2017-6-10

1. Examination time: 14:00-16:00.
2. Submission time: Please start to submit your source code by Tsinghua Web Learning (网络学堂) at least before 16:00. The submission site will be closed at 16:10. You may also submit your source code by USB drive to our TA if you cannot access internet. Please NOTE that late submissions are NOT accepted.
3. Submission requirement: Please follow the rules for our homework submission.

1. Code reuse (Credits 40%)

From Wikipedia, the definition of a trie is as follows: "In computer science, a trie, also called digital tree and sometimes radix tree or prefix (前缀) tree (as they can be searched by prefixes), is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings (以字符串为查询关键字)."

Read the given source code of a basic trie implementation and try to understand the code. Then write a subclass named MyTrie of the base class BaseTrie by public inheritance. The goal is to reuse the searching functions in class BaseTrie to search for all the strings stored in the trie structure matching a given prefix (在 trie 结构中搜索匹配某个前缀的所有字符串).

Requirements:

(1) Write MyTrie.h and MyTrie.cpp for subclass (子类) MyTrie (check main.cpp for the required interfaces); (10%)

(2) Write the destructors to avoid memory leaks (you may modify class BaseTrie if needed); (10%)

(3) Add new code into main.cpp to test class MyTrie:

(a) Randomly generate 30 character strings of 10 alphabets (a~z) in length (随机生成 30 个字符串，每个字符串长度为 10 个字符，且由字母 a 至 z 组成); (5%)

(b) Insert the 30 character strings into the object of MyTrie, with each string corresponding to a different integer key value (每个字符串对应不同整数键值); (5%)

(c) Search all the character strings starting with a given prefix string (e.g. "a"), and print the strings to visually check (通过打印信息检查结果的正确性) that results are correct. (10%)

2. Longest common subsequence (LCS) (Credits 60%)

From Wikipedia[1], the *longest common subsequence* (LCS) problem is the problem of finding the longest subsequence common to all sequences in a set of sequences (often just two sequences). For example, let X be "XMJYAUZ" and Y be "MZJAWXU", and then the longest common subsequence between X and Y is "MJAU". The longest common subsequence problem and its variants have many applications including diff, git, sequence alignment, etc. Specially, LCS is also adopted for fast evaluation of sequence-pair-based block placement (related to our individual project), which reduces the runtime from $O(n^2)$ to $O(n \log \log n)$[2].

The classic algorithm for solving the LCS problem is *dynamic programming*, the intrinsic idea of which is for reusing results obtained from previous steps (to some extent similar to the idea of OOP code reuse). You may easily find an implementation of LCS algorithm through Internet[3]. By the implementation in [3] (the source code is also attached in lcs.cpp), please finish the following requirements.

[1] https://en.wikipedia.org/wiki/Longest_common_subsequence_problem

[2] Xiaoping Tang, Ruiqi Tian, and D. F. Wong, "Fast Evaluation of Sequence Pair in Block Placement by Longest Common Subsequence Computation", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no. 12, 2001, pp. 1406-1413.

[3] http://www.geeksforgeeks.org/printing-longest-common-subsequence/

Requirements:

(1) Make a multiple-file project including *lcs.h*, *lcs.cpp*, *main.cpp*, and *makefile*. In *lcs.h*, define a class named ***LCS*** with public member function ***lcs***. In *lcs.cpp*, define the function body of ***LCS::lcs***. In *main.cpp*, include the *main()* function from [3]. Compile and run the multiple-file project with correct output (i.e., same as the output from original source code); (10%)

(2) Based on class ***LCS***, make a template class ***LCST***, which can be used for computing longest common subsequence for arrays of different data types. Please use your ***LCST*** to compute the longest common subsequence of the following two arrays, and print the correct result; (10%)

```
int XI[] = {10,20,30,40,50,60,70,80,90,100};
int YI[] = {20,40,60,80,101};
```

(3) In sequence alignment (序列比对) applications in bioinformatics, where there are only 4 characters (i.e., '*A*', '*T*', '*G*', '*C*') in the sequence, sometimes '*A*' is regarded as equal to '*T*', and '*G*' is regarded as equal to '*C*'. Please use function objects to solve this problem: define a new template class ***LCSTS***, where the member function ***lcs*** accepts an addition function object (***pred***) as parameter. You need to define two function objects, i.e., ***pred1*** and ***pred2***. If function object ***pred1*** is used, then '*A*'('*G*') is regarded as equal to '*T*'('*C*'). If function object ***pred2*** is used, then '*A*'('*G*') is regarded as unequal to '*T*'('*C*'). Please use your ***LCSTS***, along with the two function objects, to compute the longest common subsequences of the following two strings, and print the results. (20%)

```
char X[] = "AGGTACGGC";
char Y[] = "GGCCTGCG";
```

(4) Test the performance of ***LCSTS*** with ***pred1***, i.e., '*A*'('*G*') is regarded as equal to '*T*'('*C*').

(a) Randomly generate two sequences with 1000 characters, where each character belongs to {'A','T','G','C'} (随机生成两个序列，每个序列 1000 个字母，每个字母只允许在{'A','T','G','C'}内取值); (5%)

(b) Implement (实现) a straightforward enumeration method (***LCSTS2***), and along with ***pred1***, compute the longest common subsequences of the two randomly generated sequences. Recursive functions (递归函数) can be used if needed. (10%)

(c) Use QueryPerformanceCounter interface to compute runtime, and then compute and print the speedup of (***LCSTS2***) over ***LCSTS*** as "Runtime_ ***LCSTS***/Runtime_ ***LCSTS2***"; (5%)