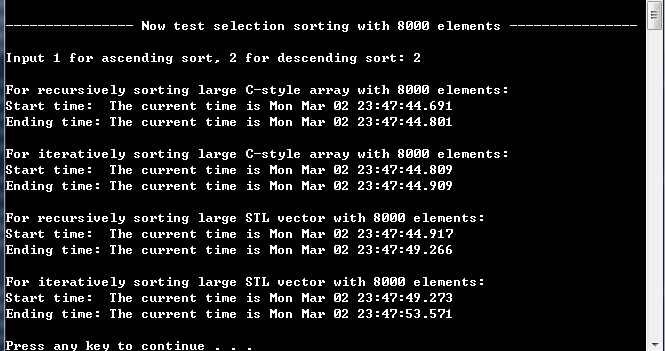
1. Issues encountered sorting large arrays and vectors with recursion. How did you fix the issue(s) encountered.

I have got a stack overflow problem when using VS2013 as my IDE. When I tried to use a data set with more than 4100 elements, the stack overflow problem occurs. While in another C++ IDE, DEV-C++, with the exactly same code, I can actually use a much bigger array up to 10000 elements. This phenomenon definitely related with the IDE itself.

With the help of Hussein, I know I can change the stack reserve size and stack commit size to a greater value, which can allocate greater size in virtual memory. Then the problem was solved.

1. The difference in performance between sorting with recursion version iteration. Which performed better, and why?

The iteration sorting performed better.



From the experiments, when the data set with 8000 elements, we can see that:

1).Recursively sort a C-style array consumes 110 ms.

2).Iteratively sort a C-style array consumes 100 ms.

3).Recursively sort a STL vector consumes 4349 ms.

4).Iteratively sort a STL vector consumes 4298 ms.

The reason is as follows: recursion function repeatedly invokes itself, even though the problem size shrinks each time, but still this will generate overhead of function calls, which will consume both extra processor time and memory space. And the iterative function does not have such a situation, so it is generally faster than the same functional recursive functions, especially for a large problem size(the recursive method also will have the potential hazard of stack overflow).

1. The difference in performance between sorting C-Style arrays versus the STL vector. Which performed better, and why?

From the experiments data, it is obvious that sorting C-style arrays has better performance. In fact, in the selection sorting algorithms, we have spent a lot of time to access the data of arrays and vectors using std::operator[]. And it turns out the accessing speed for c-style arrays is much faster than a STL vector. If we use the method .at() of vector, we can expect a more slower accessing.

Wei Liu 2015/3/3