
# Chapter 7 Sort of Linear Lists

Yonghui Wu 吴永辉
School of Computer Science, Fudan University
yhwu@fudan.edu.cn
WeChat : 13817360465

- A sorting algorithm is an algorithm that puts elements of a list in a certain order.
- Sorting algorithms are important for managing data.
- There are a large number of sorting algorithms, such as Bubble sort, Insertion sort, Selection sort, Merge sort, Heapsort, Quicksort, and so on. Such algorithms have been discussed in many classical books.

# Chapter 7 Sort of Linear Lists

- 7.1 Using Sort Function in STL
- 7.2 Using Sort Algorithms

# 7.1 Using Sort Function in STL

- The Standard Template Library, or STL, is a C++ library of container classes, algorithms, and iterators. It provides many algorithms and data structures of computer science, including sorting algorithms.
- Like many class libraries, the STL includes container classes containing objects. The STL includes the class vector, list, deque, set, multiset, map, multimap, hash_set, hash_multiset, hash_map, and hash_multimap.

- Map is a Sorted Associative Container that associates objects of type *Key* with objects of type *Data*. Map is a Pair Associative Container, meaning that its value type is pair<const *Key*, *Data*>. It is also a Unique Associative Container, meaning that no two elements have the same key.
- If there is a mapping relationship between students' names and their scores, Map can describe the relationship easily. At the head of program, there is a preprocessor directive #include <map>; and Container *mapStudent* is specified as follow:
- map<string, int>mapStudent
- Based on above statements, students' names and scores are specified with string and int respectively. All students' information "*mapStudent*[*name*]=*score*" is stored in Container *mapStudent*. The compiling system sorts students' names in alphabet order.

# 7.1.1 Hardwood Species

- **Source: Waterloo Local 2002.01.26**
- **IDs for Online Judge: POJ 2418**

- Hardwoods are the botanical group of trees that have broad leaves, produce a fruit or nut, and generally go dormant in the winter.
- America's temperate climates produce forests with hundreds of hardwood species -- trees that share certain biological characteristics. Although oak, maple and cherry all are types of hardwood trees, for example, they are different species. Together, all the hardwood species represent 40 percent of the trees in the United States.

- On the other hand, softwoods, or conifers, from the Latin word meaning "cone-bearing," have needles. Widely available US softwoods include cedar, fir, hemlock, pine, redwood, spruce and cypress. In a home, the softwoods are used primarily as structural lumber such as 2x4s and 2x6s, with some limited decorative applications.
- Using satellite imaging technology, the Department of Natural Resources has compiled an inventory of every tree standing on a particular day. You are to compute the total fraction of the tree population represented by each species.

- **Input**
- Input to your program consists of a list of the species of every tree observed by the satellite; one tree per line. No species name exceeds 30 characters. There are no more than 10,000 species and no more than 1,000,000 trees.
- **Output**
- Print the name of each species represented in the population, in alphabetical order, followed by the percentage of the population it represents, to 4 decimal places.

# Analysis

Suppose the number of the tree whose species name is $x$ is $h[x]$, the total number of trees is $n$. Firstly, the number of each species and the total number of tree $n$ are calculated based on the input, then in alphabetical order print the name of each species and its percentage of the population $\frac{h[x]}{n}$.

For this problem, STL map can be used to store $h$ and sort keys.

- In STL, there is a sort function *sort*, to sort elements in an interval. At the head of program, there must be a preprocessor directive #include <algorithm>. There are two kinds of usages: sort(*l*, *r*) and sort(*l*, *r*, *compare*). For the next two examples, such two kinds of usages are used to solve problems.

# 7.1.2 Who's in the Middle

- **Source: USACO 2004 November**
- **IDs for Online Judge: POJ 2388**

- FJ is surveying his herd to find the most average cow. He wants to know how much milk this 'median' cow gives: half of the cows give as much or more than the median; half give as much or less.
- Given an odd number of cows $N$ ($1 \leq N < 10,000$) and their milk output (1..1,000,000), find the median amount of milk given such that at least half the cows give the same amount of milk or more and at least half give the same or less.

- **Input**
- * Line 1: A single integer *N*
- * Lines 2..*N*+1: Each line contains a single integer that is the milk output of one cow.
- **Output**
- * Line 1: A single integer that is the median milk output.

# Analysis

- Sort *N* cows' milk output.
- The element in the middle is the median milk output.
- Function sort( ) in algorithm.h is used to sort.

# 7.2 Using Sort Algorithms

- Some problems require you implement sort algorithms.

# exchange two variables' values

```
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

# Bubble Sort

- Bubble sort repeatedly traverse the sequence of elements to be sorted, comparing two adjacent elements in sequence, and swapping these two elements if the order is incorrect.
- The traversal is repeated until there are no adjacent elements that need to be exchanged, which means that the sequence of elements has been sorted.

For 【 Who's in the Middle 】, Sample Input, the process for bubble sort：

- 2, 4, 1, 3, 5
- 2, 1, 3, 4, 5
- 1, 2, 3, 4, 5

# Function bubble_sort

- void bubble_sort(int arr[], int len) {
- int i, j, temp;
- for (i = 0; i < len - 1; i++) //out loop: number of traversals , *len* elements , *len*-1 traversals
- for (j = 0; j < len - 1 - i; j++) // comparing number, the *i*-th traversal, *len-i* comparing
- if (arr[j] > arr[j + 1]) //comparing two adjacent elements
- swap(&arr[j], &arr[j+1]);
- }

# 7.2.1 Flip Sort

- **IDs for Online Judge： UVA 10327**

- Sorting in computer science is an important part. Almost every problem can be solved efficiently if sorted data are found. There are some excellent sorting algorithm which has already achieved the lower bound nlgn. In this problem we will also discuss about a new sorting approach. In this approach only one operation ( Flip ) is available and that is you can exchange two adjacent terms. If you think a while, you will see that it is always possible to sort a set of numbers in this way.
- A set of integers will be given. Now using the above approach we want to sort the numbers in ascending order. You have to find out the minimum number of flips required. Such as to sort "1 2 3" we need no flip operation whether to sort "2 3 1" we need at least 2 flip operations.

- **Input**
- The input will start with a positive integer $N$ ($N \leq 1000$). In next few lines there will be $N$ integers. Input will be terminated by EOF.
- **Output**
- For each data set print "Minimum exchange operations : $M$" where $M$ is the minimum flip operations required to perform sorting. Use a separate line for each case.

# Analysis

- Suppose the given set of integers is stored in an array *a*. Initially the minimum flip operations *ans* is 0. Minimum exchange operations are simulated by using bubble sort. For each exchange, *ans*++.
- When there is no exchange operation, *ans* is the solution to the problem.

# Train Swapping

- **Source: ACM North Western European Regional Contest 1994**
- **IDs for Online Judge: UVA 299**

- At an old railway station, you may still encounter one of the last remaining ``train swappers''. A train swapper is an employee of the railroad, whose sole job it is to rearrange the carriages of trains.
- Once the carriages are arranged in the optimal order, all the train driver has to do, is drop the carriages off, one by one, at the stations for which the load is meant.
- The title ``train swapper'' stems from the first person who performed this task, at a station close to a railway bridge. Instead of opening up vertically, the bridge rotated around a pillar in the center of the river. After rotating the bridge 90 degrees, boats could pass left or right.

- The first train swapper had discovered that the bridge could be operated with at most two carriages on it. By rotating the bridge 180 degrees, the carriages switched place, allowing him to rearrange the carriages (as a side effect, the carriages then faced the opposite direction, but train carriages can move either way, so who cares).
- Now that almost all train swappers have died out, the railway company would like to automate their operation. Part of the program to be developed, is a routine which decides for a given train the least number of swaps of two adjacent carriages necessary to order the train. Your assignment is to create that routine.

- **Input**
- The input contains on the first line the number of test cases (*N*). Each test case consists of two input lines. The first line of a test case contains an integer *L*, determining the length of the train ($0 \leq L \leq 50$ ). The second line of a test case contains a permutation of the numbers 1 through *L*, indicating the current order of the carriages. The carriages should be ordered such that carriage 1 comes first, then 2, etc. with carriage *L* coming last.
- **Output**
- For each test case output the sentence: 'Optimal train swapping takes S swaps.' where *S* is an integer.

# Analysis

- Bubble sort is used to solve the problem. The number of exchanges is the solution to the problem.