Weijia Xu

Professor Arun

Algorithms & Structured Programming CS455

September 16, 2023

Consider this variant of linear search as done in class in which we are given an array A containing n 64-bit numbers and a 64-bit number x, and asked to find all in the indices in A such that A[i] equals x. This paragraph applies to the first three questions below.

1. (3 pts) What are the worst-case and the best-case running times of this algorithm? Identify worst-case and best-case inputs (one for each)

Worst- case running time: O(n). This will occur when ‘x’ is not in the array or when ‘x’ is equal to every element in the array.

Best-case running time: O(1). This will occur when ‘x’ is equal to the first element in the array.

1. (2 pts) What are the worst-case and the best-case memory usages of this algorithm? Identify worst-case and best-case inputs (one for each).

Worst-case memory usage: O(n). In the scenario where ‘x’ equals every element in the array, the algorithm will need to store n indices.

Best-case memory usage: O(1). In the best -case scenario, ‘x’ is not in the array or it’s only equal to the first element, so we don’t have to store many indices.

1. (5 pts) Implement this algorithm in Python. Run it on worst-case and best-case inputs for each of running time and memory usage.

A screen shot of a computer

Description automatically generated

**Question 4** (6 pts): Consider this problem. We are given an array Y which contains positive integers. For any two indices i, j such that i < j, define

A(i, j) = (j-i)\*min(Y[i], Y[j]). The problem is to find a pair i, j where i < j for which A(i, j) is the maximum possible.

Here is an example. Y = [8, 4, 1]. A(0, 1) = (1-0)\*min(8,4) = 4. A(0,2) = (2-0)\*min(8.1) = 2. A(1,2) = (2-1)\*min(4, 1) = 1. So the optimal answer is (i=0, j=1).

Design a greedy algorithm to solve this problem, implement it in Python, and run it on the following two inputs. Some points will be deducted if your algorithm finds good but not optimal solutions.

Y = [10, 3, 8, 4, 19, 7, 12]

Y = [5, 9, 3, 10, 4, 7, 11]

Explain what makes this algorithm greedy. What is the worst-case running time of your algorithm as a function of n, the number of elements in Y?

A screen shot of numbers and numbers

Description automatically generated

Greedy: Because it makes the locally optimal choice by moving the pointer that points to the smaller value, trying to maximize the minimum value of Y[i] and Y[j] and hence potentially increasing A(i, j).

Time Complexity: The worst-case running time is O(n2), where n is the number of elements in Y, because there have a nested loop iterating over all pairs.

Space Complexity: The space complexity of the algorithm is O(1) as we only use a constant amount of extra space to store the indices and the maximum value.