ACM Programming Contests

Coach: Dr. Joe K.W. Chong

Courtesy: Louis Siu
Kayman Lui
Alpha Lam
Francis Fok
Dr. Isaac To

Website: http://www.cs.hku.hk/~provinci

Introduction

- The ACM collegiate programming contests are the major programming challenge for the university students.
- The ultimate World Finals competition is held around April every year.
- To earn the ticket to this glory battlefield, we need to excel in the Regional competitions.
  - To be held in late October and November.

Team competition

- Each team, consisting of 3 students, works out a number of questions using a single computer.
  - Well, this needs some coordination among the members.
- Each university can send at most 3 teams to each regional contest.
  - Each team can join at most 2 regionals.
- The exact competition format will be introduced in due course.

Our schedule ...

- We'll have a team formation test by the end of September.
- The test is individual.
  - You've to work hard on your own first :-)
- Depending on your results, we'll select about 10 to 12 students to enter the regional contest(s).
  - Special training for the competition will be held after the team formation test.

Training sessions before test

- To prepare the team formation test, you can join the training sessions this month.
  - These training sessions are optional and you can join the test directly.
- Topics to be covered:
  - STL
  - Dynamic programming
  - Graph algorithms
  - Combinatorics
  - Problems require specific math concepts

Practice make perfect

- An archive of problems can be found in the site
  - http://acm.uva.es/p
- It also has an on-line judge to test your submitted program.
  - Simply register and apply for an account.
- This archive site will be a major source of training questions.

Tips for the competition

- In the competition, you're asked to write a number of programs.
- Only program correctness counts, subject to some time limits (usually 10 sec).
  - You can submit as many times as you want to the judge.
- Teams solving more problems win.
  - Tie break on smaller number of submissions.
    - Not our concern at the moment ...

Programming languages

- In the competition, you can write the programs in C,C++, Java
- We'll focus on C++ for the sake of efficient and convenient.
- One of the goodies of C++ is the Standard Template Library (STL).
  - Many useful data structures and algorithms have been implemented.
  - Their implementations are always optimal.

Efficiency

- The implementation of STL algorithm satisfies not only the requirement, but also efficiency.
- The searching, sorting, etc., algorithm are implemented in optimal time complexity.
  - O(log n) for binary search, O(n log n) for sorting, etc.
- The set and map are implemented using red-black tree.

Getting inputs ...

- Let’s see problem 483.
  - http://acm.uva.es/p/v4/483.html
- This is a typical question that requires a program to read a number of lines, and then to process each line.
- A common framework for this in C++:

```
string line;
while (getline(cin, line)){
  // .. to process a line
}
```

For each words ...

- For each line, we need to reverse every word separated by white space.
  - Tokenizing the input stream
- If the number of white space between words need not preserved in the output, string stream is a good solution.

```
istringstream ins(line);
string word;
while (ins >> word){
  // reverse word
}
```

To reverse something ...

- The last part is to reverse a word.
- You may instantly thinking about a loop to do this.

```
for (size_t i = 0; i < words.size(); ++i)
   cout << words[words.size() - i - 1];
```

- Let’s try the STL reverse() function.

```
reverse(word.begin(), word.end());
cout << word;
```

Algorithm & Iterator

- One of the simple algorithms in STL.
  - See http://www.sgi.com/tech/stl
- This function, as well as other STL algorithms, takes two iterators as arguments.
  - Imagine that iterator is class-version pointer.
  - word.begin() is pointing at the first position;
  - word.end() points to the “pass-the-end”

Container & iterator

- In C++, you can consider that any object that stores a sequence of (homogeneous) data is a container.
  - Array, strings, vector, set, ...
- To traverse a container, you can use the subscript operator or iterator.
- Accessing a container in terms of iterator allows the generic design of many STL algorithms.

Examples

- To reverse the elements of a vector.
- To sort the characters of a string
- To find the maximum ...

```
reverse(myvec.begin(), myvec.end());
sort(word.begin(), word.end();

int a[] = { 2, 9, 5, 3, 4, 0 };
sort(a, a + sizeof(a)/sizeof(a[0]);

vector<int>::iterator p;
p = max(myvec.begin(), myvec.end());
```

About iterators ...

- Due to the operator overloading of C++, we can access the object “pointed” by an iterator using the dereference operator.
  - e.g., cout << *p << endl;
- Note that iterator is usually associated with a particular container type. Its type is closely tied with the container.

```
vector<int>::iterator p;
string::iterator q;
map<string, string>::iterator r;
```

Typedef ...

- To save some typing headache, you can use typedef to create a type alias:

```
typedef vector<int> MyCont;
MyCont::iterator q;
typedef map<string, string> Dict;
Dict::iterator r;
```

- You can also define type alias for iterator type:

```
typedef vector<int> MyCont;
typedef MyCont::iterator MyCont_it;
MyCont_it p;
```

The set container

- See problem 353.
  - http://acm.uva.es/p/v3/353.html
- In this question, you're asked to find, for a given string, the number of unique substrings that is palindrome. e.g.,
  - "boy" has 3 palindrome substring
  - "adam" has 5 ...
  - "tot" has 3 ...

The STL way ...

- How to test if a string is a palindrome?
  - Please make good use of STL :-)
- To count the number of UNIQUE palindrome substrings,
  - Enumerate all possible substring
    - For each substring, check if it’s a palindrome.
    - If yes, insert into a set.
  - Print the size of the set.
- Correct?

map

- See problem 401.
- This question extends the palindrome problem a bit.
  - Some characters have mirrored version; e.g., ‘A’ and ‘A’, ‘E’ and ‘3’, ‘L’ and ‘J’, etc.
  - Some characters do not have mirrored image; e.g., ‘B’, ‘C’, ‘P’, ‘Q’, etc.
- Then we can defined mirrored string:
  - ‘3AIAE’, ‘2A3MEAS’

- To check if a string is a mirror,
  - For each character in the string, replace it with its mirrored version.
  - Check if the reversed of this string equals the original string.
- In doing the replacement, we need to find the corresponding mirrored version of a character.
  - This can be done by storing the correspondence in a map container.

Map usage

```
// declaration
map<char, char> mtbl;

// associate key-value pairs
mtbl['A'] = 'A';
mtbl['B'] = ' ';  // no mirror
...
mtbl['E'] = '3';
....
string s = "3AIAE";
string ms;
for (size_t i = 0; i < s.length(); ++i)
  ms += mtbl[s[i]];
```

Caveats

- The map is good for “database”-like container such as mapping a string to another string; e.g.,
  - u.no => name
  - Key => value

```
map<string, int> db;
db[‘one’] = 1;
db[‘two’] = 2;
...
```

Find before use?

- A common problem is to check if a given key is mapped to certain value.
  - If (db[key] == value){ ... } // dangerous
  - The testing work only if the key has been inserted into the map before.
- If you’re not sure, check the existence of the key first; e.g.,

```
map<string, int>::iterator p = db.find(key);
if (p != db.end()){
   // ...
}
```

Vector or array?

- Array is good for holding initial values.
  - The size must be fixed at compile time.
- Vector is more flexible.
  - New elements can be added (at the end) using push_back().
  - Use pop_back() to remove the last one.
- Sometimes we may use both :-)
  - Initialize an array and then copy into a vector or other container.

Exercises

- Starters: Problems 642, 10107, 10282.
- Main course: For Problems 10098 & 291, try the next_permutation() from STL.
- Dessert: Problem 195
  - Print all the permutation of a string in alphabetically ascending order.
  - An upper case letter goes before the corresponding lower case letter.

Comparator for sort()

- The sort() function by default compares two elements using the operator<().
- We can change the default comparison by plugging in different comparator.
  - In the simplest case, a comparator is just a boolean function.

```
bool myless(char a, char b){
  char la = tolower(a), lb = tolower(b);
  return la < lb || la == lb && a < b;
}
sort(mystr.begin(), mystr.end(), myless);
```

Summary

- Get familiar with the STL library.
  - Data structure: vector, map, set are of particular useful.
  - Algorithms: sort(), bsearch(), find(), count(), max(), min(), save many coding time.
    - Behavior of algorithms can be changed by plugging functional object (e.g., comparator in sort()).
  - Iterators are the interface between data structures and algorithms.

References

- “Effective STL” by Scott Meyers, Addison-Wesley.
- “Accelerated C++: practical programming by example” by A. Koenig, B. Moo, Addison Wesley.
- “The C++ programming language” by B. Stroustrup.