Manual Submissions

Assignment - In progress

Add attachment(s), then choose the appropriate button at the bottom.

Title	Assignment 6 - Spring 2017
Due	Apr 24, 2017 5:00 pm
Number of resubmissions allowed	10
Accept Resubmission Until	Apr 25, 2017 12:00 am
Status	Not Started
Grade Scale	Points (max 100.0)

Instructions

Assignment 5

Overview

For this assignment, you will complete searching/sorting tasks and efficiency analysis. No code is to be written for this assignment. Write your answers in the file assign6.txt.

Problem 1

1. Trace selection sort on the following array of letters (sort into alphabetical order):

   ```
   A X B T S Q C
   ```

   After each pass (outer loop iteration) of selection sort, show the contents of the array and the number of letter-to-letter comparisons performed on that pass (an exact number, not big-O).
2. Trace insertion sort on the following array of letters (sort into alphabetical order):

   ```
   A X B T S Q C
   ```

   After each pass (outer loop iteration) of insertion sort, show the contents of the array and the number of

letter-to-letter comparisons performed on that pass (an exact number, not big-O).

Problem 2

For each problems segment given below, do the following:

1. Create an algorithm to solve the problem
2. Identify the factors that would influence the running time, and which can be known before the algorithm or code is executed. Assign names (such as *n*) to each factor.
3. Identify the operations that must be counted. You need not count every statement separately. If a group of statements always executes together, treat the group as a single unit. If a method is called, and you do not know the running time of that method, count it as a single operation.
4. Count the operations performed by the algorithm or code. Express the count as a function of the factors you identified in Step 2. If the count cannot be expressed as a simple function of those factors, define the bounds that can be placed on the count: the best case (lower bound) and worst case (upper bound).
5. Determine what the Best Case Inputs are, and the Worst Case Inputs are, and the efficiency of your implementation
6. Transform your count formula into big-O notation by:
   - Taking the efficiency with worst case input,
   - Dropping insignificant terms.
   - Dropping constant coefficients.

a. Determine if 2 arrays contain the none of the same elements elements (assume all elements are distinct)
b. Counting total number characters that have a duplicate within a string (i.e. "gigi the gato" would result in 7 (g x 3 + i x 2 + t x 2)
c. Finding a row where every entry is 'a' in a 2-D array.

Submission

Attachments

No attachments yet

Select a file from computer Choose File no file selected or select files from workspace or site

Submit Preview Save Draft Cancel

Don't forget to save or submit!

- Office of Instructional and Research Technology
- sakai@rutgers.edu
- 848.445.8721
- The Sakai Project