CSC 6013 – Summer 2022

Coding Assignment for Brute Force Algorithms:

Selection Sort, Bubble Sort, and Polynomial Evaluation

Submit your work as a docx or pdf file through Blackboard.

1) a) Write Python code for the selection sort algorithm to sort an array into ascending order. Modify the code in the class notes to do three things:

i) After k iterations through the outer loop, the k **LARGEST** elements should be sorted rather than the k **SMALLEST** elements.

ii) After each iteration through the outer loop, print the array. After the kth iteration, you should see that the k largest elements have been placed into the last k slots of the array.

iii) Throughout the run, count the number of times two array elements are compared. Be sure to count the number of comparisons not just the number of comparisons that evaluate to True. When the algorithm concludes, display the total number of comparisons of array elements required to sort the array.

b) Check your algorithm on the problem instances:

A1 = [63, 44, 17, 77, 20, 6, 99, 84, 52, 39]

A2 = [84, 52, 39, 6, 20, 17, 77, 99, 63, 44]

A3 = [99, 84, 77, 63, 52, 44, 39, 20, 17, 6]

2) a) Write Python code for the bubble sort algorithm to sort an array into descending order by making the smallest elements "bubble up" and end up in the last entries of the array. Modify the code in the class notes to do four things:

i) After k iterations through the outer loop, the k **SMALLEST** elements should be sorted and placed at the end of the array rather than the k **LARGEST** elements.

ii) After each iteration through the outer loop, print the array. After the kth iteration, you should see that the k smallest elements have been placed into the last k slots of the array.

iii-iv) Throughout the run, count the number of times two array elements are compared and the number of times two elements are swapped. When the algorithm concludes, display the total number of comparisons of array elements and the total number of swaps required to sort the array.

b) Check your algorithm on the problem instances:

A4 = [44, 63, 77, 17, 20, 99, 84, 6, 39, 52]

A5 = [52, 84, 6, 39, 20, 77, 17, 99, 44, 63]

A6 = [6, 17, 20, 39, 44, 52, 63, 77, 84, 99]

3) Polynomial evaluation

a) Calculating the p'th power of a real number

Write a Python function

power (x, p)

to calculate the value of x^p for any real number x and any non-negative integer p.
The brute-force function should use a for loop to repeatedly multiply the value of x.
Do not use Python's exponentiation operator ** to evaluate x**p.

b) Polynomial evaluation

A polynomial can be represented as an array in which the coefficient of the k'th power of the variable is stored in array entry A[k]. For example, the array whose values are:

k	0	1	2	3	4	5	6
A[k]	12.3	40.7	-9.1	7.7	6.4	0	8.9

would represent the polynomial

$$f(x) = 12.3 + 40.7x - 9.1x^2 + 7.7x^3 + 6.4x^4 + 8.9x^6.$$

To evaluate this function at x = 5.4 we would "plug in" 5.4 for each occurrence of the variable x and perform the indicated arithmetic.

Write a Python function

evaluate (A, x)

that determines the value of f(x) for the polynomial that is represented by the corresponding array A of coefficients.
For each term of the polynomial, this function should make a call to the power() function that you wrote in part 3a.

c) Run your code to evaluate the polynomial

$$f(x) = 12.3 + 40.7x - 9.1x^2 + 7.7x^3 + 6.4x^4 + 8.9x^6$$

at x = 5.4

d) Asymptotic analysis

Determine the maximum number of multiplications that are needed for any polynomial of degree n (not just for this instance with a polynomial of degree 6). For your initial work, you can identify a sum of terms. For your final answer, give a simple expression for the sum (involving a power of n, not a summation Σ). What is the Big-Oh class for this algorithm?