GTU Department of Computer Engineering

CSE 222/505 - Spring 2022

Homework #04 Report

Şiyar Tarık Özcaner

20010400421

1. System requirements

In this homework we are asked to implement 6 recursive methods, find out their time complexities and prove 2 of them by induction method (latter 2 being extra) to improve our understanding of recursive methods.

1. Class diagrams

* Is in the directory

1. Problem solutions approach

Q1:

This method looks at indexes and one by one sees if it matches with the small string, n for bigger string’s length.

Best: Θ(1)

Worst: Θ(n)

General: O(n)

Proof:

Since it controls if starting index is the same length with the length of the array and it increases this index one by one it approaches the base case while it approaches base case it also looks contiguous block to see if it matches smaller string we can say and see that this function works.

Q2:

This method with the help of binary search finds the indexes of boundaries and returns the difference of them.

Best: Θ(1)

Worst: Θ(logn)

General: O(logn)

Standard binary search algorithm is executed, by time it decreases start and end variables’ values to be equal and by this approaches base case which also means it will return the wanted value, this observation tells us that this method does what we expect from it.

Q3:

This method sums contiguous blocks and stops until if its equal to target or bigger than it if it is equal to target it returns the starting and ending indexes of the block.

Best: Θ(n)

Worst: Θ(n^2)

General: O(n^2)

Q4:

This method looks at indexes and one by one sees if it matches with the small string, n for bigger string’s length.

Best: Θ(1)

Worst: Θ(n^log3)

Genaral: O(n^log3)

Q5:

This takes the length of the fillers and count to fill the array in every possible way with these values and it is also possible to send every possible filler size and count to get all iterations with extra operations.

Ex. 5 fillers with 3 length,

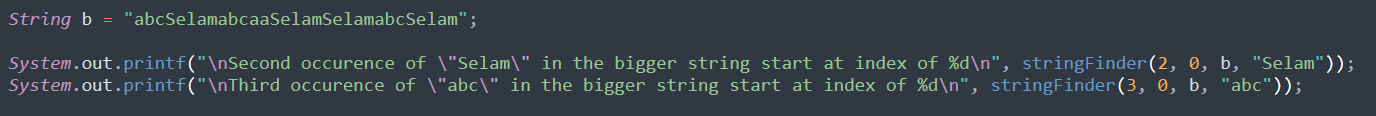
Best: Θ(1)

Worst: Θ(filler\_length\*filler\_count)

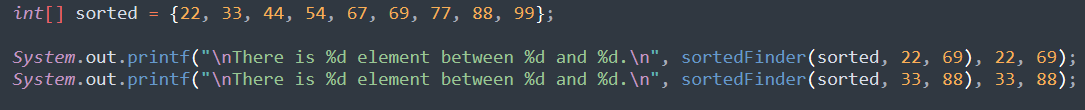
Genaral: O(filler\_length\*filler\_count)

1. Test cases

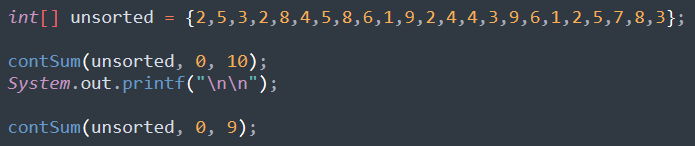
For Q1



For Q2



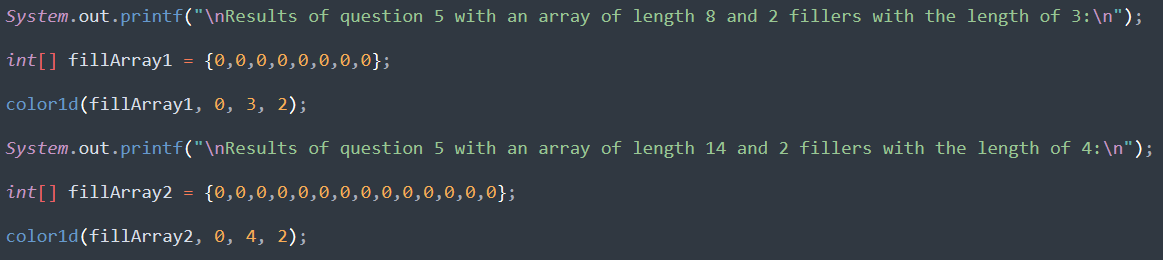
For Q3



For Q4



For Q5



1. Running command and results

make command is enough to run and see the results

