Claudiordgz

Solutions of Cracking the Coding Interview

Book Authors: Gayle Laakmann McDowel

Contents

1 Arrays and Strings . . . **2**
1.0.1 Exercises . . . 2
E-1.1 . . . 2
E-1.2 . . . 4
E-1.3 . . . 5
E-1.4 . . . 7
E-1.5 . . . 10
E-1.6 . . . 12
E-1.7 . . . 14
E-1.8 . . . 16

Arrays and Strings

The first chapter of exercises is not really the first the chapter in the book, it is more of a tiny subsection after telling you how it is in the professional world. So after some very interesting stuff you are handled some good exercises. So without any further ado... here they are.

1.0.1 Exercises

Go to Top

E-1.1

Implement an algorithm to determine if a string has all unique characters. What if you cannot use additional data structures?

Exercise E-1.1

```
#ifndef __GAYLE_CH01_EX11_HPP__
#define __GAYLE_CH01_EX11_HPP__

#include <string>

/*! @brief Implement an algorithm to determine if a string
 *  has all unique characters. What if you can not use
 *  additional data structures */
bool AllUnique(std::string const &rhs);

#endif
```

Exercise E-1.1

```
#include "ch01/ex1_1.hpp"



bool AllUnique(std::string const &rhs) {
  if(rhs.size() > 26) {
    return false;
  }
  int flag=0;
  for (std::size_t i = 0; i != rhs.size(); ++i){
    int offset = rhs.at(i) - 'a';
    offset = 1 << offset;
    if (flag & offset){
      return false;
    }
    flag = flag | offset;
  }
  return true;
}
```

Go to Top

E-1.2

Implement a function `void reverse(char* str)` in C or C++ which reverses a null-terminated string.

Exercise E-1.2

```
#ifndef __GAYLE_CH01_EX12_HPP__
#define __GAYLE_CH01_EX12_HPP__

/*! @brief Write code to reverse a
 * C Style String */
void Reverse(char *str);

#endif
```

Exercise E-1.2

```
#include "ch01/ex1_2.hpp"


void Reverse(char *str) {
  if(str){
    char *end = str;
    while(*end) {
      ++end;
    }
    --end;
    char tmp;
    while(str < end) {
      tmp = *str;
      *str++ = *end;
      *end--  = tmp;
    }
  }
}
```

Go to Top

E-1.3
Given two strings, write a method to decide if one is a permutation of the other.

Exercise E-1.3

```
#ifndef __GAYLE_CH01_EX13_HPP__
#define __GAYLE_CH01_EX13_HPP__

#include <boost/tuple/tuple.hpp>
#include <boost/iterator/zip_iterator.hpp>
#include <boost/range/iterator_range.hpp>

#include <string>
#include <vector>

/*! @brief Similar to Python's Zip function
 * Allows simultaneous iteration of multiple
 * containers */
template<class... Conts>
auto zip_range(Conts&... conts) -> decltype(boost::make_iterator_range(
  boost::make_zip_iterator(boost::make_tuple(conts.begin()...)),
  boost::make_zip_iterator(boost::make_tuple(conts.end()...)))
  )
{
  return {boost::make_zip_iterator(boost::make_tuple(conts.begin()...)),
          boost::make_zip_iterator(boost::make_tuple(conts.end()...))};
}


/*! @brief Given two strings, write a method
 * to decide if one is a permutation of the other. */
bool IsPermutation(std::string const &lhs, std::string const &rhs);

/*! @brief Inserts a char into a bit container,
 * if the char already exist in that bit container
 * then use a new one. */
void PushOffsetIntoContainer(std::vector<int> &bit_container, char const &element);

#endif
```

Exercise E-1.3

```
#include "ch01/ex1_3.hpp"
```

```
bool IsPermutation(std::string const &lhs, std::string const &rhs)
{
  bool returnVal = false;
  if(lhs.size() == rhs.size()) {
    std::vector<int> flag_lhs { 0 };
    std::vector<int> flag_rhs { 0 };
    for(auto&& str : zip_range(lhs, rhs)) {
    PushOffsetIntoContainer(flag_lhs, str.get<0>());
    PushOffsetIntoContainer(flag_rhs, str.get<1>());
    }
    if(flag_rhs.size() == flag_lhs.size()) {
      for(auto&& bitContainer : zip_range(flag_lhs, flag_rhs)) {
      if (bitContainer.get<0>() == bitContainer.get<1>()) {
          returnVal = true;
        } else {
          returnVal = false;
        }
      }
    }
  }
  return (returnVal);
}

void PushOffsetIntoContainer(std::vector<int> &bit_container, char const &element){
  int offset_ = element - 'a';
  offset_ = 1 << offset_;
  for(auto it = bit_container.begin(); it != bit_container.end(); ++it) {
    if(*it & offset_) {
      if (std::distance(it, bit_container.end()) == 1) {
        int pushIt = std::distance(bit_container.begin(), it);
        bit_container.push_back(0);
        it = bit_container.begin();
        std::advance(bit_container.begin(), pushIt);
      }
    } else {
      *it = *it | offset_;
      break;
    }
  }
}
```

Go to Top

E-1.4

Write a method to replace all spaces in a string with'%20'. You may assume that the string has sufficient space at the end of the string to hold the additional characters, and that you are given the “true” length of the string. (Note: if implementing in Java, please use a character array so that you can perform this operation in place.)

EXAMPLE
Input: “Mr John Smith”
Output: “Mr%20Dohn%20Smith”

Exercise E-1.4

```
#ifndef __GAYLE_CH01_EX14_HPP__
#define __GAYLE_CH01_EX14_HPP__

#include <string>

/*! @brief Write a method to replace all spaces
 * in a string with'%20'. You may assume that
 * the string has sufficient space at the end of
 * the string to hold the additional characters,
 * and that you are given the "true" length of the
 * string. */
void encodeSpacesStringNoFindNoInsert(std::string &rhs);
void encodeSpacesStringFind(std::string &rhs);

#endif
```

Exercise E-1.4

```
 #include "ch01/ex1_4.hpp"


void encodeSpacesStringNoFindNoInsert(std::string &rhs) {
  std::string::iterator spaceFound = rhs.end();
  for (auto it = rhs.begin(); it != rhs.end(); ++it) {
    if (*it == ' ') {
      if (spaceFound == rhs.end()){
        spaceFound = it;
      }
    }
    else {
      if (spaceFound != rhs.end()) {
        auto spaceEnd = it;
        std::string middleSequence;
        for (auto innerItr = spaceFound; innerItr != spaceEnd; ++innerItr) {
          middleSequence.append("%20");
        }
        auto SequenceBegin = std::distance(rhs.begin(), spaceFound);
        auto SequenceEnd = std::distance(rhs.begin(), spaceEnd);
        std::string sequence(rhs.begin(), rhs.begin() + SequenceBegin);
        sequence.append(middleSequence);
        sequence.append(rhs.begin() + SequenceEnd, rhs.end());
        rhs = sequence;
        it = rhs.begin() + SequenceEnd;
        spaceFound = rhs.end();
      }
    }
  }
  if (spaceFound != rhs.end()){
    rhs.assign(rhs.begin(), spaceFound);
  }
}

void encodeSpacesStringFind(std::string &rhs) {
  std::string::iterator spaceFound = rhs.end();
  std::string::iterator finishSpaces;
  bool changed = false;
  for (auto it = rhs.end()-1; it != rhs.begin(); --it) {
    if (*it == ' ') {
      if (spaceFound == rhs.end()){
        spaceFound = it;
      }
    }
    else {
      finishSpaces = ++it;
      break;
    }
  }
  rhs.assign(rhs.begin(), finishSpaces);
  for (size_t pos = rhs.find(' '); pos != std::string::npos; pos = rhs.find(' ', pos))
  {
    rhs.replace(pos, 1, "%20");
  }
```

```
}
```

Go to Top

E-1.5

Implement a method to perform basic string compression using the counts of repeated characters. For example, the string aabcccccaaa would become a2blc5a3. If the "compressed" string would not become smaller than the orig- inal string, your method should return the original string.

Exercise E-1.5

```
#ifndef __GAYLE_CH01_EX15_HPP__
#define __GAYLE_CH01_EX15_HPP__

#include <string>

/*! Implement a method to perform basic string
 * compression using the counts of repeated characters.
 * For example, the string aabcccccaaa would become
 * a2blc5a3. If the "compressed" string would not
 * become smaller than the original string, your method
 * should return the original string. */
std::string compressWord(std::string const &rhs);



#endif
```

Exercise E-1.5

```
#include "ch01/ex1_5.hpp"
#include <sstream>

std::string compressWord(std::string const &rhs) {
  auto it = rhs.begin();
  auto pastEl = *it;
  std::advance(it, 1);
  std::stringstream finalAssembly;
  int count = 1;
  for (; it != rhs.end(); ++it) {
    if (pastEl == *it){
      count += 1;
    }
    else {
      finalAssembly << pastEl << count;
      count = 1;
      pastEl = *it;
    }
  }
  finalAssembly << pastEl << count;
  std::string result = finalAssembly.str();
  if (rhs.size() <= result.size()) {
    return rhs;
  }
  else {
    return result;
  }
}
```

Go to Top

E-1.6
Given an image represented by an NxN matrix, where each pixel in the image is 4 bytes, write a method to rotate the image by 90 degrees. Can you do this in place?

Exercise E-1.6

```
#ifndef __GAYLE_CH01_EX16_HPP__
#define __GAYLE_CH01_EX16_HPP__

#include <vector>

template<class _T>
class CW {
public:
  void rotate(_T &upperLCorner, _T &upperRCorner, _T &lowerLCorner, _T &lowerRCorner){
    _T temp = upperLCorner;
    upperLCorner = lowerLCorner;
    lowerLCorner = lowerRCorner;
    lowerRCorner = upperRCorner;
    upperRCorner = temp;
  }
};

template<class _T>
class CCW {
public:
  void rotate(_T &upperLCorner, _T &upperRCorner, _T &lowerLCorner, _T &lowerRCorner){
    _T temp = upperLCorner;
    upperLCorner = upperRCorner;
    upperRCorner = lowerRCorner;
    lowerRCorner = lowerLCorner;
    lowerLCorner = temp;
  }
};

/*! Given an image represented by an NxN matrix, where each pixel in the image is
 *  4 bytes, write a method to rotate the image by 90 degrees. IN PLACE
 */
template<template<typename> class DirectionPolicy, class _Type>
class Rotate {
  DirectionPolicy<_Type> direction;
public:
  Rotate(std::vector<std::vector<_Type> > &squareMatrix){
    int n = squareMatrix.size() - 1;
    int outerEnd = squareMatrix.size() / 2;
    int innerEnd = (squareMatrix.size() + 1) / 2;
    for (int i = 0; i != outerEnd; ++i) {
      for (int j = 0; j != innerEnd; ++j) {
        direction.rotate(squareMatrix[i][j], squareMatrix[j][n - i],
          squareMatrix[n - j][i], squareMatrix[n - i][n - j]);
      }
    }
  }
};

#endif
```

Exercise E-1.6

```
#include "ch01/ex1_6.hpp"
```

Go to Top

E-1.7
Write an algorithm such that if an element in an MxN matrix is 0, its entire row and column are set to 0.

Exercise E-1.7

```
#ifndef __GAYLE_CH01_EX17_HPP__
#define __GAYLE_CH01_EX17_HPP__

#include <vector>
#include <set>

/*! @brief Write an algorithm such that if an
 * element in an MxN matrix is 0, its
 * entire row and column are set to 0. */
class ClearRowsColumns {
public:
  ClearRowsColumns(std::vector<std::vector<int> > &rectangularMatrix);

private:
  void clearCells(std::vector<std::vector<int> > &rectangularMatrix,
                  std::set<int> const &mostZeros,
                  std::set<int> const &leastZeros,
                  std::set<int> const &leastNumbers, int const&N);
  std::set<int> rowZeros;
  std::set<int> columnZeros;
  std::set<int> rowNumbers;
  std::set<int> columnNumbers;
};

#endif
```

Exercise E-1.7

```
#include "ch01/ex1_7.hpp"

ClearRowsColumns::ClearRowsColumns(std::vector<std::vector<int> > &rectangularMatrix) {
  int N = rectangularMatrix[0].size();
  for (int i = 0; i != rectangularMatrix.size(); ++i) {
    for (int j = 0; j != N; ++j) {
      if (rectangularMatrix[i][j] == 0) {
        rowZeros.insert(i);
        columnZeros.insert(j);
      }
```

```
        else {
          rowNumbers.insert(i);
          columnNumbers.insert(j);
        }
      }
    }
    if (rowZeros.size() >= columnZeros.size()){
      for (auto it = rowZeros.begin(); it != rowZeros.end(); ++it) {
        for (int j = 0; j != N; ++j){
          rectangularMatrix[(*it)][j] = 0;
        }
      }
      for (auto it = columnZeros.begin(); it != columnZeros.end(); ++it){
        for (auto jt = rowNumbers.begin(); jt != rowNumbers.end(); ++jt) {
          rectangularMatrix[(*jt)][(*it)] = 0;
        }
      }
    }
    else {
      for (auto it = columnZeros.begin(); it != columnZeros.end(); ++it) {
        for (int j = 0; j != N; ++j){
          rectangularMatrix[j][(*it)] = 0;
        }
      }
      for (auto it = rowZeros.begin(); it != rowZeros.end(); ++it){
        for (auto jt = columnNumbers.begin(); jt != columnNumbers.end(); ++jt) {
          rectangularMatrix[(*it)][(*jt)] = 0;
        }
      }
    }
}
```

Go to Top

E-1.8
Assume you have a method isSubstring which checks if one word is a substring of another. Given two strings, si and s2, write code to check if s2 is a rotation of si using only one call to isSubstring (e.g.,"waterbottle"is a rotation of "erbottlewat"

Exercise E-1.8

```
#ifndef __GAYLE_CH01_EX17_HPP__
#define __GAYLE_CH01_EX17_HPP__

#include <string>

/*!Assume you have a method isSubstring which checks if one word is a
 * substring of another. Given two strings, si and s2, write code to check if s2 is
 * a rotation of s1 using only one call to isSubstring (e.g.,"waterbottle"is a rota-
 * tion of "erbottlewat"
*/
class IsRotation {
public:
  bool check(std::string const &lhs, std::string const &rhs);
private:
  bool isSubstring(std::string const &lhs, std::string const &rhs);
};

#endif
```

Exercise E-1.8

```
#include "ch01/ex1_8.hpp"

bool IsRotation::isSubstring(std::string const &lhs, std::string const &rhs) {
  return (lhs.find(rhs) != std::string::npos);
}

bool IsRotation::check(std::string const &lhs, std::string const &rhs){
  return (isSubstring(std::string(lhs+lhs), rhs));
}
```