Lab 13: Merge Sort and Quick Sort

Lecture date 6/09/2015

Due date 6/13/2015 11:59 PM

Lab 13-1: Merge Sort

In this lab, we will implement MergeSort ADT in two different ways: iterative and recursive. For recursive approach, you need to print whenever you merge sub-lists. For iterative approach, you can print the sub-list sorted at each iteration step

1. Input
Obtain a list of elements from the given input file. An input file and the expected output are shown below.

Input.txt

```
26 5 77 1 61 11 59 15 48 19
```

The corresponding output

```
input :
26 5 77 1 61 11 59 15 48 19

iterative:
5 26
1 77
11 61
15 59
19 48
1 5 26 77
11 15 59 61
19 48
1 5 11 15 26 59 61 77
1 5 11 15 19 26 48 59 61 77

recursive :
5 26
5 26 77
1 61
1 5 26 61 77
11 59
11 15 59
19 48
11 15 19 48 59
1 5 11 15 19 26 48 59 61 77
```

2. Program description

- name : p13_1.c
- input : a list of numbers in a file (an input file name is given as a command line

argument. See the example in ″1. input″ on the first page)

- output : the corresponding result in the standard output

Lab 13-2: Quick Sort

In this lab, we will implement QuickSort ADT. Our ADT should allow three options for choosing pivot value: the leftmost element, rightmost element, the element in the middle. When you implement a function for quick sort, please print the list of elements whenever you pick a new pivot value.

1. Input

Obtain a list of elements from the given input file. In addition, the option for pivot value is given: ″leftmost″ is for the pivot value at the leftmost position; ″rightmost″ is for the pivot value at the rightmost position; ″middle″ is for the pivot value at the middle. In a line of input, the first string is for the pivot value, which is followed by a list of numbers to be sorted. Each string or number is separated by a space. An input file and the expected output are shown below.

Input.txt

```
leftmost 73 21 578 109 410 53 51 1 3216 2002 15 9 24
rightmost 73 21 19 109 410 57 51 1 3216 7000 15 9 24
middle 73 21 64 109 39 53 51 1 3216 2002 15 9 24
```

The corresponding output

```
leftmost:
(24 21 9 15 1 53 51 ) (73)(3216 2002 410 109 578 )
(1 21 9 15 ) (24)(53 51 )
() (1)(21 9 15 )
(15 9 ) (21)()
(9 ) (15)()
(51 ) (53)()
(578 2002 410 109 ) (3216)()
(109 410 ) (578)(2002 )
() (109)(410 )

result
1 9 15 21 24 51 53 73 109 410 578 2002 3216

rightmost:
(9 21 19 15 1 ) (24)(51 57 3216 7000 410 109 73 )
() (1)(21 19 15 9 )
() (9)(19 15 21 )
(19 15 ) (21)()
() (15)(19 )
(51 57 ) (73)(7000 410 109 3216 )
(51 ) (57)()
(109 410 ) (3216)(7000 )
(109 ) (410)()

result
1 9 15 19 21 24 51 57 73 109 410 3216 7000

middle:
(24 21 9 15 39 1 ) (51)(53 3216 2002 109 64 73 )
(1 9 ) (15)(21 39 24 )
(1 ) (9)()
(21 24 ) (39)()
(21 ) (24)()
(53 73 64 ) (109)(2002 3216 )
(53 64 ) (73)()
(53 ) (64)()
(2002 ) (3216)()

result
1 9 15 21 24 39 51 53 64 73 109 2002 3216
```

2. QuickSort ADT

```
struct QuickSort {
        int Capacity;
        int Size;
        ElementType *Elements;
};
```

3. Program description

- name : p13_2.c
- input : a list of operation and numbers in a file (an input file name is given as a command line argument. See the example in "1. input" on the first page)
- output : the corresponding result in the standard output

Submit to the course website (https://portal.hanyang.ac.kr) your source code and a written report. Your report should include the description of your own implementation.