Algorithms Assignment 1, [17.5 Points]
Due September 20thth, 2020 @ 23:30

Your name goes here

September 4, 2020

I declare that all material in this assessment task is my work except where there is clear acknowledgement or reference to the work of others. I further declare that I have complied and agreed to the CMU Academic Integrity Policy at the University website.
http://www.coloradomesa.edu/student-services/documents
Submissions that do not include the above academic integrity statements will not be considered.
Student Name: Your name here UID: Your UID here Date: September 4, 2020

1 Task description

Your assignment should represent your own effort. However, you are not expected to work alone. It is fine to discuss the exercises and try to find solutions together, but each student shall write down and submit his/her solutions separately. Please acknowledge collaborators, so if you worked together with classmates list their names. You must be prepared to present your solution to the class. If you are not able to explain your solution, this will be considered as if you had not done your work at all.

You must write up your answers using a word processing system like LaTeX or other PDF editing tools. Experience shows that Word is in general difficult to use for this kind of task as such no ".doc(x)" will be considered only PDF (LaTeX users the algorithms package is probably your best friend here). For a programming task, your solution must contain:

1. An explanation of your solution to the problem
2. The C++ code, in a form that compiles with g++ or clang command and runs error and warning free

3. Instructions how to run it

4. For all programming tasks, all external, STL, custom libraries (vector, list, sort, algorithm etc) are blacklisted - the chrono namespace and library is whitelisted.

1.1 Task 1. Loop Invariants. 7.5 points

Below is pseudocode for an algorithm that performs linear search. Input: Array A[1..n] of integers and an integer k.
Output: TRUE if k is found in A, FALSE otherwise.

Algorithm 1: LINEARSEARCH(A, k)

```
Result: True if found otherwise False
i:= 1
found := FALSE;
while i≤n and found=false do
    if A[i]=k then
        found:= TRUE
    end
    i:= i+1
end
return found
```

1. State a loop invariant for the while loop of the LinearSearch algorithm, and prove your claim.

2. Modify the algorithm so that it performs both forward and backward search until finding the element k.

   i) The first and the last elements of the array are checked.

   ii) If they do not contain k, the second and next to last elements are checked and so on until k is found.

   iii) State a loop invariant for the while loop of the modified algorithm and prove/support your claim.

1.2 Task 2 Comparing Sorting Algorithms. (10 Points)

You are to empirically compare two sorting algorithms, one with a worst-case running time of O(n2) and another one with a worst-case running time of O(n log n).

The question we are interested in is whether the size of the data that are sorted has an influence on the outcome of the comparison. In particular, we would like to know for which length of input arrays the second algorithm is faster than the first, depending on the size of the data in the arrays.
- All storage for this experiment must be dynamically allocated and cleaned up (memory released not leaked) at the conclusion of your program.
- Input array size should be no less than 100,000 for initial stages and should not exceed 20,000,000 for the largest size experiment (upper bound should be user defined at run time).

- 1. Write a C++ program implementing the insertion sort algorithm.
- 2. Write a C++ program implementing the merge sort algorithm.
- 3. Create two versions of each algorithm:

  (a) one for arrays of standard int values of maximum 32-bits;

  (b) another one for arrays of long long int values.

  Then compare the performance of the two algorithms

  (a) for arrays with random standard ints and

  (b) for arrays with random long long int in the range from -9,223,372,036,854,775,807 to 9,223,372,036,854,775,807.

  Gradually increase the size of the input array until the difference between the two algorithms is noticeable.
- 4. Does the size of the integers affect the performance of the algorithms?

  Which array size is the crossover point, from where on the asymptotically better algorithm has the better performance?

2 Submission

1. Report outlining experimental observations and analysis for various iterations.
2. Source code for all tasks must follow the C++ Style Guide, compile error and warning free on the g++ or clang compiler.

Last updated: September 4, 2020

Typeset using TEXShop