

Assignment 1

Searching in Arrays

Release Date	Due Date
September 14, 2017	September 28, 2017

Objectives

- Experience various techniques to search arrays
- Practice developing high-performance solutions
- Compare theoretical vs empirical complexities
- Compare linear and binary search of arrays

Problem Specification

Linear and binary searches are common techniques to search in an array. In this assignment we will get experience in implementing and comparing their performances. We will also gain experience in extending binary search. Here we go.

Write an application to solve the following problem:

- 1) Generate an $n \times n$ two-dimensional array with random lower case letters in the range $[a, z]$. Note that n is a program input with a minimal value of 2.
- 2) Search your first name. Give the indices for each of the letter that is in your first name. If a letter of your name does not appear in the random array, the index is $[-1, -1]$. Report the total time spent on searching.
- 3) Sort the two-dimensional array in ascending order (see the sample output below). You may use Java's `Arrays.sort` utility or any other built-in sorting method.
- 4) Search your first name again. This time you should use binary search. Give the indices for each of the letter that is in your first name. If a letter of your name does not appear in the random array, the index is $[-1, -1]$. Report the total time spent on searching.
- 5) Randomly replace a letter in your name $m > 0$ times and search the new name (i.e., repeat the searching of steps 2 and 4 with new names m times, and that each time a new name gets generated by first randomly finding the letter to replace in your name and replacing that letter with a random letter). Report the time spent on searching.

- 6) You should measure run-time complexities of your implementations.

Pause: If your code runs very fast (especially for small input sizes) then the elapsed time (using system supplied time functions) might be too small to be recorded. What can you do to remedy the situation?

Longer Pause: Binary search works on a sorted array. So for multiple searches, there is no point in sorting repeatedly and then searching or searching repeatedly using linear searches, but sort once and then perform multiple searches. Find the range of m for your code where the solution of m linear searches is better than the solution of sort and m binary-searches.

Sample output

Assume the input n is 3 and the name to search is *ajay*, following is the type of output we expect:

Random matrix:

```
f s m
a f d
i e q
```

```
a: [1, 0]
j: [-1, -1]
a: [-1, -1]
y: [-1, -1]
search time: 0.001 seconds
```

Sorted matrix:

```
a d e
f f i
m q s
```

```
a: [0, 0]
j: [-1, -1]
a: [-1, -1]
y: [-1, -1]
search time: 0.0005 seconds
```

Note that the second “a” in *ajay* does not exist in the array because there is only one “a”.

Output Requirements

1. If the value of n is less than or equal to 10, you must display the random and sorted two dimensional arrays.
2. If the value of n is greater than 10, please do not display any matrix. Just print the search results and the time usage.

Design Requirements

Code Documentation

For this assignment, you must include documentation for your code. This include how to compile and run your program. Also, you **must** give outputs of your program with $n=5$, $n=10$, $n=100$ and $n=1000$. If you don't provide outputs for these 4 inputs, it means your program does not execute properly. Note that your outputs include arrays for $n=5$ and $n=10$, but not for $n=100$ and $n=1000$. Choose values of m to be at least 1, 2, 4, 8, 16, 32, 64, 128, and 256.

Coding Conventions and Programming Standards

You must adhere to all conventions in the CS 3310 Java coding standard. This includes the use of white spaces for readability and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions for naming files, classes, variables, method parameters and methods. Read the material linked from our class web-pages (in case you can't recall programming styles and conventions from your CS1 and CS2 courses).

Testing

Make sure you test your application with several different values capturing different cases, to make sure it works.

Assignment Submission

- Generate a .zip file that contains all your files, including:
 - Source code files
 - Including any input or output files
 - Documentation of your code – e.g. using Javadoc if using Java
 - A brief report (in a pdf file) on your observations of comparing theoretical vs empirically observed time complexities. Note this report will include (a) a brief description of problem statement(s), (b) algorithms descriptions (if these are standard, commonly known algorithms, then just mention their names along with customization to your specific solution(s), otherwise give the pseudo-code of your algorithms, (c) theoretically derived complexities of the algorithms used in your code, (d) table(s) of the observed time complexities, and (e) plots comparing theoretical vs. empirical along with your observations (e.g. do theoretical agree with your implementation, why? Why not?).
- Don't forget to follow the naming convention specified for submitting assignments