1A2 SPNS3 NUP88 AIPL1 DLG4 MIR6883 CFAP52 MYH2 MAP2K4

OR1A1 AC118754.4 RPAIN AIPL1 DLG4 VAMP2 RP11-55L4.2 MYH2 RNU11-2P

OR1A1 SPNS2 RPAIN PIMREG DLG4 VAMP2 USP43 ADPRM MIR744

OR3A2 MYBBP1A RPAIN PIMREG DLG4 VAMP2 RP11-55L4.1 ADPRM RP11-471L13.2

OR3A2 MYBBP1A RPAIN PIMREG DVL2 MIR4521 RP11-477N12.6 RN7SL601P LINC00670

OR3A1 GGT6 RPAIN PIMREG DVL2 AURKB DHRS7C SHISA6 ARHGAP44

OR3A1 GGT6 RPAIN PIMREG ELP5 AURKB DHRS7C SHISA6 ARHGAP44

OR1R1P GGT6 C1QBP PIMREG ELP5 AURKB RP11-477N12.4 RP11-680C21.1 LINC00670

OR1E1 GGT6 C1QBP PITPNM3 ELP5 AURKB RP11-477N12.5 RP11-642C21.1

OR3A3 SMTNL2 DHX33 KIAA0753 TNK1 CTC1 GAS7 LINC00670

OR3A3 SMTNL2 DHX33 KIAA0753 TNK1 PEAS GAS7 RP11-187D20.1

HW1 tips

- Start with pseudocode
- Start writing and then improve the details
- Don’t hesitate, C++ is much faster than Python (at least for HW1)
- Get comfortable with pointers
- Output to template directly, also please make an effort to match the template

```
#include <cstdio>
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>

using namespace std;

void read_file(string filename, string& contents, int& num_lines)
{
    ifstream f;
    f.open(filename.c_str());
    string line;

    contents = "";
    num_lines = 0;
    while(getline(f, line)) {
      contents.append(line.substr(0, line.length()));
      num_lines++;
    }

    f.close();
}

int main(int argc, const char* argv[])
{
    string fn = argv[1];
    string contents;
    int num_lines;

    read_file(fn, contents, num_lines);

    cout << "Read: " << fn << "\n";;
    cout << "  * " << num_lines << " lines\n";
    cout << "  * " << contents.length() << " characters (excluding newlines)\n";

    char * contents_cstring = (char*)contents.c_str();
    for (int i = 0; i < contents.length(); i++) {
      assert(contents_cstring[i] == *(contents_cstring + i));
      assert(contents_cstring[i] == contents.at(i));
    }
    assert(contents_cstring[contents.length()] == '\0');
}
```

```cpp
#include <cstdio>
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>

using namespace std;

void read_file(string filename, string& contents, int& num_lines)
{
    ifstream f;
    f.open(filename.c_str());
    string line;

    contents = "";
    num_lines = 0;
    while(getline(f, line)) {
      contents.append(line.substr(0, line.length()));
      num_lines++;
    }

    f.close();
}

int main(int argc, const char* argv[])
{
    string fn = argv[1];
    string contents;
    int num_lines;

    read_file(fn, contents, num_lines);

    cout << "Read: " << fn << "\n";;
    cout << "  * " << num_lines << " lines\n";
    cout << "  * " << contents.length() << " characters (excluding newlines)\n";

    char * contents_cstring = (char*)contents.c_str();
    for (int i = 0; i < contents.length(); i++) {
      assert(contents_cstring[i] == *(contents_cstring + i));
      assert(contents_cstring[i] == contents.at(i));
    }
    assert(contents_cstring[contents.length()] == '\0');
}
```

```
Annes-MacBook-Pro-2:sandbox anne$ g++ example.cpp -o example
Annes-MacBook-Pro-2:sandbox anne$ ./example example.cpp
Read: example.cpp
  * 45 lines
  * 971 characters (excluding newlines)
```

If you are using Mac
(under macOS12.1, my experience):

potentially you need to install *Xcode*,
and use clang++ instead of g++

Topics for future discussion sections?

- Scalable and reproducible bioinformatics pipelines (Snakemake)
- General programming tips
- Specific languages: Python, C++, Unix tools
- Additional applications of HMMs
- Dynamic programming
- Machine learning
- Version Control/Github
- Jupyter Notebooks/Reproducibility