Programming and Data Structures Laboratory, 2018-19 Spring semester, Section 6

March 12, 2019: Tutorial and Assignment 6 (Sorting and searching, time and space complexity)

Tutorial

Write a function to sort an array of real numbers in ascending order. You can use quicksort or any other sorting algorithm.

Declare arrays of size N = 100, 500, 1000, 5000, 10000, 50000, and fill up the arrays with random numbers in the range [0, 1]. Sort each array using the function you wrote, and measure the execution time needed to sort each array. Print out each value of N and the time needed to sort the corresponding array of size N.

You can use the rand() function to generate random numbers. Use the command “man 3 rand” to know more about the rand() function. Also see http://c-faq.com/lib/randrange.html

To measure execution time, you can use the clock() function. E.g., see https://www.geeksforgeeks.org/how-to-measure-time-taken-by-a-program-in-c/

Assignment (for evaluation – write on machine and submit to Moodle before end of class)

1. [15 marks] The problem in the tutorial described above.

2. [15 marks] Write a function with the following prototype:

void sort(int A[], int size);

The function sort() takes two arguments, an integer array A, and an integer size. Assume that the number of integers in A is equal to the value of size, and they are all distinct. The function sorts the array by the following sorting algorithm.

The algorithm proceeds in iterations, starting with iteration 0. We now describe the action of the algorithm in the i-th iteration. View the array A as consisting of blocks of size 2^i, except possibly the last block, which may be smaller. Example: If A={1,3,4,7,2,5,8,9,6,10} and i=2, then the blocks are {1,3,4,7}, {2,5,8,9} and {6,10}. In the i-th iteration, each block will be sorted. Note that in the beginning when i=1, each block is of size 1 and is hence trivially sorted. In the i-th block, the algorithm merges consecutive blocks into sorted blocks of size 2^(i+1), except possibly the last block which may be smaller. In the above example, at the end of iteration 2, the blocks are {1,2,3,4,5,7,8,9} and {6,10}. The first block was obtained by merging the sorted blocks {1,3,4,7} and {2,5,8,9}. The algorithms terminates when the array is sorted. The function should print the array after each iteration.

3. [20 marks] Write a program that helps a user to maintain a dynamic set of integers. The program should start with an empty integer array of size 100, which will be used to store the set. The user should be given options to insert or delete numbers from this set. When the user wants to insert a number, the program should check whether the number already exists in the set. If yes, the program should inform the user; otherwise the element should be inserted into the array. Again, when the user wants to delete a number, the program should check whether the number exists in the set. If not, the user should be informed; otherwise the said number should be deleted from the set. Throughout the operations, the array should be maintained sorted in ascending order, and binary searching should be used to check for the presence of numbers.

An example interaction between the user (U) and the program (P):

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 1

P: Enter number to insert:

U: 43

P: 43 inserted [Note: 43 should be placed in the first element of the array]

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 1

P: Enter number to insert:

U: 21

P: 21 inserted [Note: 21 should be placed in the first element of the array, and 43 should be shifted to the second position.]

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 1

P: Enter number to insert:

U: 73

P: 73 inserted [Note: 73 should be placed in the third element of the array]

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 1

P: Enter number to insert:

U: 43

P: Number already exists

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 3

P: {21, 43, 73}

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 2

P: Enter number to delete

U: 43

P: 43 deleted [Note: 73 should be brought to the second element of the array]

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 2

P: Enter number to delete

U: 50

P: This number does not exist in the set

P: Press 1 to insert numbers, 2 to delete numbers, 3 to display the set, and 0 to exit.

U: 0

Submission instructions:

Submit one compressed file, named as **<roll number>_A6.tar.gz** or **<roll number>_A6.zip**

The compressed file should contain four source files:

<roll number>_A6_1.c, <roll number>_A6_2.c, <roll number>_A6_3.c